Autonomous Causal Generalization

Arash Sheikhlar

Dissertation submitted to the School of Technology, Department of Computer Science at Reykjavík University in partial fulfillmentof the requirements for the degree of Doctor of Philosophy

October 6, 2024

Thesis Committee:

Dr. Kristinn R. Thórisson, Supervisor
Professor, Reykjavík University, Iceland

Dr. Carlos Hernández Corbato, Committee member
Associate professor, Delft University of Technology, Netherlands

Dr. José Hernández-Orallo, Committee member
Professor, Technical University of Valencia, Spain

Dr. Marjan Sirjani, Examiner
Professor, Mälardalen University, Sweden

ISBN Print version 978-9935-539-40-3
ISBN Electronic version 978-9935-539-41-0
ORCID: 0000-0002-0568-075X
Copyright © 2024 Arash Sheikhlar

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/). You may copy and redistribute the material in any medium or format, provide appropriate credit, link to the license and indicate what changes you made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. You may not use the material for commercial purposes. If you remix, transform or build upon the material, you may not distribute the modified material. The images or other third party material in this thesis are included in the book's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the book's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Report on Arash Sheikhlar's Ph.D. Thesis and Defense

"Autonomous Causal Generalization"

Dr. Carlos Hernandez Corbato, Committee Member
Dr. Jose Hernandez-Orallo, Committee Member
Dr. Marjan Sirjani, Examiner

Summary of Scientific and Technical Contributions
The doctoral dissertation under review contributes to the study of artificial intelligence, more specifically the generalization of the experience of an autonomous agent, in the service of self-guided artificial transfer learning in the physical world. Harking back to Alan Turing's ideas about "child machines" that can learn like human babies, the work extends ideas in constructivist AI methodology and autonomous causal knowledge generation, demonstrating new ways of achieving self-guided cumulative learning. It presents a theory and implementation of software systems that can use analogical reasoning, in-situ, to learn about things, events and phenomena that they have not encountered before, without requiring intervention from its designers. The methods presented can be used to engineer new kinds of systems that can handle, in a self-guided way, novelty resulting from under-specified tasks and environments.

The work addresses many challenging topics, from philosophy, psychology and AI, whose shape, definition and interrelatedness are sharpened in the process, including causal relations, generalization, non-axiomatic reasoning, unified ampliative reasoning, goal-driven analogy and even creativity.

The work has several implications for the coming decades in AI research for how uncertainty and predictability may be implemented by autonomous systems in complex environments. Among the notable contributions to this effect are its proof of the non-generalizability of correlational models and an algorithm for autonomous cumulative transfer learning through analogy making.

Evaluation of the dissertation
This is an ambitious thesis, presenting several new ideas and proof-of-concept through a set of implemented demonstrators. Collectively, the results presented represent a substantial step towards truly autonomous systems that can handle exceptions, unpredicted events, and underspecified environments and tasks. The results have already led to several publications in venues such as Proceedings of Artificial General Intelligence and Proceedings of Machine Learning Research.

The thesis is well motivated and the technical content is sound, thoroughly contextualized in both common and not so common theories relevant to the artificial intelligence domain and intelligence research. The candidate is well aware of the relevant state of the art and shows an expert knowledge of the various models he uses and a mastery of the associated methods.

The thesis document is well structured with a good balance of background and original material, as well as detailed discussions and illustrating examples where relevant. On the whole, the thesis is clearly written, self-contained and clear. As a minor remark, it would have strengthened the thesis if the examples used to demonstrate the ideas had been a bit more complex and diverse. The candidate has incorporated the numerous comments and feedback received throughout the thesis writing, from the Thesis Committee, with great aplomb.

In conclusion, Arash Sheikhlar's thesis presents a number of original and significant scientific contributions. In particular, it contributes to the broad and difficult enterprise of cumulative learning, autonomy and non-axiomatic reasoning. The thesis clearly fulfills the requirements for a PhD degree in Computer Science, under the usual international standards.

Thesis Presentation and Defense
The candidate gave an engaging presentation of the work in a way that was informative yet accessible to a general computer science audience, in spite of its technical nature and broad scope. The accessibility of his material in the presentation led to a number of good questions from members of the audience, many who are not familiar with the field. The candidate confidently answered the questions he received in a way that showed deep understanding of the context for the work, its limitations and possible extensions.

