Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems Submitted 8/2021; published 11/2021

A Cognitive Systems Approach to Continuous Control

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

This paper presents a theory that unifies the frameworks of symbolic problem solving and continuous control at a deeper level than previously attempted. The approach builds on the earlier PUG architecture, but it also incorporates some important new ideas: symbolic concepts are grounded in numeric attributes and can match to different degrees; symbolic skills include equations for control attributes that are modulated by mismatches of target concepts; and reactive execution uses these skills to compute control values based on mismatches. We describe C^3, an implemented version of this theory and demonstrate its behavior in a simulated continuous robot environment. We conclude by discussing links to earlier work and directions for future efforts.

1. Introduction and Motivation

Early research in artificial intelligence focused on high-level tasks such as problem solving and multi-step reasoning, and the cognitive systems paradigm has carried on this tradition. However, humans also operate in a physical environment that requires them to exert fine-grained control over their effectors to achieve concrete goals. In addition, early AI efforts emphasized the role of symbolic mental structures, including the use of domain knowledge to guide behavior. The cognitive systems movement has also adopted this idea as one of its core principles. Yet physical environments have a continuous character that is difficult to handle with discrete representations alone, indicating the need for something more.

In contrast, the paradigm of cybernetics and control has directly addressed fine-grained application of effectors in the external world. Moreover, to support this ability, it has relied on continuous encodings of physical state and quantitative control equations. At the same time, the control framework offers no obvious approach to explaining high-level cognitive processes, including ones like planning that are necessary for achieving agents' physical goals. Neither does it take advantage of clear benefits offered by relational representations of situations and expertise. Clearly, neither paradigm provides a complete account of embodied intelligence, which poses an intellectual challenge for the broader research community.

In this paper, we present an alternative account of intelligent physical agents that offers a deep unification of the two views, showing that they are not incompatible. We start by reviewing how symbolic problem solving and continuous control each approach tasks that involve sequential de-

© 2021 Cognitive Systems Foundation. All rights reserved.

cision making. Next we describe a unified theory with clear postulates about the agent's representation and the mental processes that interpret them. After this, we report an architectural module that builds on these theoretical ideas, along with empirical demonstrations of its behavior in a simulated robotics environment. Finally, we review our framework's key ideas and their connections to previous research on symbolic AI and continuous control.

2. Symbolic Problem Solving and Continuous Control

Consider the scenario in Figure 1 (a), in which a robot on the left must journey to the target object on the right while avoiding collisions with the two obstacles between them. The robot can sense its distance and angle to each object as it goes through the environment, which it does by invoking actions that move it forward and change its orientation. This task, like others that involve more complex physical situations, involves making a series of decisions to produce a trajectory over time that achieves the robot's goals, like the one shown in Figure 1 (b). This holds whether the agent executes its actions reactively in the environment or plans its motion mentally in advance, although the latter requires a model of actions' effects. The AI and robotics communities have explored two broad paradigms for tackling such problems, which we review in turn.

Research on symbolic planning and heuristic search grew out of studies of human problem solving (Newell, Shaw, & Simon, 1960). This framework typically encodes states as collections of discrete elements and assumes actions cause qualitative changes that produce new states. The fact that different actions are possible in a situation produces a space of possible paths the problem solver must guide in some manner, often by examining differences between the current state and the goal description. Classic work has focused on abstract tasks like puzzles and logistics planning, which one can easily encode in discrete terms, but researchers have adapted the paradigm to motion planning tasks like that in the figure. A common approach is to discretize space into a fine-mesh grid and then applying heuristic search to find a path from the initial to the target location. For the task in Figure 1, this might lead to hundreds of grid cells and a solution paths with tens of steps. Other techniques, like probabilistic road maps (Kavraki et al., 1996), instead generate a set of discrete waypoints and then look for a path through a subset of them. This might produce only a few waypoints for our scenario, but it would not lead to the smooth motion in Figure 1 (b). Both schemes attempt to partition inherently continuous environments, which leads to large search spaces for what are reasonably simple tasks for humans.

A different paradigm – feedback control – grew out of the cybernetics movement (Bennett, 1996), which emphasized analog representations. 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' (PID) methods being a standard way to calculate control values. For settings that involve multiple objectives, a common variant involves the use of potential fields (Khatib, 1985). These compute a set of attractive or repulsive forces, whose effects are summed and provided

