Extending an Embodied Cognitive Architecture with Spatial Representation and Reasoning

Pat Langley

Center for Design Research
Stanford University

Edward P. Katz

Stanford Intelligent Systems Laboratory
Stanford University

Thanks to Mike Barley, Dongkyu Choi, Ben Meadows, Mohan Sridharan, and the Dagstuhl on spatial cognition for useful discussions. This research was supported by Grant No. FA9550-20-1-0130 from AFOSR, which is not responsible for its contents.

What is a Cognitive Architecture?

A *cognitive architecture* (Newell, 1990) is an infrastructure for developing intelligent systems that:

- Specifies facets that remain *constant* across different domains;
- Incorporates core ideas from psychological theories, such as:
  - *Short-term* memories are distinct from *long-term* stores
  - Memories contain *modular* elements cast as *symbol structures*
  - Long-term structures are accessed through *pattern matching*
  - Processing relies on *retrieval / selection / action cycles*
  - Cognition involves *dynamic composition* of mental structures

Well-known architectures include ACT-R (Anderson, 1993), Soar (Laird, 2012), and ICARUS (Langley et al., 2009).

The PUG Architecture

The recent PUG architecture (Langley et al., 2016) supports embodied agents with four core ideas:

- *Symbolic relations grounded in quantitative descriptions*
- *Relations have associated utilities that reflect tradeoffs*
- *Discrete skills have associated control equations*
- *Mental simulation creates trajectories to guide planning*

A recent extension – PUG/C – unifies symbolic and numeric processing more deeply (Langley & Katz, 2022).

PUG's Knowledge Structures

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

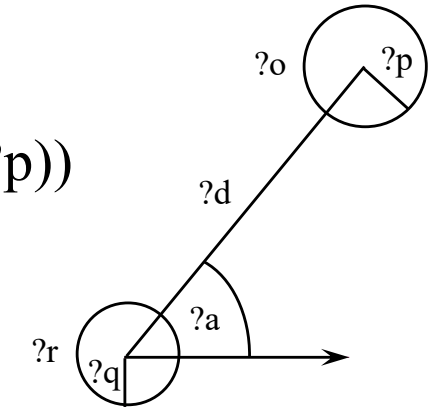
- ***Concepts*** – Define relational categories, attributes, and *veracity*
- ***Motives*** – Indicate *utility* of relations conditioned on situation
- ***Skills*** – Specify *control values* based on match to *target concepts*
- ***Processes*** – Predict *changes* in attributes given current values

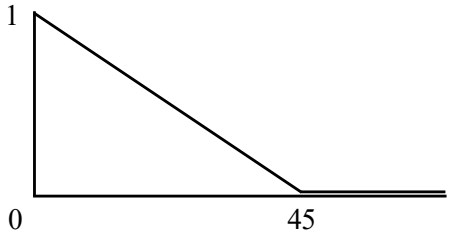
PUG uses these elements for conceptual inference, reactive control, heuristic evaluation, and plan generation.

Examples of PUG Conceptual Rules

```
((robot-at ^id (?r ?o) ^distance ?d)
   :elements    ((robot ^id ?r ^radius ?q)
                 (object ^id ?o ^distance ?d ^radius ?p))
   :veracity    ((linear ?d (+ ?p ?q) 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)) )
```



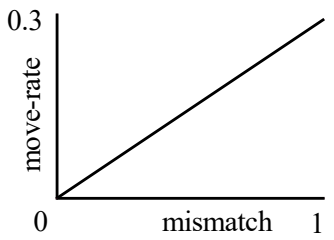


Here the function *(linear obs max min)* returns *1* if the observed value *obs* = *max*, *0* if $|obs| \geq min$, and $|obs/(max - min)|$ within that range.

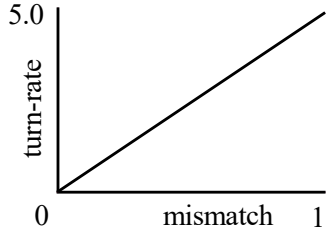
Beliefs are ground instances of concepts that relate specific entities.

