REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
11-07-2023	Final Report	15-Nov-2019 - 14-May-2023

4. TITLE AND SUBTITLE
Final Report: ACT-NOW: Autonomous Cognitive Technologies for Novelty in Open Worlds

5a. CONTRACT NUMBER
W911NF-20-2-0006

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Tufts University
Research Administration
20 Professors Row
Medford, MA 02155 -5807

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
76263-MI-DRP.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Matthias Scheutz
UU	UU	UU	UU		19b. TELEPHONE NUMBER 617-627-0453

RPPR Final Report
as of 14-Jul-2023

Agency Code: 21XD

Proposal Number: 76263MIDRP **Agreement Number: W911NF-20-2-0006**

INVESTIGATOR(S):

Name: Chitta Baral
Email: chitta.baral@asu.edu
Phone Number: 4807276047
Principal:

Name: Jivko Sinapov
Email: Jivko.Sinapov@tufts.edu
Phone Number: 6176272225
Principal:

Name: Liping Liu
Email: Liping.Liu@tufts.edu
Phone Number: 6176270556
Principal:

Name: Matthias Scheutz
Email: Matthias.Scheutz@tufts.edu
Phone Number: 6176270453
Principal: Y

Name: Michael C. Hughes
Email: Michael.Hughes@tufts.edu
Phone Number: 6176272229
Principal:

Name: Subbarao Kambhampati
Email: RAO@ASU.EDU
Phone Number: 4809650113
Principal:

Organization: **Tufts University**
Address: Research Administration, Medford, MA 021555807
Country: USA

DUNS Number: 073134835 EIN: 042103634
Report Date: 14-Aug-2023 Date Received: 11-Jul-2023
Final Report for Period Beginning 15-Nov-2019 and Ending 14-May-2023
Title: ACT-NOW: Autonomous Cognitive Technologies for Novelty in Open Worlds
Begin Performance Period: 15-Nov-2019 **End Performance Period:** 30-Jun-2023
Report Term: 0-Other

Submitted By: Matthias Scheutz Email: Matthias.Scheutz@tufts.edu
Phone: (617) 627-0453

STEM Degrees: **STEM Participants:**

Major Goals: The main objective of the project was to develop a versatile novelty-aware cognitive robotic architecture that will enable artificial agents to cope with "unknown unknowns". Specifically, we aimed to develop inference-based and learning-based methods for novelty detection, combining statistically trained subsymbolic models with knowledge-based symbolic models and utilizing a mixture of symbolic and statistical inference for detecting deviations from expectations. We used three TA1 tasks to develop these algorithms: Polycraft, Monopoly, and NLP (the last was removed in the final phase of the project). Another main goal was to also characterize the detected novelties and determine how to cope with them based on the effects they had on the agent's task. We also planned to integrate all algorithms for novelty detection, characterization, and accommodation into the unified

architecture framework to be able to utilize them in future work in different domains.

Accomplishments: Our team had the highest performing agent in the Polycraft and NLP domains, and the second highest in the Monopoly domain. Our accomplishments include:

1. Foundational work on novelty concepts
2. Architectural developments for novelty handling
3. Visual novelty detection
4. Agent novelty detection and accommodation
5. Inference-based novelty detection
6. Hybrid planning reinforcement learning for novelty accommodation
7. Multi-player novelty detection and accommodation

The details are described in the attached report.

Training Opportunities: The project allowed for the training of sizeable number of graduate students over the course of the 42 months (24 to varying degrees of involvement).

Results Dissemination: Throughout the project, we disseminated our research results in top national and international venues (a detailed publication list is included in the report).

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: The integrated novelty-aware and novelty-handling cognitive robotic architecture has been transitioned to Thinking Robots Inc.

PARTICIPANTS:

Participant Type: PD/PI
Participant: Matthias Scheutz
Person Months Worked: 4.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: Jivko Sinapov
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: Michael Hughes
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: Liping Liu
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:

National Academy Member: N

Participant Type: Co PD/PI
Participant: Chitta Baral
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: Subbarao Kampambathi
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Vasanth Sarathy
Person Months Worked: 8.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Shivam Goel
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Yash Shukla
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Gyan Tatiya
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Technician
Participant: Evan Krause
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Technician
Participant: Ravenna Thielstorm
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Technician
Participant: Marlow Fawn
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Technician
Participant: Eric Wyss
Person Months Worked: 3.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Staff Scientist (doctoral level)
Participant: Michael Jahn
Person Months Worked: 2.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Ming Shen
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Mayank Garg
Person Months Worked: 4.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Tung Thai
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

RPPR Final Report
as of 14-Jul-2023

Participant Type: Graduate Student (research assistant)
Participant: Neeraj Kalani
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Utkarsh Soni
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Mudit Verma
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Sriram Gopalakrishnan
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Sarath Sreedharan
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

RPPR Final Report
as of 14-Jul-2023

Partners

,

I certify that the information in the report is complete and accurate:
Signature: Matthias Scheutz
Signature Date: 7/11/23 5:21AM

SAIL-ON Final Report

ACT-NOW:

Autonomous Cognitive Technologies for Novelty in Open Worlds

Matthias Scheutz, Michael Hughes, Liping Liu, Jivko Sinapov
Tufts University

Chitta Baral and Subbarao Kambhampati
Arizona State University

July 11, 2023

Abstract

The goal of the project was to develop novelty-aware, novelty-handling agents that would be able to deal with a great variety of novelties in as much of a task- and domain-independent fashion as possible. For this purpose, we started with our cognitive robotic DIARC architecture as a framework for all algorithm development and code integration in the three TA1 domains: Polycraft, Monopoly, and NLP (the last domain was eventually abandoned when the TA1 team dropped out).

The main scientific innovations include a formal framework for analyzing the concept of novelty and its different forms; the combination of knowledge-based models with statistical learning models for detecting deviations from predicted outcomes: novelty detection via model prediction error; the development of novel problem solving strategies for how to handle model prediction errors (e.g., prediction failures of action outcomes) and model repairs; the synergistic combination of high-level planning with low-level reinforcement learning to quickly learn how to accommodate novelties ("RapidLearn"); a modification to the MCTS algorithms used for determining best responses after novelty introduction with accommodated novelties in large state-action spaces in multi-player stochastic environments; and finally novel neural network models for vision and agent behavior models to determine deviations from learned behaviors, and novel methods for novelty detection and accommodation in NLP domains.

We achieved top performance in Polycraft and NLP, and second best performance in Monopoly, either meeting or coming very close to all expected benchmarks, and exceeding them in several measures. We were able to demonstrate the operation of the integrated architecture in additional domains including the CARLA driving simulation as well as a physical assembly task using the Fetch robot. Throughout the project we disseminate our research in top national and international venues and we still have several project-related papers under review or in preparation. The integrated architecture has already been transitioned to Thinking Robots Inc. which is already using it in its social robot product. We already have made several datasets and tools like the NovelGridWorlds simulation available through github and plan to release the improved integrated DIARC architecture at a later point this year as well.

Objectives

The ACT-NOW project was a joint effort of the Tufts and ASU teams who collaborated on all aspects of the research and developments of the main project objective: **to develop an integrated architecture for novelty-aware and novelty-handling agents.**

We utilized the initially selected three TA1 domains: Polycraft (an action-perception domain), Monopoly (an action domain), and NLP (a perception domain) for all algorithms and technology developments in the four project years as follows:

Year 1: The goal was to start with the DIARC cognitive robotic architecture shown below as an architectural framework for all agent developments (except for the NLP domain which did not find the architectural template), and initially develop the first novelty detection and accommodation methods in Polycraft for object novelties.

Year 2: The goal was to add the Monopoly and later NLP domains and develop additional novelty methods for detecting relations, agents, actions, and interaction novelties. In particular, the goal was to combine knowledge-based with statistical methods, including learned statistical models for both perception and action.

Year 3: The goal was to develop additional methods for detecting, characterizing, and accommodating the remaining novelties (goal, environment, and events), including a comprehensive task-general method for addressing different types of expectations failures that are caused by novelties.

Year 4: The goal was to improve the performance of our agents and to characterize the novelties in as much detail as possible symbolically to allow for human understandable natural language descriptions of the characterization.

For all algorithm and code development the guiding aim was to be as task-general as possible even though we used specific tasks and environments during the project. We also aimed to demonstrate the integrated architecture on other domains at the end of the project, including simulated and physical robotics domains.

We accomplished all objectives and exceeded them in several areas.

Summary of Accomplishments

The main project outcome includes several scientific accomplishments, technical breakthroughs, as well as technical transitions and software and dataset contributions.

The scientific accomplishments include a formal framework for analyzing the concept of novelty and its different forms; the combination of top-down knowledge-based models with bottom-up statistical models for detecting deviations from predicted outcomes: novelty detection via model prediction error and deviation from known entities, properties, etc.; the extension of a cognitive natural language-enabled robotic DIARC architecture to handle all types of novelties in stochastic multi-agent environments; the development of novel problem solving strategies for how to handle model prediction errors (e.g., prediction failures of action outcomes) and model repairs; the synergistic combination of high-level planning with low-level reinforcement learning to quickly learn how to accommodate novelties ("RapidLearn"); a modification to the MCTS algorithms used for determining best responses after novelty introduction with accommodated novelties in large state-action spaces; novel neural network models for vision and agent behavior models to determine deviations from learned behaviors, and novel methods for novelty detection and accommodation in NLP domains.

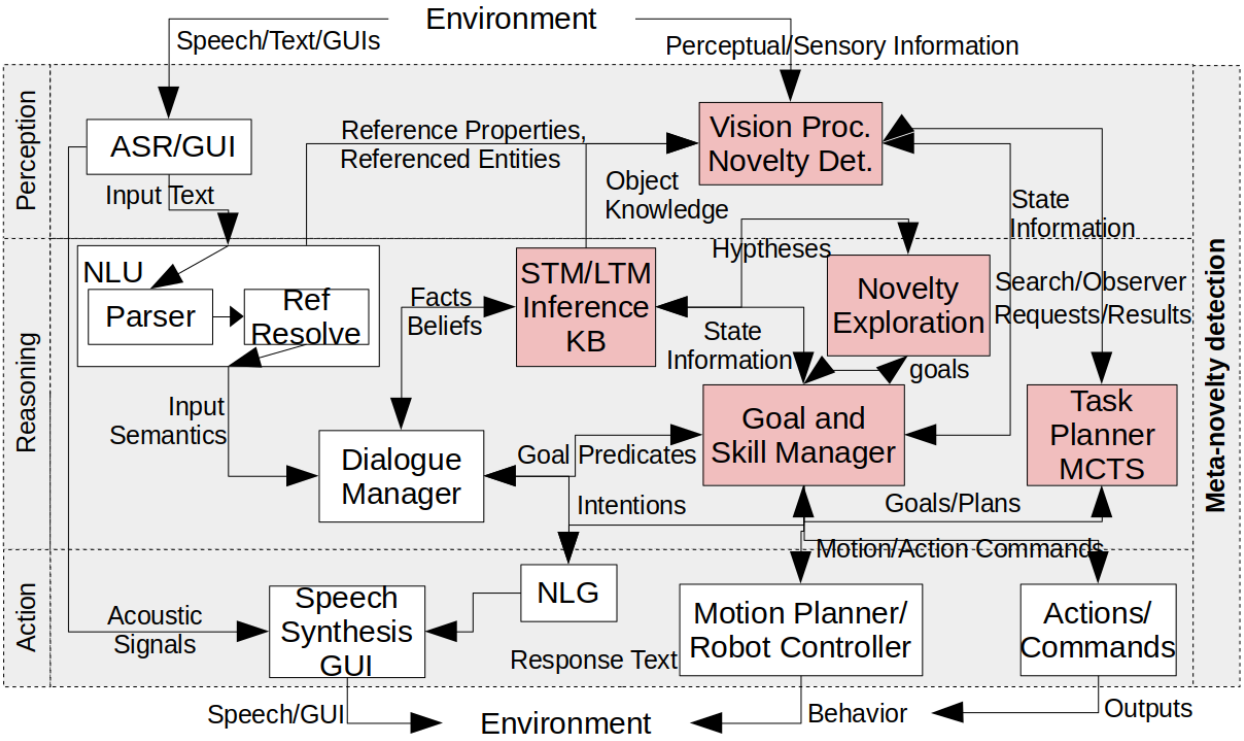
Technical breakthroughs includes significant speedup in life-long open-world learning using sequences of novelties where state-of-the-art reinforcement learning algorithms or life-long learning algorithms cannot even solve the first novelty; integration of SAIL-ON algorithms and methods into the cognitive robotic DIARC architecture which makes them immediately applicable to real-world autonomous systems; the demonstration of the performance of the integrated architecture in the Polycraft and Monopoly domains (which was among the highest in both domains); and an additional demonstration of the architecture in simulated environments (e.g., a shopping task in a Supermarket OpenAI Gym simulation and driving on country roads in the Carla simulator), and on robots (e.g., action discovery and learning on a Baxter robot and task-based human-robot interaction on a Fetch robot).

The technical transitions include the integrated architecture which was transitioned to the startup company Thinking Robots Inc. as well as the software which will be used on deployed service robots to improve robustness to unexpected changes during task performance.

Finally, software and dataset contributions include the release of various data sets (e.g., the NovelCraft dataset published in Transaction on Machine Learning Research) and open-source software packages published on github (e.g., the Novelgridworlds OpenAI Gym for novelty detection, characterization, and accommodation work, or datasets for"GAN ensemble for anomaly detection"; additional releases, in particular of the integrated DIARC architecture with all SAIL-ON integrations are currently in progress, expected for Fall 2023).

Detailed Accomplishments

We started with the DIARC cognitive robotic architecture shown below as an architectural framework for all agent developments, except for the NLP domain which did not find the architectural template, and developed various algorithms and components for the architecture to detect, characterize, and accommodate novelties.



The red components in the above diagram represent components that were either extended and augmented by new algorithms (e.g., the Vision Processing component, or the Goal and Skill Manager), or develop from scratch (e.g., the novelty exploration component or the MCTS component for multi-player novelty accommodation). Among the TA1 evaluation domains, Polycraft was unique in that it presented a combined domain in which to do perceptual novelty detection as part of performing an action-oriented task. Our novelty detection paradigm was built on the idea that both top-down knowledge-based inferences as well as a bottom-up learned statistical models are needed to detect the various types of novelty categories selected as part of the program, the former to detect state differences from rule-based expectations while the latter detects distributional differences. Novelty characterization builds on novelty detection to determine the type of novelty based on what was detected and how (e.g., a distributional shift of a set of known actions is different from encountering an unknown action). Note that novelty characterization might require exploration to determine the properties of the novelty. Finally, novelty accommodation uses novelty characterization to augment the agent's knowledge base and to update machine learning models.

In the following, we will detail the various research and development efforts performed by the team in seven sections:

1. Foundational work on novelty concepts
2. Architectural developments for novelty handling
3. Visual novelty detection
4. Agent novelty detection and accommodation
5. Inference-based novelty detection
6. Hybrid planning reinforcement learning for novelty accommodation
7. Multi-player novelty detection and accommodation
8. Evaluations
9. Publications

In addition, we put together a comprehensive video for the DARPA Open House/Family Day that showcased not only our agents detecting and handling novelties in the Polycraft/Minecraft environment, but also additional demonstrations of novelty detection in the simulated CARLA driving domain as well as the detection of expectation violations and automatic recovery in a robotic assembly task on a physica robot.

1. Foundational work on novelty concepts

Performing tasks in open worlds requires an agent to be ready for the unknown, for by definition what makes a world open is that not all ontological types or entities (widely construed to include objects, actions, events, goals, rules, etc. and their properties, properties of properties, etc.) are known in advance to the agent. For example, an agent might be confronted with an unknown change in the environment which is task-relevant if not necessary for completing the task and the agent must cope with it. Coping might take different forms, from ignoring it (either due to the failure of detecting it or because the agent does not know what to do with it), to accommodating it by working "with" or "around" it (e.g., finding alternative plans, treating it as if it were something known that the agent knows how to handle, or something that won't happen again etc.), to attempting to characterize it and, based on the characterization, accommodate it (e.g., through learning more about the entity and how it affects the agent's task performance, utilizing its properties for better task performance, understanding the counterfactual effects for future situations, etc.). The extent to which an agent is ready for open worlds, i.e., can accomplish its goals in an open world setting (given a set of initial states), we will call "open-world readiness" and is something we want to quantify. In the course of doing so, we need to distinguish different agent abilities to detect, characterize, and accommodate novelties in the open world. Some agents might be able to accommodate novelties by learning, others simply by selecting different existing behaviors, it all depends on their goals and tasks.

