Goals, Utilities, and Mental Simulation in Continuous Planning

Pat Langley, Mike Barley, Ben Meadows
Department of Computer Science
University of Auckland, Auckland, NZ

Dongkyu Choi
Department of Aerospace Engineering
University of Kansas, Lawrence, KS

Edward P. Katz
Carnegie Mellon Silicon Valley
Moffett Field, CA

This research was supported by Grant N00014-15-1-2517. We thank J. Benton, Chris Pearce, Stephanie Sage, Mike Stilman, Son To, and Charlotte Worsfold for discussions.

A Motivating Example

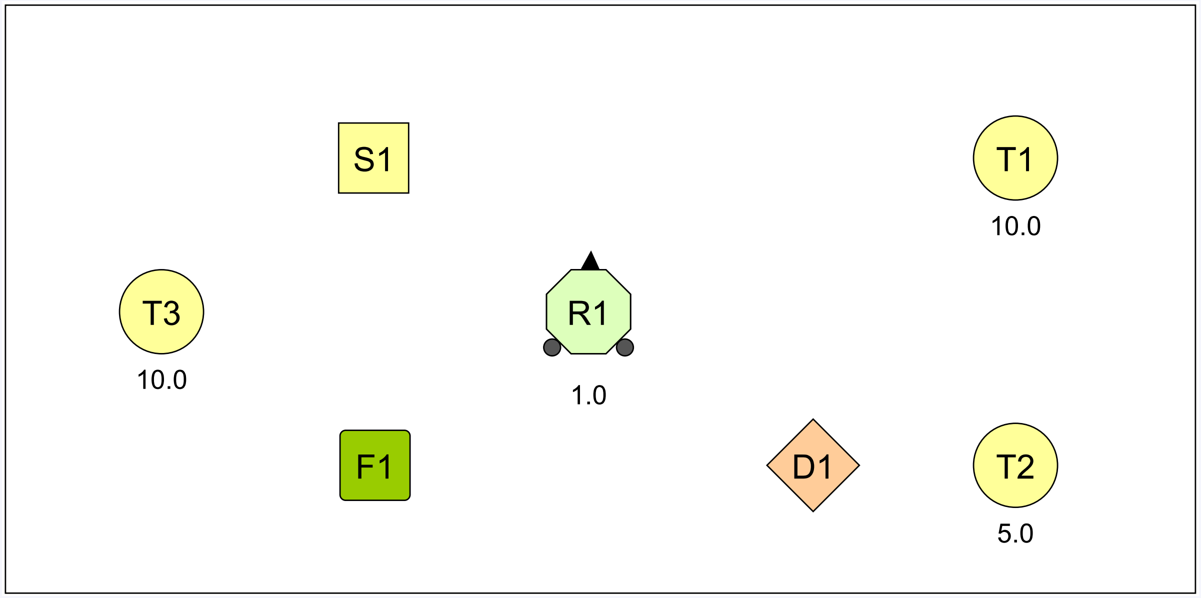
Consider a planetary mission in which an autonomous robotic agent must:

- Deposit sensors at prioritized target sites;
- Collect interesting samples that it encounters;
- Avoid the proximity of known danger areas; and
- Retain enough onboard fuel to carry out these tasks.

A mission will involve many competing goals, some mutually exclusive, that may have different value at different times.

A Motivating Example

Consider a scenario that includes two sensors, three target sites, one fuel depot, one sample, and one danger area.



Here an effective plan might collect the sample, take a detour to refuel, deliver a sensor to T3, refuel again, and deliver a sensor to T1 rather than T2, thus avoiding the danger area D1.

Background / Research Aims

The AI community has taken two main responses to pursuing objectives over time:

- Generating plans that achieve or maintain *explicit goals stated as relational literals*;
- Finding action sequences that produce high values on *numeric evaluation functions encoded as simple features*.

In this talk, we present a theory that unifies these paradigms and apply it to planning in continuous domains.

Our approach combines ideas from cognitive architectures, logical inference, teleoreactive control, and goal reasoning.