Recommendation
Arash Sheikhlar's doctoral dissertation offers several novel and challenging contributions to an ambitious research programme that advances the state of the art in artificial general intelligence and reasoning. The thesis clearly demonstrates without a doubt that the author is able to conduct research that meets the international requirements for a doctoral degree. During the defense, Arash Sheikhlar showed that he can give complex technical presentations that are accessible to a general audience. The answers to the questions he received were good, to the point, and showed scientific maturity. We are very happy to recommend that the candidate be awarded the PhD degree in Computer Science from Reykjavik University.

The thesis work is accepted by this Committee "as is" with no requests for additions or modification.

Reykjavik, August 9, 2024,

Dr. Carlos Hernandez Corbato	**Dr. Jose Hernandez-Orallo**	**Dr. Marjan Sirjani**
		

Autonomous Causal Generalization

Arash Sheikhlar

October 6, 2024

Abstract

For any agent to effectively learn how to achieve its goals via interaction with environments, it must have causal reasoning capabilities. Causal reasoning enables an agent to predict actions' consequences and hypothesize the necessary conditions for taking appropriate actions. Effective mechanisms for generalizing causal knowledge and reasoning lead to more adaptive and autonomous artificial intelligence (AI) systems. While the topic of generalization has extensively been researched in AI over the past few decades, the question of which mechanisms are required to enable causality-based agents to leverage their familiarity with tasks in order to generalize their knowledge when trying to achieve their goals has remained to be addressed. In this thesis, we present a cumulative learning-based generalization mechanism that allows AI agents to use their familiarity with experienced situations to guide their causal hypothesis generation and exploration processes, thereby making their task planning more flexible and well-informed. The mechanism also makes AI systems more flexible and imaginative by enabling them to exploit both good and bad analogies to invent fresh approaches to new tasks. Constructivism and causality serve as the primary foundations of this work's methodology. To validate the proposed mechanism, we have implemented it into a general machine intelligence (GMI) aspiring control system called Autocatalytic Endogenous Reflective Architecture (AERA) and tested it on multiple robot learning tasks as a proof of concept. We have also compared the extended AERA to another GMI aspiring system called Non-Axiomatic Reasoning System (NARS) in terms of their generalizability. The mechanism presented showcases its efficacy through empirical evidence and analytical evaluation.

Keywords: General Machine Intelligence, Artificial general intelligence, Causality, Generalization, Knowledge Representation, Autonomy, Reasoning

Algrím fyrir alhæfingar um orsakasambönd

Arash Sheikhlar

October 6, 2024

Útdráttur

Gervigreind kerfi (e. artificial intelligence systems, AI) sem eiga að ná markmiðum í fjölbreyttu umhverfi, og sem læra sjálf gegnum vélrænt nám, þurfa að tákna orsök og afleiðingu þannig að það nýtist til sjálfstæðrar hugsunar og röksemdarfærslu. Sé sú krafa uppfyllt geta kerfin fært rök fyrir afleiðingum aðgerða sinna, spáð fyrir um nauðsynlegar forstendur þeirra, og séð fyrir hvernig best sé að ná settum markmiðum. Nytsamlegar alhæfingar um orsakatengsl eru forsenda þess að gæða gervigreind kerfi sjálfstæðri aðlögunarhæfni (e. autonomy). Þótt ýmsar aðferðir til framleiðslu alhæfingarreglna hafi verið skoðaðar í gervigreind á undanförnum áratugum er spurningunni enn ósvarað hvernig gervigreind geti nýtt til þess eigin reynslu af orsakasamhengi í fyrri verkefnum. Hér kynnum við aðferð fyrir alhæfingagerð í vélrænu námi sem gerir gervigreind kleift að nota uppsafnaða þekkingu sína til að spinna nýjar orsakatilgátur og sannreyna þær með skipulagðri íhlutun við umhverfið. Aðferðin gerir gervigreind kerfi betur í stakk búin að aðlagast nýjum aðstæðum og nýta betur þekkingu í nýjum verkefnum. Aðferðin gerir gervigreindarkerfi sveigjanlegri og hugmyndaríkari með því að leyfa þeim að nýta bæði góðar og slæmar samlíkingar og myndlíkingar til að finna upp nýjar aðferðir við verkefni. Aðferðin á rætur að rekja til hugsmíðahyggju (e. constructivism) og sjálfstæðrar táknunar orsakasamhengis (e. autonomus generation of causal knowledge). Til að sannreyna aðferðina höfum við innleitt það í kerfi sem er hluti af rannsóknum á alhliða gervigreind (e. general machine intelligence/artificial general intelligence) og kallast Autocatalytic Endogenous Reflective Architecture (AERA). Sýnt er hvernig aðferðin virkar í nokkrum verkefnum. Þá er AERA borin saman við Non-Axiomatic Reasoning System (NARS) hvað varðar alhæfingargetu, þar sem hugmyndafræðilegur grunnur NARS er vel fallinn til slíks samanburðar. Aðferðin og virkni hennar er metin með bæði empírískum og analýtískum hætti.

