An Architectural Perspective on Planning, Execution, and Justified Agency

Pat Langley
Institute for the Study of Learning and Expertise
Palo Alto, California, USA

This research was supported by ONR Grant N00014-15-1-2517. This talk reports work done jointly with Dongkyu Choi, Mike Barley, Ben Meadows, and Edward Katz.

Key Ideas of the Talk

We can use a *cognitive architecture* to create intelligent agents that are able to:

- *Generat*e plans in the presence of conflicting objectives
- *Execute* these plans and *monitor* them for anomalies
- *Explain* and *justify* their behavior in terms of social norms

In this talk, I report a new architecture that demonstrates the first two abilities and should support the third.

Autonomy and Cognitive Architectures

The Nature of Autonomy

Autonomous artifacts are becoming ever more widely deployed in the form of:

- Self-driving cars
- Delivery drones
- Military robots

These require more than object recognition / control; they need:

- *Knowledge* about high-level relations and activities
- *Reasoning* over this content to update beliefs
- *Planning* with this knowledge to achieve goals

Truly autonomous agency depends on ***cognitive processing***.

Cognitive Architectures

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

- Makes strong assumptions about representations and mechanisms
- Incorporates ideas from psychology about the nature of cognition
- Contains modules that share memories and representations
- Comes with a programming language to build intelligent agents

A cognitive architecture is all about *mutual constraints*, in that it aims for a *unified theory* of the mind.

Example: A Planetary Rover

Consider another agent – a planetary rover sent on missions in which it must:

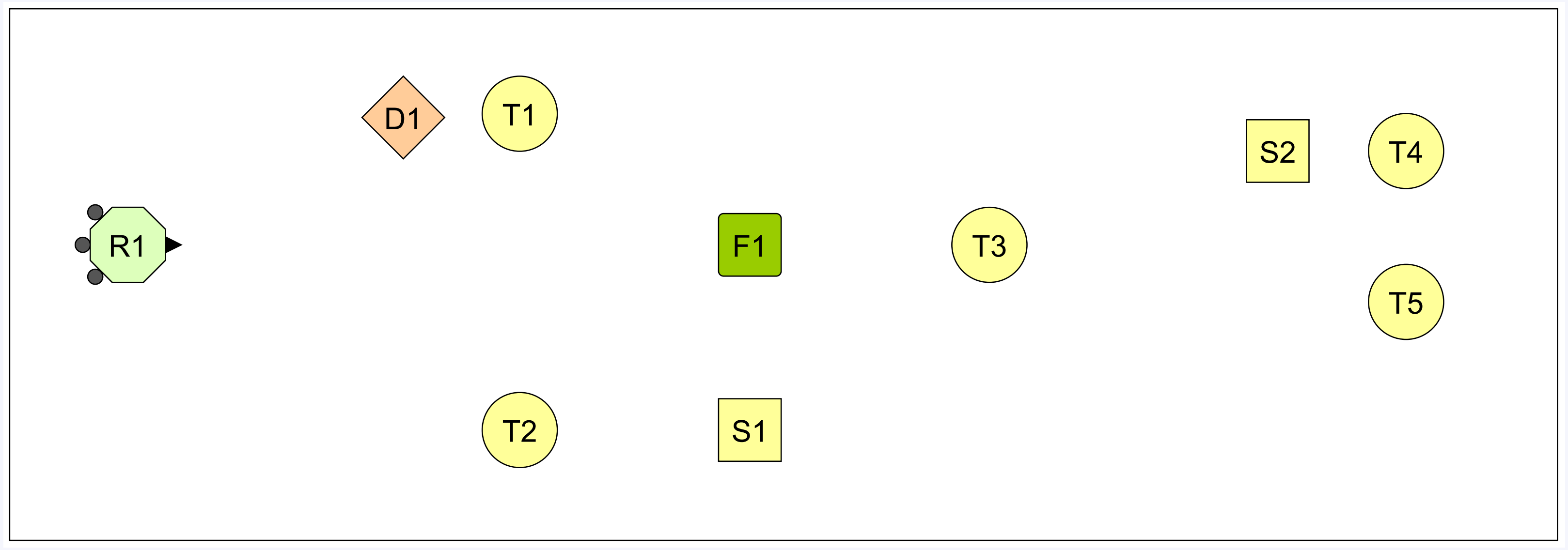
- Deposit sensors at various target sites
- Collect interesting samples it encounters
- Avoid the proximity of danger areas
- Retain enough fuel to carry out these tasks

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

We want computational agents that support all these capacities.

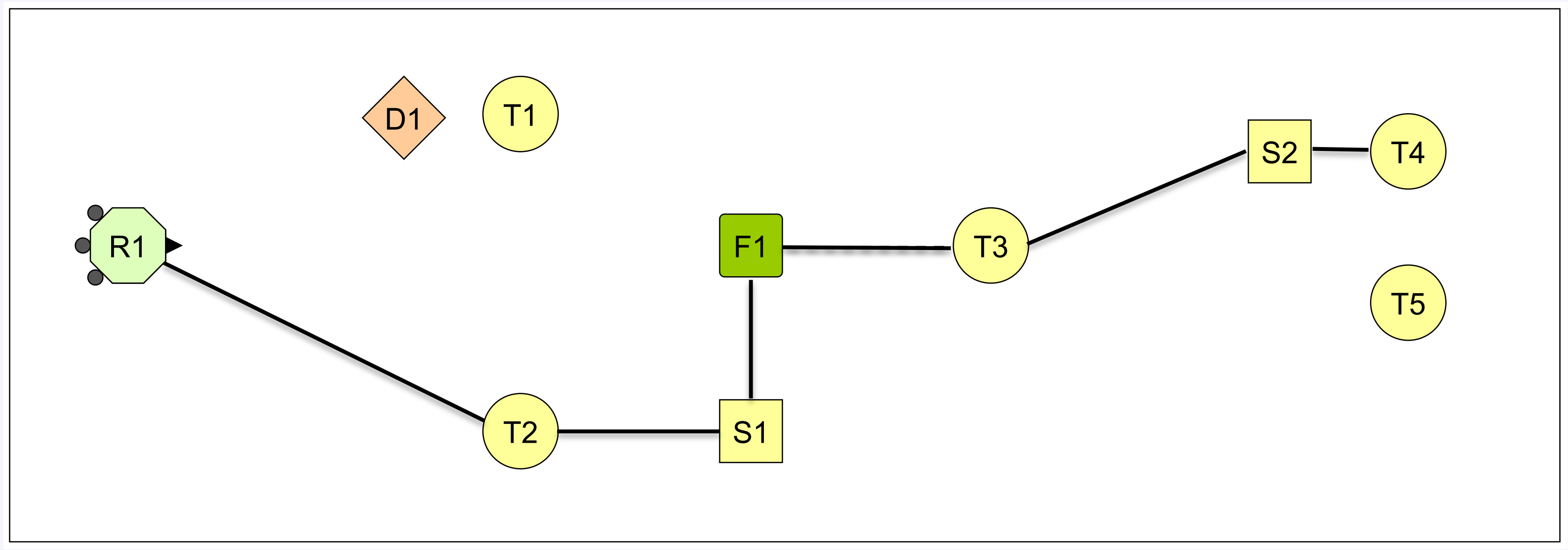
A Rover Scenario

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



A Rover Scenario

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



Here an effective plan might deliver a sensor to T2, collect S1, refuel at F1, deliver a sensor to T3, collect S2, and deliver a sensor to T4. The agent would bypass T1, as it is near D1.

Target Abilities

To operate in such complex, continuous domains, intelligent agents must:

- Reason about *qualitative* structures and *quantitative* attributes
- Decide which *goals* to adopt and determine their *priorities*
- Handle *conflicting* and even *inconsistent* objectives
- Balance *tradeoffs* among goals in a *situation-aware* manner

These are crucial in any domain that requires autonomy, from driving to planetary exploration to disaster relief.

What type of cognitive architecture can support these abilities?

Inference and Planning in the PUG Architecture

The PUG Architecture

We have developed a new architecture – PUG – that supports planning in continuous domains with conflicting goals.

The framework incorporates four core theoretical postulates:

- States are *relational structures* with *quantitative attributes*
- *Symbolic goals* are annotated with *numeric utilities*
- Goals and their utilities are *conditioned on states*
- Planning uses *numeric simulation* to guide heuristic search

This is a *hybrid* architecture that combines symbolic with numeric processing.

PUG’s Knowledge Structures

PUG incorporates four distinct types of generic knowledge:

- *Compositional rules* – define relational concepts and associated numeric attributes
- *Specialization rules* – discriminate among subclasses of more general concepts
- *Goal-generating rules* – specify when goal instances should be active and their utilities
- *Operators* – encode 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: 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: 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: 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 compositional rules to generate composite objects (e.g., *vector*) with derived numeric attributes
- Matches specialization rules against objects to infer specialized relations (e.g., *at*, *facing*)
- Matches goal rules against these derived beliefs to generate active goal instances
- Calculates utilities of satisfied active goals and combines them to compute a state's utility

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

Examples: 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)

PUG’s Planning Mechanisms

PUG carries out forward search through a space of partial plans. On each step it:

- Finds each operator that is applicable in the state
- Uses numeric simulation to predict its trajectory
- Calculates the utility of each operator over time
- Selects the best alternative to extend the partial plan

Goal-based utility guides a heuristic depth-first search; plans need *not* satisfy all goals.