Target Abilities

We desire intelligent agents that reason about and plan their activities in continuous domains by:

- Operating over space and time in contexts with *competing* and even *inconsistent* objectives;
- Assigning different *importance* or *value* to objectives based on the agent's *situation*;
- Reasoning about activities that involve change in *qualitative* structure and *quantitative* attributes; and
- Producing reasonable behavior that *balances tradeoffs* among different objectives in a situation-aware manner.

These abilities are crucial in domains that require autonomy, from planetary exploration to disaster relief.

Theoretical Assumptions

Our account of these abilities relies on five central theoretical assumptions:

- *Goals are the locus of utility*. Each goal has a score that denotes desirability, with state utility distributed among them.
- *Both goals and utilities are conditional*. Goals activate when conditions are met and utilities are functions of the situation.
- *Mental structures include symbolic and numeric content*. They specify both relations and numeric attributes that affect utilities.
- *Goal-oriented utilities play a key role in planning*. Utilities that accrue over a trajectory guide choices during heuristic search.
- *Operator utility is estimated by mental simulation*. This applies operators until termination, matching goals to compute scores.

We have implemented these tenets in PUG, a new architecture for planning in continuous domains.

Knowledge in the PUG Architecture

PUG incorporates four distinct types of generic knowledge:

- *Compositional rules* that define relational concepts and their associated numeric attributes;
- *Specialization rules* that discriminate among subclasses of more general concepts;
- *Goal-generating rules* that specify when goal instances should be active and what utility to assign them; and
- *Operators* that encode models of actions' immediate effects and final results under given conditions.

Together, these provide the content PUG uses for conceptual inference, goal creation, and plan generation.

Examples of PUG Conceptual Rules

Compositional rule:

```
((vector ^id (?r ?o) ^from ?r ^to ?o ^distance ?d ^angle ?a)
   :elements     ((robot ^id ?r ^xloc ?x1 ^yloc ?y1 ^orient ?f)
                  (object ^id ?o ^xloc ?x2 ^yloc ?y2))
   :binds        (?d  (*distance ?x1 ?y1 ?x2 ?y2)
                  ?a  (*angle ?x1 ?y1 ?x2 ?y2 ?f) )
   :tests        ((< ?d 100)))
```

Specialization rules:

```
((at ^id (?r ?o))
   :specializes  (vector ^id (?r ?o) ^distance ?d)
   :tests        ((< ?d 0.2)))

((at-ahead ^id (?r ?o))
   :specializes  (at ^id (?r ?o) ^angle ?a)
   :test         ((> ?a −0.01) (< ?a 0.01)))
```

Examples of PUG Goal-Utility Rules

Achievement rule:

```
((sensor-at ^id (?sensor ?target))
   :type          achievement
   :conditions    ((object ^id ?target ^type target ^priority ?p))
   :function      (* 100.0 ?p))
```

Maintenance rule:

```
((not (vector ^id (?r ?o) ^from ?r ^to ?o))
   :type          maintenance
   :conditions    ((object ^id ?o ^type danger) (robot ^id ?r)
                   (vector ^id (?r ?o) ^from ?r ^to ?o ^distance ?d))
   :tests         ((< ?d 20))
   :function      (/ 0.1 (+ (sqrt ?d) 0.01)))
```

Examples of PUG Operators

```
((move-to ?r ?o)
  :elements     ((robot ^id ?r ^xloc ?x1 ^yloc ?y1 ^orient ?a ^fuel ?f)
                  (object ^id ?o ^type ?t ^xloc ?x2 ^yloc ?y2))
  :conditions   ((facing ^id (?r ?o) ^distance ?d) (not (at ^id (?r ?o))))
  :tests        ((> ?d 0.2))
  :changes      ((robot ^id ?r ^fuel (− ?f 0.01) ^xloc (+ ?x1 (dx ?a))
                             ^yloc (+ ?y1 (dy ?a))))
  :results      ((at ^id (?r ?o))))

((turn-left-to ?r ?o)
  :elements     ((robot ^id ?r ^orient ?f) (object ^id ?o ^type ?t))
  :conditions   ((to-left ^id (?r ?o) ^angle ?a) (not (at ^id (?r ?o)))
                  (not (at-with-no-sensor ^id (?r ?other))))
  :changes      ((robot ^id ?r ^orient (+ ?f 1)))
  :results      ((ahead ^id (?r ?o)) (not (to-left ^id (?r ?o)))))
```

Inferring Beliefs, Goals, and Utilities

When PUG encounters a state during the planning process, it:

- Matches composition rules against primitive objects and generates composite objects (e.g., *vector*) with derived numeric attributes;
- Matches specialization rules against primitive and composite objects that denote specialized relations (e.g., *at*, *facing*);
- Matches goal rules against these derived beliefs to generate goal instances that are active in the state; and
- Calculates the utilities of all satisfied active goals and combines them to compute the state's utility.