Efnisorð: Alhliða gervigreind, orsakatengsl, alhæfing, þekkingartáknun, sjálfstæði, röksemdafærsla

Acknowledgements

This research was partially funded by the Icelandic Institute for Intelligent Machines, Cisco Systems, Reykjavik University Research Fund, and the Department of Computer Science at Reykjavik University. With the completion of this thesis, the author gratefully acknowledges the Department of Computer Science at Reykjavik University for their assistance. Special thanks go to Kristinn R. Thórisson for his encouragement, support, and time he dedicated to guiding this thesis in the right direction. Carlos Hernández Corbato, José Hernández-Orallo, Stefán Ólafsson, and Marjan Sirjani also provided helpful comments and feedback. Lastly, the AERA team members Jeff Thompson, Leonard Eberding, and Chloe Schaff deserve recognition for their invaluable feedback.

Contents

List of Figures

List of Abbreviations

Ph.D.	Doctor of Philosophy
AI	Artificial Intelligence
AGI	Artificial General Intelligence
GMI	General Machine Intelligence
AERA	Auto-Catalytic Endogenous Reflective Architecture
NARS	Non-Axiomatic Reasoning System
LHS	Left-hand side
RHS	Right-hand side

Chapter 1

Introduction

1.1 Introduction

Creating machines that can autonomously grow their knowledge by interacting with their environment, much like a human child, has been a goal of Artificial Intelligence (AI) since Turing first proposed the idea (1950). The interaction of such machines calls for taking action, leading to changing the state of environments, which may cause the machines to face unfamiliar situations. So, they must be able to create hypotheses about what their actions might cause in novel but partly familiar situations based on their prior causal knowledge. New hypotheses can be created by generalizing prior experience about the tasks based on how familiar they are, as it is not always practical to perform all possible experiments to learn to solve the tasks. This limitation is often due to the lack of sufficient resources. Therefore, designing effective generalization mechanisms enables the creation of intelligent machines that adapt under insufficient knowledge and resources (P. Wang, 2019).

General-purpose AI systems must be capable of generalizing their knowledge in an autonomous manner (Thórisson, 2021b). In this work, we investigate the requirements for autonomous generalization of causal knowledge and introduce a mechanism that allows AI systems to learn to take appropriate actions in novel situations based on extending their causal understanding. Our methodology is based on the constructivist AI framework (Drescher, 1991; Thórisson, 2012), where an AI system is a learning controller, i.e., an agent, that grows its knowledge from a small "seed" through interaction with its surroundings and learns through feedback loops based on a scalable, non-axiomatic, temporal knowledge representation and ampliative reasoning (unified deduction, abduction, analogy, and induction). In our study, an agent is an embodied AI system interacting with its environment through its I/O devices. It must be autonomous, meaning that it can generalize its

knowledge of its own accord and in line with its goals. This section briefly describes the terms generalization, autonomy, and causality, the fundamental notions of this research.

Generalization
Generalization means extending the scope of an agent's current knowledge so that it fits the requirements of performing new tasks (Sheikhlar et al., 2022). One way of generalizing knowledge is to create general hypotheses from specific experiences, which might be accurate or inaccurate and, thus, must be evaluated in action and revised if necessary. This implies that agents need to validate their created general hypotheses, which must be an essential consideration when designing effective generalization methods for autonomous agents (Sheikhlar and Thórisson, 2024).

Analogical reasoning enables humans to create hypotheses about how to solve novel but partly familiar tasks based on comparing relevant pieces of knowledge with the experienced tasks, choosing the proper knowledge pieces to be transferred, generalizing the knowledge pieces through adjusting/pruning them, and applying the new knowledge pieces through taking relevant actions (Sheikhlar et al., 2022)[1]. In other words, analogy involves analyzing a phenomenon's familiar and unfamiliar aspects and, accordingly, generalizing the relevant pieces of knowledge and taking relevant actions when novel situations/tasks are encountered (Thórisson, 2021b). Similarly, if designed and implemented effectively, the analogy process can guide AI agents' hypothesis generation and subsequent interactions, enabling them to come up with sensible solutions when facing partially familiar and uncertain situations/tasks.

