A Unified Formalism for Constructing Embodied Agents

Pat Langley
Institute for the Study of
Learning and Expertise

Edward P. Katz
Stanford Intelligent Systems Laboratory
Stanford University

Mohan Sridharan
School of Computer Science
University of Birmingham

Thanks to Mike Barley, Dongkyu Choi, and Ben Meadows for useful discussions. This research was supported by AFOSR Grant No. FA9550-20-1-0130 and Grant No. N00014-23-1-2525, which are not responsible for its contents.

Cognitive Architectures

A cognitive architecture (Newell, 1990) provides a framework for intelligent systems that:

- Makes *theoretical assumptions* about the representations and mechanisms underlying behavior
- Incorporates many ideas from *cognitive psychology* (e.g., symbol structures, stable vs. dynamic memories, cyclic operation)
- Contains distinct *modules* that access / alter the *same memories* and representations
- Provides a *programming language* with a high-level syntax that reflects its theoretical assumptions

This talk focuses on the formalism of PUG, an architecture for embodied intelligent agents.

PUG's Knowledge Structures

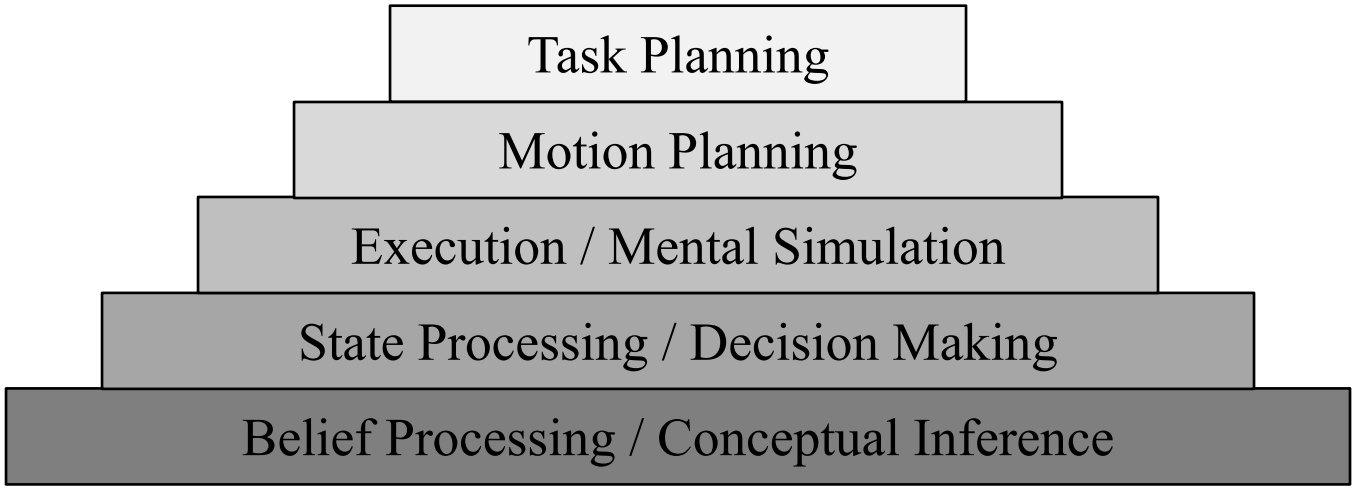
PUG incorporates four distinct types of generic, long-term knowledge structures:

- ***Concepts*** – Define relational categories, attributes, and *veracity*
- ***Motives*** – Indicate *utility* of relations conditioned on situation
- ***Skills*** – Specify *control values* based on match to *target concepts*
- ***Processes*** – Predict *changes* in attributes given current values

The architecture uses these elements for conceptual inference, goal reasoning, teleoreactive control, and plan generation.

PUG's Layered Processes

Like other cognitive architectures, PUG operates in *cycles* that use knowledge to produce new short-term structures.