Examples of PUG Skills

```
((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.

Intentions are ground instances of skills that involve specific entities.

PUG's Layered Processes

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

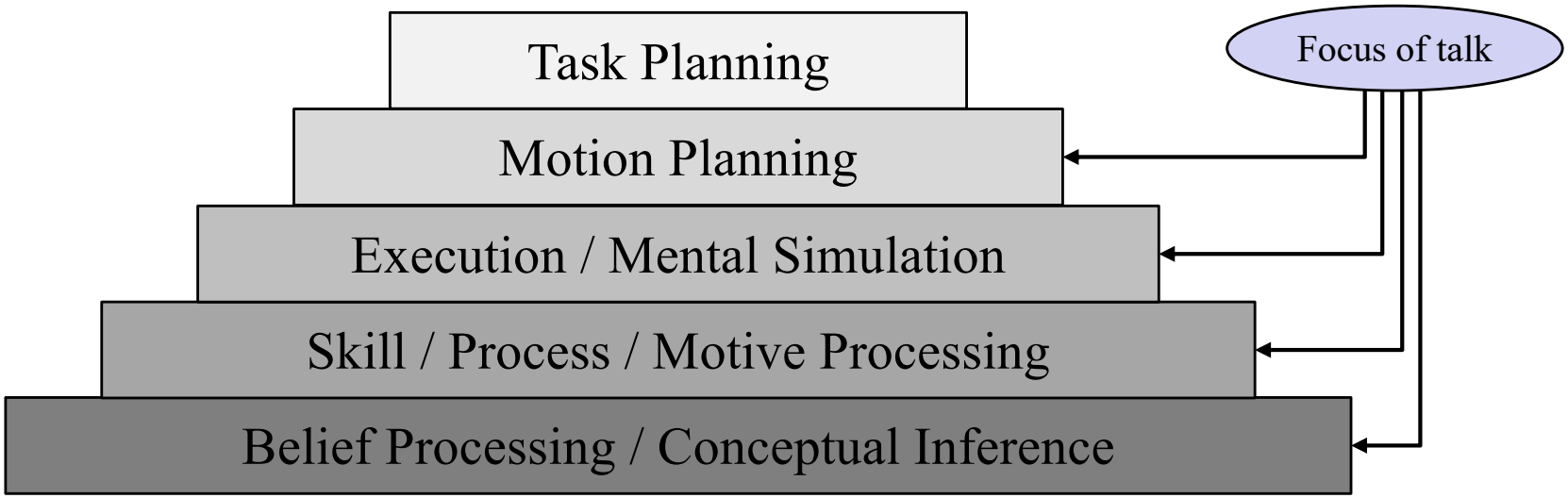
The framework differs in that it relies on five processing levels:

- ***Belief processing*** – Inference from perceptions / predictions
- ***State processing*** – Applies skills, processes, motives
- ***Execution / Mental simulation*** – Generates trajectories
- ***Motion planning*** – Heuristic search for an intention set
- ***Task planning*** – Search for a sequence of motion plans

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

PUG's Layered Processes

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



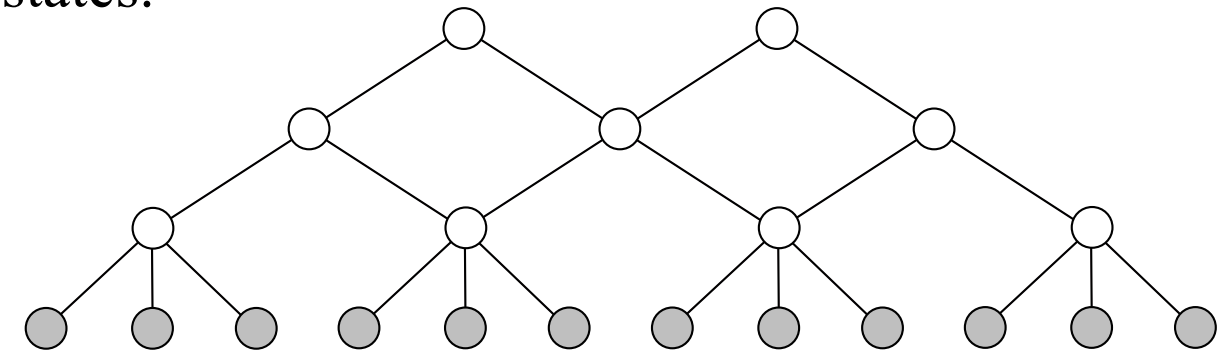