Autonomy
An autonomous agent can perform its tasks automatically without a necessary reliance on external guidance and supervision. Autonomous agents require proper seed knowledge and interaction mechanisms provided at design time that enable them to perform their tasks, adapt themselves to novel circumstances, and bootstrap learning new pieces of knowledge whenever required (Thórisson, 2021b). An important benefit of incorporating autonomy in AI systems is that it allows the AI system designers not to preprogram all required operational knowledge for the systems, which is, in fact, close to impossible for complex tasks and environments, such as physical tasks, as there exist many situations and scenarios that systems may encounter but can not be predicted at design time. Therefore, autonomy must be a

[1]We consider analogy a process that does not necessarily require intentional efforts to make analogies for the sake of analogy. Our concept of analogy relies on comparisons between processes or phenomena as a natural process that occurs in light of achieving a purpose.

fundamental assumption for AI agents that are used in environments with unpredictable features.

Causality
Effective interaction with environments demands the agents' ability to predict how their actions change the state of their surroundings. Additionally, they need to be capable of hypothesizing the most salient conditions and reasons for choosing specific actions. Such capabilities require an agent to create and use causal knowledge. The knowledge about causal relations has other features, such as that it has to do with the physical properties of environments that do not easily alter when the agents try to solve different variations of tasks in the same environments (Pearl, 2009a). Therefore, causality can be a solid foundation for the representation of knowledge and its transfer when the agents deal with tasks that are novel but partly familiar (Sheikhlar et al., 2021). However, although causal knowledge has to do with the physics of environments, an autonomous agent must assume the knowledge to be a defeasible hypothesis that can be modified or even falsified as the agent accumulates contradictory evidence.

1.2 Research questions

A proper formulation of causality provides a knowledge representation that allows for task-independent transfer of knowledge and generalization over task variations (Pearl, 2018). One way to learn causal relations is to intervene in environmental properties (Peters et al., 2017a). Therefore, our first question is:

- **Q1**: How useful and robust is the causal knowledge learned from an agent's interventions for task-independent generalization?

The learned causal knowledge by an agent can be composed of a set of causal relations between events, actions, and environmental properties. Different relations in different contexts may or may not be relevant to the agents' goal achievement, leading us to ask:

- **Q2**: How can an agent generalize the aspects of its causal knowledge relevant to its goal-achievement, allowing it to generate new hypotheses about solving novel tasks efficiently?

According to Wang's (2020) theory of intelligence, reasoning is a fundamental aspect of a general-purpose, autonomous mind. This theory suggests that learning and reasoning are not separate but rather intertwined. In other words, hypothesizing new pieces of knowledge and drawing conclusions are

inherently linked. Also, extending the scope of knowledge calls for utilizing the existing knowledge to generate new hypotheses, which might be correct or incorrect (Thórisson, 2021b). So, our last research question is:

- **Q3**: What are the essential processes for reasoning and learning in a unified manner, allowing an agent to autonomously learn from correctly or incorrectly generated causal hypotheses?

To summarize, a machine seeking autonomous generalization must have causal knowledge representation, reasoning, and generalization mechanisms, allowing it to generalize relevant pieces of knowledge efficiently and perform reasoning and learning in a unified manner to generalize the knowledge effectively.

Research aim

While the concepts of knowledge generalization and transfer learning have been extensively studied in AI, most machine learning (ML) based approaches rely on the heavy involvement of human programmers and their prior knowledge about the tasks, limiting the autonomy of their proposed methods, as studied in (Sheikhlar et al., 2020). Besides, there is no holistic approach towards the design of general-purpose AI systems where interconnections are considered between different cognitive information processing parts such as reasoning, learning, and resource management.

Therefore, the primary aim of this research is to develop a new mechanism for an agent that unifies the following.

- (a) to hypothesize new, generalized conditions for different actions the autonomous agent can take for performing new tasks that have never been encountered before,
- (b) to verify and evaluate the correctness of the generated hypotheses via direct interventions made by the agent's actions while helping the agent to perform the novel tasks and achieve its goals,
- (c) to unify the outcomes of the agent's actions via integrating newly hypothesized pieces of knowledge with the existing knowledge.