Open-world learning is also not required for novelty detection, characterization, and accommodation. An agent might detect a novelty, characterize it relative to its goals and decide that it can be ignored and does not need to be remembered, which is a form of accommodation that does not require learning (the agent might go through the same reasoning over and over again if it repeatedly encounters the same type of novelty). Open-world learning thus is orthogonal to novel detection, characterization, and accommodation and can occur irrespective of novelty-awareness (e.g., through simple reinforcement, Hebbian learning, etc.). We do, however, claim that open-world learning based on novelty detection, characterization, and accommodation can be beneficial to agents if done right (it can also be detrimental to the agent's goals if it ends up chasing down novelties and trying to characterize them, even though they have no relevance for its tasks). For example, while novelty detection and characterization might produce overhead initially (compared to other ways of coping with novelties), it might provide longer-term benefits and thus putting in the effort to characterize novelties that seem task-relevant and to store the characterization might pay off later when that knowledge can lead to better accommodation (e.g., discovering and remembering how a screwdriver can be used to open a can of paint could be helpful for using a pry bar). We start by analyzing the concept of novelty: what it is and what it is not; what different

types there are; how it can be detected, characterized, and accommodated; and how it could be evaluated in artificial agents.

1.1 Concepts of Novelty

Novelty can arise in many different ways, from novel objects with novel properties, to novel relations and contexts, to novel actions and events, to novel goals and tasks, to novel constraints and rules, and so forth, even in closed worlds. For example, suppose a closed-world agent encounters an unforeseen fault during task performance (such as data corruptions, sensor failure, insufficient compute power, etc.) which causes a change in its performance. The agent then may or may not be able to detect the change, and even when it detects the change, it might not be able to characterize the fault and find a way to address it. Similarly, in open worlds, agents ought to handle novelty when it presents opportunities for improving performance, especially when it enables goal accomplishment (which would not have been possible without utilizing the novelty). We thus need a definition of “novelty” that can distinguish novelty from other related phenomena (such as surprising non-novel states of affairs).

Novelty is not a property of entities. Entities in the world do not have labels that say “novel” on them. Rather, novelty is always relative to an agent’s cognitive system and past experiences, which is essential for how we approach novelty in the design of our algorithms and agent architectures. Novelty thus is an intrinsically relational concept: something is “novel” with respect to some reference frame describing or reference system instantiating entities (e.g., a set of images that does not contain the image of a particular object or type of object); it is a relation between an agent and an environment; more precisely, a relation between the agent’s belief state and a representation the agent cannot accommodate with its past memories or inferences from those memories.

The reference system is typically realized in an epistemic agent, i.e., as the agent’s internal representations, stored or inferred, of certain entity instances and types: something is “novel” for the agent if the agent does not have an internal representation of it and cannot generate a representation for it (e.g., by way of making inferences, or through other types of internal processes generating combinations of representations). I.e., when an agent has previously encountered an entity, being able to recall the encounter makes a subsequent encounter not novel. However, if the agent does not remember encountering the entity previously and can also not identify the entity or its type through inference, the encountered entity will be novel for the agent (and it will continue to be novel for as long as the agent does not form memories or acquire the requisite knowledge about the entity, regardless of whether an external observer watching the agent’s repeated encounters with the entity expects the agents to know the entity after the first encounter).

For agents that do not form (long-term) memories, every encounter will be novel whether the agent is aware of it or not, while for others (that remember every past experience and can make inference based on those memories, say) nothing will be novel because they can already represent all possible encounters (e.g., an obstacle-avoiding robot that for each obstacle configuration in its perceptual occupancy grid has a policy for how to avoid it). For many entities it does not even make sense to ask whether something is novel for them (e.g., a lamp), since they do not have perceptions or internal processes that generate representations of entities, while for others it might not be interesting to ask that question because encountering novelty does not cause any changes in the agent's behavior (e.g., the agent might remember the novel encounter, but without using it to adapt its internal processes). Yet other agents are able to detect novelty and adjust their internal processes based on it (e.g., as many machine learning algorithms do automatically without explicitly noticing novelty).

Since novelty is in the mind of the beholder, i.e., agents observing or contemplating an entity, action, process, etc., we need to keep in mind whose novelty we are evaluating when we design artificial agents that can detect, characterize, and accommodate novelty: the agent's or ours? Typical machine learning tasks require our agents to detect what we – the agent designers and evaluators – consider "novel in the data", i.e., novel items in evaluation data relative to a set of given items in training data that does not contain these items. Our reasoning is that any agent familiar with the training set ought to be able to determine that the evaluation data contains novel items. Whether this is actually so, however, will depend on a variety of factors, not the least the agent's discrimination capabilities.

We introduce the concept of "original" vs. "derived" novelty to elucidate an important distinction regarding the agent's cognitive capabilities: for some agents, entities they have not per se experienced will not be novel as these agents can construct patterns representing or denoting such entities (e.g., by combining patterns resulting from other previously experienced entities), while others not capable of such (hypothetical) constructions will always find unexperienced entities novel. We will thus count an item as only novel for an agent when the agent has not experienced the item before (i.e., cannot recall it) and cannot actually derive it from its knowledge base, which leaves us with four cases:

- the agent has experienced it before and remembered it, hence it is not novel
- the agent derived it without having experienced it, so if and when it experiences it, it will not be novel (note that this does not exclude the possibility that the experience of the novelty will have other aspects the agent did not foresee that would be novel)

- the agent could in principle derive it based on its knowledge and derivation processes if it did it for long enough (but practically not derivable), hence practically the item is novel
- the agent could not derive it given its cognitive capabilities, hence it is novel

When we intend to evaluate whether an agent detected novelties and accommodate them in a task, it is possible and in some cases will be extremely likely, that the agent detected novelties that we were not aware of or novelties that we did not consider novel because we did not understand what the agent did and did not know. What can appear to be accidentally novel or not novel at all to us, might be genuinely novel to an agent which raises the question of what it means to evaluate an agent-relative concept like novelty.

"Novelty" is different from other related concepts that are often conflated with novelty:

- "Surprise": is based on forming expectations about states, entities, properties, etc. based on past experience; being surprised by the outcome is one way for an agent to detect novelty which ought to be maximally surprising (as it will have had zero probability based on the agent's internalized distribution); however, novelty may or may not be surprising to an agent
- "Outlier" or "distributional change": whether a distributional change really involves a novelty will depend on whether the range of the distribution is novel (e.g., "we didn't know we could have negative values"), whether the shape of the distribution is novel (e.g., "we didn't know we could have zig-zag probability density functions"), or whether there are novel elements in the distribution (which is different from the above "negative values" case where we know what those values are, just not that the particular distribution could range over them, and different from the shape of the distribution above where we also knew the values but just not how they could be distributed). So even in the novel distribution case the novelty does not lie in the change of the distribution per se (because I can learn about a change of the distribution in a biased die by observing it long enough; but even without learning that particular change I know that there can be biases and what possible distributional shapes they can take for die rolls, so while I might not expect the different distribution with that particular die, I am not caught by surprise that it could exhibit the one it does precisely because of my knowledge about die rolls and biased dice). Rather, a distributional novelty would have to be an unknown mathematical form which the agent has not experienced before and could not have derived from its knowledge.
- "Anomaly": is the mismatch between an expectation/prediction (based on a law, a regularity, a distribution, etc.) and an observation that does not match the expectation/prediction; the difference between "noise" and "anomaly" then is that

anomalies are systematic, repeatable, and thus indicative of lack of knowledge in the law, regularity, or distribution used to predict future behavior whereas noise may indicate a one-off outcome; anomalies may or may not be novel depending on what the anomalous outcome is and how it relates to the underlying law, regularity, distribution, etc.; note that an agent witnessing an anomaly once might encounter novelty (e.g., a novel state) but when the anomaly occurs again it is no longer novel, while still being an anomaly; critically, if the agent does not update its model for making predictions under which the specific outcome was anomalous, it will continue to be anomalous, even though it will not be novel.

- Novelty and similarity: a red apple that differs slightly in terms of color from another red apple is still an apple, and still a red apple, because it is similar enough (based on some notion of similarity); hence, an agent that has seen the first red apple will not have seen a novel property instantiated in the second apple , even though the second apple's redness differs slightly from the first one unless one includes "trivial novelty" which counts any quantitative deviation in redness as different (note that it will have a seen a novel apple instance though, but different instances of the same type are usually not a concern)

Overall, it's essential to distinguish the concept of novelty from the process of how novelty can be detected by an agent (e.g., whether and how "surprise", "anomaly", and "similarity" can be used to detect novelty).

1.2 Novelty detection and characterization

While the concept of "novelty" is different from those of "surprise", "anomaly", "prediction", "distributional outlier", "expectation failure", and other related concepts, many of the related concepts (some which implicitly refer to architectural mechanisms) can be used to detect novelty.

Forming expectations based on past experience and being surprised by the outcome, for example, is one way for an agent to detect novelty which ought to be maximally surprising (as it will have had zero probability based on the agent's internalized distribution). However, surprise does not entail detecting novelty: surprise requires the formation of expectations, but just noticing that an expected outcome did not happen is not sufficient for detecting novelty (e.g., if the agent cannot characterize or classify the actual outcome, if it only knows that it does not know it, or if it cannot determine what about the outcome is different, it could only hypothesize that something novel might have happened). For example, if an agent cannot distinguish between two types of objects A and B given a set of objects but notices that other agents seem to have consistently clear preferences when they deal with these objects (e.g., in terms of value of the object), it might infer that there must be something about the objects it is not perceiving or cannot discriminate (e.g., think of the subtle difference in patterns

between poisonous and edible mushrooms). Even without forming expectations, something to be judged as being out-of-distribution can indicate novelty (although if the distribution is continuous, then it is unclear whether out-of-distribution item is truly not in the distribution or the distribution is not represented well enough to include the item).

Being told about something new or being in a situation where something novel is expected is another way to learn about novelty (this does not require surprise, but it still requires the ability to compare to stored memories). Generating an internal combination of symbols in a way that has not happened before and that is meaningful w.r.t. to types of entities is another way to discover novelty without surprise. While some methods such as forming expectations based on past distributional knowledge is a good way for detecting pattern-based novelties, other types of novelties such as semantic or conceptual novelties cannot be simply found based on pattern-based similarities. There is also the case of meta-novelties, i.e., novelties arising due to presence or absence of novelties at some object level (e.g., the expectation to see new pictures at an exhibit when only old known ones are displayed).

Based on the novelty hierarchy, we have some qualitative measures of difficulty of different types of novelty, e.g., relational novelty involves entities in which the relation is grounded, hence recognizing relational novelty at the very least requires the recognition of the involved entities (I cannot determine the relation between two entities if I cannot recognize them as entities; I might be able to determine the relation without being able to determine the types of the entities, but at the very least I need to be able to see them as entities that can be related to each other). For example, consider "symmetry" and "asymmetry" as properties of relations: It seems that an agent could not discover/detect the properties of "symmetry or asymmetry" of a relation without being able to detect the particular relation (e.g., "next to" or "taller than"), which, in turn, would require it to detect entities that could be in the "next to" or "taller than" relations. Note that there is a complex interplay based on the agent's employed ontology and formalism for representing it that will potentially trade off different ontological categories.

When novelty is detected, an agent might opt to characterize it, i.e., determine what it is in order to be able to utilize it and possibly better accommodate it (now or in the future). However, what ontological types are detected or are relevant for an agent will essentially depend on the agent's goals, and hence the question arises what we mean by novelty characterization when it is again a agent-relative and not absolute notion, one that depends on the agent's ontology, representational formalism, goals, and possibly other aspects. For example, consider the tree in Polycraft that only yields one log (instead of two). From the agent pursuing the Pogostick goal it might be characterized as a "poor" tree because it only yields one log, whereas from the clean-up lumberjack agent's perspective it might be characterized as a "light" tree because only one log needs to be cleaned up. Or it could be considered a different kind of object, a shrub, say, which looks identical to a tree. In either case, the new object must have a

property, potentially unobservable to the agent, that distinguishes it from trees according to Leibniz' principle of the identity of indiscernibles: "x is identical to y iff for every property F, object x has F if and only if object y has F" (for otherwise it would be the same object). Note that as a result of either internal difference (i.e., novelty characterization relative to the agent's goals) or external differences (i.e., novelty characterization to difference in hypothesized types) we get two conceptually different novelty accounts. Moreover, if the tree yielding only one treelog is perceptually indistinguishable from other trees yields it could be interpreted as either "tree novelty" – in this case the tree has a hidden, unobservable property that makes is only produce one log – or an "chop action novelty". I.e., the above can also be construed as a different action outcome of the "chop" action (e.g., and thus as the "chop" affordance of the tree): what distinguishes two perceptually indiscernible trees is that one yields one tree log as action outcome of "chop" while the other yields two (the two trees are identical in all other aspects). Conversely, the "chop" action could be viewed as non-deterministic on the same tree, sometimes yielding one, other times yielding two tree logs (e.g., depending on how hard the tree is chopped; maybe the agent does not have enough control of how it chops trees and small deviations in low-level behavior, possibly unobservable to the agent, have an effect on the outcome). The action outcome then also points to the construal of differences in "events": "one treelog appearance" event vs. the "two treelog appearance" event. Similarly, the chop action can be viewed as an interaction with the tree rather than applying an action to an object, hence "interactions", too, can be traded off with object affordances, events, object properties, etc. Even objects themselves could be viewed as "action outcomes" of "look" actions in context (if I look over there from here I see a tree, i.e., the tree appears because I looked there from here). Hence, the look action can have different non-deterministic outcomes.

All of this points to the difficulty of assessing and evaluating novelty characterizations because even within the same ontological and representational frameworks the same novelty might receive different characterizations by different agents. Since novelty characterization depend on the agent's tasks and goals, the agent's ontology, and the agent's representational repertoire, it is, like novelty itself, as an agent-relative notion not directly measurable, unless a particular representational format is explicitly imposed on the agent and the agent has an explicit goal (as part of its task) to generate novelty characterizations in the specified format. This is important because external evaluations of novelty characterizations without requiring a shared vocabulary would have otherwise to resort to behavioral measures and those will likely have to measure "novelty accommodation" and thus conflate "novelty characterization" with "novelty accommodation". If characterization accuracy, or more generally, success is to be evaluated, it is thus critical to make it made part of the agent's task and to require the agent (again as part of the task) to produce the proper representational

description in the prescribed ontology, as otherwise there is no incentive for the agent to characterize novelty, let alone in the format used for evaluation (for an agent might well be able to detect novelties and accommodate them, even in cases where proper accommodation is necessary for task accomplishment, but without ever characterizing novelties, or characterizing them in any way understandable for human observers). Also note that novelty characterization may or may not involve learning (e.g., the agent may be able to describe what is novel about a novel item without storing that description and using it for its operation).

We need to distinguish “novelty accommodation” from “novelty characterization” in that accommodation, in the simplest case, does not require “characterization”, just to deal with novelty in some way (without necessarily attempting or even being able to characterize it). While “characterization” requires the agent to somehow “understand” novelty in terms of its conceptual system (i.e., accommodate it at the conceptual level), “novelty accommodation” in the sense used here only means to be able to deal with it behaviorally, i.e., work with or around it for the purposes of the agent’s task (of course, one way to accommodate it is precisely to characterize it).

Detecting novelty does not automatically “accommodate it”, the same way it does not automatically lead to characterization (e.g., noticing an outlier to a distribution or an out-of-distribution element does not update the distribution which might be difficult or itself unknown for certain types of distributions). Whether novelty is, will, or ought to be accommodated will depend on the agent’s goals (a driving agent with the goal to get to a location as quickly as possible might notice traffic cones on the street as novel objects, but might end up treating them simply as obstacles that it needs to avoid without generating new representations for them or attempting to determine their properties, functions, etc.). Novelty accommodation, thus, is, just as novelty detection dependent on the agent’s goals, knowledge, etc. depending on the task, the same agent might characterize and accommodate the same “novel experience” differently. Note also that accommodation can happen with and without learning (e.g., avoiding novel obstacles does not necessarily require the agent to learn new behaviors).