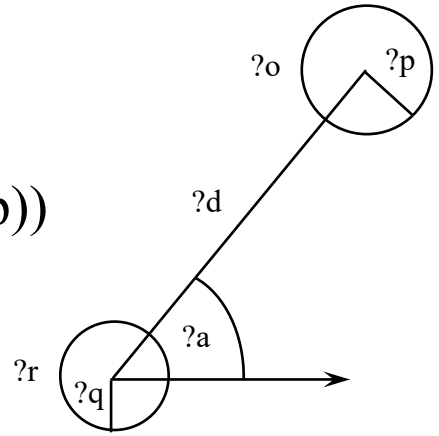


These levels are organized in a ***cascaded*** manner, with each one using results produced by those below it.

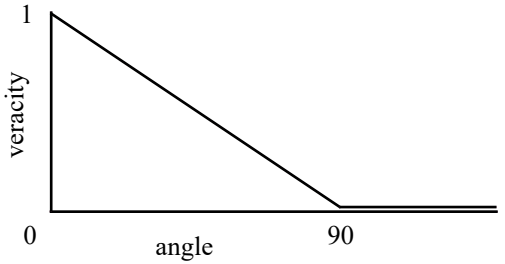
PUG Concepts

PUG encodes knowledge about entities and relations as *concepts*.

```
((robot-at ^id (?r ?o) ^distance ?d)
   :elements   ((robot ^id ?r ^radius ?q)
                (object ^id ?o ^distance ?d ^radius ?p))
   :binds      (?e (– ?d (+ ?q ?p)))
   :veracity   (cond ((> ?e 10.0) 0.0)
                     (t (– 1.0 (/ ?e 10.0)))) )
```




```
((robot-facing ^id (?r ?o) ^angle ?a)
  :elements   ((robot ^id ?r)
               (object ^id ?o ^angle ?a))
  :veracity   (cond ((< ?a 0.0) (– 1.0 (/ ?a –90)))
                    ((> ?a 0.0) (– 1.0 (/ ?a 90)))
                    ((= ?a 0.0)  1.0)) )
```



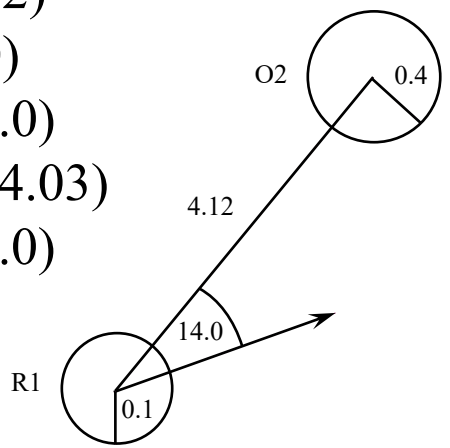
PUG Beliefs and Veracities

Beliefs are ground instances of concepts that relate specific entities.

Perceptions:	**Veracity**
(robot ^id R1 ^radius 0.15 ^move-rate 0.0 ^turn-rate 0.0)	1.00
(object ^id O1 ^distance 0.56 ^angle 0.0 ^radius 0.4)	1.00
(object ^id O2 ^distance 4.12 ^angle 14.03 ^radius 0.4)	1.00
(object ^id O3 ^distance 6.0 ^angle 0.0 ^radius 0.4)	1.00

Inferred Beliefs:

(robot-at ^id (R1 O1) ^distance 0.56)	0.99
(robot-at ^id (R1 O2) ^distance 4.12)	0.64
(robot-at ^id (R1 O3) ^distance 6.0)	0.46
(robot-facing ^id (R1 O1) ^angle 0.0)	1.00
(robot-facing ^id (R1 O2) ^angle 14.03)	0.84
(robot-facing ^id (R1 O3) ^angle 0.0)	1.00
(close-to-colliding ^id (R1 O1))	0.99