The triad of hypothesizing, verification, and unification leads to an efficient knowledge generalization and an effective guided experimentation (a.k.a. exploration), where the general hypotheses created steer the agent's subsequent actions. As such a generalization process aligns with goal achievement, the agent spends fewer resources exploring multiple possibilities.

To summarize, the primary contribution of this research is to introduce a reasoning mechanism by which AI agents can extend the scope of their

causal knowledge to make it more applicable for planning and exploring environments for new pieces of knowledge.

1.3 Research hypotheses

For autonomous learning, knowledge representation must allow an agent to generate and verify its hypotheses through both observations and interventions (Pearl and Mackenzie, 2018). It must also enable the agent to learn the importance of relations and properties in particular situations. Such knowledge representation can effectively generalize beyond task variations. Therefore, we propose our first hypothesis that

- **H1**: A causal knowledge representation can be generalized over variations of tasks with new goals, initial conditions, etc, thus increasing adaptivity when encountering novel tasks compared to correlational knowledge representations.

As knowledge acquisition in partially familiar environments must occur in a gradual manner, a cumulative learning process is called for, which involves automatically and seamlessly integrating new information with existing knowledge (Thórisson and Talbot, 2018). In fact, autonomous generalization should be a side effect of identifying and utilizing familiarities of environments via using analogies while gradually accumulating and testing knowledge through interaction with environments (Thórisson, 2021b). Therefore, our second hypothesis is

- **H2**: A goal-driven analogical reasoning-based knowledge accumulation enables an agent to generalize knowledge while hypothesizing and exploring different solutions for goal achievement. Such guided exploration based on defeasible analogy-based hypotheses is more efficient and effective than unguided random exploration.

A goal-driven analogy-making approach provides an agent with high flexibility in reasoning by creating multiple hypotheses about how to solve novel tasks (Sheikhlar and Thórisson, 2024). Yet, some of these hypotheses might be incorrect due to over-generalization. Thus, our third research hypothesis is

- **H3**: A causality-based system that integrates goal-driven analogy-making with learning from prediction failures can cumulatively learn from correct or incorrect analogies and generalize its knowledge in an effective manner. This calls for ampliative reasoning within a constructivist AI framework, enabling the agent to unify and use different types of reasoning, i.e., deduction, abduction, induction, and analogy.

Ampliative reasoning based on goal-driven analogy-making steers the agent's exploration processes, allowing the agent to unify the agent's new experiences with existing knowledge and gradually improve its hypothesis generation.

1.4 Contribution and impact

Designing an autonomous mechanism that can incrementally improve a causality-based learner's knowledge acquisition and planning over time is a challenge in AI research that still needs to be addressed. This research is one of the first attempts to present a mechanism for analogical reasoning to guide causal knowledge generalization with the aim of improving the planning and exploration for agents.

The thesis will have several contributions to the fields of intervention-based causal AI, autonomous control, and general machine intelligence, detailed as follows:

- **C1**: We propose an intervention-based causal discovery algorithm for learning an invariant model and evaluate the model's robustness in solving new tasks (Sheikhlar et al., 2021). We study the limitations of such invariant models and why more comprehensive knowledge representations are called for.

- **C2**: We present a theory that relates an agent's familiarity with situations to the predictability of intervention outcomes in those situations. The theory is explained in Section 5.2 and in detail in (Sheikhlar et al., 2020).

- **C3**: We design and implement a goal-driven analogy mechanism based on the notion of familiarity, integrated with a causality-based general machine intelligence (GMI) aspiring control architecture, Autocatalytic Endogenous Reflective Architecture (AERA) (Nivel, Thórisson, Dindo, et al., 2013). The mechanism extends AERA's abductive and inductive reasoning capabilities, allowing it to generalize its causal models, identify the significance of environmental properties, and generate and try multiple plans for goal achievement. The mechanism is described in Section 5.5 and partly by papers (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022).

- **C4**: We evaluate the proposed mechanism in terms of extending AERA's reasoning capabilities by testing it with a few motor skills learning tasks within a robot simulation environment and comparing its generalization features with those of another GMI-aspiring system. The evaluations are described by Sections 7.4 and 7.5 and partly by paper (Sheikhlar and Thórisson, 2024).