These levels are organized in a ***cascaded*** manner, with each one using results produced by those below it.

Conceptual Inference in PUG

At lowest level, *conceptual inference* derives beliefs 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

The next level – *state processing* – uses motives to assign ***utilities*** to beliefs and states.

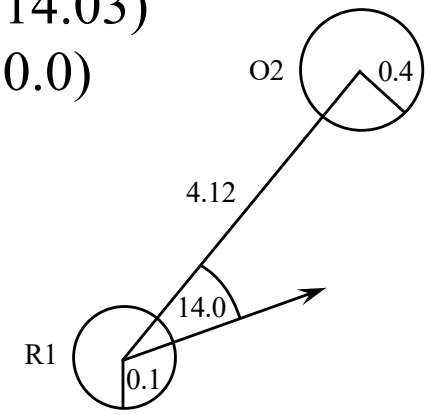


Examples of PUG Beliefs

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 2.0 ^angle 0.0 ^radius 0.4)	1.00	0.0
(object ^id O2 ^distance 4.123 ^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 2.0)	0.85	0.0
(robot-at ^id (R1 O2) ^distance 4.12)	0.62	0.0
(robot-facing ^id (R1 O1) ^angle 0.0)	1.00	0.0
(robot-facing ^id (R1 O2) ^angle 14.03)	0.69	0.0
(robot-facing ^id (R1 O3) ^angle 0.0)	1.00	0.0
(approaching ^id (R1 O1))	0.91	−20.0

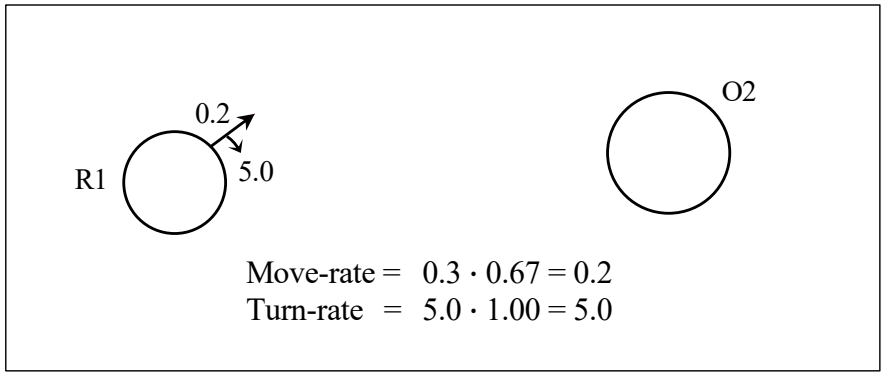