Maintenance goals contribute on each cycle they are satisfied; *achievement* goals only contribute when first satisfied.

Examples of PUG Beliefs

Primitive objects:

```
(robot ^id r1 ^xloc 0.0 ^yloc 0.0 ^orient 180.0 ^fuel 1.0)
(object ^id t1 ^type target ^priority 1.0 ^xloc 2.0 ^yloc 0.0)
(object ^id t2 ^type target ^priority 2.0  ^xloc −2.0 ^yloc 0.0))
(object ^id d1 ^type danger ^xloc −1.0 ^yloc 1.0)
```

Composite relations:

```
(vector ^id (r1 t1) ^from r1 ^to t1 ^distance 2.0 ^angle −180.0)
(vector ^id (r1 t2) ^from r1 ^to t2 ^distance 2.0 ^angle 0.0)
(vector ^id (r1 d1) ^from r1 ^to d1 ^distance 1.41 ^angle −45.0)
```

Specialized relations:

```
(to-right ^id (r1 t1) ^from r1 ^to t1 ^distance 2.0 ^angle −180.0)
(ahead ^id (r1 t2) ^from r1 ^to t2 ^distance 2.0 ^angle 0.0)
(to-right ^id (r1 d1) ^from r1 ^to d1 ^distance 1.41 ^angle −45.0)
```

Planning in the PUG Architecture

The PUG planning module operates in discrete cycles that take one of five meta-level actions:

- If there are enough acceptable plans or no further choices, then halt and return the candidates ranked by their utilities.
- If the plan for the current node is acceptable, then store it and select another node in the search tree for elaboration.
- If the current plan is unacceptable (e.g., involves a loop), then mark it as failed and select another node in the search tree.
- If the current plan has no operator instances, then find operators applicable in the current state and generate scores for each one.
- If the current plan has untried operator instances, then select one with the best utility and apply it to generate a new state.

Module parameters control the details of planning, but here we report on heuristic depth-first search.

Selecting Operators During Search

However, some of these parameters relate directly to the theoretical postulates presented earlier:

- When selecting among operator instances, PUG uses utility as determined by goals matched over their simulated applications.
- To this end, it simulates each operator's trajectory, calculates the utilities of matched goals on each state, and averages them.
- If a given simulation halts before producing its expected results, then the operator instance fails.
- To rank alternative plans, PUG sorts acceptable candidates by the average utilities over their entire simulated trajectories.
- For a plan to be acceptable, its trajectory’s average utility must exceed those for ancestors in the search tree.

To clarify these ideas, we must explain how mental simulation aids calculation of expected utilities.