Parameters include maximum length, nodes to be considered, and number of solutions desired.

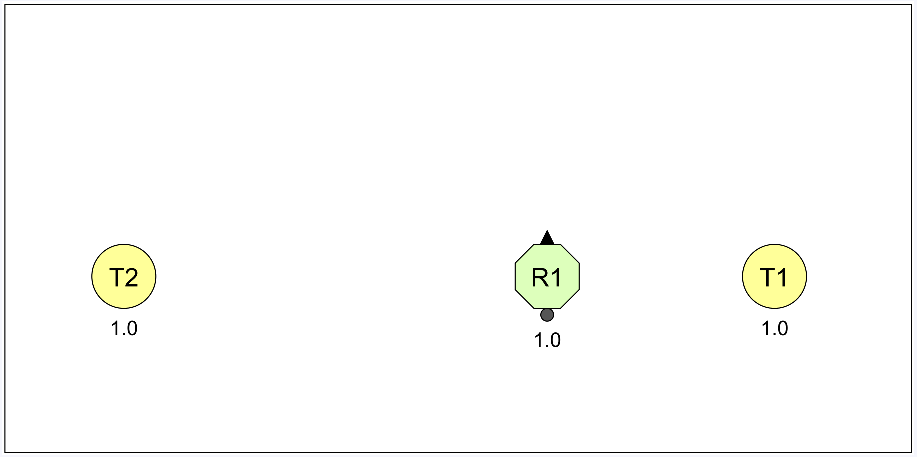
Mental Simulation and Utility

PUG determines the utility for each operator O instance by:

- Applying O repeatedly, starting from the current state
- Calculating the utility on each time step T by:
  - Deriving beliefs that hold on that time step
  - Generating goal instances with satisfied conditions
  - *Adding* the utility of each matched *positive* goal
  - *Subtracting* the utility of each matched *negated* goal
- Terminating on reaching a state that matches an operators' results or fails to match its conditions

The planner uses the *average utility* as its evaluation metric during operator selection and backtracking.

Demonstration Scenarios 1 and 2

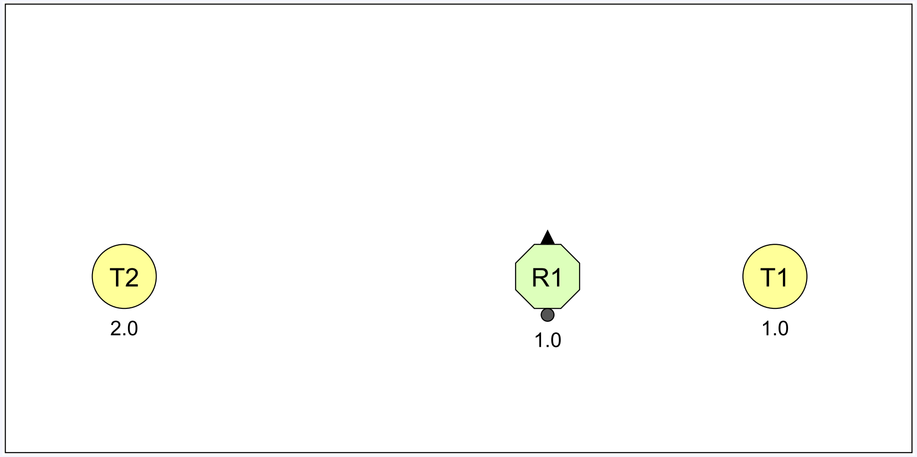


One sensor, two targets, with T2 more distant from R1 than T1.

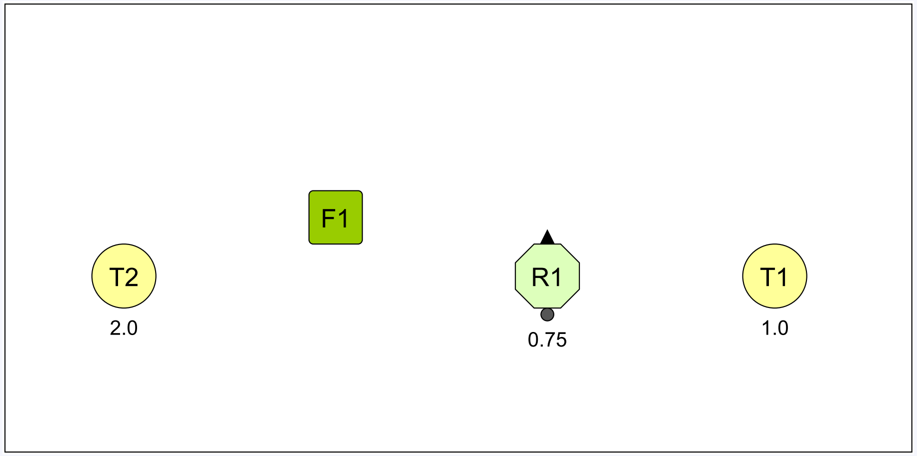
Here PUG assigns higher utility to a plan that delivers the sensor to T1 because it is closer.

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.

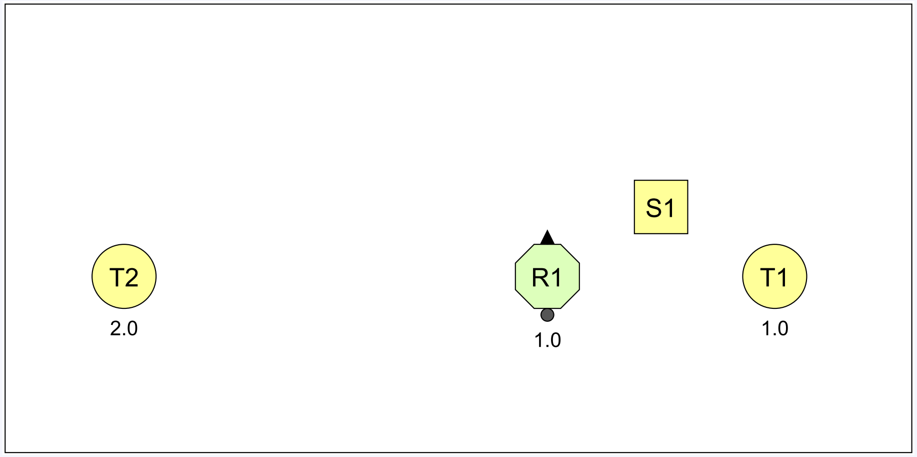


Demonstration Scenarios 4 and 5



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

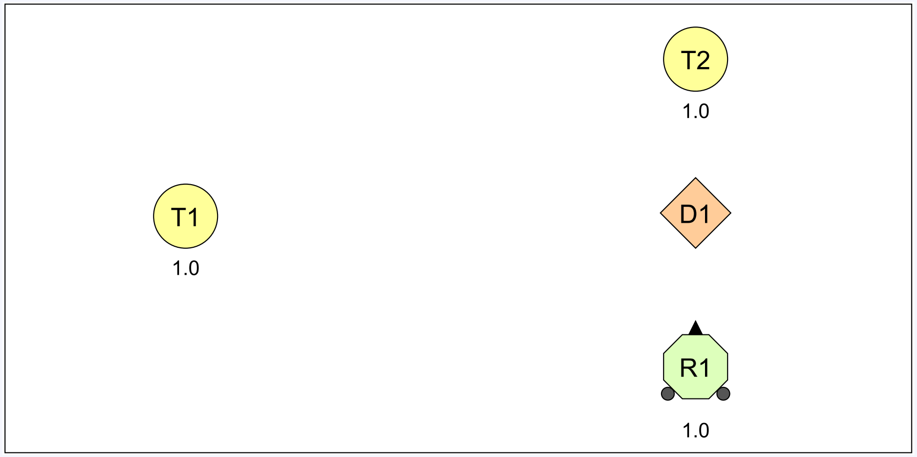
Here PUG prefers a plan to visit F1 to refuel, then delivers the sensor to the higher priority T2.



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

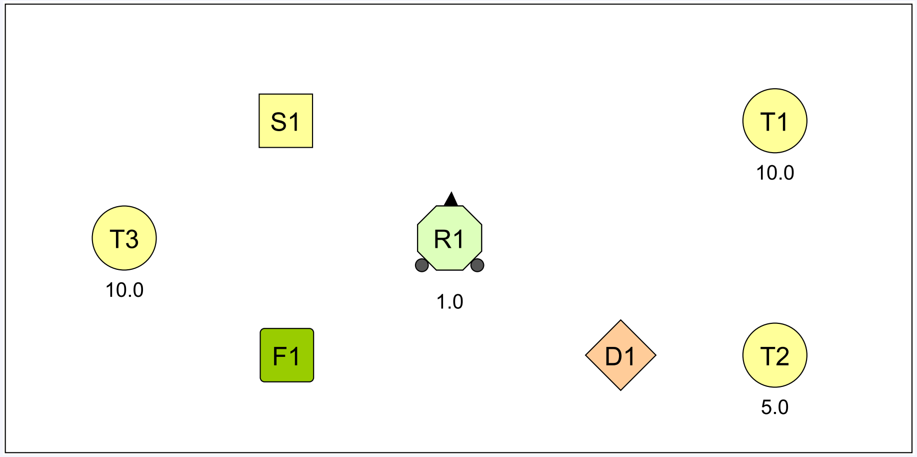


Two sensors, two targets, with a danger area D1 to avoid.