Continuous Control in PUG

When PUG carries an out an active intention associated with skill S, whether mentally or externally, it:

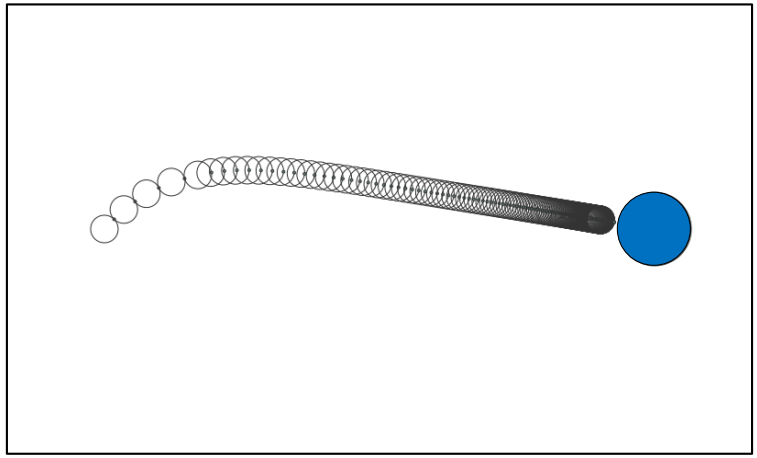
- Checks that S's conditions match the current beliefs
- Finds degree of mismatch M to S's target belief
- Ensures the mismatch does not fall below threshold
- Else inserts M into S's equations to find control values

If multiple intentions apply, then PUG takes the vector sum of control values (as with potential fields).

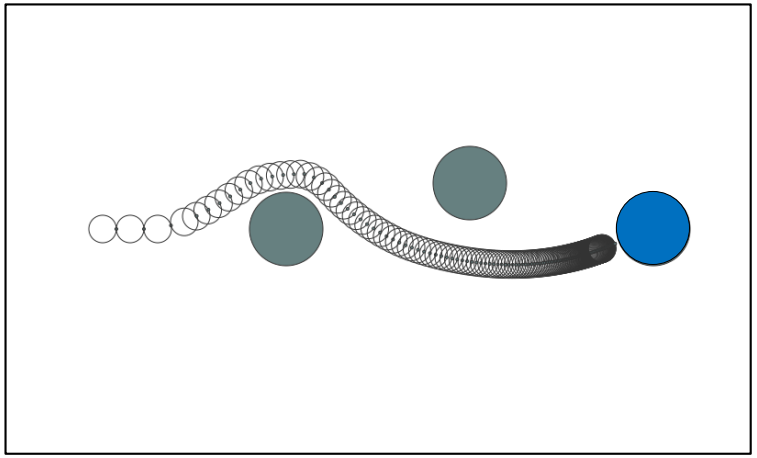


Example of PUG/C Motion Plans

PUG/C invokes skills, processes, and motives repeatedly, in the world or in ***mental simulation***, to generate ***motion trajectories***.



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



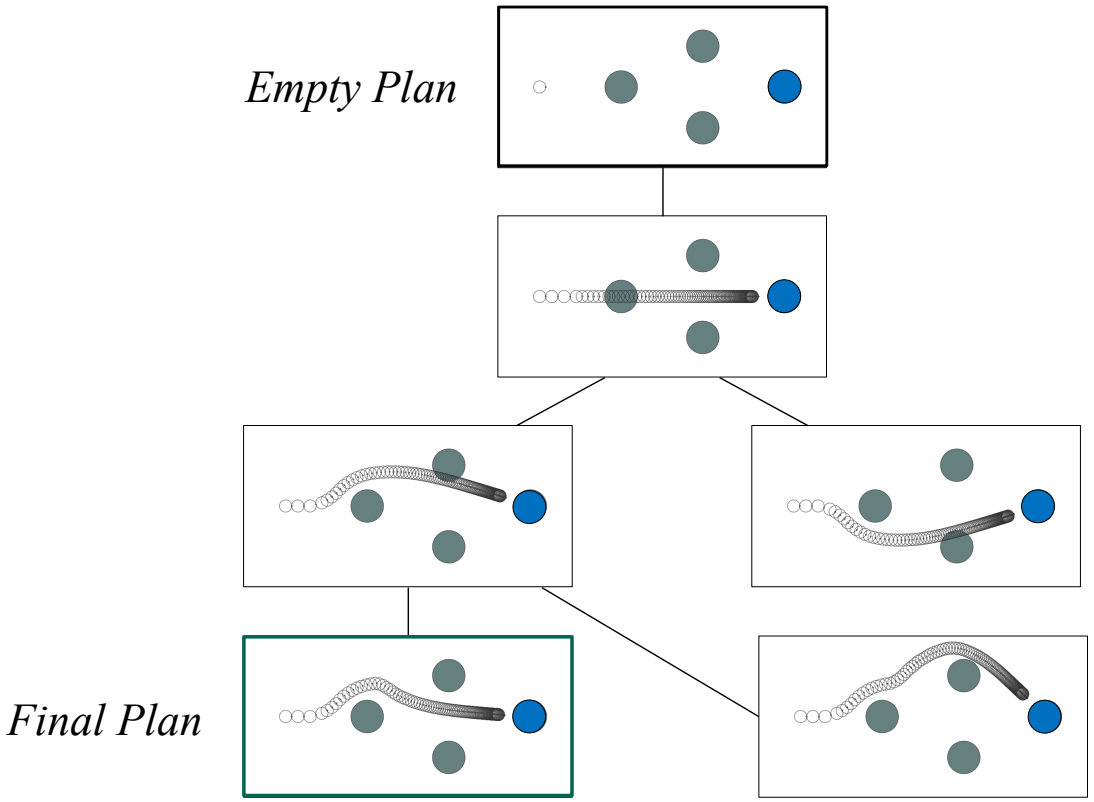
(move-to R1 O3)
(turn-to R1 O3)
(avoid-on-left R1 O1)
(avoid-on-right R1 O2)