1.3 Core requirements for accounts of novelty

Successful accounts of novelty must, at the very least do the following:

- define novelty as a relation between an agent and an entity: Novelty is not a property of entities (concrete or abstract), but rather an epistemic relation between an agent and an entity to whom the entity is or is not novel. The theory needs to define what it means to be novel for agents by making recourse to their architecture and the functional capabilities enabled by the architecture (e.g., whether there are memories of past experiences that can be retrieved, whether there are inference processes that can generate patterns, etc.).

- not solely focus on patterns: Whether an agent has seen a particular pattern or whether it can classify the pattern as belonging to a particular pattern type is often not relevant for semantic or conceptual novelties which do not depend on the particulars of patterns so much as they depend on finding the right interpretation in the right domain (mapping from patterns to concepts).
- define novelty at different levels of abstraction: Especially with novelty that is not pattern-based (e.g., conceptual novelty that might be expressed in familiar patterns), it is important to determine the conceptual spaces in which novelties are detected, where the patterns are a mere means for communication, but not the domain themselves (e.g., just take any formal theory with countably many formulas over an uncountable domain).

1.4 Evaluating open-world readiness

Since novelty and thus whether it is being detected by an agent intrinsically depends on the agent, the question arises how we – as outside observers of the agent – can tell whether the agent has detected novelty and whether it accommodated it? This is the same question that arises with our fellow humans because, unlike artificial agents where in some cases "looking inside" might be possible – it is definitely not possible to look inside people's heads (certainly not at present). But if we cannot look inside an agent, there is no direct way to assess whether it detected novelty, nor whether it "accommodated" novelty in any meaningful sense of the term; i.e., an agent might encounter something that "ought" to be novel for it (because the agent has not encountered that entity, an object say, before), but unless this encounter leads to behavior that counterfactually would have been different, there is no way for us to tell whether it detected it or whether it only had the "opportunity" to detect it; the same goes for accommodation. Of course, we could ask the agent if it can handle questions, but that presumes that the agent forms a memory of the encounter and that that memory is accessible to introspection in a way that it can be expressed in a human-understandable form. As designers of novelty detection and accommodation evaluation tasks, we need to be careful both in the way we introduce novelties and with the kinds of novelties we introduce, to allow us to at least determine when an agent has the "opportunity" to detect novelty; this is the best we can hope for, short of the agent itself allowing for a conversation about its experiences after the task. An "opportunity to detect novelty" arises in the case of perceivable objects and actions, say, when the objects or actions are in the agent's perceptual range and when the agent either has the perceptual capability to discriminate them directly or indirectly via context-based inference; hence, agents that have no way of detecting a particular novelty cannot be blamed for not discovering it – nothing is ever novel for a lamp. Similarly, whether an agent truly accommodated novelty and not just accidentally acted in a way that seems to suggest it did, can only be determined indirectly via the agent's behavior in a task

where accommodating the novelty is the only way for the agent to succeed; i.e., a task needs to be designed with a modification in a way that the agent (who has learned to solve the task without the modification, say), has to detect and accommodate the novelty in order to solve it.

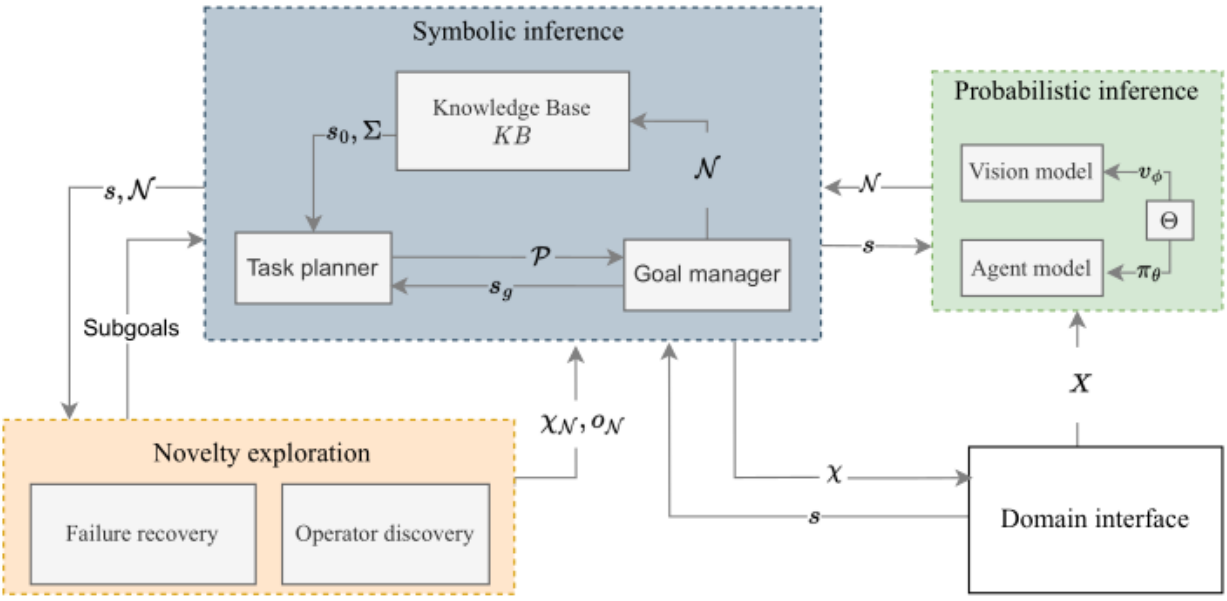
We should thus design evaluations with the above constraints in mind: first, make modifications to the task settings that require the agents to detect novelties (i.e., aspects that we know the agent was not exposed to because they were not present in the training/practice setting the agent used to learn the task) and then record novelty detection opportunities (to the extent that we can) and determine how far in the task the agent got. Note that we should not use action costs or "shortest solution" as performance measures for novelty detection and accommodation, for novelty detection will often mean to become suboptimal, to explore and learn about the novelty without pursuing the task goals. It
is, of course, possible to compare the agent's performance to best possible performance assuming an agent (incidentally) acted optimally (e.g., because it was able to consult an oracle about what to detect and what to do with it); however, such baselines are themselves meaningless because such agents are unrealistic, but they can serve as a yardstick to compare the performance of different agents to each other (especially when those agents failed at the task, which might be likely depending on the complexity of the novelties they have to detect and accommodate).

It is also important to keep in mind that, precisely because novelty is an agent-centric property, what is novel for one agent might not be novel for another agent. This also applies when we compare human agents (e.g., the designers of artificial agents evaluating an artificial agent's ability to handle novelty) to artificial agents. When we, as agent designers, make statements about something being novel (without qualifying for whom) there are three interpretations: (1) we believe it is novel for the artificial agent under consideration based on what we know about the agent and its operation; (2) we believe it's novel for any agent placed in particular settings (based on the properties of the setting which is such that there would have been no opportunity for any agent to experience or derive the novelty); or (3) the item is novel from our perspective (and if our perspective is the yardstick as agent designers it it mutatis mutandum novel for the artificial agent). It is important to not conflate these three notions. (1) can typically be assessed and potentially be established if the operation of the agent is clear enough (which might not be the case with self-learning and self-improving systems such as deep neural networks where internal representations cannot be clearly isolated and assessed; in that case, it might be necessary to develop externalized evaluations such as forcing the agent to come up with a known or new label depending on whether it something is known or novel). (2) needs to be establish through formal arguments based on precisely characterizing what agents can potentially know and why the situation will generate something new (e.g., "diagonalizing out" of an

agent's knowledge can produce something that the agent provably cannot know). (3) is usually established through reflection on one's own knowledge and then simply stipulated about the artificial agent (and might, as a result, be false: a "novel object" in an image for the human designer might not be "novel" for the artificial agent given its internal representations of objects).

2. Architectural developments for novelty handling

We introduce the architectural framework with the essential components depicted below for developing novelty-aware agents:



Through their function and interactions, these components define the agent's knowledge repository *K*, its inference algorithms $\vdash$, and operate its sensors ζ and effectors Ξ. The agent has a DOMAIN INTERFACE component that mediates its interactions with the environment through the sensors ζ and effectors Ξ (this will typically be itself a set of components for different sensors and effectors but we are not concerned with those architectural details in this work). Sensory information is organized into symbolic state descriptions $s \in S$ and passed on to other components for processing, including recognized actions of other actors in the environment. Additionally, subsymbolic perceptions X (e.g. images) acquired by the sensors are also made available by the DOMAIN INTERFACE. The knowledge base $KB \in K$ stores facts using about the environment as extracted from state description obtained from the domain interface, as well as the operators available to the agent. The GOAL MANAGER and TASK PLANNER are the symbolic inference mechanisms of the architecture. The GOAL MANAGER detects novelty by comparing inferred expected state descriptions from rules and facts in KB to state descriptions obtained through the DOMAIN INTERFACE. The TASK PLANNER generates plans for the agent to execute in the environment. The GOAL MANAGER also detects novelty due to plan failure. Whenever novelty is detected, it is submitted to the KB for further inference and exploration.

The subsymbolic knowledge repository $\Theta \in K$, along with the VISION MODEL and AGENT MODEL components, make up the probabilistic inference algorithms of the agent. The VISION MODEL receives subsymbolic state descriptions (e.g., images) and detects out-of-distribution samples using statistical methods. The AGENT MODEL receives symbolic state descriptions and, with statistical methods, monitors the behavior of other actors in the environment to detect when their behavior is inconsistent with the agent's expectations. When either component detects novelty, it sends a description of the novelty to the KB. The NOVELTY EXPLORATION component is comprised of symbolic novelty exploration algorithms, as well as a reinforcement learner that can learn policies to form new executors if necessary. It accepts a description of the novelty and symbolic state from the KB. It returns sub-goals to the GOAL MANAGER that aim to explore the environment or specifications of operators that can be used to aid in solving task T . The operation of these components is described in detail in subsequent sections.

2.1 Symbolic inference

Symbolic inference is performed by the GOAL MANAGER and TASK PLANNER on facts and rules from the KB to solve task T and to detect and characterize novelties .

2.1.1 Knowledge Base

The knowledge base $KB \in K$ stores facts about symbolic state descriptions using the language . It is populated with facts and rules defined a priori and through interaction with the environment. This may include objects, actors, functions, symbolic state representations, and a semantic type hierarchy used to describe planning tasks. During execution, other components in the architecture may assert, retract or query facts in KB to detect novelty, create plans, or store new knowledge. The KB is part of the knowledge repository against which novelties are detected. As a result, when information about a detected and explored novelty is asserted into KB, subsequent encounters with it will no longer be novel for the agent. This is not limited to novelties detected using symbolic inference, as novelties detected with probabilistic inference are also eventually expressed in and stored in KB.

2.1.2. Task Planner

The TASK PLANNER is one of the two symbolic inference mechanisms utilizing the KB. Once a desired goal state description sg is received from the GOAL MANAGER, the TASK PLANNER retrieves a planning domain definition Σ (operators, predicates, etc.) and an initial state description $s0$ from the KB to form a planning task T and searches for a plan $P \triangleright T$. Whether a plan is found or not, along with the plan itself, is then submitted to the GOAL MANAGER for further inference and execution.

2.1.3. Goal Manager

The GOAL MANAGER is responsible for managing the agent's top-level goal(s), which can consist of multiple concurrent goals, and is responsible for detecting novelties in symbolic state descriptions. The component submits goals to the TASK PLANNER and performs inference on state descriptions, a process during which novelties of different types may arise due to planning failure, execution failure, or unexpected aspects of state descriptions. Once a novelty is detected, the GOAL MANAGER produces a description of it and asserts it to the KB. Depending on the novelty type, the NOVELTY EXPLORATION component (see Section 2.3) may be invoked. The GOAL M ANAGER is also the primary component communicating with the DOMAIN INTERFACE.

Prohibitive novelties that prevent task completion. Prohibitive novelties are most often encountered due to the failure of the TASK PLANNER. For a given goal, the GOAL MANAGER consults the TASK PLANNER for a plan to execute. If the TASK PLANNER fails to find a plan for a particular goal, assuming that the goal is achievable based on the agent's knowledge, then the failure may be due to an prohibitive novelty.

Obstructive novelties that make task accomplishment harder. Obstructive novelty is most likely encountered due to plan execution failure. Once a plan sequence has been generated, the GOAL MANAGER enters an execution phase where it verifies that each operator's preconditions are met. In this inference task, the D OMAIN INTERFACE is consulted to verify that the preconditions ψo of an operator $o \in O$ hold in the current symbolic description of the environment state. If inference on prior states indicates that all of the operator's preconditions ψo should be met, but they are not in the current state, this constitutes a novelty. If all preconditions are met, the executor χo of the operator is sent to and executed by the DOMAIN INTERFACE component. The agent may receive feedback through the DOMAIN INTERFACE component indicating the success or failure of the executor. In case of failure when success was anticipated, this again constitutes novelty. Additionally, obstructive novelty may occur even when an executor succeeds, if its effects are not consistent with the agent's expectations.

Beneficial and nuisance novelties that either help or distract. Success in the task does not eliminate the possibility of novelty. The GOAL MANAGER may still encounter beneficial or nuisance novelties during its inference of the expected state of the world. If the inferred state does not match the observed state of the world solicited from the DOMAIN INTERFACE, then that still constitutes a novelty, regardless of its effect on the task. Otherwise, plan execution continues until all operators in the plan have been executed.

In all cases where novelty is detected, a symbolic description of it is asserted to KB and sent to the NOVELTY EXPLORATION component, which utilizes different exploration policies depending on the way novelty is encountered.

2.2. Probabilistic inference

The architecture's probabilistic inference components are used to detect novelty in the subsymbolic state representations X as well as in statistical relationships between elements of the state descriptions *S*. The knowledge repository $\Theta \in K$ stores the agent's knowledge of these modalities in the form of machine learning model parameters. The first component for probabilistic inference is VISION MODEL, which implements a visual novelty detector based on images from the DOMAIN INTERFACE. The second component of probabilistic inference is an AGENT MODEL that models the behavior of other actors in the world to detect deviations from their known behaviors If either module detects model deviations with sufficient confidence, a symbolic representation of the hypothesized novelty is submitted to the knowledge base for further inference and exploration.

2.2.1. Vision Model (see also Section 3)

The VISION MODEL is responsible for detecting visual novelty. At every step, the DOMAIN INTERFACE provides a 2-dimensional color image of the agent's current view of the world. The VISION MODEL's task is to determine whether each new input image represents a plausible view of the known standard environment, or a different "novel" distribution. Some authors refer to similar problems as anomaly detection or out-of-distribution detection (Ruff, Kauffmann, Vandermeulen, Montavon, Samek, Kloft, Dietterich and Müller, 2021). Visual novelties could include new object types, new agent appearances, or new backgrounds. Scene composition properties could also change (sizes, frequencies, relative locations, etc.). To solve the detection problem, our VISION MODEL takes a deep autoencoder approach, following Abati, Porrello, Calderara and Cucchiara (2019). Deep autoencoders are often applied to visual novelty detection for their ability to learn effective representations of data through self-supervision (Pang, Shen, Cao and Hengel, 2022). The model consists of two component neural networks: an encoder $e\phi$ with weights ϕ that maps image X to a code vector $z \in \mathbb{R}C$, and a decoder $d\phi'$ with weights ϕ' that maps code vector z back to an image X. After a model is suitably trained (see below), given a new image $\tilde{X}$, the model assesses possible novelty via reconstruction error: $r(\tilde{X}; \phi,\phi')=||\tilde{X}-d\phi'\ (e\phi\ (\tilde{X}))||2$, which measures Euclidean distance between the input and its reconstruction. Intuitively, a well-trained model should have low reconstruction error for images that represent the normal environment used for training, while images containing novelty will yield higher error. The effectiveness of this novelty detection method is determined

by the separability of the normal and novel reconstruction error distributions (Richter and Roy, 2017). The VISION MODEL can pass the reconstruction error signal directly to the Symbolic Inference. Naturally, we can also apply a threshold τ to produce binary detection decisions, where "novelty" is called if $r(\tilde{X}) > \tau$ and "normal" otherwise. The value of τ can be selected on a validation set containing labeled examples of normal and novel data to maximize a performance metric of interest.