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.



Key Results

These runs support our aims for PUG's planning behavior, as they show it:

- Reasons over space and time in pursuit of multiple and even conflicting goals;
- Reasons about both qualitative and quantitative aspects of the agent's state;
- Calculates operator and plan utilities that vary according to situation details; and
- Produces reasonable behavior that balances tradeoffs among different objectives.

Effort on tasks varies, with PUG considering from 12 to 110 plans, but our focus is on *ranking*, not search.

Extending PUG to Execution and Monitoring

An Extended Architecture: PUG/X

We have also extended PUG to *execute* and *monitor* the plans it generates by:

- Using mental simulation, inference, and goal creation to form expectations for each time step
- Comparing the expected states to predicted ones on each step
- On detecting anomalies – disagreements between expectations and observations– replanning from the current state

Alternation between plan generation and execution continues until a plan succeeds or no revised plan emerges.

Mental simulation is deterministic and so adds only a constant factor to computation costs.

Four Types of Anomalies

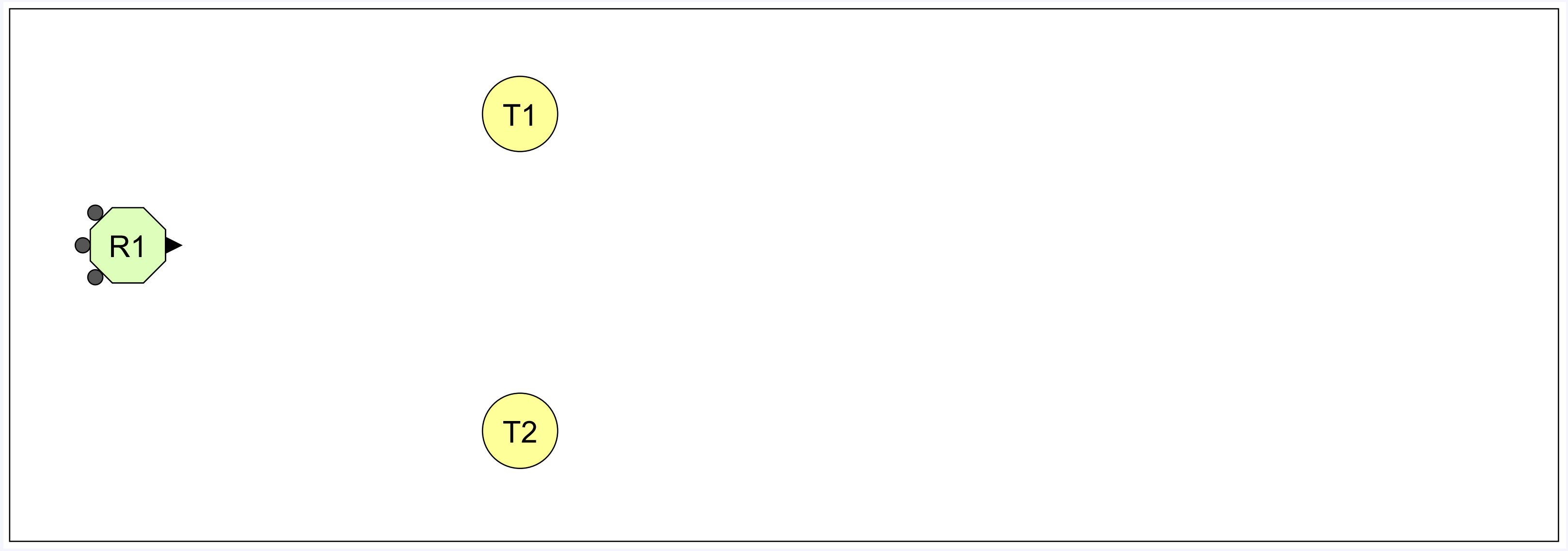
The PUG/X architecture posits four types of anomalies that can arise while monitoring:

- An operator’s *conditions* are not matched when expected
- An operator’s *results* are not produced when expected
- An operator’s *utility* differs ‘enough’ from expected values
- *Goals* generated for observed states differ from expected ones

Any of these discrepancies leads the agent to search for a new plan from the current state.

An Altered Rover Scenario

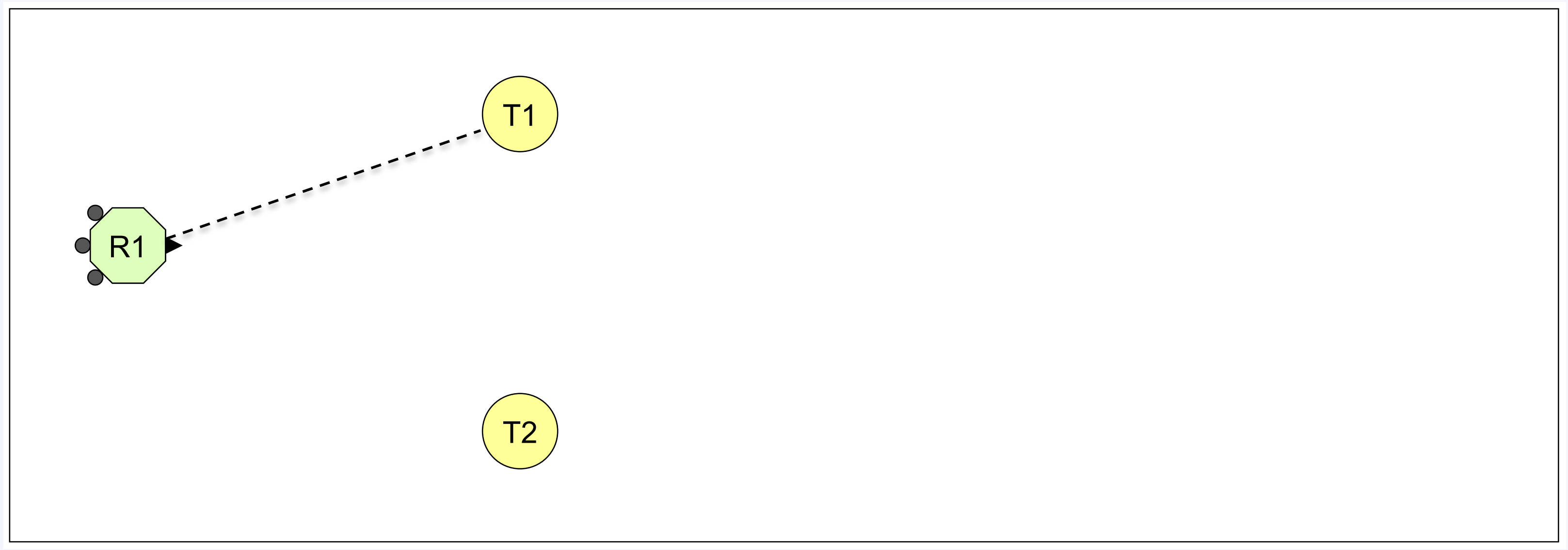
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

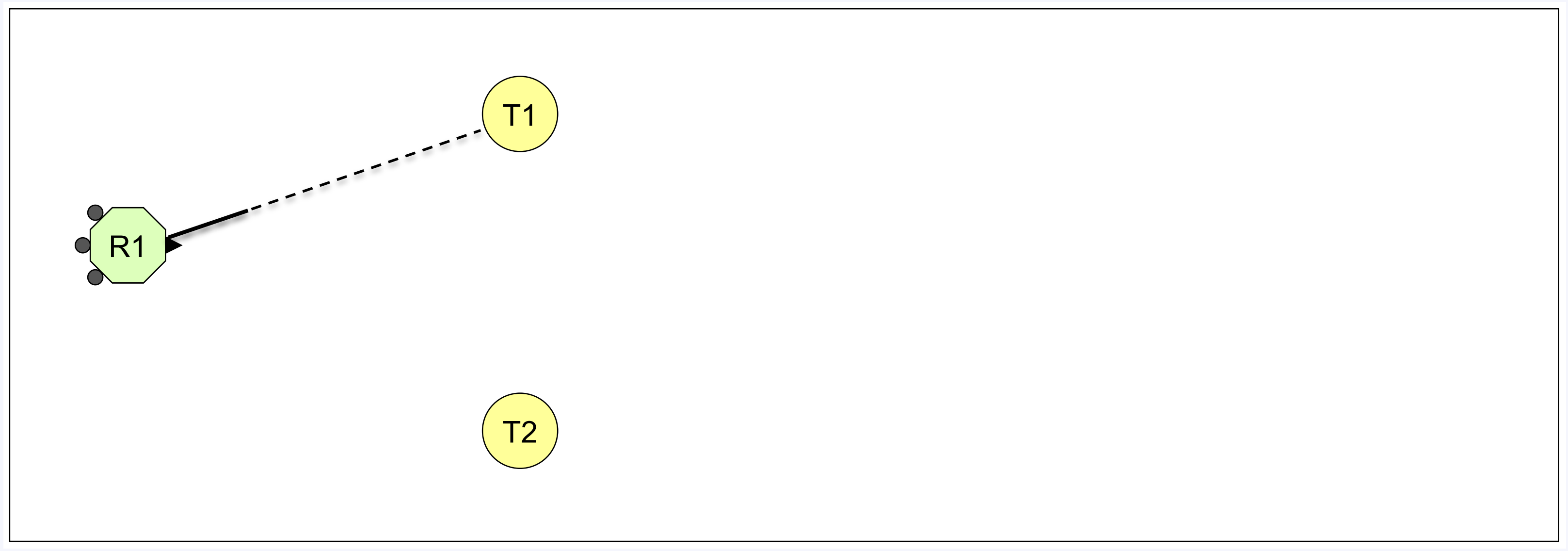
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