Autonomous causal generalization by cumulative learning and analogy-making is a significant contribution to various practical fields, including autonomous control, cognitive robotics, and any field that uses AI systems within partially unknown environments. More specifically, it becomes useful for tasks where causal reasoning-based AI agents cannot be fully pre-trained due to unforeseen conditions that might occur within the environments they interact with. This approach provides flexibility when facing novelty, an inherent property of complex physical tasks. The introduced mechanism guides an agent towards pruning out irrelevant pieces of knowledge through interaction and learning relevant pieces over time. The mechanism can be a foundation for guided exploration for agents, presenting multiple solutions with varying confidence levels for task-solving. In summary, this approach is a practical step towards higher levels of autonomy and generality in AI systems, and due to its flexibility, it can be used in a variety of applications.

1.5 Organization of thesis

The thesis has eight chapters. Excluding the introduction, the second chapter covers the theoretical and methodological framework of the thesis, which is briefly discussed in our paper (Sheikhlar and Thórisson, 2024). Chapter three investigates related work, discussing transfer learning methods, planning frameworks, causality literature, and analogical reasoning methods. Some sections of chapter three include content from related work sections of our published papers (Sheikhlar et al., 2021; Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2020; Sheikhlar et al., 2022). Chapter four describes the problem of causal generalization in detail. The section *Causal relations: designer's perspective* of this chapter is adapted from the causal interventions' formalization in (Sheikhlar et al., 2021). Chapter five outlines our proposed autonomous cumulative transfer learning (ACTL) mechanism. The sections related to analogy-making, a key element of the ACTL mechanism, are adapted from our papers (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2020; Sheikhlar et al., 2022). Chapter six describes the details of implementing the ACTL mechanism in AERA. Chapter Seven provides an empirical, analytical, and comparative evaluation of the generalization capabilities of AERA and non-axiomatic reasoning system (NARS). A part of AERA's implementation results and the related evaluations can be found in (Sheikhlar and Thórisson, 2024). Chapter eight concludes the thesis by briefly discussing the results, implications, and future work.

Chapter 2

Theoretical & Methodological Framework

2.1 Introduction

When it comes to Artificial Intelligence (AI), setting forth an explicit methodological framework from the beginning becomes useful, as it leads to the investigation's directedness, coherence, and clarity of concepts being studied. This chapter, therefore, introduces the direction and the methodology of this research, which is built upon constructivism that allows cognitive agents to grow their knowledge through interaction and reasoning (Piaget et al., 1951; Thórisson, 2012).

Disclosure: This chapter provides a more comprehensive description of methodological framework sections in our papers (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022).

2.2 Child machine and constructivism

Turing (1950), around seven decades ago, suggested that the first step toward realizing human-level intelligence in machines is to make them mimic a human child's learning and cognitive processes. He proposed that such a machine could learn from scratch by observing its environment, interacting with it, and writing and modifying its own code. This work follows Turing's suggestion of building machines capable of gaining knowledge from a tiny "seed", based on which the knowledge grows gradually through experiments (Thórisson, 2012).

Piaget (1951), in his constructivist theory, posits that human children, through interaction with their surroundings, build and adjust mental mod-

els that represent their understanding of the world. This knowledge creation and integration that occurs at different stages of cognitive development involves using logic as a tool for organized thinking, creating hypothetical scenarios, and performing reasoning over abstract concepts. According to Piaget, a learning human mind acquires and constructs its knowledge and understanding from experiences; teachers do not "instruct" their students so much as they guide them in their construct and their cognitive development and growth.

Piaget's theory has inspired AI researchers to build new methodologies for general machine intelligence development (Drescher, 1991; Thórisson, 2009). Our work is based on the principles of constructivist AI, introduced by Thórisson (2012), addressing how an agent can effectively grow and accumulate its knowledge via direct interaction. Some of the essential principles of constructivist AI are briefly discussed in this chapter.

2.3 Learning controllers and feedback loops

A learning controller is an agent with a physical or virtual body, e.g., an electro-mechanical robot or a virtual character in a simulation environment, that can interact with its surroundings using its sensors and actuators and learns to control things through manipulating them in its desired way (Thórisson, 2021b). According to Conant and Ashby's theorem (1970), "every good regulator of a system must be a model of that system". A good learning controller processes information to create better models of itself and its environment via taking actions and analyzing the outcomes (Thórisson, 2012). Feedback loops allow agents to take actions based on their current models and evaluate whether and how the actions lead to achieving specific goals, enabling the agent to learn new models or adjust the existing ones based on assessing how accurate the utilized models are in terms of goal achievement (Sheikhlar et al., 2022).