The VISION MODEL can be trained in advance on a dataset of images depicting the "normal" environment. Given N training images, we seek encoder and decoder weights ϕ, ϕ' that minimize the total reconstruction error: $\sum n\ r(Xn;\phi,\phi)$. This can be solved via stochastic gradient descent (Abati et al., 2019), and the optimal weights ϕ, ϕ stored within Θ for later processing. If the VISION MODEL detects abnormal images, it generates a description that is sent to KB. The description includes an identifier that allows the KB to associate a visually-detected novelty with the state description of the environment state in which it was observed.

2.2.2. Agent Model (see also Section 4)

The AGENT MODEL is a crucial part of the architecture when operating in multi-agent environments. It uses probabilistic inference over symbolic state descriptions and is responsible for maintaining knowledge about other actors. The aim of the AGENT MODEL is to evaluate whether facts about other agents contained in symbolic state descriptions are consistent with its knowledge of those agents' behavior. One major factor we consider here is the types of other agents.

Behavioral modeling with behavioral cloning. The main approach of the AGENT MODEL is to model other agents' behaviors via behavioral cloning (Bain and Sammut, 1995; Daftry, Bagnell and Hebert, 2016; Argall, Chernova, Veloso and Browning, 2009) which is an offline reinforcement learning technique that learns policies from a dataset of actor trajectories. That dataset can be collected from repeated interactions with the environment and the policies are learned using supervised learning.

Suppose other actors Φ identified in the state descriptions have actor-types in a set Q, also provided in the state description. For each type $q \in Q$, the AGENT MODEL learns the policy $\pi(a|s, q)$ of an agent of type q. It is implemented with a neural network nnθ (s, q), which outputs the probability of actions in the action space Aq of the agent of type q. Here the input (s, q) is represented as a feature vector. If necessary, other representations such as graphs and text can also be consumed by neural networks. If complete knowledge of another actor's state is not known, then a pseudo-state s' can be inferred using knowledge in . Then the model becomes nn$\theta(s',q)$ and the action likelihood can be estimated. The model is trained via supervised learning using the true actions each actor took at that state (described in symbolic representations). The resulting model parameters θ are stored in Θ and used to

evaluate action likelihoods during execution. The exact architecture, hyper-parameters, training procedure and feature representation is also application dependent and is discussed in the experiment section.

Unlikely action detection. During operation, the AGENT MODEL uses the learned neural network to monitor an actor's behaviors and detect unlikely actions. Our agent may encounter actors that deviate from their actor-type's policy. Such an event is inconsistent with the agent's in the sense that the actors exhibit behavior that is unlikely under the actor policies encoded in Θ. Such novelty can be detected by evaluating the likelihood of observed actions using nnθ . Focusing on a single actor of type $q \in Q$ and its action $a \in Aq$ from a symbolic state description $s \in$, the likelihood under the learned model is computed by ℓ = nnθ (s, q). Then, ℓ is compared to previously encountered values to decide if a deviates from the known distribution. For instance, if nnθ is trained offline with a dataset of trajectories from the environment, then a portion of the dataset can be used for validation to set a threshold below which actions are considered inconsistent with the model for that actor type.

Actor type classification. Using the model nnθ, the AGENT MODEL can also classify actors into the known actor types Q. During operation, a voting scheme over the Q known actor types is used to classify actors into types using their action likelihoods, see the algorithm below:

Algorithm 1 Voting-based Actor Classification

Inputs:
$D = \{(s_i, a_i) | i = 1..M\}$ ▷ state and actions observations
nn_θ ▷ Learned actor policies
Q ▷ Set of known actor types
procedure:
$\mathbf{v} \leftarrow \mathbf{0}, \mathbf{v} \in \mathbb{N}_0^{|Q|}$ ▷ Vote vector
while not empty(D) **do**
    $(s, a) \leftarrow D.pop()$
    $b \leftarrow argmax_{q \in Q} [\text{nn}_\theta(s, q)]_a$ ▷ Vote for actor-type b
    $v_b \leftarrow v_b + 1$
end while
return $q \leftarrow argmax(\mathbf{v})$ ▷ Plurality Vote

The predicted types can be compared with the types extracted from the symbolic description to determine if some actors deviate from their actor types. If the the AGENT MODEL detects novelties due to unexpected actor behavior, it can provide feedback to the rest of the architecture for accommodation. It produces a symbolic description v that includes the state description in which novelty was detected, any unlikely actions detected for each actor in the environment and an indicator of the inferred agent-types

using the vote-based classifier, if they deviate from those expected. The novelty description is submitted to the KB and used for accommodation if necessary.

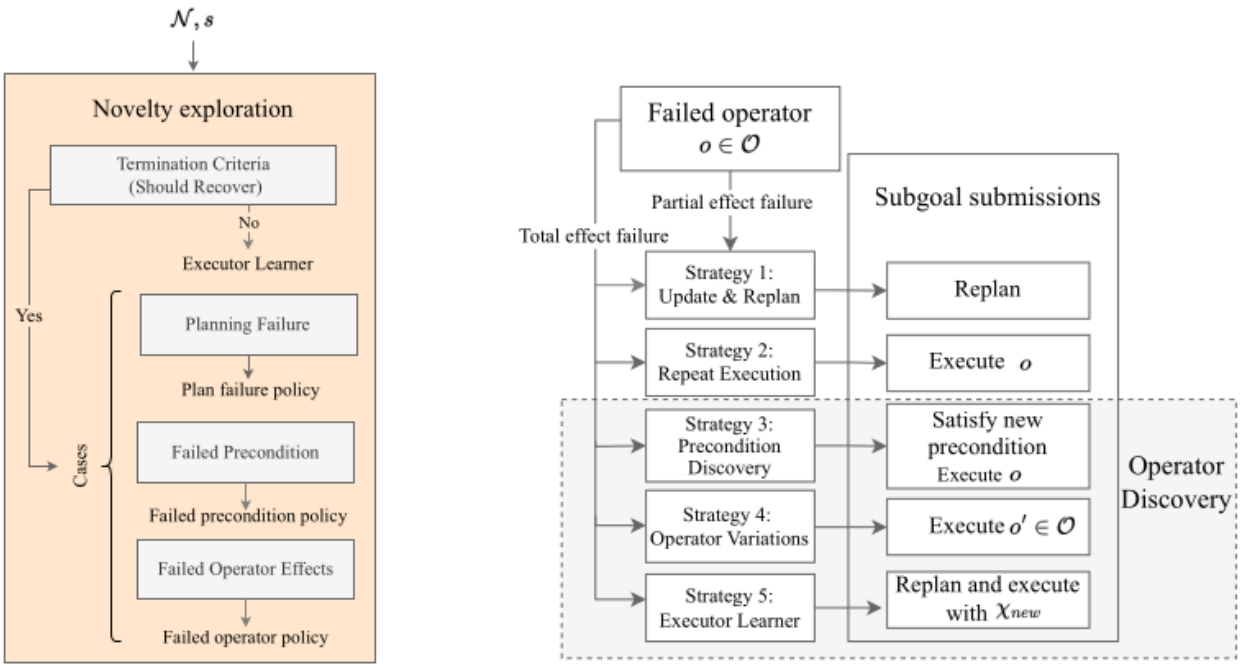
2.3. Novelty Exploration Component

The NOVELTY EXPLORATION component receives symbolic descriptions of novelties from the GOAL MANAGER and is responsible for generating exploration strategies depending on the type of novelty encountered. These include heuristic search strategies on the symbolic descriptions of states (and on failed operators) as well as knowledge-guided reinforcement learning-based exploration to learn new executors for existing failed operators.

Knowledge discovery. If the NOVELTY EXPLORATION component receives a novelty description before the agent attempts to generate a plan, then the type of the novelty (i.e. how it may impact its ability to solve the task) has not yet been determined. In that case, a knowledge discovery routine is utilized to gather information about the encountered novelty. The exploration strategy is dependent on the symbolic description of the novelty and can involve exploratory subgoals and operators related to the novelty. If additional information is gathered, it is appended to the symbolic novelty descriptions and, if necessary, used for additional exploration and recovery. For instance, if a novel actor is encountered, the NOVELTY EXPLORATION component would submit subgoals to the GOAL MANAGER that would involve interactions with the novel actor. These subgoals are generated using the agent's type hierarchy: known operators applicable to known actors are used on the novel actor. Such interactions may reveal additional information about the novelty and aid in further failure recovery depending on the novelty type.

4.3.1. Failure recovery

The FAILURE RECOVERY component is responsible for deploying recovery strategies for novelty accommodation and is invoked when novelty causes a planning or execution failure. Depending on the novelty description it receives, it employs various recovery strategies to address the particular failure. The left part of the figure below illustrates the agent's recovery policies for different failure cases attributed to novelty. Special attention is paid when novelty is attributed to the effect failure of an operator. The right part of the figure shows in detail the strategies employed when an operator's known effects are inconsistent with the state description after its execution.

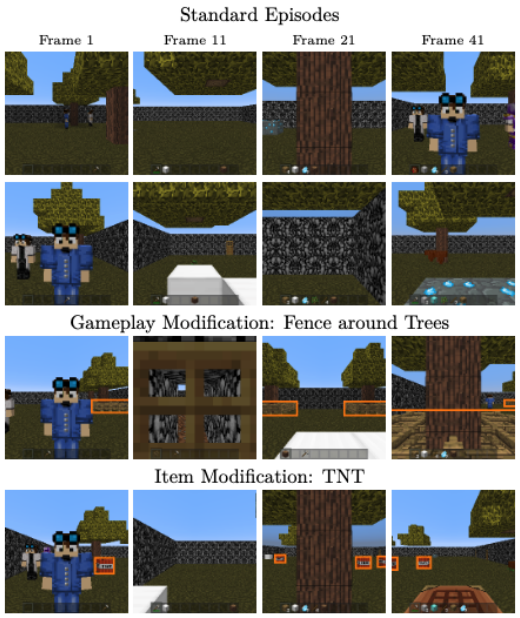


Depending on the severity of the failure, the agent employs different recovery strategies. If the state description is consistent with only a subset of the operator's effects (partial effect failure), then the KB is updated with the new effects, and the agent attempts to replan. If instead, none of the operator's effects are observed (total effect failure), the agent attempts to discover new operators that may assist it in solving the task. This process involves hypothesizing new preconditions, a heuristic search over known operators to uncover unknown effects, and a reinforcement-learning based exploration methodology that learns new executors for failed operators (see also Section 5). These strategies may be executed in the order presented, or may be applied to novelties at any order depending on the domain implementation and novelty descriptions. Overall the failure recovery policies are general and flexible, as each may be involved in recovery from multiple novelty types and they can be composed to recover from complex failures.

3. Visual novelty detection

3.1 New benchmark dataset: NovelCraft

We developed and published (Feeney et al. 2023) a new open-access dataset, which we call NovelCraft, for assessing novelty detection and accommodation from both visual and multi-modal observed data. This dataset is built from prerecorded imagery obtained from an agent exploring the Polycraft POGO environment released by our TA1 partners at UT-Dallas. This dataset makes visual detection in this challenging open-world environment accessible to researchers around the globe who want to focus on visual processing rather than build fully interactive agents.




```
{"player": {
  "pos": [55, 17, 43],
  "facing": "WEST",
  "yaw": 90,
  "pitch": 0
}, ...
"entities": {
  "103": {
    "type": "EntityTrader",
    "pos": [46, 17, 43],
  }, ... },
"map": {
  "45,17,52": {
    "name": "plastic_chest",
    "facing": "north"
  },
  "47,17,56": {
    "name": "log",
    "axis": "y",
    "variant": "oak"
  },
  "59,17,62": {
    "name": "air",
  }, ...
}}
```

Example contents of our new multi-modal NovelCraft dataset. Left: Select image frames from two normal episodes (rows 1-2) and two novel episodes (rows 3-4). Typical episodes contain 40-60 frames. Detection is challenging as only a few images in novel episodes actually contain novel objects, here outlined in orange when visible. Right: Example symbolic information available at each frame. At each frame, we record a complete JSON representation of the player state and all objects and artificial agents in the world (x,z,y position in 3D, orientation, etc.). This includes objects outside the player's view recorded in images.

Our NovelCraft dataset addresses the program goal of **principled characterization and detection.** It contains imagery of an open world scene in a MineCraft-like world from the perspective of an agent attempting to complete a task (build a pogo stick). The

movement of the agent ensures that novelties are depicted from a variety of perspectives. Our data is grouped into episodes, where each episode is one attempt at the task by the agent. Within each episode, the open world scene is consistent and evolving over time, with sensory information captured at each step of the agent completing the task. Thus computer vision models can be tested on data representative of an integrated AI system operating in an open world without having to spend time and computational resources rerunning the environment and agent. Labels are provided for each episode denoting whether there is novelty and what the novelty is. This enables evaluations to measure novelty detection performance and the ability of models to infer novel concepts.

This dataset should substantially improve evaluation practices in the research community compared to what is currently done in the status quo. Most existing methods for novelty detection are evaluated on datasets like CIFAR-10 and ImageNet, which were originally intended for supervised classification of visual object types. However, these datasets tend to be object-focused and do not have images representative of open world scenes, where novelties may not be large, centered, or fully visible in the images. Some recent works, like the Common Objects in Context (COCO-2014) dataset, have developed datasets that depict complex visual scenes similar to open worlds. These datasets include only one image per scene, which limits their applicability to an AI system operating in a consistent open world scene over time.

3.2 Evaluation of Detection on NovelCraft

In the NovelCraft paper (Feeney et al. 2023), we evaluate visual, symbolic, and multimodal models' ability to algorithmically detect novelties during an episode. Visual models are provided with image data, symbolic models are provided with JSON data representing the state of the world, and multimodal models ensemble the outputs of each of those models. The symbolic model detects novelty by extrapolating from previous world states, so we test the model when given a sequence of 2 or 5 previous states to extrapolate from.

			at TPR 95%			at PPV 95%		
Model	AUROC	AUPRC	TNR	PPV	Delay	TNR	TPR	Delay
Visual (AE Patch)	81.62	**95.61**	31.34	**88.72**	**5.32**	80.60	66.47	14.41
Symbolic (2 step)	76.12	94.18	4.48	84.96	10.39	**98.51**	5.04	33.24
Symbolic (5 step)	81.51	**95.28**	11.94	85.97	10.19	**98.51**	5.04	32.89
Multimodal (2 step)	80.46	94.65	**34.33**	**89.15**	7.44	**98.51**	5.04	33.14
Multimodal (5 step)	**85.17**	**95.83**	**34.33**	**89.15**	6.93	77.61	**75.07**	**12.72**

Table 1 (from Feeney et al 2023): Episodic novelty detection performance (higher is

better for all but delay) on test episodes where the novelty is detectable by both visual and symbolic methods. Multimodal models perform better than either visual-alone or symbolic-alone models.

Overall, we find that multimodal ensembles that combining algorithmic novelty detection techniques from both vision and symbolic data types tend to reach the highest level of performance, especially in “area-under-the-curve” metrics. This arises from an ensemble’s ability to take advantage of visual and symbolic methods’ differing performance on novelties at different levels of abstraction. We further find that models with more context perform better (5 step is recommended over 2 step).

Another key takeaway from our work is that different methods may be recommended depending on whether false positives or false negatives are more important to minimize. Looking at detector performance in the true positive rate (TPR) equals 95% regime represents the case where finding all novelties is most important (prioritizing models that avoid false negatives). The multimodal 5-step model performs the best, with the visual model getting similar performance while slightly reducing the number of frames in an episode before the novelty is detected (delay). Looking at the regime when positive predictive value (PPV, aka precision) equals 95% represents the case where being confident that a predicted novelty is really novel is most important (prioritizing models that reduce false positives). The multimodal 5-step model provides the best overall performance, but the symbolic and multimodal 2-step model can better identify normal episodes though at the expense of novel episodes.