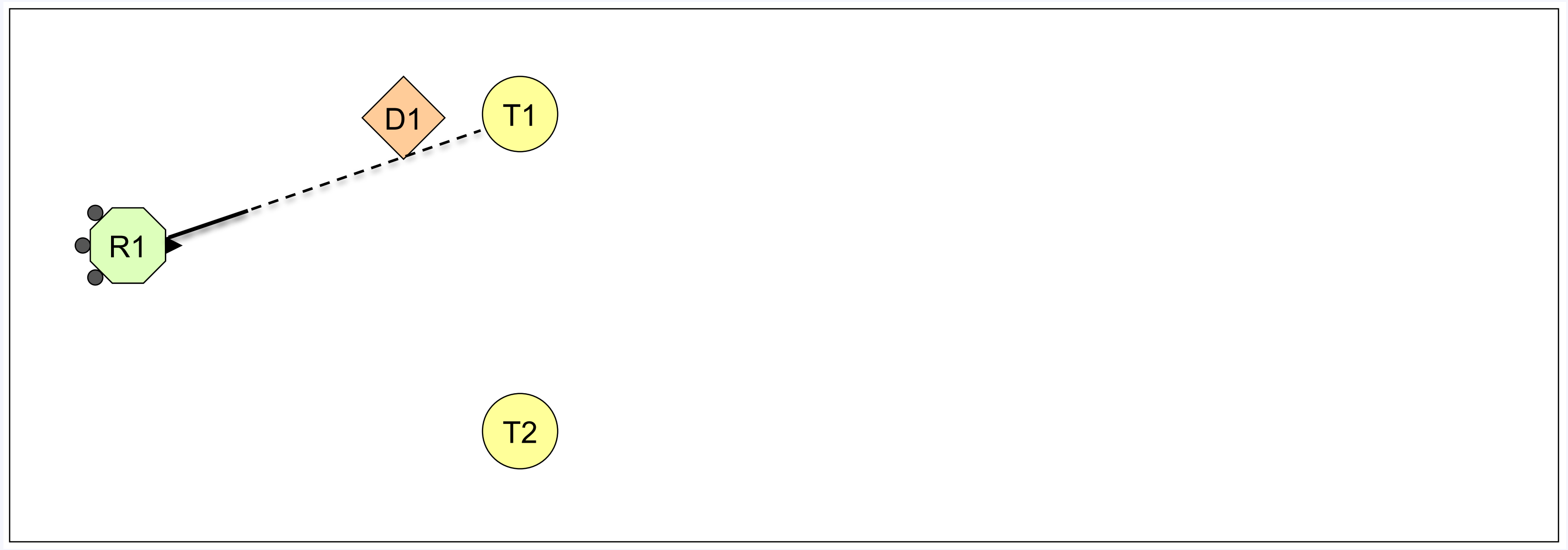
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

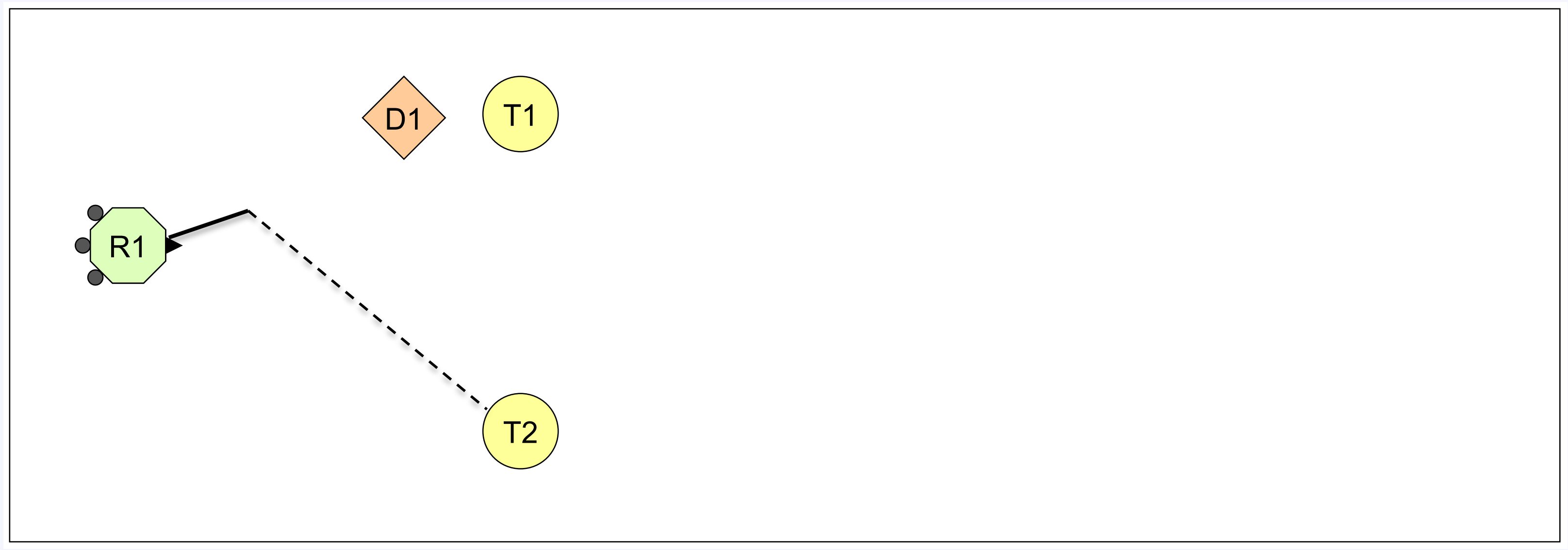
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

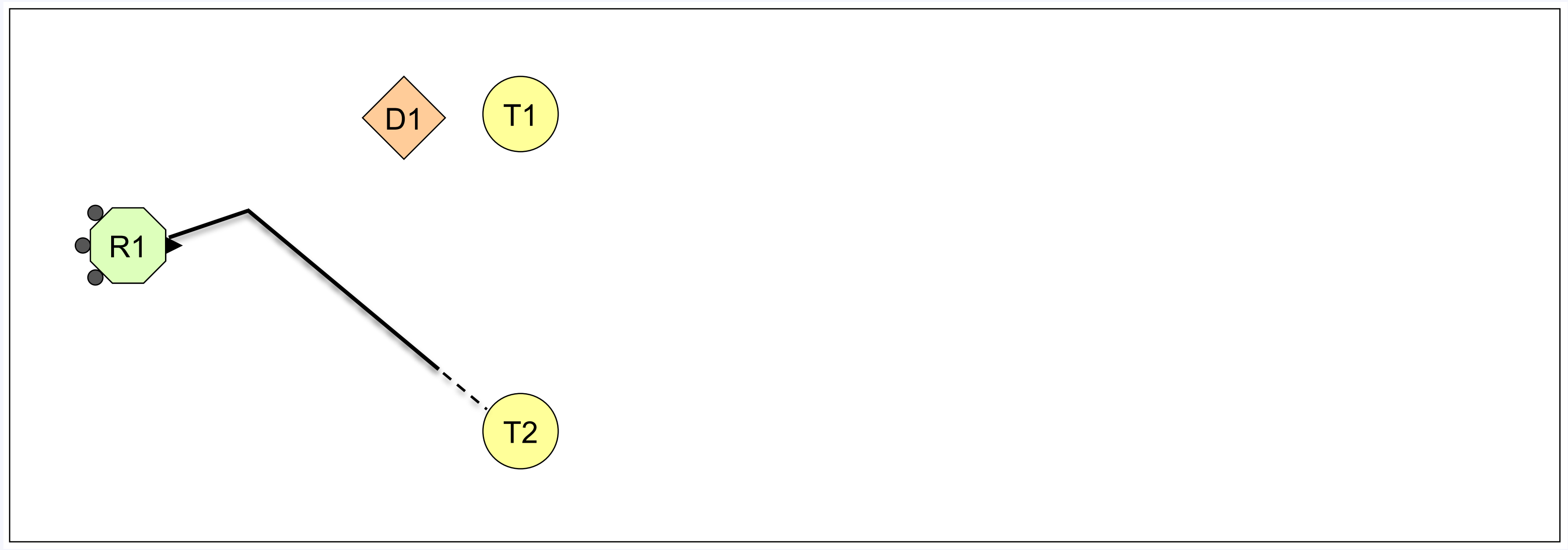
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