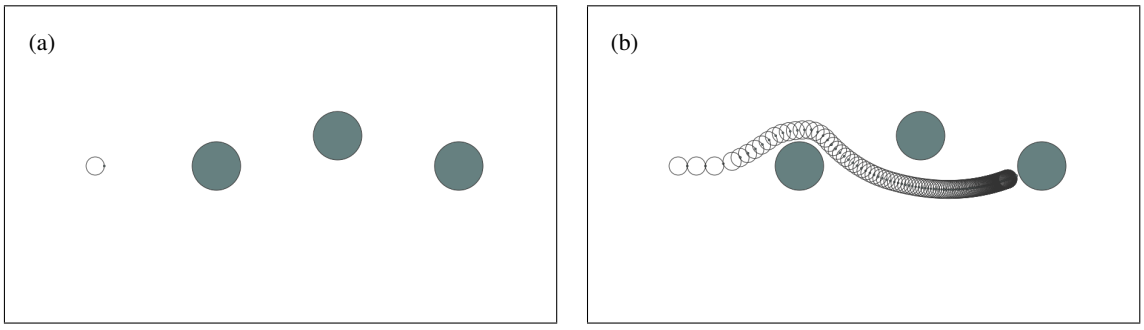

Figure 1. (a) A scenario in which a robot must approach a distant object while not colliding with two closer objects and (b) a trace of reasonable behavior that satisfies these aims. The problem requires reasoning about qualitative relations but also depends on continuous control.

to control equations, leading to smooth behavior like that in Figure 1 (b). However, these approaches all assume a single set of formulae suffice to determine agent behavior in a given setting.

In summary, each paradigm has strengths and weaknesses. Continuous control excels at low-level tasks that require fine-grained, adaptive execution, and it can even be used for motion planning when a predictive model is available, but it assumes that one set of control equations is appropriate throughout the space. In contrast, symbolic search is well suited to high-level task planning that requires heuristic choice among discrete sequences of coarse-grained actions. These differences have encouraged researchers to combine ideas from the two frameworks. *Hybrid control* methods provide a finite-state graph of 'modes', each with its own control equations (Goebel, Sanfelice, & Teel, 2009). This breaks the environment into a set of discrete regions, but it requires careful engineering to specify the mode structure and boundaries. Another approach uses a symbolic problem solver to find a plan with discrete operators, then invokes a continuous controller to generate low-level behavior for each abstract step (e.g., Laird et al., 2012; Salvucci, 2006),

These responses provide techniques for *integrating* discrete problem solving with continuous control, but they do not attempt to *unify* them in a single framework. One effort along in the latter direction is the PUG/X architecture (Langley et al., 2017), which uses symbolic skills to search for abstract multi-step plans, but which also associates difference equations with each skill that it uses to simulate continuous trajectories and evaluate candidate solutions. Skills also refer to symbolic concepts that are grounded in numeric attributes of perceived objects, which the system uses to infer relational beliefs about the agent's situation. Despite these features, PUG/X does not support error-driven control and it can execute only one skill at time, which means that it cannot handle scenarios like the one in Figure 1. An improved theory would retain the architecture's key contributions but also import ideas from the control literature to support more adaptive behavior.

3. A Cognitive Theory of Continuous Control

As just noted, we desire a theory of continuous control that unifies ideas from these two distinct traditions. Any theory should address phenomena that it aims to explain. Here we are concerned with characteristics of human behavior in physical environments, including the ability to:

- Combine symbolic relations with numeric attributes to describe the current situation;
- Encode discrete actions with continuous parameters to specify fine-grained activities;
- Infer both relational descriptions and quantitative details of the agent's situation;
- Exhibit adaptive control that takes into account both the agent's situation and its objectives;
- Carry out multiple activities in parallel while balancing their contributions to agent objectives.

In this section, we present such a unified computational account of these capabilities. We focus first on the theory's commitments about the representation of mental structures and then turn to its claims about the processes that interpret and manipulate them.

3.1 Representational Postulates

Our theory makes a number of commitments about the cognitive elements that underlie continuous control. Each statement reflects the assumption that mental constructs combine symbolic relations with quantitative attributes. A few earlier frameworks, like PUG/X, have attempted this unification, but none have explored its potential fully. The theory's first tenet identifies two types of knowledge structures on which cognitive control depends:

- *Cognitive control draws on two types of long-term mental elements: concepts and skills*.

Briefly, the first define the terms an agent uses to describe its environmental situation and objectives, whereas the second specify how to achieve those aims.

Additional postulates provide details about the form and content of these long-term mental structures. In particular, we hold that:

- *Concepts encode generic symbolic relations among entities that are grounded in numeric attributes associated with those entities*.

This assumes that the agent's perceptual system produces a set of entities or objects, each with an associated set of quantitative attributes. Both unary categories (e.g., tower, disk) and relational concepts (e.g., facing, between) are defined in terms of these features. This is directly related to claims that cognition is *embodied*. The notion of grounded symbols lends itself to another theoretical statement about the nature of these categories:

- *Concepts are graded in that they can match against situations to different degrees*.