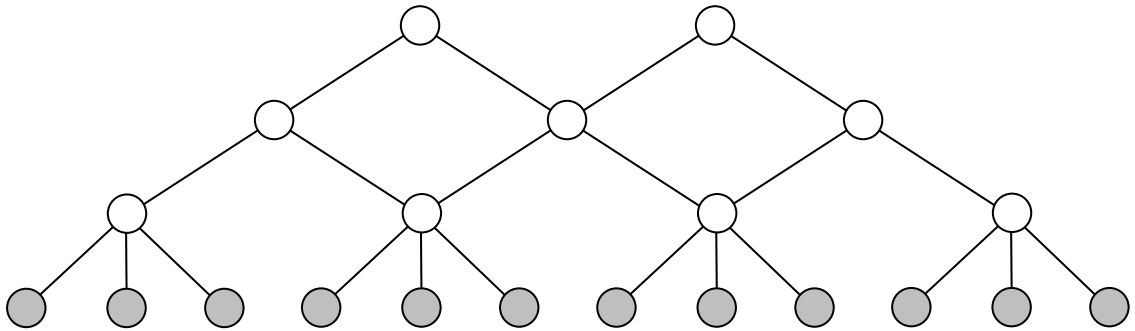


Conceptual Inference in PUG

At the lowest level, *conceptual inference* derives beliefs that are consistent with perceptions / predictions:

- Matches conceptual rules to infer beliefs like (*robot-at R1 O1*)
  - Computes ***values*** of numeric attributes associated with beliefs
  - Calculates ***veracity*** (degree of match) for each inferred belief
- Applies this recursively to generate the full deductive closure

Together, the resulting beliefs describe the current ***state*** as a point in N-dimensional space.



PUG Motives

The framework encodes knowledge about agent goals as *motives*.

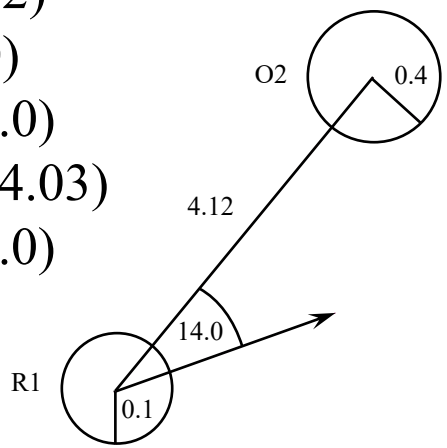
```
((robot-at ^id (?r ?o))
  :conditions  ((robot ^id ?r ^radius ?rr)
                (object ^id ?o ^type target ^distance ?d ^radius ?or))
  :utility     (cond ((< ?d (+ ?rr ?or 0.25))  10.0) (t  0.0))
  :type        achievement )

((close-to-colliding ^id (?r ?o))
  :conditions  ((robot ^id ?r ^radius ?rr)
                (object ^id ?o ^type obstacle ^distance ?d ^radius ?or))
  :utility     (cond ((< ?d (+ ?rr ?or 0.20))  −20.0) (t  0.0))
  :type        maintenance )
```

An *achievement* motive assigns utility to a belief only on its first match, where a *maintenance* motive does so repeatedly.

PUG Beliefs and Utilities

Beliefs also include *utilities* computed by any applicable motives.

Perceptions:	**Veracity**	**Utility**
(robot ^id R1 ^radius 0.15 ^move-rate 0.0 ^turn-rate 0.0)	1.00	0.0
(object ^id O1 ^distance 0.56 ^angle 0.0 ^radius 0.4)	1.00	0.0
(object ^id O2 ^distance 4.12 ^angle 14.03 ^radius 0.4)	1.00	0.0
(object ^id O3 ^distance 6.0 ^angle 0.0 ^radius 0.4)	1.00	0.0
Inferred Beliefs:		
(robot-at ^id (R1 O1) ^distance 0.56)	0.99	0.0
(robot-at ^id (R1 O2) ^distance 4.12)	0.64	0.0
(robot-at ^id (R1 O3) ^distance 6.0)	0.46	0.0
(robot-facing ^id (R1 O1) ^angle 0.0)	1.00	0.0
(robot-facing ^id (R1 O2) ^angle 14.03)	0.84	0.0
(robot-facing ^id (R1 O3) ^angle 0.0)	1.00	0.0
(close-to-colliding ^id (R1 O1))	0.99	−20.0