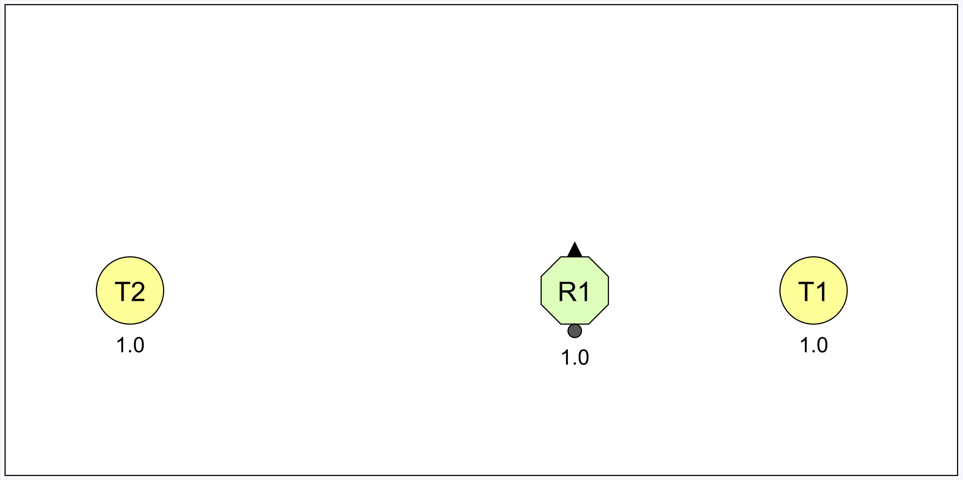
Mental Simulation / Utility Projection

PUG determines the average utility for an operator instance by:

- Applies the operator repeatedly, starting from the current state;
- Calculates the utility of each successive state by:
  - Deriving beliefs that hold in that situation;
  - Generating goal instances with satisfied conditions;
  - Adding the utility of each positive goal that is matched;
  - Subtracting the utility of each negated goal that is matched.
- Terminates on reaching a state that matches the operators' results or that fails to match its conditions.
- Divides the accumulated utility by the number of time steps.

The planner uses the resulting average as its evaluation metric during operator selection.

Demonstration Scenarios 1 and 2

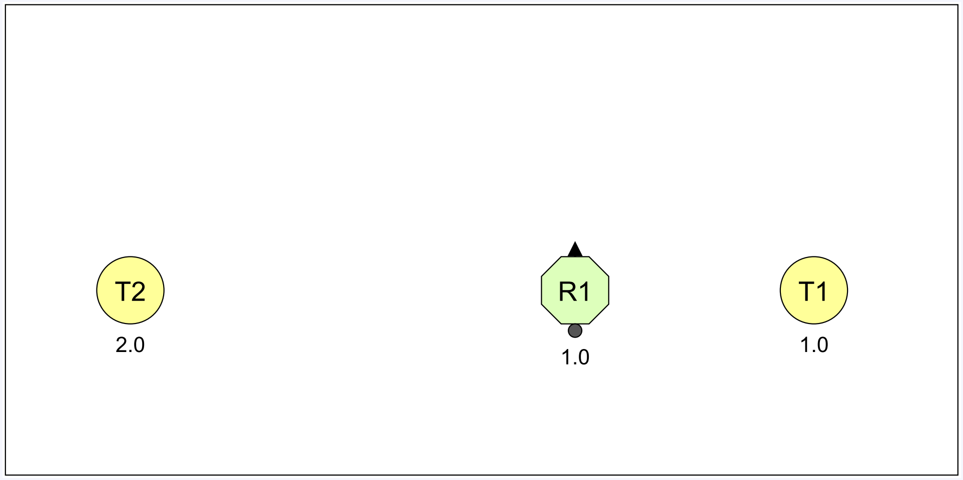


One sensor, two targets, with T2 more distant from R1 than T1.

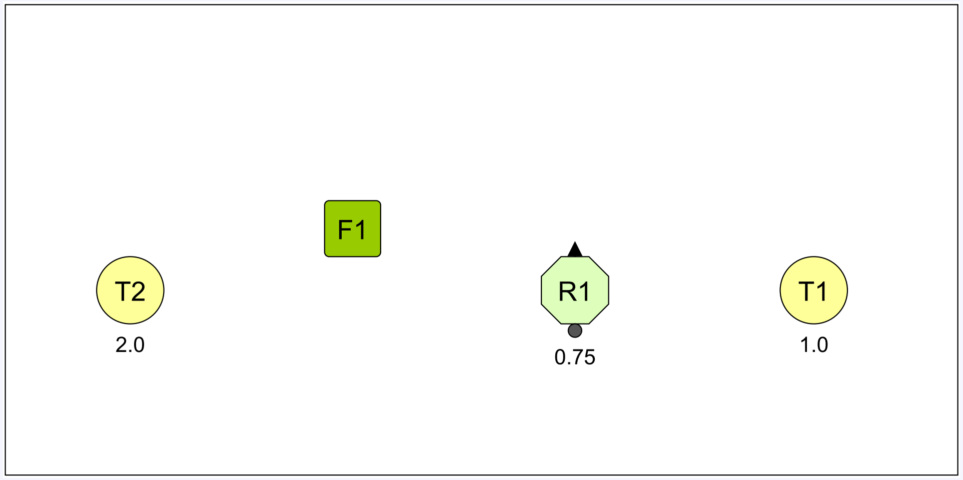
Here PUG assigns higher utility to a plan that delivers the sensor to T1 because it is closer.