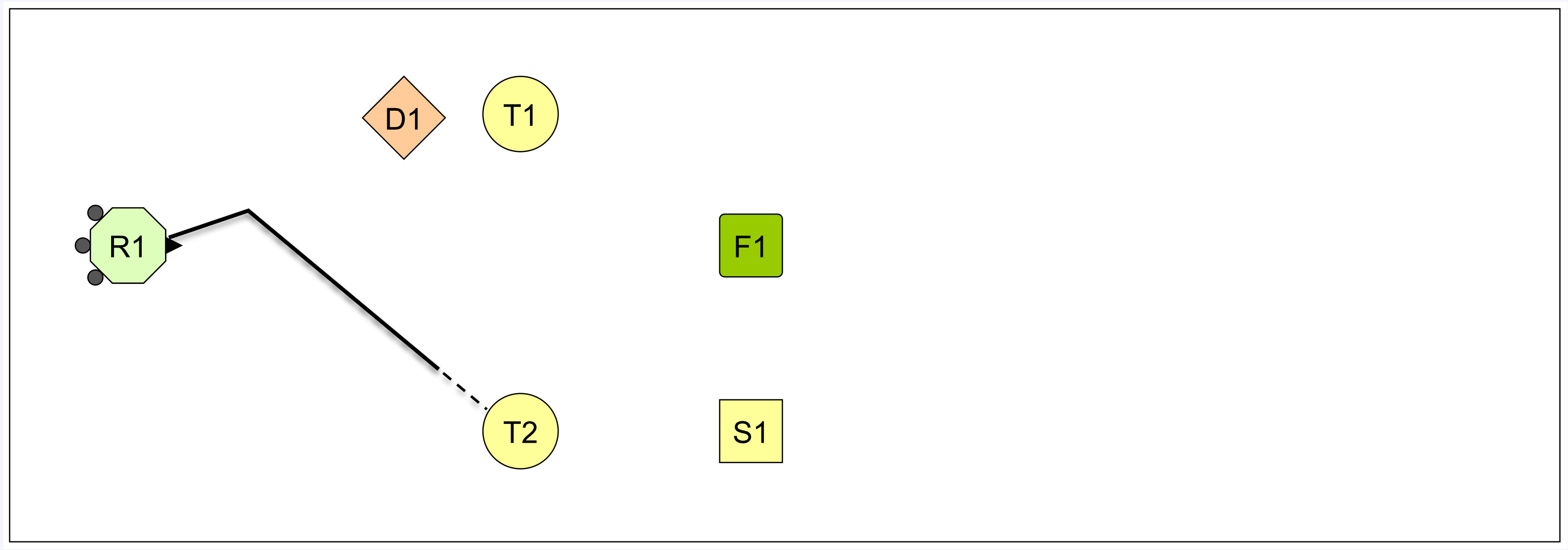
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

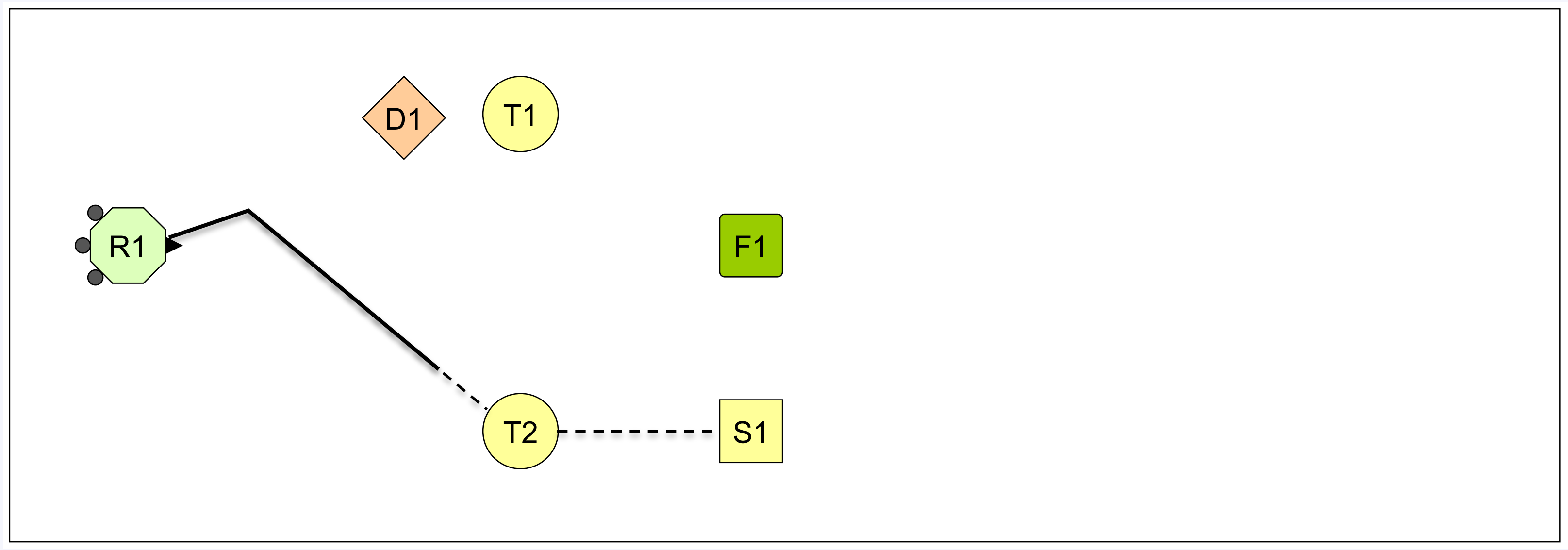
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

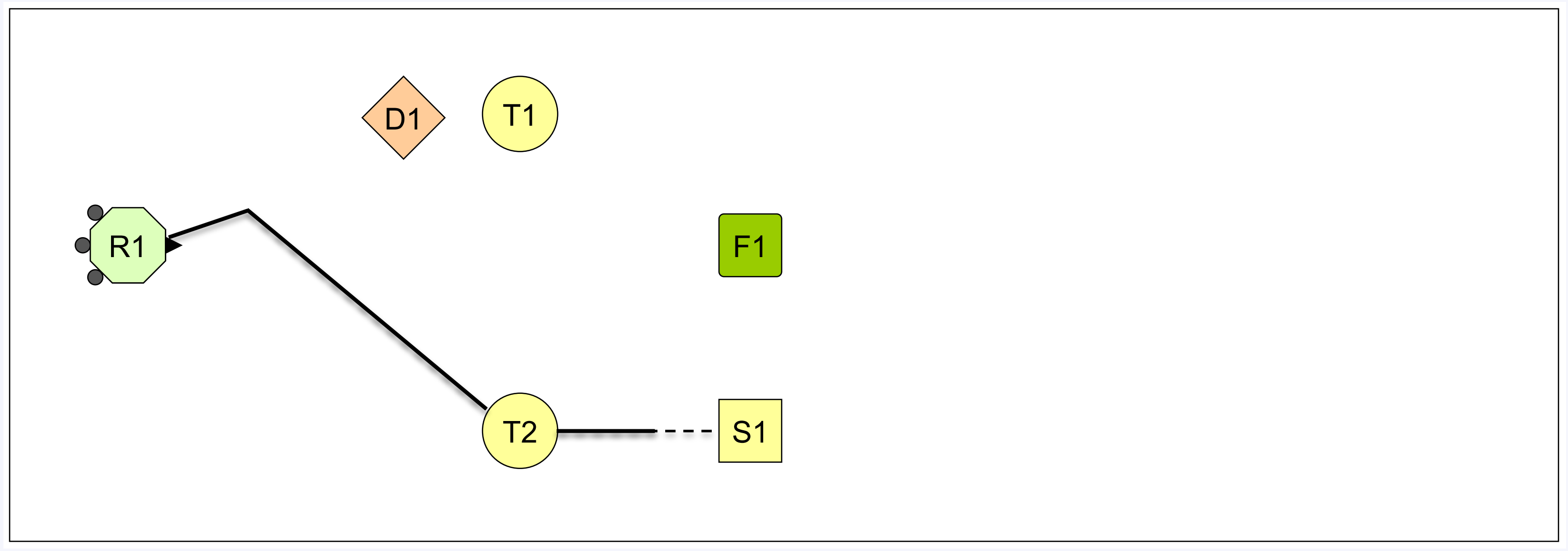
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