3.3 Evaluation of novelty characterization on NovelCraft

Moving beyond simple visual novelty detection, we investigated *generalized category discovery* (GCD) is a task that provides a principled approach to characterization of visual novelties. GCD assumes a set of normal data labeled with examples of different classes. Given a set of unlabeled data, some of which may be novel, GCD aims to group the unlabeled data into both the normal classes and new novel classes. In the context of an integrated AI in an open world environment, the goal of GCD is to determine whether incoming data is from a normal environment, a novel environment that has been seen before, or a novel environment that is new to the agent.

When evaluating GCD methods on NovelCraft (Feeney et al. 2023), we found that current GCD models need to be made more robust to class imbalance to be suitable for an open world environment setting. Our experiments utilized a long tailed class frequency distribution, which assumes that there are a few normal classes with lots of data and a mix of normal and novel classes with significantly less data. The model’s strong performance on normal data, listed as labeled in the table, can be attributed to successfully recognizing data from normal classes with lots of data.

Although some of the less frequent classes are successfully clustered, future work is needed to achieve this result consistently and raise the accuracy for novel, listed as new in the table, data.

3.4 Integration of vision into DIARC agent

To provide algorithmic novelty detection for our team's integrated AI system, DIARC, we provided a visual autoencoder novelty detection model. Our experiments found that autoencoders perform surprisingly well in open world settings, being more robust and outperforming state of the art novelty detection models for object-focused datasets. Autoencoders also perform well under a variety of thresholding regimes, making the model flexible enough to be quickly adjusted to satisfy the needs of our agent development team.

Our TA1 partner from UT-Dallas generously provided an evaluation of our visual novelty detection module on the subset of their novelties that include new, visually distinct objects in the environment. This includes novelties such as trees with different colored bark or new fence objects. Novelties that vary visual stimuli seen in the normal environment, such as trees producing more logs when broken without changes to the tree or log visuals, are not included. This evaluation also disabled the agent team's novelty detection system in order to evaluate the visual novelty detection module in isolation. However, the agent is still informed by the visual novelty detection module in order to determine when exploration of the environment is necessary.

method	***M1: Fn_CDT***	***M2: CDT%***	***M2.1: FP%***
Patch AE @ 36 mo	6.70	27.8%	58.3%
Patch AE @ 42 mo	**3.44**	**56.6%**	**34.1%**

Table: Progress of our vision-only agent at novelty detection on Polycraft via a third-party evaluation by TA1, in terms of SAILON metrics. First row represents the agent at 36 months, final row represents our performance after 42 months.

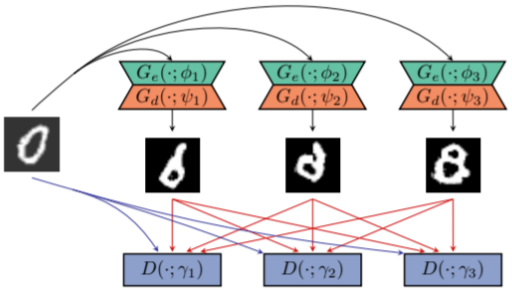
Our initial evaluation at 36 months revealed two weaknesses of the visual novelty detection module. First, we found that our validation set of novelties provided by UTD was significantly less challenging than the novelties UTD tested us on. This led us to more aggressively reduce the number of false positives when choosing a threshold for our model. Second, we found that the novelties we performed worst on were very different from the novelties in our validation set. Increasing the variety of novelties in our validation set through further data collection allowed us to more accurately predict how the model would generalize to UTD's evaluation. Our final evaluation showed significant improvement and suggests that our vision module complements the DIARC agent's other novelty detection modules moving forward.

4. Agent novelty detection and accommodation

We developed new machine learning algorithms to detect novelties encountered by the agent. In particular, we have innovated models for detecting novelties in the i.i.d. and time series settings. We also have investigated methods to update classifiers in a gradient-free fashion.

4.1 GAN ensemble for novelty detection

In our work of detecting novelties, we innovated Generative Adversarial Networks (GANs) to detect novelties more accurately. GANs are powerful models for detecting novelties in the data, however, they are hard to train: the training procedure of a GAN model is a min-max problem, whose optimal solutions are hard to reach. In the novelty detection task, the discriminator is usually used to compute representations to distinguish normal examples and novel ones. If the discriminator is not well trained, then its ability in detecting novelties is compromised.



The structure of a GAN ensemble model. Each generator is an autoencoder, which perturbs a normal example to generate a fake example. Each generator is criticized by multiple discriminators, and each discriminator receives fake samples from multiple generators.

We addressed this problem by using an ensemble of generators and discriminators in the GAN model to greatly improve the model performance over previous GAN-based novelty detection models. In our work, we used multiple discriminators and generators in the GAN for novelty detection. In our ensemble model, a generative model was an auto-encoder: it perturbed a normal example to generate a “fake” example. Each discriminator received “fake” samples from multiple generators, and the discriminator was trained to distinguish these fake examples from normal examples. Therefore, all these samples provide better profiling of normal examples, and thus a discriminator’s ability in detecting novel instances is enhanced. To further improve the detection performance, we also concatenate learned representations from all discriminators to enrich the information.

We have conducted an extensive empirical study of the proposed model. Compared with a previous GAN model, the representations from a GAN ensemble better distinguished abnormal examples from normal ones. We tested our model on several novelty-detection tasks, and the proposed model significantly outperformed previous GAN-based detection models.

4.2 Novelty detection in time series

The detection of novelties also needs to consider the context of the data. This is especially true in the tasks of detecting novelties from time series data. For example, a hot day is normal, a hot day after a cold day with freezing temperature is not normal. Therefore, a novelty detection model needs to be able to model the relationship between data entries.

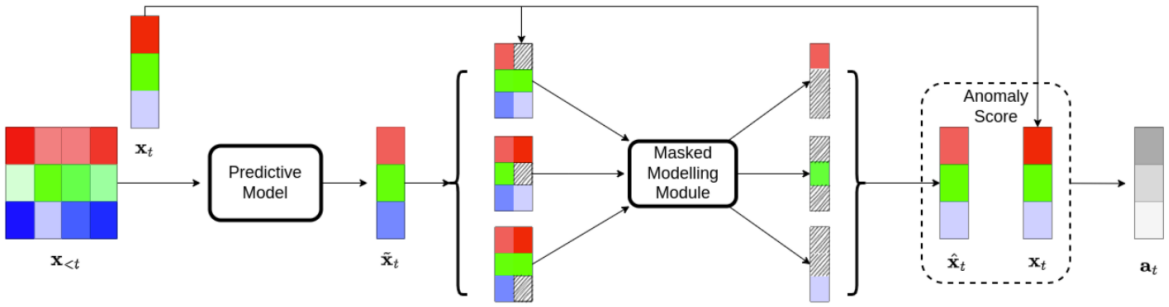


Illustration of masked modeling strategy. The current observed variables in a time step are masked *K* times, and each mask (dashed) produces a separate prediction. These predictions are assembled to form the final prediction, which is compared to the observed value to get the anomaly score.

In this part of the work, we devised a new novelty detection model that was able to model both temporal relationships and relationships between feature entries each time step. To capture the temporal relationship, we used an autoregressive predictive model. To capture relationships between feature entries, we proposed a new asked training strategy. In each step, we masked out some data entries and let the model predict these masked entries based on other data entries and previous steps. With the new training strategy, the model had an enhanced ability to capture relationships between data entries in the current time step. It also captured the relationship between data in the time step and data from previous time steps through the autoregressive model. We further developed a transformer-based neural network as an autoregressive predictive model. This new model had stronger abilities to learn from the data than recurrent neural networks.

We tested this model on Polycraft and Monopoly tasks. Our experiments indicated that the strategy of masked training does improve the performance of novelty detection.

4.3 Fast adaptation after detecting novelties

Once a model detects a novel object, it should be updated to remember the object and not detect it as a novelty in the future. Ideally, the model update should be fast and need little training, instead of running further optimization iterations to learn new concepts. Learning new concepts has been studied in the field of continual learning. However, the

focus of the research in this field is to avoid forgetting previously learned concepts, and it rarely concerns the computation issue.

In this part of the work, we devised a learning model that was updated without gradient-based optimization. It combines the strength of neural networks and K-nearest-neighbor classifiers (KNN). We used a neural network as a flexible feature extractor and did not need to update it in new tasks because representation learning

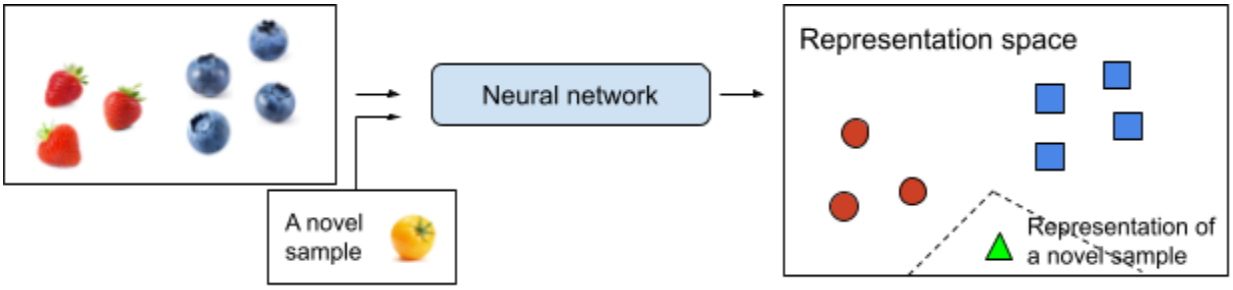


Illustration of gradient-free model update. When a novel example is present the classifier, its representation from the neural network will be stored in the KNN classifier for further predictions.

could be done on the training set. We used a KNN classifier as a classification head because it requires little computation to update the classifier: we only needed to add representations of a few new examples. When an example of a seen novelty type was fed to the classifier, it gets its representation from the neural network and then is classified by the KNN into the known novelty type. Compared to a pure neural model, which needs expensive computations to get updated, our model has a clear advantage of low cost in model updates.

We tested our model in a series of benchmark tasks and showed that our model has the ability to adapt to new classification tasks.

5. Inference-based novelty detection

The main methodology that was used in novelty detection was having two modules, one for detecting novelty with respect to the environment and another with respect to the agent which we use both in Monopoly and Polycraft, even though the details of the detection were different given the differences in the tasks and settings (e.g., observability, stochasticity, etc.). The first covered the environment and event novelties in M42 and the second covered the goal novelties in M42. For the first we developed an accurate simulation of how the environment would change in a normal situation and compared that with the observation about the environment. When there was a difference between our expectation based on the simulation with our observation, novelties were detected. For the second, we developed a simulation of the normal agent behavior and compared that with the observation about the agent. When there was a difference – beyond a well-designed threshold – between our expectation based on the simulation with our observation, novelties were detected.

For novelty characterization, we first detected if it was an agent related novelty (such as a goal novelty) or not; if not then if an environment related novelty was detected, then we analyzed the difference between the environment expected and the environment observed and based on that made the characterization of whether it was an environment novelty or an event novelty,

For novelty accommodation we used a two stage MCTS (Monte Carlo Tree Simulation) rollout method (to be described in Section 7).

5.1 ASP-based novelty detection

Our ASP novelty detection component for Monopoly is an inference-based novelty detection module where we perform simulations based on the environment states and actions (we used a similar module for Polycraft as well). Specifically, given the current game board state, a sequence of actions, and a ground truth next game board state, we infer the next state by executing actions on the current state with ASP. If the inferred next state deviates from the ground truth next state, our ASP novelty detection module will report novelty. The intuition is that if there is any novelty during inference, for example, a new sell factor or a new color for a property, the inferred next state using the current knowledge will deviate from the ground truth next state. Advantages of using ASP simulator to perform inference include that ASP rules can be automatically generated from PDDL descriptions in any action domain, and a large varieties of novelties can be covered without caring too much about what those novelties are and their novelty hierarchy since we only care about whether deviation exists. Below we first give a glimpse of how ASP can be used for reasoning about the next state and how that code can be minimally modified to infer a novelty. In the Monopoly game, to reason

about the next state, the ASP code will first define the game parameters through facts such as the following:

```
dice_value(1..6).
player(player1;player2).
cash(1..1000).
asset("B&O_Railroad").
penalty(50).
```

Then rules of the following form are used to define actions and fluents.

```
action(sell_property(P,X)) :- player(P), asset(X).
fluent(asset_owned(P,V)) :- player(P), asset(V).
```

Properties of actions, such as their pre-conditions, and their effects are defined using rules of the following kind:

```
%Executability of selling assets
:- occurs(sell_property(P,V), T), player(P),
asset(V), time(T), not holds(asset_owned(P,V),T).

%Effect of selling assets
not_holds(asset_owned(P,V),T+1)
holds(asset_owned(P,V),T),
occurs(sell_property(P,V),T),
player(P), asset(V),time(T).

not_holds(asset_mortgaged(P,V),T+1) :- holds(asset_owned(P,V),T),
occurs(sell_property(P,V), T),
player(P), asset(V), time(T).

holds(current_cash(P,X+Y),T+1) :- holds(current_cash(P,X),T),
occurs(sell_property(P,V),T),
not holds(asset_mortgaged(P,V),T),
asset_price(V,Y), player(P), asset(V), time(T).

not_holds(current_cash(P,X),T+1) :- holds(current_cash(P,X),T),
occurs(sell_property(P,V),T),
```

```
not holds(asset_mortgaged(P,V),T),
asset_price(V,Y), player(P), asset(V), time(T).

holds(current_cash(P,X+Y),T+1) :- holds(current_cash(P,X),T),
occurs(sell_property(P,V),T),
holds(asset_mortgaged(P,V),T),
asset_m_price(V,Y), player(P), asset(V), time(T).

not_holds(current_cash(P,X),T+1) :- holds(current_cash(P,X),T),
occurs(sell_property(P,V),T),
holds(asset_mortgaged(P,V),T),
asset_m_price(V,Y), player(P), asset(V), time(T).

%Executability of paying jail fine
:- occurs(pay_jail_fine(P), T), player(P), time(T),
not holds(in_jail(P), T).

:- occurs(pay_jail_fine(P), T), player(P), time(T),
not holds(current_cash(P, _), T).

:- occurs(pay_jail_fine(P), T), player(P), time(T),
holds(current_cash(P,X),T), X < 50.

%Effect of paying jail fine
not_holds(in_jail(P), T+1) :- holds(in_jail(P), T),
occurs(pay_jail_fine(P), T),
player(P), time(T).

not_holds(current_cash(P, X), T+1) :- holds(current_cash(P,X),T), holds(in_jail(P),
T), occurs(pay_jail_fine(P), T), player(P), time(T).

holds(current_cash(P, X-50), T+1) :-holds(current_cash(P,X),T), holds(in_jail(P),
T), occurs(pay_jail_fine(P), T), player(P), time(T).
```

The inertia rules are expressed as follows:

```
holds(F,T+1) :-
```

```
fluent(F), holds(F,T),
not not_holds(F,T+1), time(T).
not_holds(F,T+1) :- fluent(F), not_holds(F,T),
not holds(F,T+1), time(T).
```

The initial state is defined using holds facts with respect to time point 0 such as:

```
holds(in_jail(player1), 0).
holds(current_cash(player1,500),0).
```

An action occurrence at time point 0 is then defined as a fact in the following form.

```
occurs(pay_jail_fine(player1),0).
```

Now when a complete ASP program with rules and facts of the above kind is run, we get an answer set from which we can determine the state of the world at time point 1. Suppose that the answer set has the facts:

```
holds(in_jail(player1), 0).
occurs(pay_jail_fine(player1),0).
holds(current_cash(player1,500),0).
holds(current_cash(player1,450),1).
while our next observation gives us:
obs(current_cash(player1,477),1).
```

The discrepancy between our prediction about player1's current_cash being 500 (at time point 1) is different from our observation that player1's current_cash is 477. This suggests there is a novelty. This can be determined by the following two simple rules.

```
discrepancy(F,T) :- fluent(F), time(T),
holds(F,T), not observed(F,T).
discrepancy(F,T) :- fluent(F), time(T),
not holds(F,T), observed(F,T).
```

While the above could have been implemented in any language, including in the simulator language, which we also implemented in Python, having it in ASP, makes it easier for us to take the next step, which is to find out what the novelty is. In ASP, we have to modify the above ASP code by adding the following and removing "penalty(50)" (referring to the jail fine in the Monopoly game) from the original code.

```
oneto200(1..500).
1 { penalty(X) : oneto200(X)} 1. %choice rule
:- obs(current_cash(P,X),1),
holds(current_cash(P,Y),1), X!=Y, player(P).
```

In the above, the first fact and the choice rule defines the range of penalty that we are exploring. If we had just those two rules, we would get multiple answer sets with a penalty ranging from 1 to 500. The constraint (the last ASP rule) then eliminates all the answer sets where the observation about current_cash does not match with the holds. In the answer set that remains, we get the penalty value that would make the observation match with the holds, thus allowing us to figure out the novelty with respect to the penalty. In this particular case, the program will have the answer set with “penalty(23)" thus characterizing the novelty that the penalty is now 23. The agent uses a novelty identification module to characterize the novelty. This module has several sub-modules (which can be run in parallel), each focused on determining a specific novelty type. We add two modifications that are used for hypothetical reasoning about the effect of an action. The first modification allows the ASP module to replace parameters with constant values by choice values. For the second modification, constraints are added to remove answer sets where the predicted game board state does not match the observed game board state. The resulting answer sets give us the parameter values which reconcile the predicted game board state and the observed game board state. Now we can update our ASP code that was used for hypothetical reasoning by simply replacing the earlier value of the parameter with the new value.