2.4 Causality

Pearl (2009a) considers cause-effect relationships as "autonomous mechanisms" that have strong connections to the physics of environments and can be manipulated separately. He asserts that causality has the following "rungs": 1) association, which has to do with an agent's ability to make predictions based on correlations; 2) interventional reasoning, which is about inferring how the actions change the world; and 3) counterfactual reasoning, which refers to reasoning about imaginary scenarios that the agent has not yet experienced but could have experienced under different conditions (Pearl and Mackenzie, 2018). Thórisson (2016) adds explainability and recreation

ability to the levels of understanding, which its higher levels enable agents to not only learn from interventional reasoning but to utilize counterfactuals to make new guesses about how to deal with situations that might happen in the future. These imply that learning controllers with causal understanding are required to realize higher levels of intelligence in machines.

2.5 Assumption of insufficient knowledge and resources

Wang (2009b) argues that the Assumption of Insufficient Knowledge and Resources (AIKR) must be considered for the design of general-purpose AI systems. According to this assumption, knowledge is assumed to be insufficient, implying that it must not have fixed axioms (P. Wang, 1995). Instead, the knowledge pieces must be considered as hypotheses with confidence values that can alter when the agent collects evidence for the hypotheses. Also, according to AIKR, knowledge is defeasible and can be falsified as contradictory evidence is collected. This implies that an agent dealing with novelties must be able to base its reasoning on insufficient collected information, uncertain hypotheses, and limited resources (e.g., time)(Sheikhlar et al., 2021).

2.6 Ampliative reasoning

Reasoning components used by generally intelligent agents must not be independent of each other, as the components may require to be connected or integrated to accumulate knowledge effectively and efficiently (Thórisson, 2021b). More precisely, a unified reasoning mechanism is called for based on the following principles (Thórisson, 2021b).

- (a) deduction allows an agent to use hypothesized rules/knowledge pieces to reason forward from actions toward changes in the environment's properties,
- (b) abduction allows for reasoning backward from the desired states to potential actions or the conditions under which the actions can be taken,
- (c) analogy can be used for comparing different hypotheses with faced situations/tasks, analyzing the similarities and differences, and hypothesizing new pieces of knowledge to be applied to the novel situations/tasks.
- (d) induction allows the agent to use different learning mechanisms to hypothesize new rules/knowledge pieces.

The above-mentioned reasoning components together refer to *ampliative reasoning*[1] (Thórisson, 2021b; P. Wang, 2013), by which the learning controllers can generalize their knowledge autonomously and learn to focus their attention by excluding unimportant aspects of environments from consideration and taking into account the important ones when taking their actions (Sheikhlar et al., 2020).

2.7 Real-time processing and temporal grounding

It is essential for learning controllers to explicitly incorporate *time* into their knowledge representation, as only in this way will it be possible to calculate the precise timing of the predicted situations and infer the correct sequences of actions that need to be taken during a generated plan. This is a critical feature of real-time controllers, which take into account an agent's ability to accurately predict behaviors using its current knowledge (Sheikhlar et al., 2022).

Real-time processing is also essential for controllers that constantly verify their predictions via their existing knowledge in parallel with pursuing an already generated plan. In other words, while monitoring prediction via deductions, a real-time controller may generate plans via abduction and, at the same time, generate new hypotheses through induction. The reason is that the agent must not ignore monitoring its predictions when achieving its goals or put learning on hold when planning. This means that some (or even all) of abduction, deduction, analogy, and induction components may occur simultaneously and require precise timing. The requirement for real-time control is that the knowledge representation must be based on fine-grained models, as prediction monitoring of complex models is impractical (Nivel and Thórisson, 2013b).