One sensor, two targets, with T2 having twice the priority of T1.

Here PUG assigns higher utility to a plan that delivers the sensor to T2, despite its greater distance.



Demonstration Scenarios 4 and 5

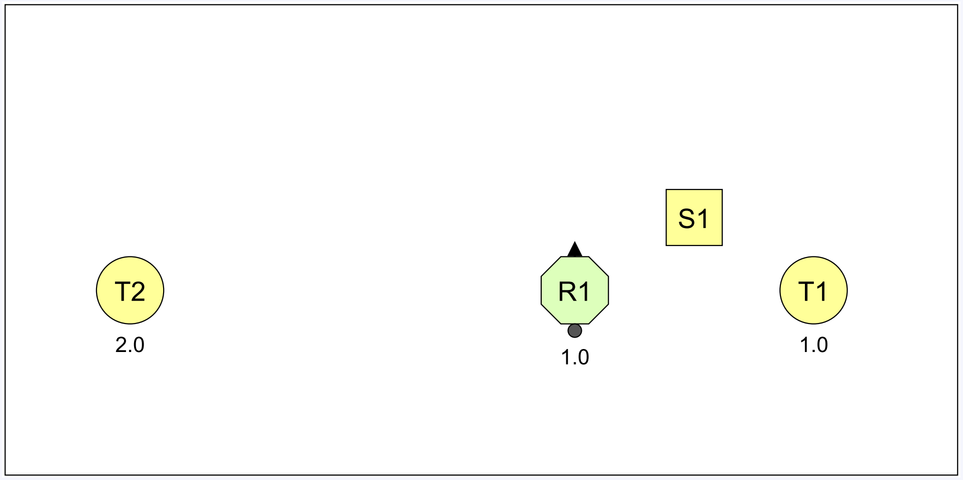


One sensor, two targets, with R1 lacking enough fuel to reach T2.

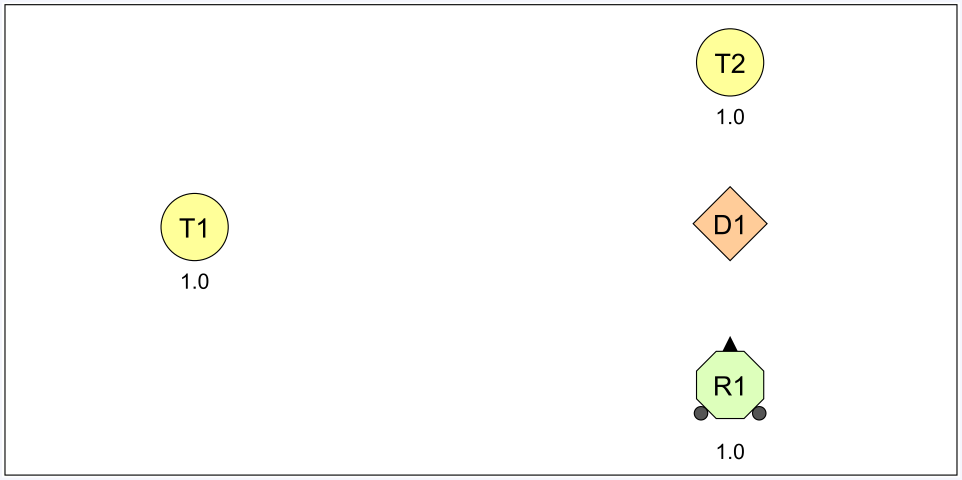
Here PUG prefers a plan to visits F1 to refuel, then delivers the sensor to the higher priority T2.

One sensor, two targets, higher priority for T2, and one sample.

Here PUG favors a plan in which it collects the sample S1 before delivering the sensor to T1.