We illustrate the novelty characterization process using ASP as follows. For an action whose effects or preconditions are changed to cause a novelty, to detect the changed preconditions, ASP rules can be written to enumerate the various possible preconditions, and then constraints would be written to pick the right precondition, and similarly for effects. An ASP sample code snippet (in Clingo) to discover effects is as follows:

```
not_holds(f,0).
0 { causes(a,f); causes(a,neg(f)) } 1.
holds(F, T+1) :- occurs(A,T), causes(A,F).
occurs(a,0).
:- not holds(f,1).
#show causes/2.
```

Here knowing f is not true at 0, and observing an occurred at time step 0, and f is true at time step 1, one can use the above code to infer "$causes(a, f)$". An ASP sample code snippet to discover preconditions is given below. Using that, one can infer

```
"prec(a,neg(p))".
0 { prec(a,p); prec(a,neg(p)) } 1.
1 { holds(p,0) ; not_holds(p,0) } 1.
fails(A,T) :- prec(A,F), time(T), fluent(F),
not holds(F,T).
fails(A,T) :- prec(A,neg(F)), time(T), fluent(F),
not not_holds(F,T).
:- not holds(p,0). %observed that p was initially true
:- not fails(a,0). %observed that a failed in time 0
#show prec/2.
```

The ASP enumerate, and test-based modules were very good at detecting small changes; they not only detected a novelty but often predicted the exact novelty, e.g., some novelties, such as rent price change, can be easily detected and characterized.

5.2 Natural language author-attribution domain

Novelty detection and accommodation in the natural language domain was one of the domains that we had proposed in our original proposal. In the first phase of SAIL-ON we did not have an external TA1 team that would generate challenges for our TA2 effort. When a TA1 team developed an NL challenge on "author attribution" in Phase 2 of the SAIL-ON program, we were the only TA2 team that worked on it but ceased work as the TA1 team dropped out in Phase 3. Our M30 results are shown below:

U.S. NAVAL RESEARCH LABORATORY

M30 Results: VAR & NLTI (PAR Government)

Targets	Relation	M30 Results: VAR			Phase 2 Targets	M30 Results: NLTI	Phase 2 Targets
		Kitware	Raytheon BBN	SRI	VAR	Tufts	NLTI
M1: FN_CDT	≤	152.9	91.4	120.5	100	97.3	164
M2: CDT%	≥	39.1%	85.0%	62.8%	75%	81.4%	86%
M2.1: FP%	≤	60.8%	3.4%	9.1%	16%	18.6%	14%
M3: NRP_s	≥	0.79	0.73	0.76	0.83	1.43	0.90
M4: NRP_G		0.40	0.71	0.79	0.85	1.43	0.92
AM1: OPTI		55.1%	53.9%	55.3%	52%	58.9%	56%
AM2: APTI		56.1%	54.4%	56.4%	51%	59.3%	56%

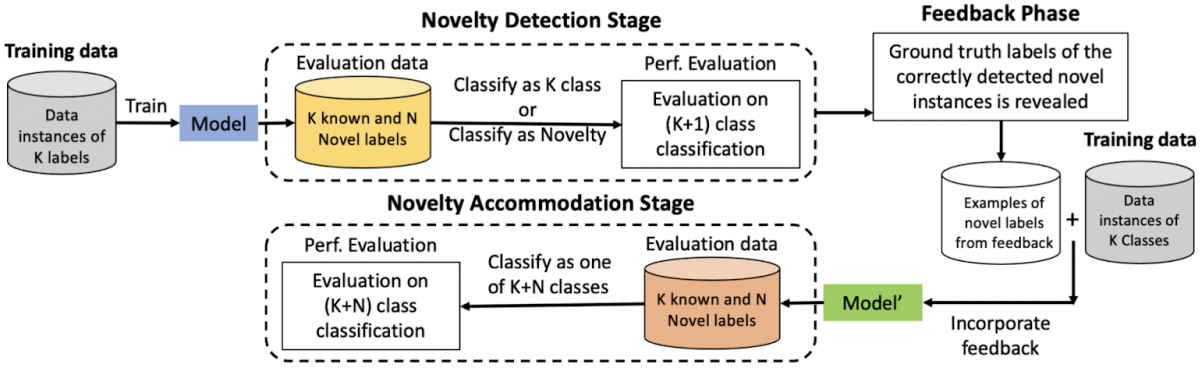
Observations

- Kitware: First (non-pathological) results for detection measures
- Raytheon BBN: Good overall performance
- SRI: Good accommodation results
- Tufts: Achieved 5 of 7 targets

While the PAR-UIC team did not officially continue for the third phase. However, we continued working with them informally which resulted in the following paper.

1. Neeraj Varshney, Himanshu Gupta, Eric Robertson, Bing Liu and Chitta Baral. A Unified Evaluation Framework for Novelty Detection and Accommodation in NLP with an Instantiation in Authorship Attribution. Findings of ACL 2023. https://arxiv.org/pdf/2305.05079.pdf

The main contribution of this paper was a unified evaluation framework for Novelty Detection and Accommodation in NLP.



The above figure illustrates our multi-stage pipelined formulation of NoveltyTask. Initially, examples of a set of K labels ('known labels') are provided for training a classification system. The first evaluation stage i.e. the 'novelty detection' stage consists of evaluation instances from the K known labels and 'N' novel labels. For each instance, the system needs to either classify it to one of the K known classes or report it as novel (not from any of the K known classes) i.e. the system is evaluated on a (K + 1) class classification problem. This stage is followed by a Feedback phase in which the ground truth label of the novel instances that the system correctly reports as novel is revealed. The system then needs to leverage these new examples (of the novel labels) for the second evaluation stage (novelty accommodation) in which it is evaluated on a (K + N) class classification problem.

5.2.1 Natural Language Domain (independent of an external TA1 evaluation):

Novelty detection and accommodation in the natural language domain was one of the domains that we had proposed in our original proposal. In the first phase of SAIL-ON we did not have an external TA1 team that would generate challenges for our TA2 effort. Nevertheless, we worked on it on our own and this led to the following papers:

2. Neeraj Varshney, Swaroop Mishra, Chitta Baral. Investigating Selective Prediction Approaches Across Several Tasks in IID, OOD, and Adversarial Settings. Findings of ACL 2022. https://aclanthology.org/2022.findings-acl.158.pdf

3. Neeraj Varshney, Swaroop Mishra, Chitta Baral. Towards Improving Selective Prediction Ability of NLP Systems. ACL 2022 RepL4NLP Workshop. https://arxiv.org/pdf/2008.09371.pdf

Without external evaluation we followed the existing terminology used in the NLP domain. For example, we modeled the notion of "novelty detection" in terms of "selective prediction". The idea behind is that when a novelty is detected the model being not sure of its ability to predict accurately would decline to predict; thus the term "selective prediction". We studied this in the above two papers with respect to 17 datasets across multiple NLP tasks such as Natural Language Inference (NLI), Duplicate Detection, and QA tasks and under In-Domain (IID),Out-Of-Domain (OOD), and Adversarial (ADV) settings.

We then studied novelty accommodation in two ways. Initially we studied OOD generalization methods in NLI, Question Answering and Image Classification tasks in the following paper.

4. Tejas Gokhale, Swaroop Mishra, Man Luo, Bhavdeep Singh Sachdeva, Chitta Baral. Generalized but not Robust? Understanding the Effects of Out-of-Domain Generalization Methods. Findings of ACL 2022. https://aclanthology.org/2022.findings-acl.213.pdf

We then explored novelty accommodation, which we referred to as “Post-Abstention”, that allows re-attempting the abstained instances with the aim of increasing coverage of the system without significantly sacrificing its accuracy. We provided a mathematical formulation of this task and then explored several methods to solve it. Comprehensive experiments on 11 QA datasets showed that these methods lead to considerable risk improvements–performance metric of the Post-Abstention task– both in the in-domain and the out-of-domain settings. Our paper on this is accepted in ACL 2023.

5. Neeraj Varshney and Chitta Baral. Post-Abstention: Towards Reliably Re-Attempting the Abstained Instances in QA. ACL 2023. https://arxiv.org/pdf/2305.01812.pdf

5.2.2 Vision-language domain (independent of an external TA1 evaluation):

We also worked on developing robust methods in various vision-language tasks. These methods are useful in novelty accommodation. Following are two papers of ours on this.

6. Tejas Gokhale, Abhishek Chaudhary, Pratyay Banerjee, Chitta Baral, Yezhou Yang. Semantically Distributed Robust Optimization for Vision-and-Language Inference. Findings of ACL 2022.

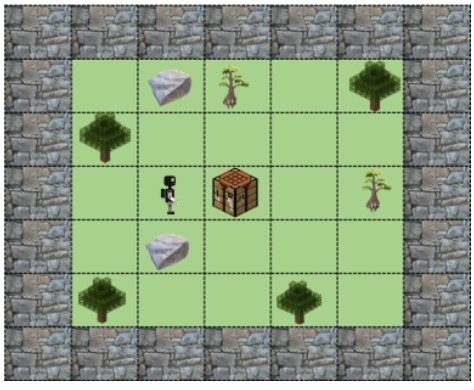
7. Tejas Gokhale, Rushil Anirudh, Bhavya Kailkhura, Jayaraman J. Thiagarajan, Chitta Baral and Yezhou Yang. Attribute-Guided Adversarial Training for Robustness to Natural Perturbations. AAAI 2021.

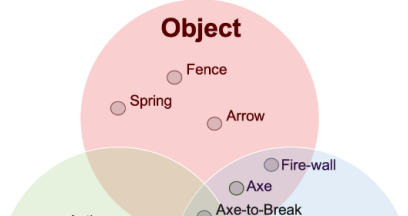
6. Hybrid planning reinforcement learning for novelty accommodation

We developed several novel algorithms that would allow us to handle novelties we could not easily detect or characterize based on a novel hybrid planning reinforcement learning (RL) framework, initially the SPOTTER algorithms and later a specialization and refinement with the RapiLearn algorithms, and also the biplex planning framework. The idea is that the planner should plan for discoveries and if it does not have any operators that apply in a world changed by the introduction of a novelty, it needs to make a discovery to find out what has changed. As described in Section 2, we had developed an elaborate framework for handling unexpected changes and events that includes characterizing them and recovering from them. But sometimes it was not possible to determine how to get past an impasse, in which case the hybrid framework resorted to different types of RL algorithms to detect possible actions that could lead again to plannable states from which the planner could find a path to the goal state. In particular, when facing a prohibitive novelty that results in the failure of execution of known actions, the agent can still use its domain knowledge as well as any observations it made about the failure to formulate an RL problem on-the-fly and try to reach a state from which it can recover. We extended the framework from a tabular representation to rich and continuous sub-symbolic spaces that require function approximation to learn successful policies. To evaluate the proposed methods, we developed NovelGridWorlds, a 2D environment designed to mimic the Polycraft environment that our agent was tested on throughout the program. The NovelGridWorlds environment contains versions of all tasks used in the evaluation but it also features simpler crafting and navigation problems.

6.1 The NovelGridWorlds environment for benchmarking novelty-aware agents

We developed “NovelGridworlds”, an OpenAI Gym environment framework for developing and evaluating AI agents that can detect and adapt to unknown sudden novelties in their environments. The environment was designed to mimic the Polycraft domain at the gridworld-level representation of the domain, an example visualization of a simple 2D world is seen on the figure to the right. The environment was originally designed so that we can perform internal evaluation of our across a wide-spectrum of novelties. The environment also provides an interface for specifying a variety of tasks in addition to the main task used in the Polycraft environment, allowing us to test our agent on crafting

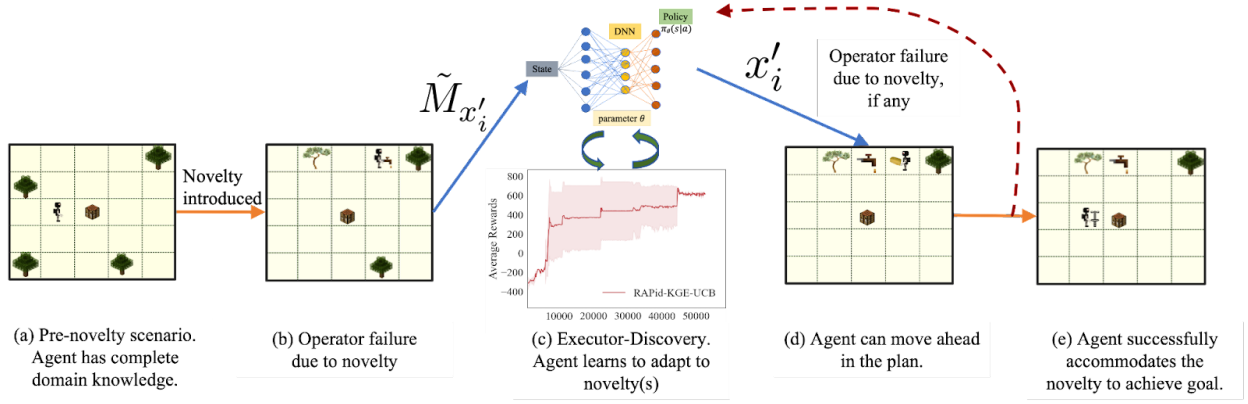




tasks that are simpler as well as more complex compared to the original pogo stick crafting domain.

The environment affords an easy to use interface for implementing novelties according to the custom-designed hierarchy shown to the right. This interface allows the user to specify novelties that affect objects, actions, and their attributes. For example, one of the novelties consists of a novel action called “jump”, allowing the agent to move faster and hop over obstacles. Another action novelty is that of action remapping, where the control outputs of the agent are “scrambled” to call different actions as compared to the pre-novelty scenarios. A novelty may affect actions, objects, and attributes at the same time – e.g., the “ax to break” novelty is implemented as a novel action, performed with a novel object, resulting in a novel affordance.

The environment also affords multiple types of sub-symbolic representation of the world which are often used by the research community for evaluating reinforcement-learning algorithms. These representations include image-based top-down views of the world for use by CNNs, as well local-view, LiDAR-like representations which mimic a physical robot’s partial observation of the world. The environment was used for evaluating our RaPID-learn framework and algorithm for knowledge-guided RL, where an agent formulates an RL problem on-the-fly when it faces a novelty that breaks down its solution based on the pre-novelty domain knowledge, shown in the figure below.

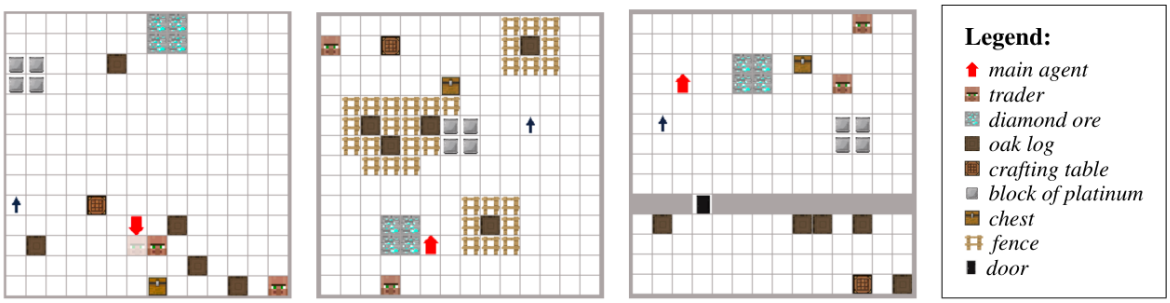


In September of 2022, the environment was cloned into NovelGridWorld 2.0 and extended as follows:

1. The 2.0 version supports multiple-agents that include a competitor crafting agent as well as agents that can be used to trade items with.
2. The new environment supports the notion of partial-observability at the symbolic level; it can include multiple rooms as well as objects (e.g., a “safe”) that may contain items that are unseen until interacted with.