Concurrent Intentions

Each trajectory follows deterministically from an intention set (i.e., skill instances) that constitute a ***motion plan***.

Heuristic Search for Motion Plans

PUG/C pursues greedy search through a space of motion plans.

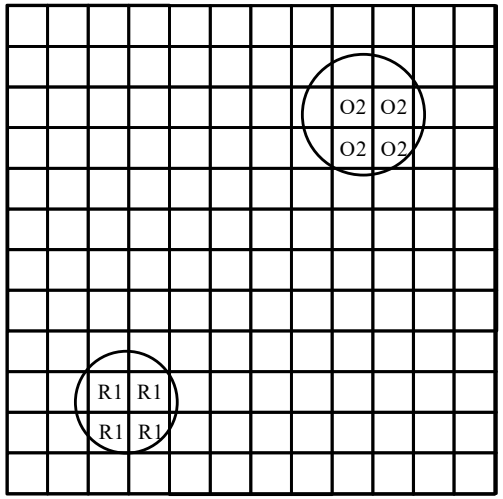


The architecture uses mental simulation to produce trajectories and motive-generated utilities to evaluate them.

Adding Place Knowledge to PUG

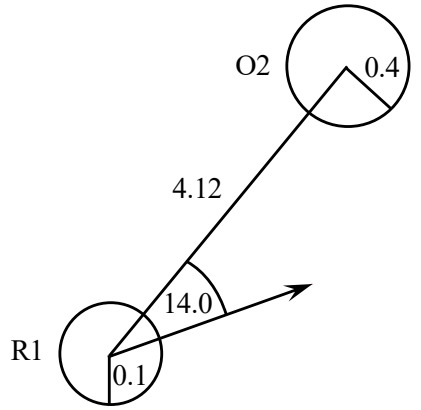
PUG supports basic spatial abilities, such as avoiding obstacles, but it cannot:

- Reason about knowledge of ***places***
- Use ***topological networks*** of places



Robotics often uses discretized, world grids, but we maintain that humans:

- Describe situations in terms of ***nearby objects***
- Specify these objects with ***continuous attributes***