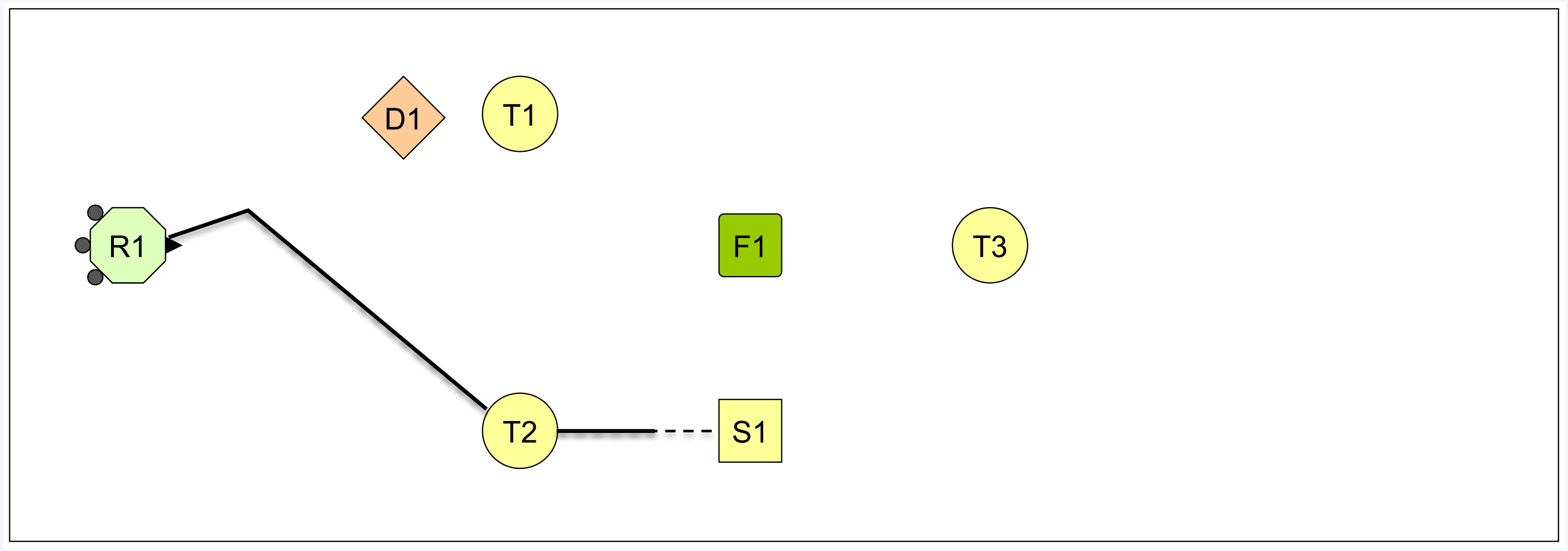
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

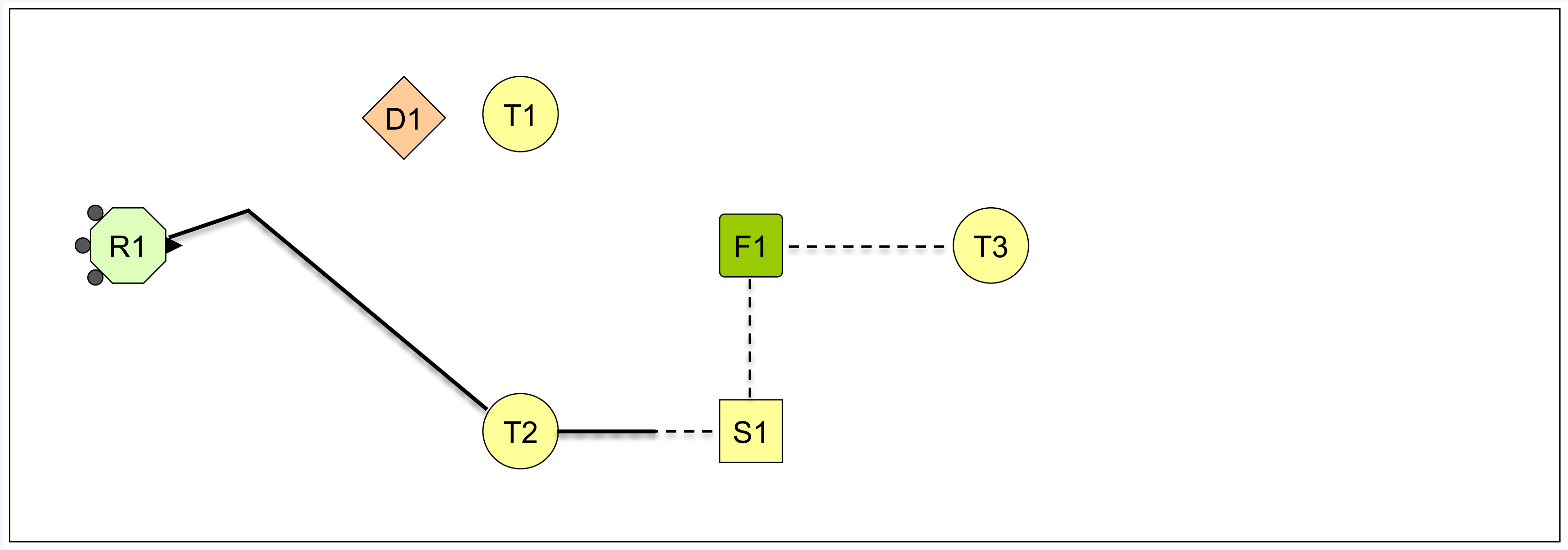
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

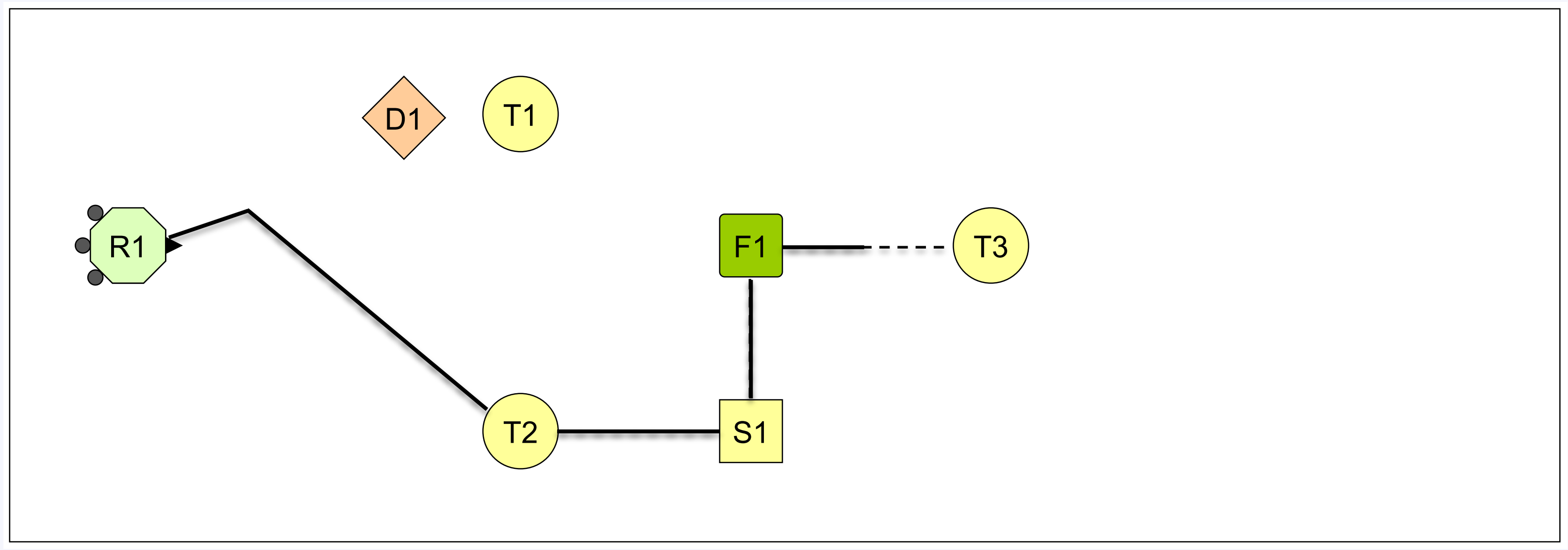
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

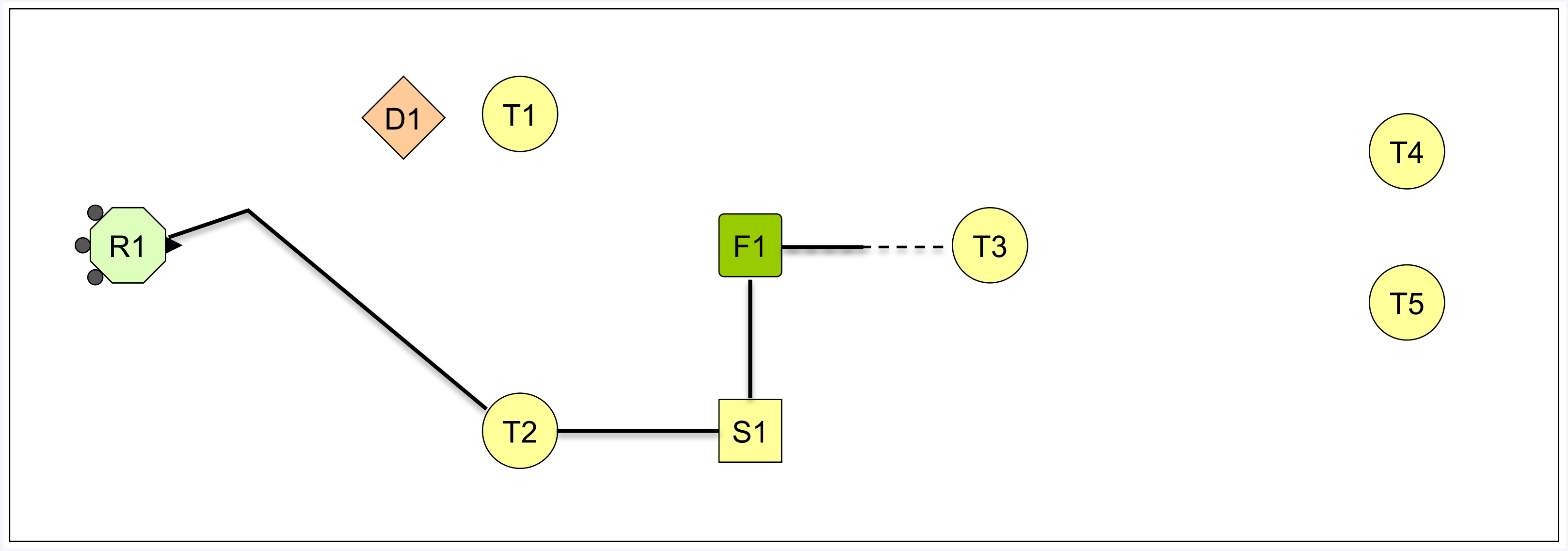
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