3. The 2.0 version also provides a code-free interface for specifying certain classes of novelties, in addition to the code-based interface which requires the user to specify the novelty in Python code.

Some example worlds generated with NovelGridWorlds 2.0 are seen below:



(a) Moving Traders (b) Across the Fence (c) Multi Rooms

The environment will be submitted as a paper to the 2024 AAAI Conference.

6.2 The hybrid planning reinforcement-learning framework

When prohibitive or obstructive novelties are attributed to operator failure, the agent needs to discover new operators that may allow it to solve the task.

Precondition discovery. The first strategy to discover new operators involves precondition exploration. For a given failed operator $o \in O$ with preconditions ψo , a new operator o' is constructed with the same effects and executor as o but with a new set of preconditions. The new preconditions $\psi o'$ are constructed using a priority-based order of all possible preconditions encountered in O. The preconditions are added to $\psi o'$ one by one, and goals are submitted to
the GOAL MANAGER to satisfy them and attempt the new operator o' . If the operator succeeds, the broken operator o is replaced with this new operator oi and added to KB.

Operator variations. If the precondition discovery fails to produce a working operator, the agent actively searches for known operators with unknown effects. This process is guided by the agent's type hierarchy and has two phases. In the first phase, the agent attempts (compatible) operators with the same parameters as the failed one, and in the second phase, prioritizes operators that act on different parameters of the same type as the failed operator. If any operator produces unexpected effects, a new operator o' is added to KB, and the agent re-plans. For instance, consider a scenario where traders are no longer available to interact. In this scenario, when the interact operator fails. The

agent may attempt to interact with the pogoist as it belongs to the same type as the traders. If both precondition discovery and operator variations fail to produce an operator that enables a successful plan, then the agent attempts to learn a new executor for the failed operator using knowledge-guided reinforcement learning.

Due to the exponentially large search space of precondition discovery and operator variations, the agent cannot exhaustively search that space. After expending some effort in those directions to no avail, a more intelligent search strategy is employed to create a new executor for the failed operator. That way, the agent can discover ways to achieve a desired effect that would assist it in solving the task. The executor learning component employs RL with a reward function encoding the failed operator's desired effects. A new operator is then created with preconditions derived from the state description where the failure occurred and effects identical to the failed operator. The learned policy is used as the new operator's executor, and the agent can use it to reach a state from which it can plan and complete the task. The proposed algorithm called RAPid-learn uses knowledge-guided exploration informed by the novelty. The component receives symbolic state descriptions $s \in S$ and novelty descriptions from the goal manager and temporarily guides the agent's behavior to explore the environment.

Algorithm 2 ***Executor learner*** $(T, \mathcal{P}, \omega_o, s_f) \rightarrow x_{new}$

Inputs:
$T = \langle \Sigma, s_0, s_g \rangle$ ▷ Symbolic Planning Task
$\mathcal{P}$ ▷ Plan $\mathcal{P} = \{o_1, o_2, ..., o_{|\mathcal{P}|}\}$
N_{eps} ▷ Number of episodes
$\eta, \omega_o, \tilde{s}_f$
Procedure:
$\gamma(s)$: initiation indicator ▷ computed from s_f
$\beta(s)$: termination indicator ▷ computed from ω_o
Construct MDP $\mathcal{M} = \langle \tilde{S}, \tilde{A}, R, \gamma \rangle$ using T
for N_{eps} *episodes* **do**
    $\pi_o^{new} \leftarrow$ ***Train***$(\mathcal{M}, T, \beta)$ ▷ Train in $\mathcal{E}$
    if $success(\pi_\mu^o, T) > \eta$ **then**
        $\chi_{new} \leftarrow \langle \gamma, \pi_\mu^o, \beta \rangle$ **return** χ_{new}
    end if
end for
return *failure*

The procedure for learning a new executor is described in Algorithm 2.

Knowledge-guided exploration of novelties. An important feature of the executor learner is its knowledge-guided exploration strategy. Using a description of novelties , the exploration strategy of the RL learner is biased towards states and actions that are related to the novelty. For instance, if the presence of a novel object is detected in the environment, the RL learner may be encouraged to explore actions on that object. The knowledge-guided exploration

improves the efficiency of exploration and makes the plan recovery process easier. Once a new policy $\pi\mu o$ for the executor χo is learned, the resulting operator is stored in the KB. The parameters of the policy μ are stored in and retrieved when χo is executed.

7. Multi-player novelty detection and accommodation

For multi-player novelty detection, characterization and accommodation we used Monopoly as the main task which is a multi-player adversarial board game with up to four players. For the SAIL-ON program evaluation, agents were tested with one novelty injected per trial, where each trial is 100 games of Monopoly. The novelty could be added in any one of the 100 games and persists for the remaining games. The novelty could be changing the number of properties in a set required for Monopoly, the rent of a property after building a hotel on it, the order of properties on the board and such. The set of possible novelties is *not* shared with us by the evaluation/test team, and so it is left to us to make the agent as robust and adaptable to novelties as possible.

We developed the novelty accommodation component as a policy controlled by a state-value function. The value of a state is primarily determined by the expected short and long term reward obtained from that state. Typically, approaches that use a value function for game-playing agents – like Monte Carlo Tree Search – either simulate trajectories to the end and backpropagate the terminal state value to compute the starting state's value, or they use a limited lookahead with an evaluation function that captures the value of the rest of the trajectory. We use the simplest form of the limited-lookahead approach where we just lookahead by one-step and then evaluate the next state by approximating the expected short and long term returns that would result by taking the action. The reason for planning with a one-step lookahead was that in the game of Monopoly, a single roll of the die, or a chance card, or an adversary's decision could change the entire value of a state. So to compute the value of a state accurately with simulated actions, requires considering a very large set of branches from an extremely wide and deep tree that includes many possible combinations of dice rolls, combination of player decisions, auction bids, and more. It should be noted that each turn of a player also includes what are called out-of-turn moves by other players, which further increases the branching factor of the search tree.

If one had a very accurate mental model of adversaries, the possible branches of the search tree might become more manageable. Additionally, pre-training a large neural network for state evaluation is not viable since our agent would have to handle novelties or modifications to the game (the space of which we did not know). Lastly, the evaluators impose a max time limit of 3 hours per full-game, so simulating enough MCTS rollouts for each action did not seem feasible. Rather than requiring an accurate and complete model to rollout and evaluate each state, we consider long-term consequences with simplifying assumptions (to be described below). The state value includes the current monetary value of possessions, potential short and long term gains, as well as the future benefit of monopolized properties. Importantly, the evaluation function is largely parameterized with game attributes (that can change) and has few tuned constants; this helps make it robust to game variations. We will first go over the

evaluation function. We will then provide some examples of the state attributes that are tracked and updated in V(s) to accommodate for novelty.

The value of a state should consider the current (monetary) value of owned properties as well as the potential for future earning as possible future rewards. Thus, the evaluation function we propose is a linear combination of four terms i.e., V$(\mathrm{s}) =$ Massets $+$ Rs $+$ Rl $+$ Mmonopoly. Each of these terms is described below:

Massets: Property value of all the agent's assets that are not currently mortgaged. Each property can be mortgaged with the bank for cash. We can buy back the property from the bank for the mortgaged amount plus interest on the mortgage.

Rs: short term expected gain in funds computed as the difference between expected rent the agent will get for the properties that it owns in state s and the expected rent it would owe to other players based on current ownership of properties over the next k turns. The expectations are computed over the probabilities of each player landing in a particular position in the next k turns. This is akin to a rollout with the strong relaxation (assumption) that no more properties will be bought or developed. To be specific, let G be the set of all agents, $\mathrm{g}1$ be our agent, P(g) denote the properties owned by agent g, $\mathrm{r}(\mathrm{p})$ denote the rent of property p, and $\mathrm{P}\ \mathrm{r}(\mathrm{g}, \mathrm{p}, \mathrm{k})$ denote the probability that an agent g will land on a property p in the k^{th} turn from state s, then Rs is computed as $\mathrm{Rs} = \sum_{k} \sum_{g \in G - g_1} \left[\sum_{p \in P_{g1}} Pr(g,p,k) * r(p) - \sum_{p \in P(g)} Pr(g1,p,k) * r(p) \right]$

Rl: the expected long term change in funds. The computation for this term is similar to Rs, except that the probability of an agent landing on a property is assumed to be uniform over all properties. Note that the long term gain is calculated for k full loops/passes around the board (not turns). The value of k for both Rs and Rl was taken as 5.

Mmonopoly: A monopoly gain term is computed to incorporate the monetary benefit our agent would get for monopolizing and improving all properties of the same color. The purpose of this term is to drive our agent towards taking actions that would let it gain a monopoly on a color and subsequently perform maximal improvements on its properties. To compute Mmonopoly, we start by calculating the expected funds, F, our agent would have after going around the board (full loop) k times ($\mathrm{k} = 5$ in our implementation) from its current position as F $=$ cash possessed $+\ \mathrm{k} *$ go increment $+$ Rl where the last term Rl is also computed for k loops around the board. Now let C$(\mathrm{g}1)$ be the set of all colors such that our agent owns at least one property of that color. Then for each $\mathrm{c} \in$ C$(\mathrm{g}1)$, we compute the combined potential rent for that color (Rc) that our agent will get from all the properties of that color if it spends all of F in buying all the properties of color c followed by im- proving each of the properties as much as possible with the remaining amount from F. This potential rent value is then scaled down based on how many properties the agent actually possess (currently) for that color. For

example, if we own 1 out of 3 blue properties, then the potential value from that color should be much less than if we own 2 out of 3 red properties. The scaled potential value $R^s c$ is computed as $R^s c = Rc\ /2^{P(c)-P(g1)}$ where P(c) is the total number of properties of color c. We use an exponential function in the denominator to value color sets that are closer to completion significantly more than others. Since the set size can change as part of game novelties, we think this is prudent. Finally, the monopoly component of the state evaluation, Mmonopoly, is simply computed as the maximum $R^s c$ over all the $\mathrm{c} \in C(g1)$. What this monopoly term does for the agent is to allow it to eschew buying new or bidding for proper- ties if that amount can be used to complete and develop a monopoly.

Another complication for the agent is that it must try to avoid bankruptcy in the face of a lot of stochasticity from the game. So even if the expected value of a policy is high, if it risks bankruptcy then a lesser-value policy that minimizes the risk of bankruptcy might be preferred. Concretely, at any state s, the agent considers if each possible move from the set of possible moves $\mathrm{m} \in M$ with cost $C(\mathrm{m})$ satisfies the following conditions:

Condition 1: cashcurrent $+$ Rnext $-$ $C(\mathrm{m}) \geq \mathrm{cash}$min where cashcurrent is the current amount of money our agent possess, Rnext is the expected change in cash due to rent after one round, cashmin denotes the absolute minimum amount needed to protect against bankruptcy. This covers misfortunes from the Chance and Community chest cards that the agent might draw.

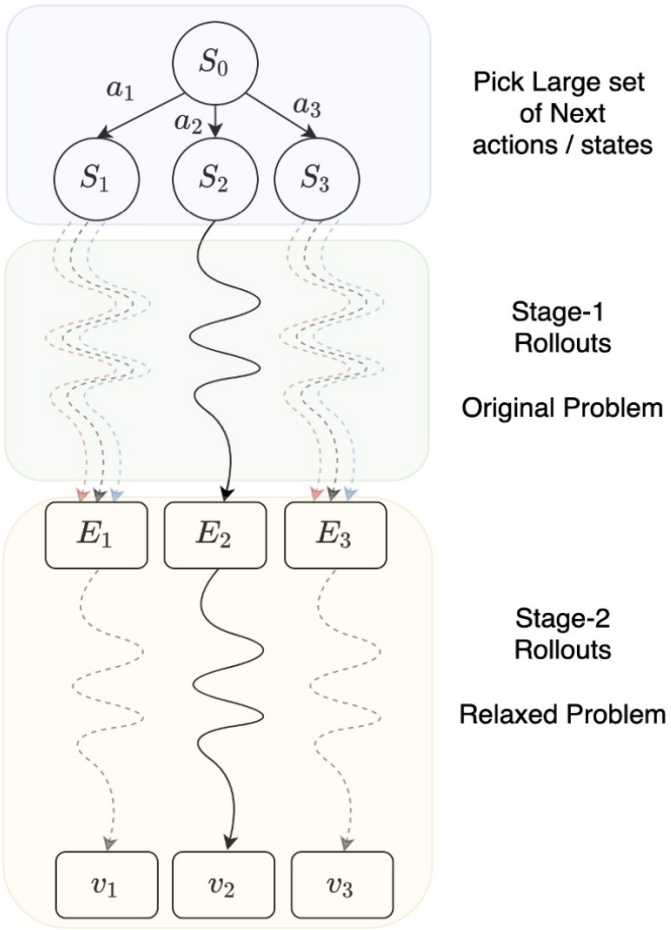
Condition 2: cashcurrent $+$ Rowed $+$ worthscaled $-$ $C(\mathrm{m})$ $-$ Rworst > 0 where Rowed is the expected income from charging rent that our agent will get in the next round, worthscaled is some mortgage value of all properties our agent owns, and Rworst is the maximum possible rent our agent could be charged in the next round. This protects against bankruptcy from landing on an adversary's property. Both the above conditions were used to prevent the agent from aggressively spending its cash and going bankrupt. Once we have pruned the moves in M, our agent simply chooses the move such that $\mathrm{m} = \arg\max \mathrm{m}\ V\ (\mathrm{s}'\mathrm{m})$ where $\mathrm{s}'\mathrm{m}$ is the next state after simulating the move m.

7.1 Novelty Detection and adaptation

To perform well in the SAIL-ON evaluation, our agent needs to detect the novelties introduced and adapt state evaluation accordingly. The novelties that the agent was tested on were hidden. To adapt, we maintain knowledge of the expected values for game-board attributes like property rent, dice out- come likelihood, etc. The evaluation function is parameterized with such attributes and is updated once a change is detected. Some of these values are provided directly as the state information and thus we keep track of the current values of these attributes by observing the state. For other at- tributes, the agent needs to observe the outcome of certain actions (like selling a property) to infer how the relevant at- tributes changed. In attribute value changes, we

also detect novelties related to dice. This includes addition/deletion of a die, additional sides added to the dice, and the distribution of rolling any number on each die. The first two are inferred by observing the dice rolls in the game. For the last one, we model the distribution of rolling any number as Dirichlet distribution and use MAP estimates to update this distribution. This updated dice distribution is then used to compute the probability function $\Pr$, as used to compute Rs.

As mentioned, accommodation used a 2-stage rollout based controller for efficient online action selection. It involved a (a) model-based controller to account for novelty and (b) a parameterized simulator module updated at runtime with detected novelties



Stage - 1 rollout: In the stage-1 roll-out we rolled out the original problem, but kept it short to avoid high branching factor

Stage -2 rollout: In the stage-2 rollout we rolled out a relaxed domain (such as just going through the board, and only paying and collecting rents, but not indulging in any other activities) which had a low computational cost and allowed for long horizon value computation. Some of the strategies for problem relaxations that we experimented with are domain independent and could be used in other domains are: limiting adversarial actions, and removing stochasticity from the stochastic dynamics.

7.2 Evaluation and Results

Our agent was evaluated against other teams in the SAIL-ON program by the USC TA1 team. Evaluation consists of multiple trails, where each trial consists of 100 games of monopoly against 3 baseline agents. The baseline adversarial agents were programmed by the evaluation team to serve as the competition baseline. The behavior

of the baseline agent is described in (Haliem et al. 2021) as the ”simple baseline agent” in that work. During each trial, a novelty is injected during one of the 100 games, and kept for the remaining games.