Naturally, this match score is linked to numeric attributes used to define concepts. For instance, the relation *behind* can be satisfied by two objects to a greater or lesser extent based on their distance and angle to the perceiving agent. The idea that some entities or situations fit a concept description better than others is akin to the notion of *typicality* in categorization (Rosch & Mervis, 1975).

In addition, the theory of cognitive control specifies how long-term, stable concepts relate to their short-term, dynamic counterparts:

- *Beliefs are concrete instances of concepts that specify symbolic relations among entities and have associated numeric attributes*.

For example, an agent might hold the belief *(between R1 O1 O2)* that, from robot *R1*'s perspective, it is between objects *O1* and *O2*, where *between* is defined in conceptual memory. However, because concepts can be satisfied to a greater or lesser extent, we also postulate:

- *Beliefs specify the degree to which their corresponding concepts match against a situation.*

In this case, the degree to which this belief matches the concept will depend on relative distances and angles to the robot. As the agent or objects move in the environment, these quantities may change, causing the match score to change in response, but the basic relation remains the same.

Of course, beliefs seldom occur in isolation; they invariably appear in combination with others. Following standard usage, we will refer to a collection of beliefs that the agent holds at the same time as a *state*. In classic AI planning research, a state is simply a conjunction of relations, which is also the case in our theory. But we can also view a state as a point in an N-dimensional space, where each dimension refers to a belief and each value denotes the degree to which it matches the corresponding concept. Note that the state encodes a *model* of the environment that the agent holds in memory. It may not be an accurate depiction of the actual situation, and it may even represent a scenario that the agent believes is not true but it desires to achieve.

The theory also includes postulates about agent activities that can lead to changes in its beliefs, its mental states, and its external environment. The first such claim is that:

- *Skills encode generic discrete activities but include continuous equations for control attributes.*

For example, an agent might have a skill for approaching a target object or turning to face that object. One would include an equation for calculating the value of a control attribute for the rate of forward motion or the force to be applied in that direction. The other would have an analogous expression for the angular rate or rotational force. These are similar to equations used in feedback control systems, but here they are associated with symbolic skills reminiscent of operators in planning systems. Moreover, the theory places constraints on their form:

- *Each skill has an associated target concept and its control equations are functions of the degree to which this concept is mismatched.*

For example, a robotic agent's skill for moving to an object might include a *robot-at* relation it aims to achieve, whereas one for turning to an object might have a *robot-facing* 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.

However, just as an agent must instantiate its general concepts to encode particular beliefs, so it must concretize its generic skills:

- *Intentions are instances of skills that refer to specific entities, including a target belief.*

For instance, the agent might desire to approach the rightmost object in Figure 1. In this case, the intent might be *(move-to R1 O3)* and the target belief might be *(robot-at R1 O3)*. As with beliefs, the structures of intentions are stable over time, but their quantitative elements, such as the mismatch of their target and the values of control attributes, will vary. In many situations, it is important to use multiple skills in parallel, so the agent must retain a *set* of intentions.[1]

1. Clearly, some intentions are mutually exclusive, such as avoiding an obstacle on the left and the right. We will assume here that such knowledge is invoked at higher levels of the agent architecture.

3.2 Processing Postulates

Our theory of cognitive control also makes commitments the mechanisms that operate over mental structures. As in most treatments of both cognition and control, these involve the repeated use of long-term content to update dynamic elements. This leads to the first postulate about processing:

- *Cognitive control operates in two-stage cycles that involve repeated application of conceptual inference and skill execution*.

This statement moves beyond the recognize-act cycle assumed by production system architectures (Neches, Langley, & Klahr, 1987), as it divides iterative cognitive behavior into two distinct stages. This separation comes closer to Nilsson's (2001) design for a *tower architecture* and to ICARUS, which relies on a similar separation between inference and execution.

The theory makes additional, more detailed statements about each of these processing stages. The first of these postulates claims that:

- *On each cycle, conceptual inference matches concepts against percepts and beliefs to generate new beliefs that provide a model of the agent's environment*.

This assumes that, on each cycle, the perceptual system produces a set of entities, each described as a collection of attribute-value pairs. The inference process matches concept definitions against these percepts, or against existing beliefs, to produce new beliefs that make up the current state. This phase has much in common with deductive reasoning systems, the key difference being that it calculates a degree of match for each belief. We will not take a stance on details, such as whether processing is data driven or guided by expectations, or whether it produces all derivable beliefs.

The theory incorporates similar clams about the mechanism of skill execution, which builds on results of the conceptual inference process:

- *On each cycle, skill execution evaluates control equations for each applicable intention to determine the values of control attributes*.

This is a variation on feedback control, in that calculation of the control value is a function of an error signal. However, this signal comes from the degree of mismatch between the target concept (e.g., `(robot R1 O3)`) and the corresponding belief.