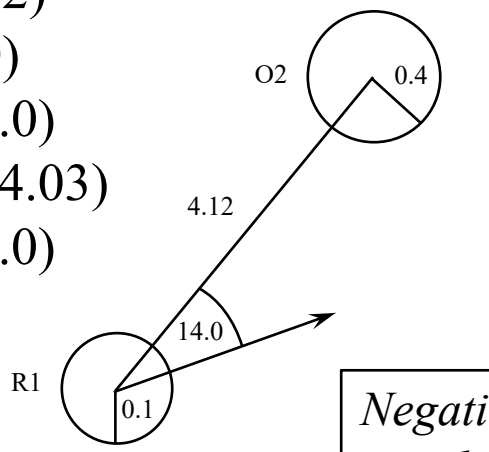
PUG Beliefs and Utilities

Beliefs also include *utilities* computed by any applicable motives.

Perceptions:	**Veracity**	**Utility**
(robot ^id R1 ^radius 0.15 ^move-rate 0.0 ^turn-rate 0.0)	1.00	0.0
(object ^id O1 ^distance 0.56 ^angle 0.0 ^radius 0.4)	1.00	0.0
(object ^id O2 ^distance 4.12 ^angle 14.03 ^radius 0.4)	1.00	0.0
(object ^id O3 ^distance 6.0 ^angle 0.0 ^radius 0.4)	1.00	0.0

Zero because robot not yet at O3

Inferred Beliefs:		
(robot-at ^id (R1 O1) ^distance 0.56)	0.99	0.0
(robot-at ^id (R1 O2) ^distance 4.12)	0.64	0.0
(robot-at ^id (R1 O3) ^distance 6.0)	0.46	0.0
(robot-facing ^id (R1 O1) ^angle 0.0)	1.00	0.0
(robot-facing ^id (R1 O2) ^angle 14.03)	0.84	0.0
(robot-facing ^id (R1 O3) ^angle 0.0)	1.00	0.0
(close-to-colliding ^id (R1 O1))	0.99	−20.0

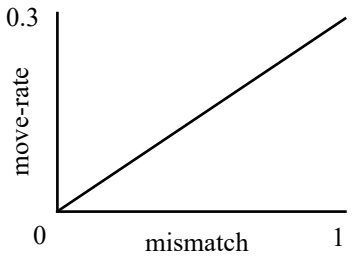


Negative because robot too close to object O1

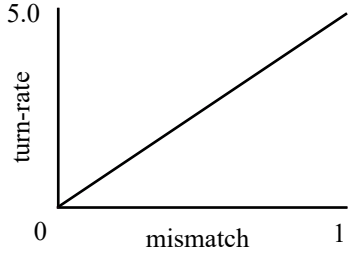
PUG Skills

Skills encode knowledge for how to achieve agent objectives.

```
((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))) )
```