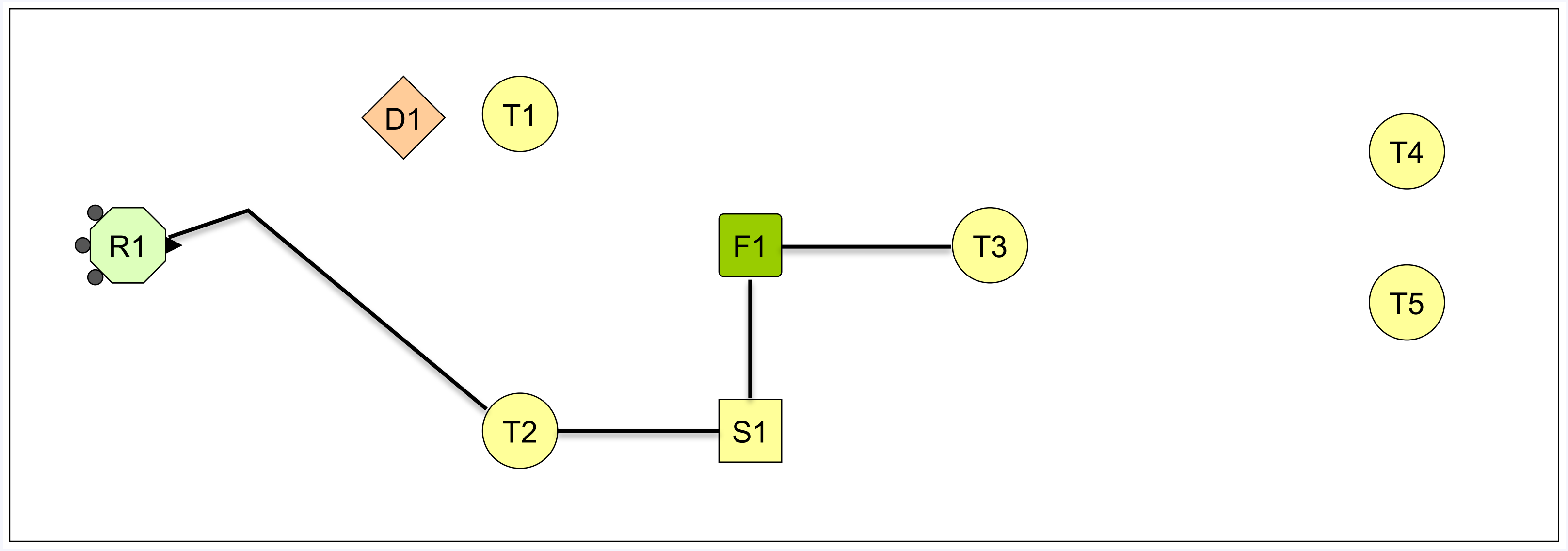
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

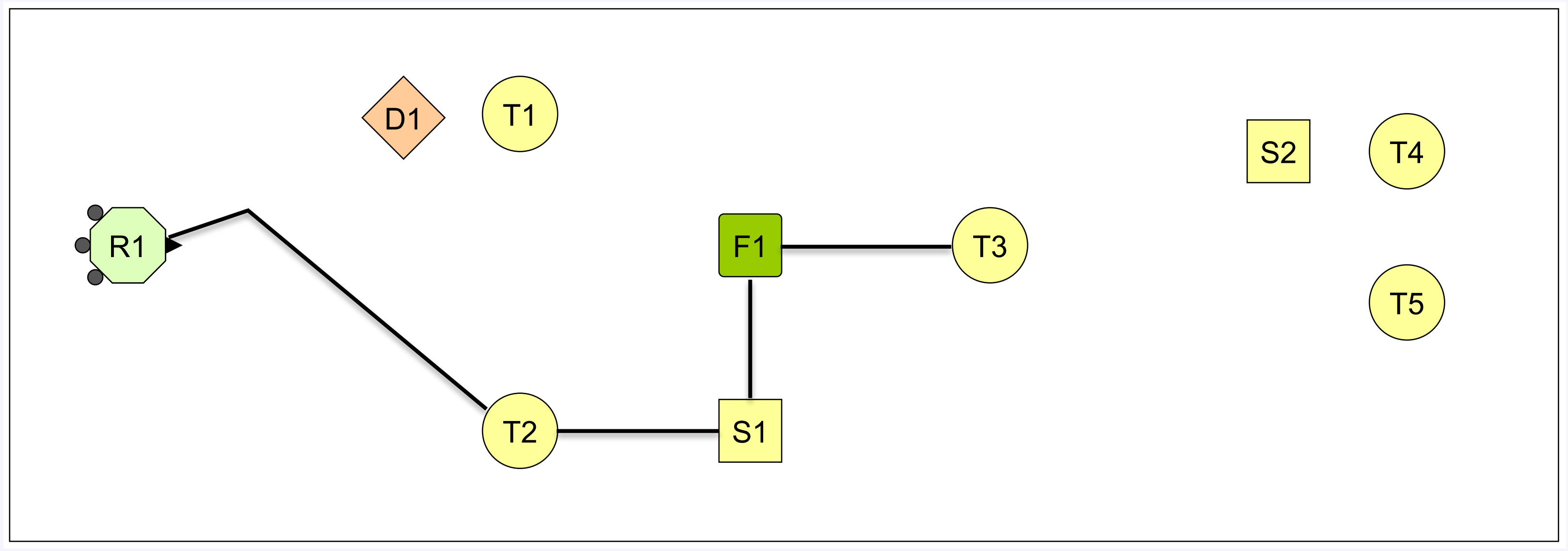
Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

An Altered Rover Scenario

Consider our motivating scenario with three sensors, five target sites, one fuel depot, one sample, and one danger area.



Here PUG/X builds an initial plan to deliver a sensor to T1, then shifts to T2 on detecting D1, extends it to include S1, decides to refuel on sensing T3, and finally adds S2 and T4.

Empirical Demonstrations

We have demonstrated PUG/X's operation on scenarios with:

- Nominal plan execution in which no surprises arise
- Anomalous operator conditions
  - Robot veered slightly to right, samples moved on their own
- Anomalous operator termination
  - Robot advances more slowly than expected
- Anomalous utilities
  - Increased radiation from danger area
- Unanticipated goals
  - Newly observed target sites, samples, danger areas

One scenario required extended operation: the agent continued to detect new (20) target sites and replan in response.

Some Caveats

The current implementation of PUG/X makes six simplifying assumptions:

- The agent can execute only one operator at a time
- Operators have deterministic effects on the environment
- Operator descriptions are usually (but not always) accurate
- Environmental changes are typically due to agent actions
- Once it perceives object O, it has complete information on O
- Execution halts while the agent is generating a new plan

The assumptions are not central to the theoretical framework and we will address them in future work.

Intellectual Precursors

Our research incorporates ideas a number of earlier efforts on plan generation, execution, and monitoring:

- Cognitive architectures (Soar, Prodigy, ICARUS, MIDCA)
- Three-tiered architectures and teleoreactive systems
- Integrated planning and execution (PRS, SIPE, CASPER)
- Goal reasoning (Talamadupula et al., 2010; Roberts et al. (2015)

The PUG/X architecture is distinctive in its combination of quantitative simulation with goal-oriented utility.

Review of Key Ideas

I have presented a theory for plan generation, execution, and monitoring in physical settings that posits:

- *Symbolic goals are the loci of numeric utility*, with goals and utilities varying by situation
- Mental structures – long and short term – comprise *symbolic relations* and *numeric attributes*
- Planning and monitoring rely on *quantitative mental simulation*
- *Replanning occurs when monitoring detects anomalies* involving operator conditions, results, utilities – or *goals*

We have incorporated these ideas into the PUG/X architecture and demonstrated it on scenarios with unexpected events.