Recall also that the agent can hold more than one intention at the same time, which raises issues about how to handle their interactions. Here the theory claims that:

- *To determine the values for control attributes on each cycle, skill execution sums the values computed for each intention*.

For instance, if one intention has the target of being at the object A but another wants to avoid object B, the first may indicate that the agent turn in one direction and the second may state the opposite. In such cases, the two skills' influences on control attributes are added, in much the same way as the vector sums that play a central role in potential field approaches to control.

4. The C^3 Architectural Module

We have incorporated these ideas into C^3 (*Cognitive Continuous Control*), a module designed for use in the PUG/X architecture (Langley et al., 2017). This earlier system also relied on grounded concepts and embedded continuous behavior within symbolic skills, but it did not support graded

Table 1. Two C^3 concepts for the robot domain, each of which includes relational head, a set of observed entities, and an expression for veracity (degree of match) as a function of the entities' attributes. The match function $(pyramid\ obs\ max\ min)$ returns 1.0 if the observed value $obs = max$, 0.0 if $abs(obs) \geq min$, and $abs(obs/(max - min))$ within that range. The function is symmetrical for positive and negative values.

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

categories, feedback control, or parallel intentions. We start by discussing C^3's cognitive structures and then clarify how it interprets them to achieve an agent's objectives in an environment.

4.1 Representation in the C^3 Module

C^3 provides a programming language that supports the construction of cognitive control systems. The syntax covers both long-term knowledge structures that remain stable over time and short-term elements that can change during processing. This extends the PUG/X notation in important ways. Table 1 shows four *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 its own identifier and set of numeric attribute values, and a `:tests` field that contains Boolean tests that must be satisfied for the concept to match. An optional `:binds` field introduces new variables defined as arithmetic combinations of ones in the `:elements` field that can be used in the head or the `:tests` field. Most novel is a `:veracity` field that specifies how to compute the concept's degree of match as a function of bound variables; this supports the notion of graded category membership.

Table 2 presents examples of *percepts* and *beliefs* that describe the scenario in Figure 1 from the robot's egocentric perspective. The three percepts – primitive beliefs that come directly from the environment – 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 six 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 a symbolic predicate; they include attribute values derived from their constituent entities. Moreover, each belief includes a `:veracity` score that reflects the degree to which it matches its respective concept. For instance, `(robot-facing R1 O2)` has the score 0.842 because the instantiated veracity expression `(pyramid 7.125 0.0 45.0) = 0.842`. This score, along with derived attribute values, can change over time while the symbolic aspects remain stable.

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 ^fuel 50 ^radius 0.15) [1.0]
(object ^id O1 ^distance 2.0 ^angle 0.0 ^radius 0.4) [1.0]
(object ^id O2 ^distance 4.0311 ^angle 7.125 ^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 4.0311) [0.632]
(robot-facing ^id (R1 O1) ^angle 0.0) [1.0]
(robot-facing ^id (R1 O2) ^angle 7.125) [0.842]
(robot-facing ^id (R1 O3) ^angle 0.0) [1.0]
```

The C^3 module also provides a syntax for *skills* that describe how to achieve the agent's objectives. Table 3 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 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*. [2]

Just as beliefs in C^3 are instances of general concepts, so *intentions* are instances of general skills. Each intention provides a skill name and arguments, along with a mismatch score for the target belief and values for the associated control attributes. The symbolic aspects remain unchanged over time, while the numeric values vary as the robot or its environment changes. For instance, for the scenario in Figure 1, the agent might have the intention set `((move-to R1 O3) (turn-right-to R1 O3)(turn-right-to R1 O3)(avoid-on-left R1 O1 O3) (avoid-on-right R1 O2 O3))`. The target mismatch and control values for each intention would vary over time, while its symbolic relation would remain the same.

4.2 Processing in the C^3 Module

Like many cognitive systems, C^3 operates in discrete cycles, but this occurs at two levels of processing. At the highest level, the system imports percepts from an external environment, invokes conceptual inference to generate beliefs that describe its current situation, uses intentions to compute control values, and exports the results to affect the environment. In this sense, C^3 is a classic teleoreactive controller that is driven both by goals (the target beliefs in active intentions) and by the situation (observed percepts and inferred beliefs). However, the module is distinctive in terms of what takes place in its inner cognitive cycle.

2. The mismatch score is simply one minus the match score reflected by the target belief's `:veracity` field.

Table 3. Two skills for the robot domain, each of which includes a relational head, a set of observed entities, arithmetic tests, control equation, and a target concept. The `move-to` skill influences the control attribute `move-rate`, while `turn-right-to` affects `turn-rate`. A third skill, `turn-left-to`, is not shown.