- Rely on ***egocentric***, ***object-centered*** coordinates

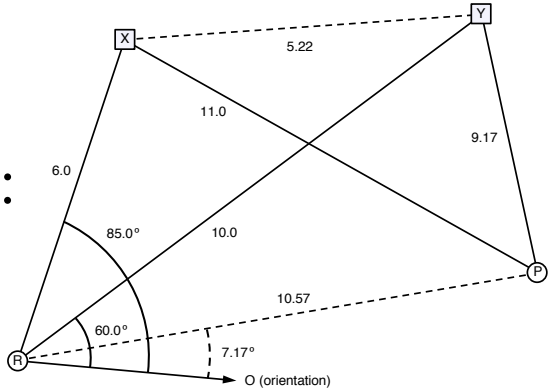


We can use the existing architecture to encode and use such spatial content.

Place Knowledge in PUG

We can represent spatial content as existing PUG structures:

- A place is a ***virtual object*** defined by:
  - Agent ***distances*** to *two* reference objects
- An instance of place P is a belief that includes:
  - The agent's derived ***distance*** and ***angle*** to P



To generate an instance of place P on a time step, PUG/C:

- ***Calculates*** distance / angle to P using reference objects
- ***Infers*** degree of match for beliefs like *(robot-at R P)*

PUG uses these belief structures for higher-level processing.

A Sample Place Definition

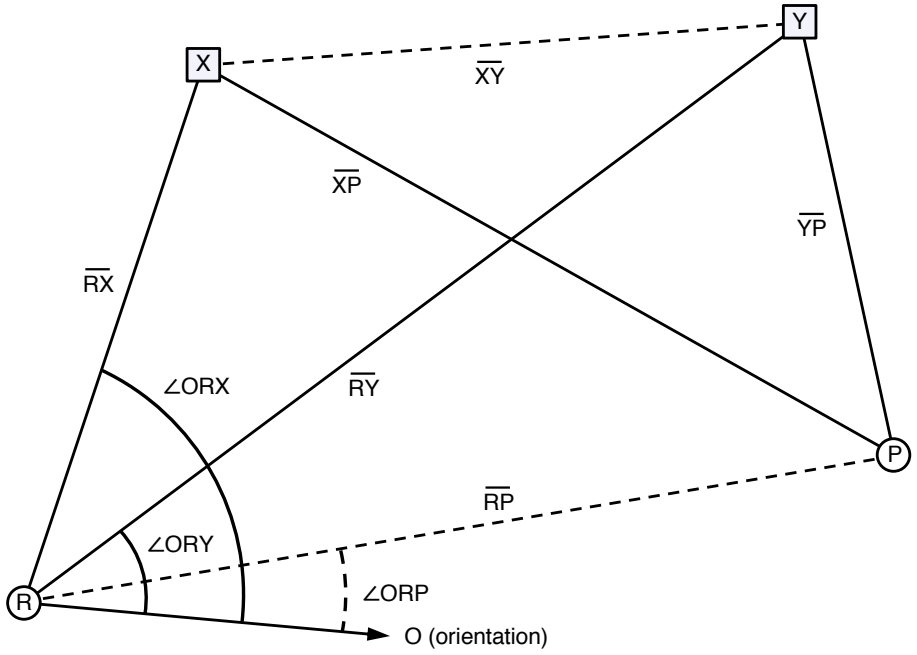
Here is a conceptual rule that defines a place V1 in terms of reference objects O1 and O2:

```
((object ^id V1 ^distance ?d3 ^angle ?a3 ^type virtual)
  :elements    ((object ^id O1 ^distance ?d1 ^angle ?a1)
                (object ^id O2 ^distance ?d2 ^angle ?a2))
  :binds       (?d3 (*distance-to-virtual ?d1 ?d2 ?a1 ?a2 11.0 9.17)
                ?a3 (*angle-to-virtual ?d1 ?d2 ?a1 ?a2 11.0 9.17)))
```

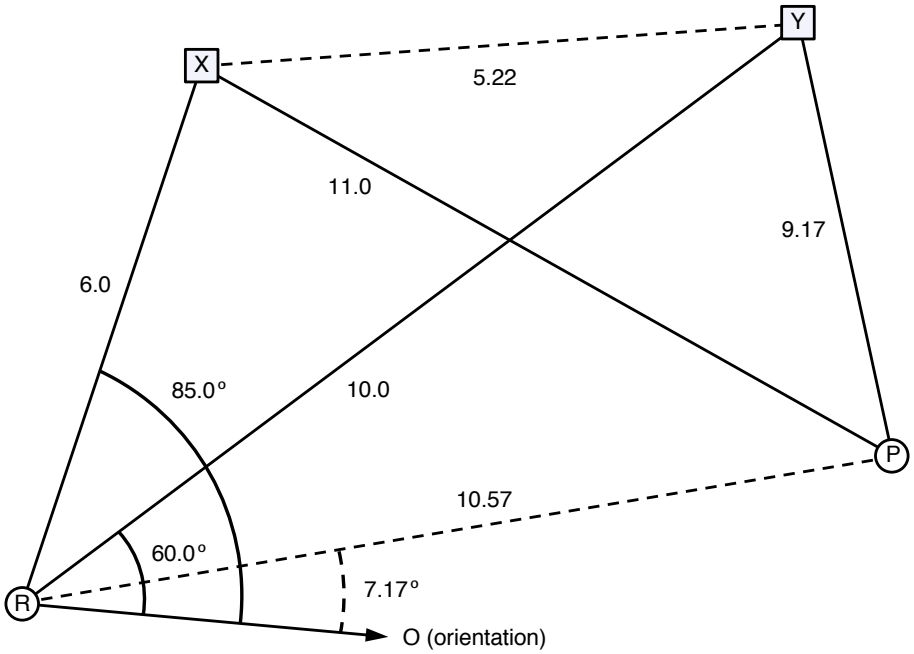
This states that V1's distance to object O1 is 11.0, while its distance to object O2 is 9.17.

The two functions **distance-to-virtual* and **angle-to-virtual* compute V1's distance and angle to the agent.

Distance and Angle to a Virtual Object



$\overline{RP} = SAS(\overline{RY}, |SSS(\overline{XY}, \overline{XP}, \overline{YP}) - SSS(\overline{XY}, \overline{RX}, \overline{RY})|, \overline{YP})$
where $\overline{XY} = SAS(\overline{RX}, |\angle ORX - \angle ORY|, \overline{RY})$
and where $\angle ORP = \angle ORY - SSS(\overline{RY}, \overline{YP}, \overline{RP})$
and where $SSS(a, b, c) = \text{acos}((a^2 + c^2 - b^2) / (2 \cdot a \cdot c))$
and where $SAS(b, A, c) = \sqrt{b^2 + c^2 - 2 \cdot b \cdot c \cdot \cos(A)}$

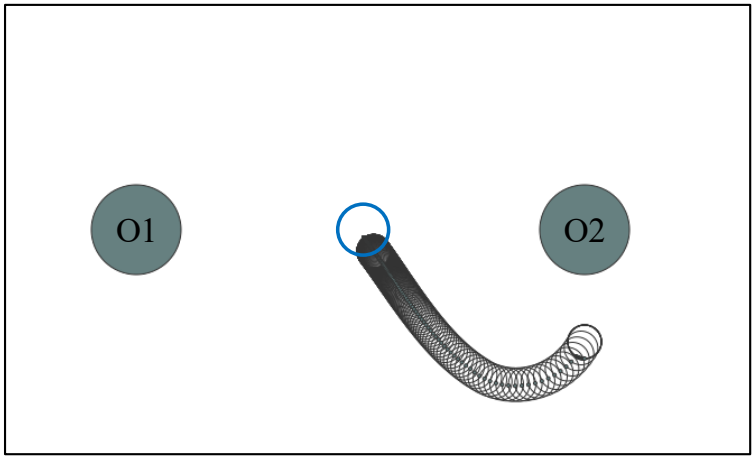


Here R's *distance* to virtual object P is $\overline{RP} = 10.57$
and R's *angle* to the same object P is ∠ORP = 7.17°

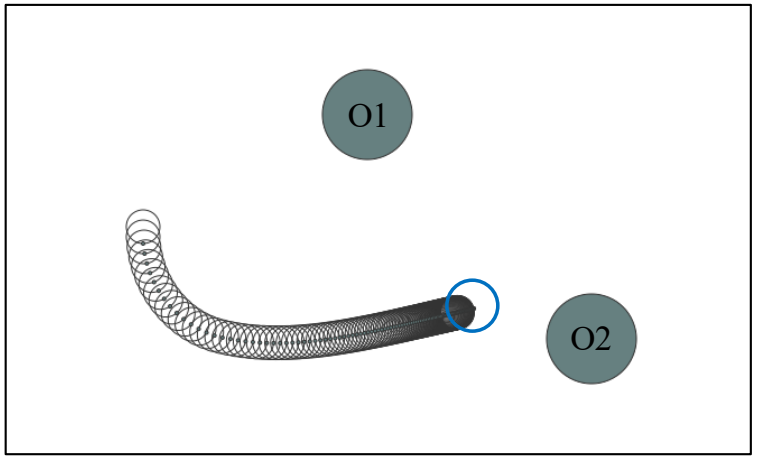
Only two reference objects are necessary because R itself serves as a third point.

Trajectories to Virtual Objects

PUG/C can use the definition of a virtual object to guide its skill applications and to simulate a trajectory there.



(move-to R1 V1)



(move-to R1 V2)

On the left, robot R1 moves to V1, midway between O1 and O2.

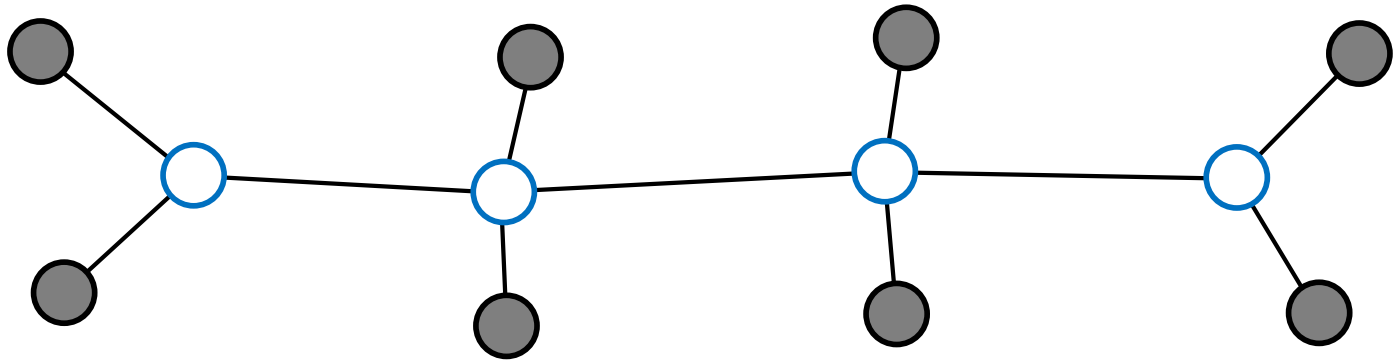
On the right, R1 moves to V2, which is closer to O2 than to O1.

Maps as Collections of Places

A *map* is a collection of place definitions that include distances to *other places*.

- E.g., P's definition might include distance to another virtual object, *Q*, and vice versa.