Demonstration Scenarios 8 and 10

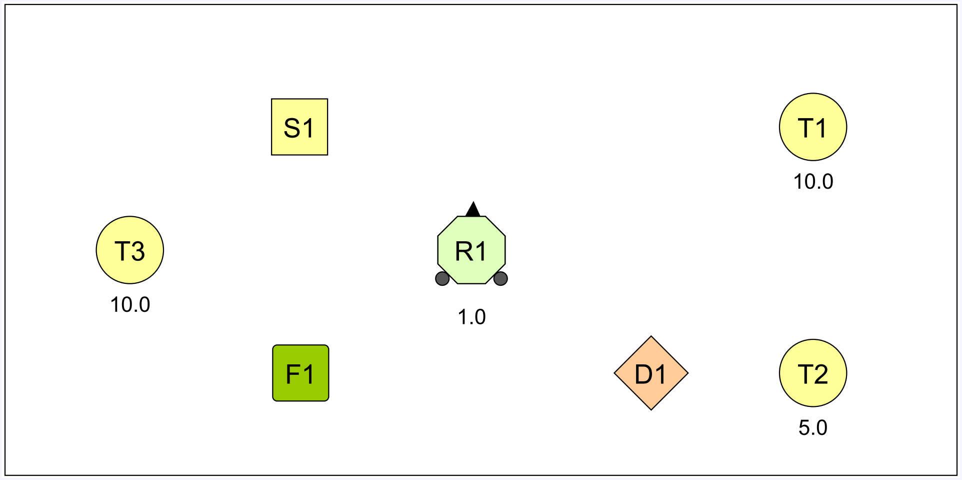


Two sensors, two targets, with a danger area D1 to avoid.

Here PUG bypasses D1 by first delivering one sensor to T1 and then delivering the other to T2.

Two sensor, three targets, a fuel depot, sample, and danger area.

Here PUG collects sample S1, refuels at F1, delivers a sensor to T3, refuels again, and delivers a sensor to T1, avoiding D1.



Summary of Demonstrations

These runs support our aims for PUG's behavior, showing that the architecture:

- Operates over space and time in pursuit of multiple and even conflicting goals;
- Calculates operator and plan utilities that vary according to situation details;
- Reasons about changes both in qualitative relations and in quantitative attributes; and
- Produces reasonable behavior that balances tradeoffs among different objectives.

Effort on these tasks varies, with PUG considering from 12 to 110 plans, but our focus here is on ranking rather than search.

Related Research

Our approach incorporates ideas from cognitive architectures, teleoreactive operators, logical inference, and heuristic search.

But how do our theoretical tenets relate to previous research?

- *Goals are the locus of utility, which guides plan generation.*
  - This combines insights about problem solving with behaviorism.
  - Partial satisfaction planning (Benton, 2009) adopts similar ideas.
- *Both goals and utilities are conditional / situation dependent.*
  - Choi (2011) and Talamadupula et al. (2010) on goal creation.
  - Roberts et al. (2015) on goal activation, deactivation, refinement.
- *Utility calculation via symbolic and numeric mental simulation.*

Our main contribution is the combination of these ideas into a unified and viable framework.

Directions for Future Work

Our initial results are encouraging, but in our future research we should:

- Extend framework to support *hierarchical methods* that PUG can use alone or combine with primitive operators (To et al., 2015).
- Support *satisficing* behavior (Simon, 1956) by halting upon finding a plan with acceptable utility.
- Explore other forms of utility, including ones that handle both *stochastic* operators and *discounting* over time.
- Augment PUG to *execute* and *monitor* its continuous plans, and to *adapt* them when utilities diverge from expected values.

These extensions should produce a more complete theory of planning with goals, utilities, and mental simulation.

Concluding Remarks

We have presented a new theory of planning in physical settings that posits:

- Symbolic goals are the distributed loci of numeric utility;
- Goals and utilities vary with the agent's situation;
- Mental structures comprise symbolic relations and numeric attributes; and
- Planning relies on mental simulation to calculate utilities and evaluate alternative plans.

We also reported PUG, an architecture that uses these tenets to handle scenarios with competing goals and limited resources.

End of Presentation