```
((move-to ?r ?o)
  :elements     ((robot ^id ?r ^fuel ?f)
                 (object ^id ?o ^angle ?a))
  :tests        ((> ?f 0.0) (> ?a -90) (< ?a 90))
  :control      ((robot ^id ?r ^move-rate (* 0.3 (- 1.0 VERACITY))))
  :target       ((robot-at ^id (?r ?o))))
((turn-right-to ?r ?o)
  :elements     ((robot ^id ?r ^fuel ?f)
                 (object ^id ?o ^angle ?a))
  :tests        ((> ?f 0.0) (< ?a 0))
  :control      ((robot ^id ?r ^turn-rate (* -10.0 (- 1.0 VERACITY))))
  :target       ((robot-facing ^id (?r ?o))))
```

This lower-level processing involves two stages. First the inference module finds all conceptual rules with elements whose types match against a subset of the current percepts. For each match, the system computes any derived attribute values, calculates the `:veracity` score, and, if this exceeds a global 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 exceed a threshold, which 0.5 by default. This procedure continues until no more conceptual matches occur; the resulting set of beliefs and percepts serve as the agent's model of the environment. 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.

In the second stage, C^3 examines its 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 the intention I's target belief and its associated `:veracity` score V. The interpreter substitutes V for the symbol `VERACITY` in I's control equations; it also substitutes the values for bound variables into these expressions. Next the system evaluates the instantiated expression to compute its contribution to the value for each control attribute. For example, given the intention `(move-to R1 O1)` and the `:veracity` score 0.519 for target belief `(robot-at ^id (R1 O1))`, the mismatch for would be `1 - 0.519 = 0.481` and the value for the control attribute `move-rate` would be `0.3 * 0.481 = 0.144`.

When more than one intention has satisfied conditions on a given cycle, the module computes the target mismatches and control values for each one. Different intentions may affect the same control attributes, in which case C^3 takes the vector sum, much as in potential field approaches to continuous control. The system passes these quantities to its effectors, typically in a simulated environment that serves as the external world. The two-stage procedure of conceptual inference followed by control occurs repeatedly to produce a trajectory through the state space over time. This continues until the intentions' target beliefs achieve high enough `:veracity` scores or until behavior reaches a steady state in which no changes occur.

As noted earlier, we have designed C^3 for incorporation into PUG/X, an agent architecture that combined a less flexible approach to continuous control with higher-level mechanisms to search for sequences of intentions that achieved complex goals. The new module itself does not carry out such task planning, but we should be able to substitute the new version for existing software in a straightforward manner. This should let PUG/X carry out the generation and execution of more robust task-level plans. Nevertheless, they are not the focus of the current paper, so we we will delay discussing them until the closing remarks.

4.3 Implementation and Use Details

We have implemented C^3 in Steel Bank Common Lisp. The module supports the syntax outlined above for skills, concepts, intentions, and beliefs, the first three of which can specify strategies for cognitive control. The software interprets these structures in a cyclic manner, calling on a submodule for conceptual inference to generate beliefs and another for reactive skill execution that applies intentions, calculates control values, and combines their effects. Thus, C^3 incorporates the postulates presented earlier about representations and mechanisms that underlie cognitive control. Moreover, it provides a programming language for goal-directed behavior in continuous domains.

To apply C^3 in a specific scenario, the user loads a file that contains a set of concepts and skills, along with an initial state that is provided to the simulated environment. One runs the module by calling it with two arguments: a set of intentions and the desired number of cycles. On each cycle, the interpreter retrieves percepts from the simulator, invokes conceptual inference to generate beliefs, uses matched intentions to determine values for control attributes, and passes their sums to the simulator, which updates the external world. This produces a trajectory over time through the environmental state, along with corresponding paths for the agent's beliefs and control attributes.

5. Empirical Demonstrations