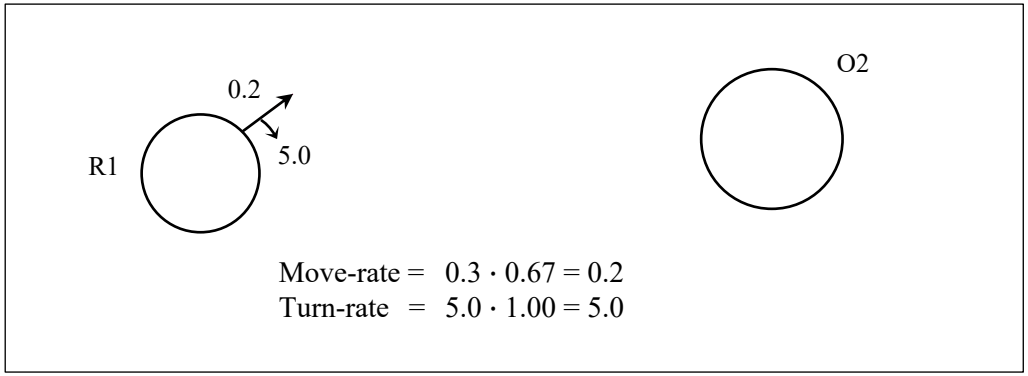
Here the symbol *$mismatch* stands for one minus the *veracity* of the matched target concept.

State Processing / Decision Making

When the architecture carries an out an intention associated with skill S, it:

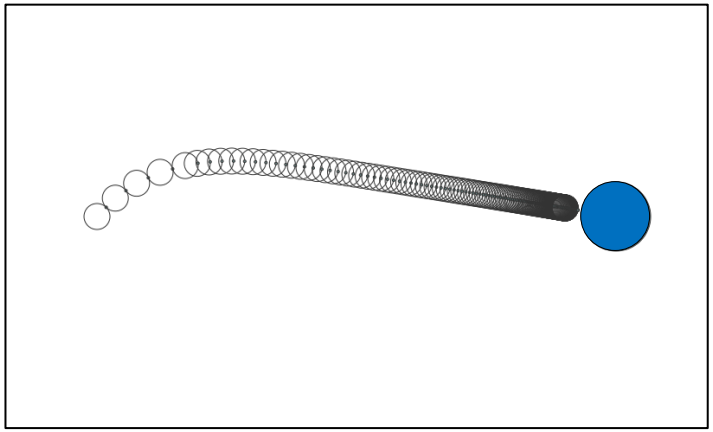
- Checks that S's conditions match the current beliefs
- Finds degree of *mismatch* M to S's target belief
- Ensures the mismatch does not fall below threshold
- Else inserts M into S's equations to find control values

When *multiple* intentions apply, then PUG takes the *sum* of their control values (as with potential fields).

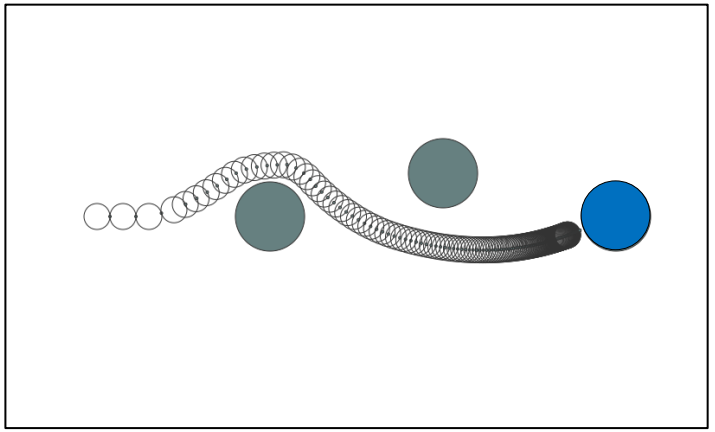


Teleoreactive Execution

PUG applies skills on each cycle of state processing to generate agent behavior in the environment.



(move-to R1 O1)
(turn-to R1 O1)



(move-to R1 O3)
(turn-to R1 O3)
(avoid-on-left R1 O1)
(avoid-on-right R1 O2)

Concurrent Intentions

A particular trajectory follows deterministically from a set of agent *intentions* (i.e., skill instances).