The following metrics help compare agent performance: (1) Pre-Novelty Win-Percentage (PNWP): The ratio of games won before any added novelty. (2) Novelty Detection Accuracy (NDA): This is the percentage of trials in which the novelty was correctly detected, and without a false positive before the novelty was added. (3) Novelty Reaction Performance (NRP): To compute this, the win ratio of our agent after the novelty was added is divided by the win ratio of the baseline agent before the novelty was added. These measures were not defined by us, but by the evaluation group, and directed by discussions in the SAIL-ON pro- gram. For every measure, our agent was evaluated with different classes of novelties, these are: Class Novelties (CN) such as new classes of objects like property classes, or new classes of actions; Attribute Novelties (AN) such as changes in the mortgage rate, rent costs; and Representation Novelties (RN) such as changes to the position of properties, and the color sets to which they belonged. Within each type of novelty, the evaluators further classified them into easy, medium and hard. As mentioned, we do not have more details about the specific type and distribution of novelties that our agent was evaluated on, as this information is currently hidden from us to evaluate agent adaptation better.

The results shown in the table below compare our performance and the performance of the best competitor for NDA and Win percentages for the different novelty settings. With respect to the win ratio of our agent, our pre-novelty win ratio (PNWR) was 76.48%, i.e. the 3 other base- line agents combined only won less than a quarter of the games when there was no novelty injected. This win rate represents the efficacy of our agent design/playing algorithm for the standard Monopoly game. In comparison the win rate for the next best team was 63.61%. Our agent performs better before novelty was added to the game, and also in most set- tings after novelty was added to the game. The only setting in which our agent did not get the best result was for ”NRP- CN-hard” (Novelty Reaction performance for hard class- novelties). This reflects our agent’s ability to accurately capture both the short and long-term effects of actions, as well as, how well it can adjust for novelties in the game while making decisions by modifying the evaluation function during the gameplay. The evaluators also ran a special test to see how well our agent performs against another instance of our agent, and a baseline agent. The two instances of our agent won 40.75 and 39.43% of the games on average (over many trials).

Novelty type	NDA(%)	Best competitor NDA(%)	Win-rate (%)	Best competitor win rate (%)
None(PNWP)	-	-	76.48	63.61
CN-easy	40	20	71.1295	58.448
CN-medium	46.67	20	64.8895	61.867
CN-hard	48.00	24	28.899	43.173
AN-easy	90.32	80	70.473	62.335
AN-medium	90	90	94.432	68.5425
AN-hard	75	20	61.503	46.1825
RN-easy	52	3.33	74.9905	58.448
RN-medium	32	13.33	78.0975	50.8625
RN-hard	32	20	81.5815	67.6585

8. Evaluations

Below is the novelty hierarchy as it was used eventually for evaluations:

Open World Novelty Hierarchy			
Single Entities	Phase 1	1	**Objects:** New classes, attributes, or representations of non-volitional entities.
Single Entities	Phase 2	2	**Agents:** New classes, attributes, or representations of volitional entities.
Single Entities	Phase 2	3	**Actions:** New classes, attributes, or representations of external agent behavior.
Multiple Entities	Phase 2	4	**Relations:** New classes, attributes, or representations of static properties of the relationships between multiple entities.
Multiple Entities	Phase 2	5	**Interactions:** New classes, attributes, or representations of dynamic properties of behaviors impacting multiple entities.
Complex Phenomena	Phase 3	6	**Environments:** New classes, attributes, or representations of elements independent of specific entities.
Complex Phenomena	Phase 3	7	**Goals:** New classes, attributes, or representations of external agent objectives.
Complex Phenomena	Phase 3	8	**Events:** New classes, attributes, or representations of series of state changes that are not each the direct result of volitional action by an external agent or the SAIL-ON agent.

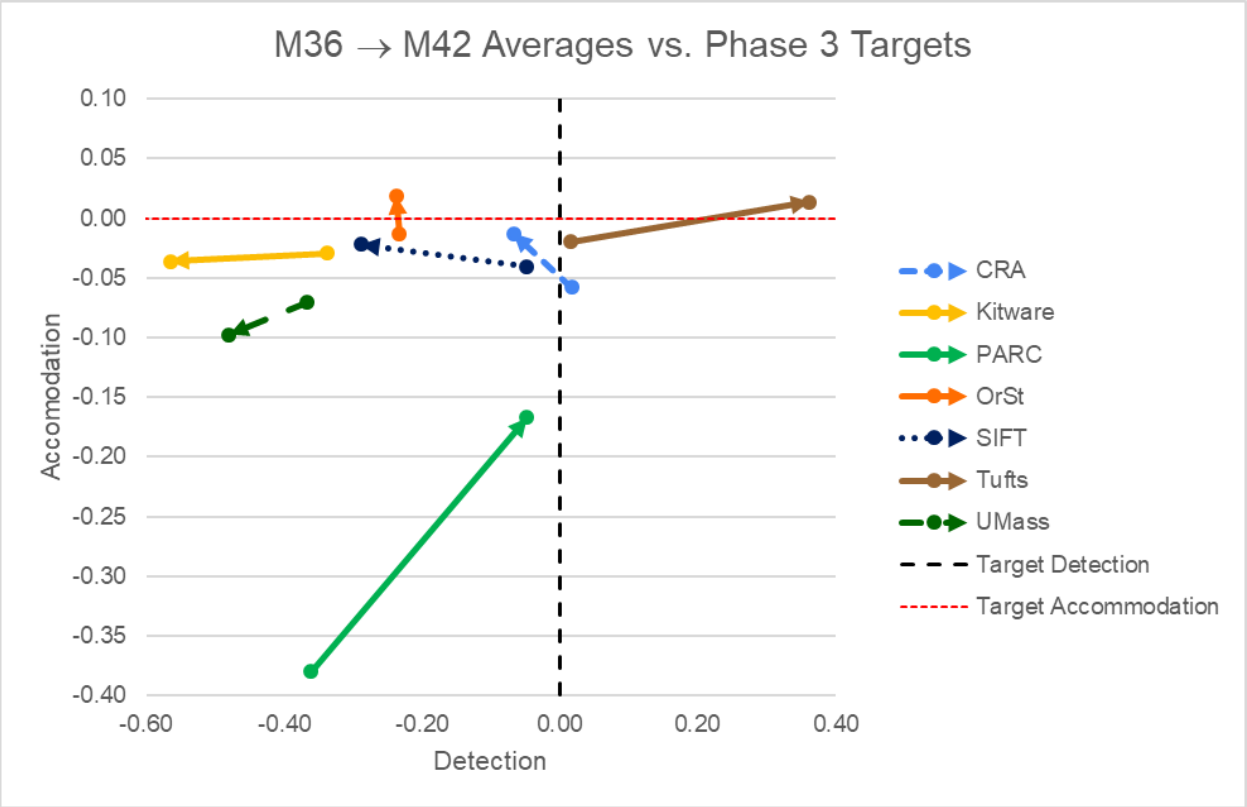
Here we only show the final evaluation results from M42 evaluations for both the Polycraft and the Monopoly domains compared to the other teams that participated in those domains.

	Pogo Stick				Monopoly		
#Trials (Levels 6-8)	270				207 (61,75,71)	225	
TA2	CRA	PARC	SIFT	Tufts	CRA	Tufts	UMass
M1:	0.8	3.3	1.0	0.7	1.3	0.8	7.8
M2: CDT%	87.0%	63.3%	98.1%	92.6%	66.7%	91.6%	36.0%
M2.1: FP%	0.0%	1.5%	0.0%	0.0%	3.4%	2.2%	58.7%
NHP	68.5%		34.8%	65.2%	27.1%	91.1%	32.9%
$ONRP_S$	0.59	0.41	0.73	0.75	1.20	1.27	1.40
$INRP_S$	0.61	0.43	0.71	0.75	1.22	1.27	1.43
$OPTI_S$	71.7%	48.0%	71.3%	74.5%	88.3%	92.6%	96.6%
$APTI_S$	74.1%	54.6%	72.5%	74.3%	86.6%	91.6%	95.6%
$IPTI_S$	74.2%	60.7%	73.6%	77.9%	87.1%	90.9%	96.7%

Exceeds Stretch	17%
Achieved Target	26%
Close (1.0, 2%, 0.1)	16%
Did Not Achieve	40%

As can be seen, our agent achieved or exceeded all measures in the Polycraft domain and achieved or came close to all measures in the Monopoly domain (and note that

these evaluations are only performed on the Phase 3 novelties). Moreover, in Polycraft our agent had the best performance in every measure except for CDT (where SIFT was better). We also had the biggest improvements in performance gains of all teams from the M36 to the M42 evaluations.



We were also the only team that performed a separate evaluation on the visual novelty detection subtask in Polycraft:

method	*M1: Fn_CDT*	*M2: CDT%*	*M2.1: FP%*
Vision @ 36 Months	6.70	27.8%	58.3%
Vision @ 42 Months	**3.44**	**56.6%**	**34.1%**

Results on only visually-detectable novelties (those that generate visually distinct objects). Thanks to Eric Kildebeck and UTD for help.

9. Publications

Feeney, P., Schneider, S., Lymperopoulos, P., Liu, L., Scheutz, M., and Hughes, M.C (2023). "NovelCraft: A Dataset for Novelty Detection and Discovery in Open Worlds". Transactions on Machine Learning Research.

Feeney, Patrick, and Michael C. Hughes (201). "Evaluating the Use of Reconstruction Error for Novelty Localization." ICML 2021 Workshop on Uncertainty & Robustness in Deep Learning.

Gizzi, E., Guaman, M., Lin, W., Sinapov, J. (2021). "A Framework for Creative Problem Solving Through Action Discovery". In Proceedings of RSS 2021 Workshop on Declarative and Neurosymbolic Representations in Robot Learning and Control.

Gizzi, E., Hassan, A., Lin, W., Rhea, K., and Sinapov, J. (2021). "Toward Creative Problem Solving Agents: Action Discovery through Behavior Babbling." In proceedings of the IEEE International Conference on Development and Learning (ICDL), Beijing, China, August 23-26.

Gizzi, E., Nair, L., Chernova, S., and Sinapov, J. (2022). "Creative Problem Solving in Artificially Intelligent Agents: A Survey and Framework". Journal of Artificial Intelligence Research 75.

Goel, S., Shukla, Y., Sarathy, V., Scheutz, M., and Sinapov, J. (2022). "RAPid-Learn: A Framework for Learning to Recover for Handling Novelties in Open-World Environments". In Proceedings of International Conference on Development and Learning (ICDL).

Goel, S., Tatiya, G., Scheutz, M., and Sinapov, J., (2021). "NovelGridworlds: A Benchmark Environment for Detecting and Adapting to Novelties in Open Worlds." In proceedings of the 2021 AAMAS Workshop on Adaptive Learning Agents (ALA).

Gokhale, T., Anirudh, R., Kailkhura, B., Thiagarajan, J.J., Baral, C. and Yang, Y. (2021). "Attribute-Guided Adversarial Training for Robustness to Natural Perturbations". In Proceedings of AAAI.

Gokhale, T., Banerjee, P., Baral, C. and Yang Y. (2020). "MUTANT: A Training Paradigm for Out-of-Distribution Generalization in Visual Question Answering". EMNLP 2020.

Gokhale, T., Chaudhary, A., Banerjee, P., Baral, C. and Yang, Y. (2022). "Semantically Distributed Robust Optimization for Vision-and-Language Inference". Findings of ACL 2022.

Gokhale, T., Mishra, S., Luo, M., Sachdeva, B.S., and Baral, C. (2022). "Generalized but not Robust? Understanding the Effects of Out-of-Domain Generalization Methods". Findings of ACL 2022.

Gopalakrishnan, S., Soni, U., Thai, T., Lymperopoulos, P., Scheutz, M., and Kambhampati, S. (2021). "Integrating Planning, Execution and Monitoring in the presence of Open World

Novelties: Case Study of an Open World Monopoly Solver." In 31st International Conference on Automated Planning and Scheduling.

Guan, L., Verma, M., Guo, S., Zhang, R., and Kambhampati, S. (2021). "Widening the Pipeline in Human-Guided Reinforcement Learning with Explanation and Context-Aware Data Augmentation." Advances in Neural Information Processing Systems 34.

Han, X., Chen, X., & Liu, L.-P. (2021). "GAN Ensemble for Anomaly Detection". Proceedings of the AAAI Conference on Artificial Intelligence, 35(5), 4090-4097.

Han, X., Chen, X., and Liu, L. (2021). "Gan ensemble for anomaly detection." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 5.

Lorang, P., Goel, S., Zips, P., Sinapov, J., and Scheutz, M. (2022). "Speeding-up Continual Learning through Information Gains in Novel Experiences". In Proceedings of 5th Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL) at International Joint Conference of Artificial Intelligence (IJCAI), 2022.

Lymperopoulos, P., Li, Y., and Liu, L. (2022). "Exploiting Variable Correlation with Masked Modeling for Anomaly Detection in Time Series." NeurIPS 2022 Workshop on Robustness in Sequence Modeling.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. (2022). "Cross-Task Generalization via Natural Language Crowdsourcing Instructions". Proceedings of ACL.

Mishra, S., Mitra, A., Varshney, N., Sachdeva, NS, Clark, P., Baral, C., and Kalyan, A. (2022). "NumGLUE: A Suite of Fundamental yet Challenging Mathematical Reasoning Tasks". Proceedings of ACL.

Muhammad, F., Sarathy, V., Tatiya, G., Gyawali, S., Goel, S., Guaman, M., Sinapov, J. and Scheutz, M. (2021). "A Novelty-Centric Agent Architecture for Changing Worlds". In proceedings of the 2021 ACM Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)

Pal, K., and Baral, C. (2021). "Investigating Numeracy Learning Ability of a Text-to-Text Transfer Model". Findings of EMNLP.

Sarathy, V., Kasenberg, D., Goel, S., Sinapov, J. and Scheutz, M. (2021). "SPOTTER: Extending Symbolic Planning Operators through Targeted Reinforcement Learning". In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (pp. 1118-1126).

Shukla, Y., Thierauf C., Hosseini R., Tatiya G., and Sinapov J. (2022). "ACuTE: Automatic Curriculum Transfer from Simple to Complex Environments". In Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

Sreedharan, S., Soni, U., Verma, M., Srivastava, S. and Kambhampati, S. (2022). "Bridging the Gap: Providing Post-Hoc Symbolic Explanations for Sequential Decision-Making Problems with Inscrutable Representations." Proceedings of the 10th International Conference on Learning Representations.

Thai, T., Shen, M., Varshney, N., Gopalakrishnan, S., Soni, U., Scheutz, M., Baral, C. and Sinapov, J. (2022). "An Architecture for Novelty Handling in a Multi-Agent Stochastic Environment: Case Study in Open-World Monopoly." AAAI Spring Symposium.

Thai, T., Soni, U., Verma, M., Gopalakrishnan, S., Shen, M., Garg, M., Kalani, A., Vaidya, N., Kambhampati, S., Varshney, N., Baral, C., Sinapov, J., and Scheutz, M. (2023). "Methods and Mechanisms for Interactive Novelty Handling in Adversarial Environments". Extended abstract, Proceedings of AAMAS.

Varshney, N. and Baral, C. (2023). "Post-Abstention: Towards Reliably Re-Attempting the Abstained Instances in QA". Proceedings of ACL 2023.

Varshney, N., Banerjee, P., Gokhale, T., and Baral, C. (2022). "Unsupervised Natural Language Inference Using PHL Triplet Generation". Findings of ACL.

Varshney, N., Gupta, H., Robertson, E., Liu, B. and Baral, C. (2023). "A Unified Evaluation Framework for Novelty Detection and Accommodation in NLP with an Instantiation in Authorship Attribution". Findings of ACL.

Varshney, N., Mishra, S., and Baral, C. (2022). "Investigating Selective Prediction Approaches Across Several Tasks in IID, OOD, and Adversarial Settings. Findings of ACL.

Varshney, N., Mishra, S., and Baral, C. (2022). "Towards Improving Selective Prediction Ability of NLP Systems". Proceedings of ACL 2022 RepL4NLP Workshop.