In Section 3, we listed some abilities that people exhibit when pursuing physical activities. These included using both symbolic and numeric content to describe situations and actions, inferring situations from percepts, exerting fine-grained control appropriate to situations, and carrying out activities in parallel. We have presented a theory of cognitive control that addresses these abilities, along with C^3, an architectural module that incorporates its key assumptions. However, theories require more than internal coherence; they also need empirical support. In this section, we report demonstrations of the module's behavior on scenarios that benefit from cognitive control.

For this purpose 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. However, it has access to intrinsic values associated with its own body, such as fuel level. 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.

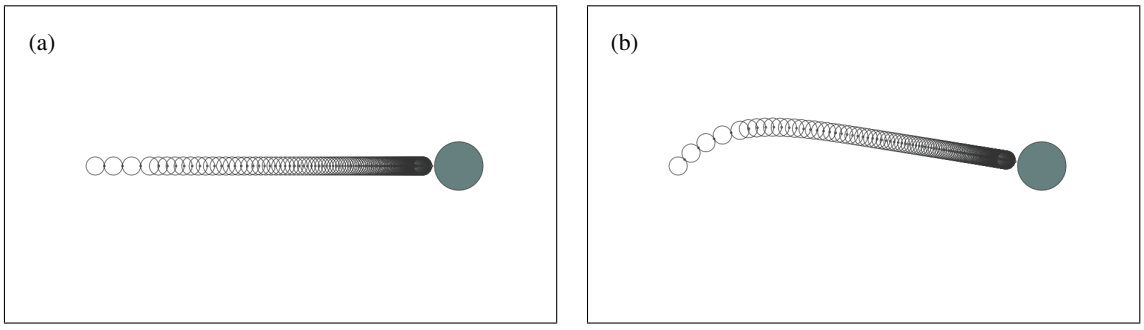

Figure 2. Two scenarios in which a robot must approach a nearby object: (a) when the robot is already facing the target, it simply moves forward until it reaches the object; (b) when the robot is facing elsewhere, it turns and moves forward simultaneously. Traces show the robot's pose at equal time intervals, illustrating that its position and orientation change more slowly as it approaches the objective.

5.1 Approaching a Target Object

Our first demonstrations show that the module supports the basic ability to approach a target object in the absence of other factors. We provided the system with two concepts `robot-at` and `robot-facing` from Table 1, along with the skills `move-to`, `turn-left-to`, and `turn-right-to` from Table 3. We also gave it three intentions (instances of these skills) – `(move-to R1 O1)`, `(turn-left-to R1 O1)`, and `(turn-right-to R1 O1)` – the first of them with `(robot-at R1 O1)` as a target belief and the latter with `(robot-facing R1 O1)`. Figure 2 (a) shows the behavior produced by these knowledge structures when `O1` is initially 6.0 units away from the robot and has an angle of zero. In this case, conceptual inference reveals that `(robot-facing R1 O1)` has a perfect match score of 1.0, so the two turning-related intentions do not come into play. However, the belief `(robot-at R1 O1)` has a `:veracity` of 0.423, causing `(move-to R1 O1)` to set the value for the control attribute `move-rate`. This starts at 0.3 but, once the robot comes with 5.1 units of `O1`, the control decreases in proportion to the distance, as seen by the smaller steps in the figure. The module continues for 145 cycles before the robot reaches the target object and halts without ever invoking the two intentions for turning.

A more challenging scenario requires the robot to turn to face the target object while approaching it. In this case, we gave C^3 the same concepts, skills, and intentions as before, but the initial orientation is such that `O1` is 45 degrees to the right. Figure 2 (b) traces the robot's behavior during this run. Conceptual inference shows that `(robot-facing R1 O1)` has an imperfect match score (initially 0.0), so the system invokes the intention `(turn-right-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-right-to R1 O1)` becomes inactive after cycle 137, as the robot is facing `O1` at this point and it needs only to move forward.

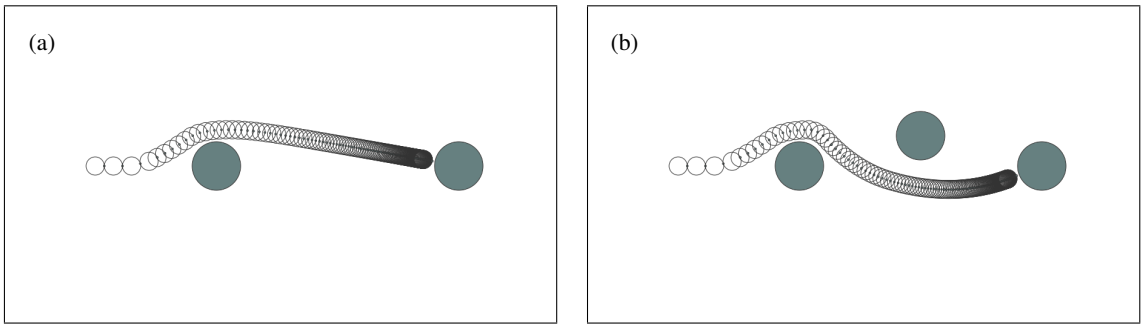

Figure 3. Two scenarios in which a robot must approach a target object while (a) avoiding one obstacle and (b) avoiding two obstacles along the way. These runs rely on additional skills for turning to the left and right around obstructions. Traces show the robot's position and orientation at equal time intervals. Table 2 reflects the agent's percepts and inferred beliefs on the first cycle of the second run.

5.2 Avoiding Obstacles

We have also tested our module for cognitive control 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 previous runs. However, we also 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 added two skills for avoiding objects on the left and on the right, both with `approaching` as their 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 3 (a) shows the resulting behavior when the agent must avoid a single obstacle on its way to the target, whereas Figure 3 (b) displays a trace when it must bypass two of them. In the first run, the intention `(avoid-on-left R1 O1 O3)` 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`. In the second run, the system repeats its initial behavior, but its trajectory diverges when it comes close to `O2`, the second obstacle. Starting on cycle 14, `(avoid-on-left R1 O2 O3)` causes the robot to turn in the other direction, but its influence ends by cycle 18, when a 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. A third scenario with five obstacles produced similar results. These traces provide more evidence that C^3 combines intentions in an effective manner, reproducing the behavior of potential fields in classic control systems.

5.3 Following a Moving Target

To demonstrate further C^3's support for adaptive cognitive control, we presented it with the same concepts, skills, and intentions as in prior scenarios, but with moving target objects. Figure 4 (a)

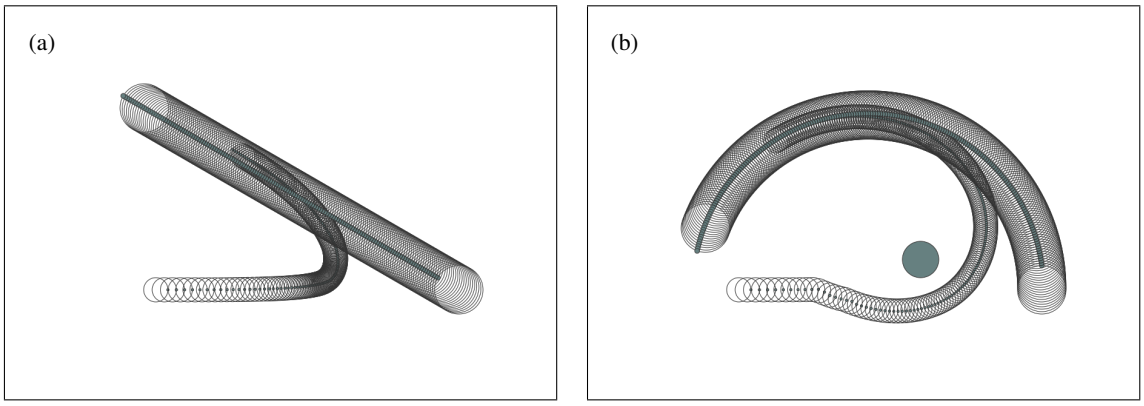