PUG Processes

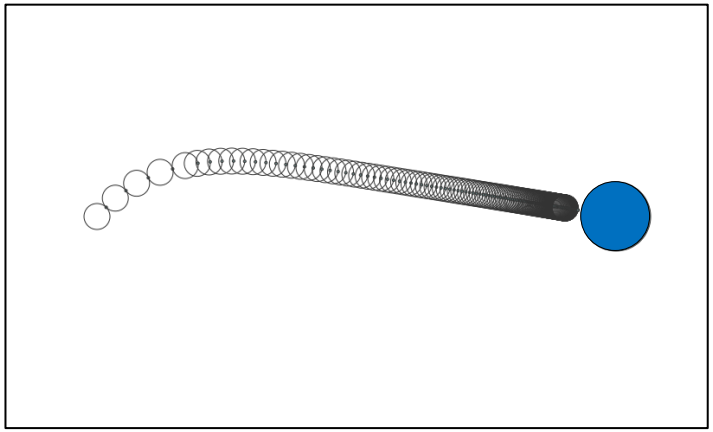
Processes let the agent generate predictions about future states.

```
((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 (- ?t))) )
```

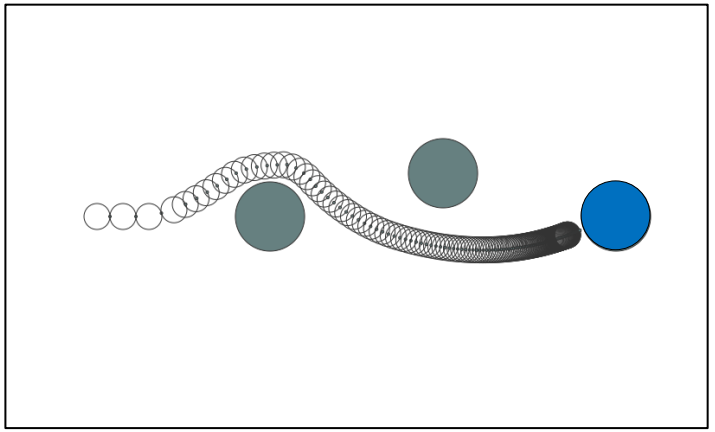
Here the functions **dd* and **da* calculate change in the robot's *distance* and *angle* relative to an object as it moves forward.

Mental Simulation

PUG can combine its skills and processes to *simulate* motion trajectories in the agent's mind.



(move-to R1 O1)
(turn-to R1 O1)