Explainable, Normative, and Justified Agency

A Motivating Example

Suppose that Dan drives a friend, Eve, with a ruptured appendix to the hospital. On the way, he:

- Exceeds the speed limit
- Weaves in and out of traffic
- Slows at red lights but runs them
- Detours briefly onto a sidewalk
- Yet retains control and avoids collisions

Dan later defends his actions because Eve's life was in danger, so reaching the hospital was more important than traffic laws.

We will say that, in this scenario, Dan exhibits ***justified agency***.

More on Autonomy

Autonomous artifacts are becoming ever more widely deployed in the form of:

- Self-driving cars
- Delivery drones
- Military robots

But before such systems can gain widespread acceptance, they must first be able to:

- *Explain* their behavior in understandable terms;
- Follow the laws, customs, and morals of *society*.

We claim that ***both*** abilities are required for justified agency.

Explainable Agency

When we make a decision, we can often explain the choices we considered and why we selected one over others.

Definition:

- *An intelligent system exhibits* ***explainable agency*** *if it can provide, on request, the reasons for its activities.*

Examples of explainable agency:

- Why did you prefer driving route A to work over route B?
  - Route A had fewer traffic signals and it was still pretty short.
- Why did you swerve suddenly into the next lane?
  - It was the only way to avoid hitting a fallen tree limb.

Facets of Explainable Agency

An explainable agent should be able to answer questions about:

- Alternatives considered / selected (e.g., routes, lanes)
- Reasons selected / criteria used (e.g., shorter, less crowded)
- Responses in other situations (e.g., near ambulance)

The agent should answer questions about both its *generation* and *execution* of plans.

Previous research:

- Explainable expert systems (Swartout, 1991)
- Explainable reactive execution (Johnson, 1994; van Lent, 2004)
- Explainable planning (Smith, 2012; Fox et al., 2017)

PUG/X and Explainable Agency

PUG/X already stores most content needed for self explanation:

- Each node in the search tree is a partial plan
  - Including options for extending it and associated utilities
- Execution always follows one of these stored plans
  - Monitoring compares traces to the plan's expectations

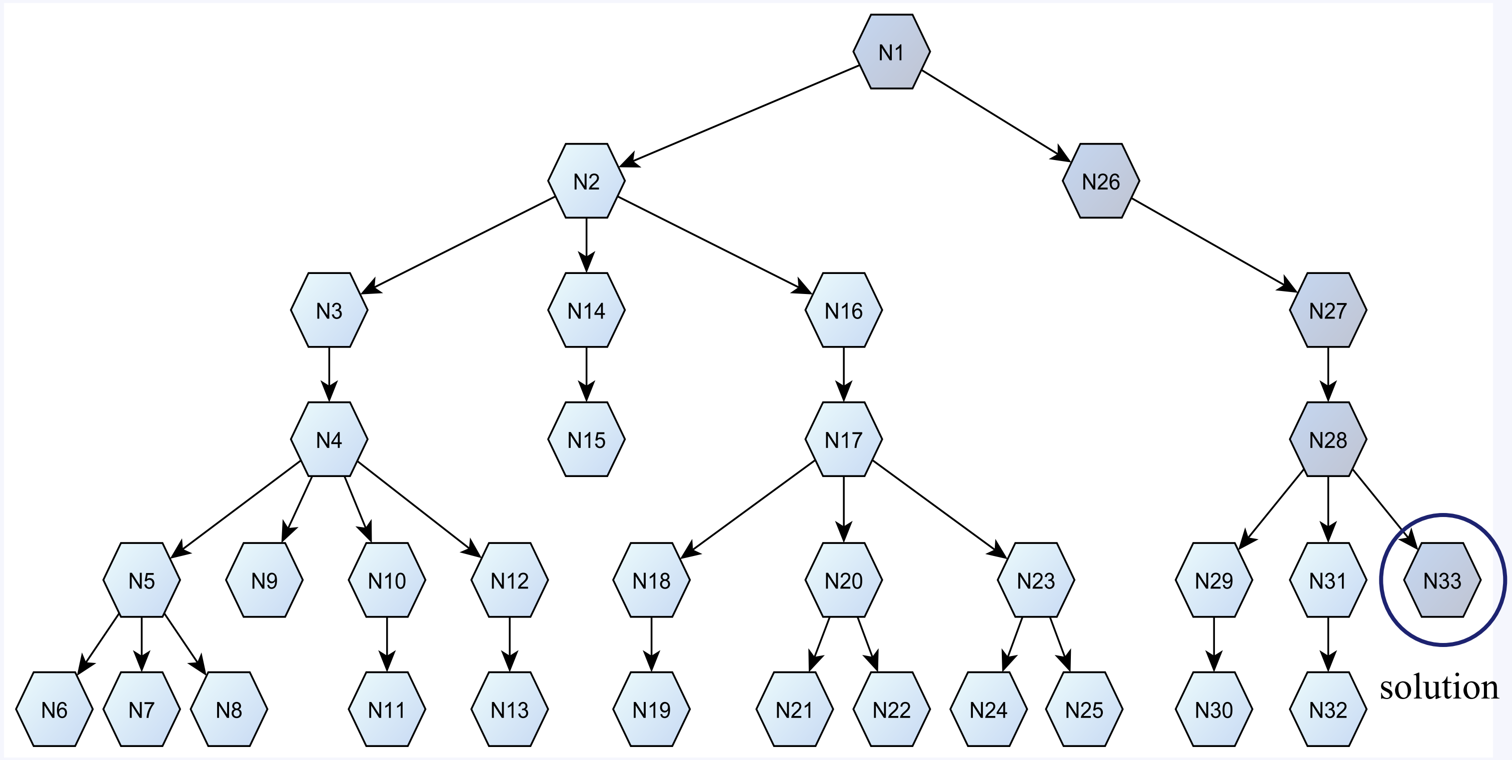
If we store anomalies when they interrupt execution, we have the key elements of an episodic memory.

An extended PUG would *index* this content, *retrieve* relevant decisions as needed, and *communicate* it to answer queries.

These should let the architecture exhibit explainable agency.

PUG/X and Explainable Agency

During plan generation, PUG generates nodes in a search tree. Each node is a partial plan that elaborates on its parent.



The search tree includes alternative choices and their scores.

Normative Agency

Humans are driven by goals, but they must also operate within their society’s norms.

Definition:

- *An intelligent system exhibits **normative agency** if, to the extent possible, it follows the norms of its society.*

Examples of normative agency:

- Paying for food rather than stealing it
- Saluting to a superior officer
- Waiting in line rather than cutting ahead
- Recycling to help the environment

These all canalize people’s behavior in certain directions.

Facets of Normative Agency

A normative agent's behavior should take into account:

- Formal laws (e.g., obey traffic signals)
- Military orders (e.g., get up at reveille)
- Informal customs (e.g., Pittsburgh left turn)
- Moral tenets (e.g., favor life over property)

Different norms may conflict, so that the agent must handle tradeoffs among them.

Previous research:

- Legal reasoning (e.g., Branting, 2000)
- Moral reasoning (e.g., McLaren, 2005; Deghani, 2008; Mikhail, 2007; Iba & Langley, 2011; Malle et al., 2015)

PUG/X and Normative Agency

The PUG/X architecture can already specify physical criteria:

- Goals are conditioned on qualitative relations
  - E.g., stop if the red is light, avoid collisions
- Utilities incorporate quantitative attributes
  - E.g., faster travel is better, closer calls are worse

These can encode many social norms, including tradeoffs, but we must extend it to support:

- Mitigating factors that modulate acceptability
- Goals for others' mental states (e.g., minimize pain)

Together, these should let PUG/X exhibit normative agency.

Justified Agency

When we make a decision, we can often state the choices we considered and how norms influenced our selection.

Definition:

- *An intelligent system exhibits **justified agency** if it follows society's norms and explains its activities in those terms.*

Examples of justified agency:

- Stealing food to help a starving child (and explaining why)
- Disobeying an order that you consider illegal (and . . .)
- Cutting in line to avoid missing a flight (and . . .)
- Breaking traffic laws for a medical emergency

Justified agency is most interesting when norms ***conflict***.

The Character of Justified Agency

Three major design issues arise in devising justified agency:

- Generating, storing, and using explanations
- Encoding, using, and combining norms
- How to integrate explanations with social norms

We have considered the first two issues, but what of the third?

Consider a plausible hypothesis:

- *Any intelligent system that supports explainable agency and normative agency* ***will also*** *exhibit justified agency.*

If we include social norms in our agent's goals and values, then we get justified agency with no extra effort.

Testing the Hypothesis

The hypothesis does ***not*** follow logically from our definitions.

- Justified agency requires the ability to explain decisions and reason about norms, but they may not be sufficient.
  - *Agency may be more complex than assumed*
  - *Norms may demand richer forms of explanation*

To test it, we must construct explainable and normative agents, combine them, and measure their ability to justify.

Simulated domains for urban driving, robotic rescue, and other mission-oriented settings support such studies.

Intelligent Agents of the Future

Designing and constructing justifiable agents is an important step toward replicating the full range of human intelligence.

The ultimate demonstrations of such autonomous artifacts would be:

- Self-driving cars that sway judges in traffic court
- Police drones that defend themselves in civil suits
- Military robots that win court martials about their actions in combat

We encourage other AI researchers to pursue this audacious vision of explainable, normative, and justified agency.