Figure 4. Two scenarios in which a robot must (a) approach a target object that is moving in a straight line and (b) approach a target that is moving in an arc while avoiding an obstacle. These runs rely on exactly the same skills as those in Table 3. Traces show the robot's pose at equal time intervals, illustrating that its position and orientation change more slowly as it approaches an objective, while targets move at constant speed.

shows a behavioral trace when the robot is initially facing the target `O1`, so it starts to move in its direction. However, the object is moving in a straight line, at a constant rate, to the upper left quadrant. The intentions `move-to` and `turn-left-to` cause the robot to turn in pursuit, but the agent does not know what physical processes are at work, so it cannot predict `O1`'s movements. As a result, the robot adjusts its values for `move-rate` and `turn-rate` in response, but it continues to lag behind the moving target, even though it has the ability to change position more rapidly. A more intelligent strategy would anticipate the object's motion and 'head it off at the pass', but that would require encoding and using a predictive model.

In another test, we provided the agent with the concepts, skills, and intentions described earlier for avoiding obstacles, then placed it in a pursuit scenario that required them. Figure 4 (b) shows a trace of the robot's motion when the target object `O1` moves in a arc rather than a straight line and when the agent encounters an obstruction in its path. The robot avoids this object by veering to the right, but the target object's trajectory curves in the other direction. The robot eventually maneuvers past the obstacle, but the detour has set its chase back considerably. Afterwards, the agent makes better progress and gradually comes closer to its prey without ever quite reaching it. As in the first pursuit, the agent never halts this activity because the target's motion continues unabated. This run provides further evidence that C^3's approach to cognitive control is effective at combining multiple intentions to balance different objectives.

6. Related and Future Research

Our theory of cognitive control incorporates multiple ideas from the literature, but it also moves beyond them in important ways. As we noted in Section 2, there has been considerable work on combining symbolic and numeric processing for sequential decision making, but none has attempted a deep unification. 'Hybrid' control systems (Bennett, 1996) embed continuous control within dis-

crete finite-state networks, with different equations for each mode, but they do not join the two paradigms at the control level. Similarly, extensions to the Soar (Laird, 2012) and ACT-R (Salvucci, 2006) architectures use feedback control in physical settings, but they remain separate, with discrete processing calling on continuous computation as a subroutine. Planning formalisms like PDDL+ (Cashmore et al., 2020; Fox & Long, 2006) augment symbolic operators with difference equations that apply over multiple cycles, but they do not support adaptive control. The ICARUS (Langley et al., 2009) and PUG/X (Langley et al., 2017) architectures support durative skills with continuous effects, but have similar limitations. In contrast, our framework offers a deeper unification of symbolic and continuous control.

On another front, there have been many efforts to combine inference with control. However, nearly all have 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, ICARUS, and PUG/X provide explicit mechanisms for elaboration of sensed percepts to form higher-level beliefs, but they do not 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 for adaptive regulation. Fuzzy controllers (Katz, 1997; Zadeh, 1968) employ membership functions in much the same way as C^3 but, again, encode only low-level state descriptions. Instead, our theory of cognitive control augments symbolic inference of relational descriptions with a degree of match to provide a continuous feedback.

A third key idea is that sequential action is guided by differences between the current and desired state. Of course, continuous control methods rely crucially on an error signal 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. Skills and associated target concepts also serve as the source of potential fields that are combined to balance the agent's competing objectives.

Still, these comparisons are not entirely justified. Recall that our venture into cognitive control was motivated largely by limitations of the existing PUG/X architecture. That framework incorporated durative skills that produced continuous effects, with their application conditioned on conceptual inferences that were grounded in percepts. However, the previous system did not support either adaptive control or degrees of conceptual match, much less use the latter to guide the former, and it could execute only one skill at a time. We intend the current theory, and its implementation in C^3, as a more flexible replacement for the lower levels of the earlier architecture, not as a complete account of goal-directed behavior in the large.