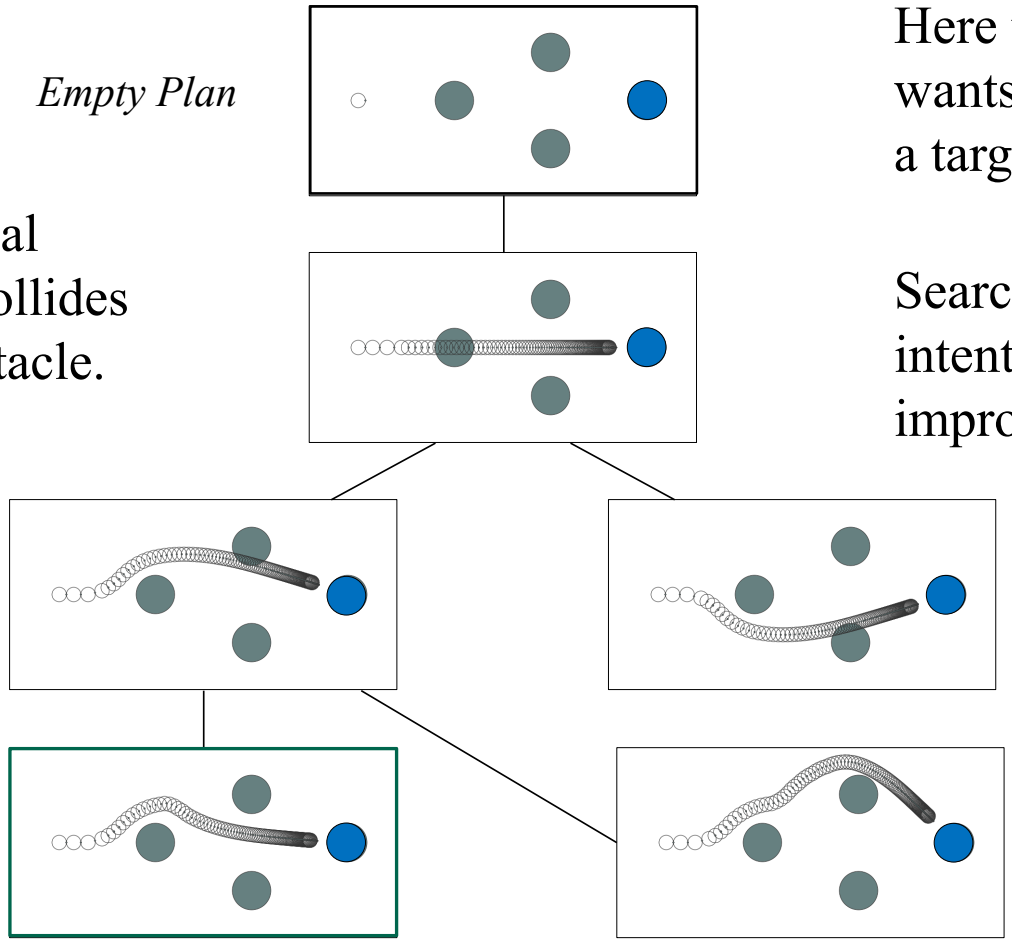


(move-to R1 O3)
(turn-to R1 O3)
(avoid-on-left R1 O1)
(avoid-on-right R1 O2)

Concurrent Intentions

One can view each such simulated trajectory as a ***motion plan***.

Heuristic Search for Motion Plans



PUG embeds the search for motion plans in a higher-level search for task plans driven by agent objectives.

Related Research

Our framework incorporates ideas from earlier work on goal-directed agents that operate time:

- *Robotic architectures*, which integrate state inference, reactive control, and planning (Kortenkamp & Simmons, 2008)
- *Cognitive architectures*, which provide high-level formalisms for modular knowledge elements (Langley, Laird, & Rogers,1997)
- *Cognitive robotics*, which offers logic-based formalisms to encode expertise for goal-driven activities (Ferrein & Lakemeyer, 2008)
- Kuipers' (2000) *spatial semantic hierarchy*, which attaches numeric control laws to qualitative regions with target states

PUG also borrows from research on logical inference, continuous control, decision theory, task planning, and heuristic search.

Advantages of the Framework

PUG does not support new abilities or more robust operation, but the framework should:

- Ease the construction of robotic systems
  - *By offering high-level programming constructs*
- Enable explainable agency in such systems
  - *By storing and indexing traces of decisions*
- Provide inductive bias to aid learning
  - *By increasing sample efficiency and interpretability*

Thus, we predict that users will find it a useful environment for developing knowledge-based embodied agents.

Directions for Future Work

Despite PUG's promise, we should still extend the framework along multiple fronts:

- Complete integration of motion planning with task planning
- Introduce more flexible methods for inference and control
- Incorporate temporal constraints into concepts, skills, and motives
- Elaborate conceptual formalism to encode spatial knowledge
- Embed skills in hierarchical structures for temporal abstraction
- Support stochastic skills and processes to encode uncertainty

We should also improve the system's interface so users can trace and analyze agent behavior more easily.

Summary Remarks

PUG is a cognitive architecture for embodied intelligent agents that incorporates:

- Concepts, motives, skills, and processes that have both ***symbolic*** and ***numeric*** elements
- ***Cascaded processing*** with layers for inference, state processing, mental simulation, motion planning, and task planning
- Skills' target concepts match to different degrees, which serve as error signals that enable continuous control
- Mental simulation and utility calculation support ***heuristic search*** for both motion and task plans

The framework borrows from earlier ones but combines their ideas in innovative ways.