- Such a set of connections specify a *topological network* of distinct places.

PUG/C can use this knowledge to create motion plans between places, as well as larger-scale *task* plans.



Related and Future Research

Our research incorporates ideas from multiple prior efforts:

- Cognitive architectures (Soar, ICARUS, teleoreactive systems)
- Error-driven feedback control and potential fields
- Egocentric encodings and topological networks

However, PUG/C embeds them in a unified architecture with a high-level programming language.

Future work will test PUG in dynamic settings (e.g., CARLA) and support complex shapes (e.g., using RCC8).

Summary Remarks

PUG/C is a cognitive architecture for embodied, human-like agents that incorporates:

- Concepts, motives, skills, and processes that have *symbolic* and *numeric* elements
- *Cascaded processing* with layers for beliefs, states, mental simulations, motion plans, and task plans
- Reasoning about virtual objects – *places* – in terms of their distances to other objects
- Topological *maps* encoded as places that refer to each other and allow long-distance navigation

Future work will extend PUG to dynamic environments and richer spatial representations.

References

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

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

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.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In I. J. Cox & G. T. Wilfong (Eds.), *Autonomous robot vehicles*. New York: Springer.

Kuipers, B. J., & Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, *8*, 47–63.

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

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

Yeap, W. K. (2011). How Albot0 finds its way home: A novel approach to cognitive mapping using robots. *Topics in Cognitive Science*, *3*, 707–721.