This observation suggests a number of topics for future research, These involve at replicating PUG/X's abilities for predicting future states, evaluating them, and generating multi-step plans that would achieve the agent's objectives, but doing so with a substantially improved module for cognitive control. These include:

- Adding a new type of generic structure – *processes* – that lets the agent predict changes to the environment as a function of the current state and values of control attributes (for the agent's actions) or from the current state alone (for natural forces like wind and friction);
- Mentally applying instances of skills (intentions) and processes in an iterative manner to produce a *mental simulation* – a sequence of environmental states and inferred beliefs – that predicts results of the agent's behavior over time;
- Introducing another type of long-term structure – *motives* – that specify how to calculate the utility for a given belief as a function of the current situation and using a sequence of such quantities to compute an overall score for the agent's trajectory;
- Carrying out heuristic search through a space of *motion plans* by considering alternative sets of agent intentions, mentally simulating the trajectories they produce, using using motives to calculate their utilities, and comparing the results to make a selection; and
- Conducting higher-level heuristic search through a space of *task plans* by examining alternative sequences of motion plans, mentally simulating the trajectories they generate, computing their utilities with motives, and selecting one or more high-scoring candidates.

Taken together, these would embed our new approach to cognitive control in an agent architecture that supports both motion planning at the continuous level and task planning at the symbolic level. At the same time, it would provide this framework with flexible mechanisms for feedback control that are nevertheless unified with classic ideas from the cognitive systems literature.

7. Closing Remarks

In this paper, we presented an account of sequential decision making that unifies ideas from symbolic problem solving and continuous control. After reviewing these two paradigms and their limitations, we described a new theory of goal-directed physical activity, first introducing postulates about cognitive structures and then claims about processes that operate on them. The framework assumes that symbolic concepts are grounded in numeric attributes, discrete skills incorporate control equations and target concepts, conceptual inference elaborates perceptions into relational state descriptions, reactive execution modulates calculation of control attributes by the degree to which target concepts match, and combines the effects of multiple skills by summing control values. Thus, it enriches ideas from the two traditions in ways that offer abilities that go beyond either in isolation.

In addition, we described C^3, an implementation of our theory for cognitive continuous control. The system provides a syntax for encoding concepts and skills, as well as an interpreter that combines this knowledge with percepts to control embodied agents. We reported demonstrations of the C^3's operation in a simulated robotics environment, including scenarios that required the agent to approach target objects, to reach targets while avoiding obstacles, and to pursue moving targets. These runs provided evidence that the module support the desired abilities we enumerated at the outset. After this, we examined the framework's relationship to earlier research that combines symbolic and numeric processing for sequential control. We also considered the role that C^3 might play within a more complete architecture for intelligent physical agents. More work remains, but the initial evidence suggests that our approach to cognitive control has substantial promise.

References

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

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.

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

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

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.

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

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.

Laird, J. E., Kinkade, K. R., Mohan, S., & Xu, J. Z. (2012). Cognitive robotics using the Soar cognitive architecture. *Proceedings of the AAAI-12 Cognitive Robotics Workshop*. Toronto, CA.

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., Choi, D., & Rogers, S. (2009). Acquisition of hierarchical reactive skills in a unified cognitive architecture. *Cognitive Systems Research*, *10*, 316–332.

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.

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