2.8 Scalable knowledge representation

The knowledge representation for a learning controller must handle ampliative reasoning in real time under AIKR. Therefore, it must be compositional and thus scalable to represent complex behaviors/processes via chains and hierarchies of representations of simpler behaviors/processes. A compositional knowledge representation has the potential to be controlled, analyzed, analogized, and scaled entirely or partially. It also eases task-independent

[1] The term 'reasoning' can be ambiguous, as reasoning in different AI systems might involve different information processing paradigms. 'Ampliative reasoning,' however, emphasizes the collective role of deduction, abduction, induction, and analogy, allowing for creating and verifying new hypotheses and knowledge, as explained in (Thórisson, 2021b). Note that the definition and integration of these components can vary in different systems.

reasoning by dividing the causal knowledge of an agent up into smaller, meaningful pieces and then connecting them in multiple different ways for reasoning in various contexts and tasks (Belenchia et al., n.d.). A dynamic, interlinked network of causal relations enables the agent to learn and solve variations of tasks in a more flexible manner (Sheikhlar et al., 2022). Also, such a representation enables a learning controller not only to represent but also to manipulate environments' properties at the desired level of detail. This can only be achieved if the causal relations are captured and formulated as peewee- (small) size models to represent simple relations, which can also be used as building blocks of representing complex relations in different task-environments (Sheikhlar et al., 2022) [2].

2.9 Summary

In this chapter, we went over the key principles of constructivist AI and their necessity for the design and development of agents learning from direct experience and hypothesizing new knowledge via reasoning. Constructivism is based on agents' cognitive growth and interaction with environments, independently selecting, hypothesizing, and using knowledge in novel situations/tasks. Learning agents call for feedback loops, allowing them to learn from their interventions. They also need causality, providing a foundation for reasoning and generalizing causal relations both through and beyond their direct experiences. Such causal reasoning must be under the AIKR; that is, the knowledge has degrees of truth and is falsifiable. The commitment to AIKR allows AI systems to adapt to novel situations where sufficient knowledge is unavailable. Learning agents also require ampliative reasoning, which unifies multiple reasoning types, including deduction, abduction, induction, and analogy, essential for effective and efficient knowledge acquisition and generalization. The knowledge representation and reasoning processes must also consider temporal aspects of environments, allowing for predicting action sequences at the right time and enabling the controllers to monitor, plan, and generate hypotheses simultaneously. Lastly, such a learning controller calls for a scalable and compositional knowledge representation by which an agent can learn and perform complex tasks. Such representation supports the manipulation and comparison of knowledge at multiple levels of detail, allowing for flexible and adaptable learning and planning.

[2] The reason for using the term 'task-environments' instead of only 'tasks' or 'environments' is that, from a human task designer's perspective, it is not always trivial to distinguish the elements/variables/relations of each, especially for agents exploring complex environments for solving complex tasks (Belenchia et al., 2022).

Chapter 3

Related Work

3.1 Introduction

As detailed in the previous chapter, achieving autonomous knowledge transfer and generalization necessitates a holistic approach that unifies multiple reasoning types while taking practical assumptions, e.g., Wang's (2009b) Assumption of Insufficient Knowledge and Resources (AIKR), into account. This chapter delves into the current methods used in machine learning (ML), automated planning, and symbolic AI research that are related to this context while analyzing why each method falls short of meeting the requirements for autonomous causal generalization. The chapter also explores the methods regarding causality and analogical reasoning, as these notions are closely related to the mechanism that will be introduced later in the thesis.

Disclosure: The transfer learning and analogy sections of this chapter are partly adapted from the related work section of our papers (Sheikhlar et al., 2021; Sheikhlar et al., 2020) and (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022), respectively.

3.2 Transfer learning

The closest notion to generalization in ML is often referred to as transfer learning (TL). A well-known example is deep transfer learning (DTL) approaches based on stand-alone Artificial Neural Networks (ANNs), which are primarily domain-dependant and supervised approaches (Tan et al., 2018; Yosinski et al., 2014), making DTL techniques overlook the possibility of self-guided, autonomous transfer of knowledge (Sheikhlar et al., 2020). Yet, we briefly explore a few ANN-based robot planning methods later in another section 3.4. This section, however, mainly explores reinforcement learning

(RL) techniques that allow learning agents to take action and learn from how their actions affect their environment. This section is partly adapted from the related work sections of the papers (Sheikhlar et al., 2021; Sheikhlar et al., 2020).

3.2.1 Reinforcement learning

In reinforcement learning (RL), the rewards guide the agent's subsequent interventions (a.k.a. actions) in environmental variables/properties via feedback, eventually allowing it to find the optimal solutions for goal achievement (Sutton and Barto, 2018). In this context, transfer learning approaches, referred to as reinforcement transfer learning (RTL), allow an RL-based agent to interact with an environment while performing a particular source task, acquire some pieces of knowledge, and then use that knowledge in order to solve other variations of the source task in analogous environments (Taylor and Stone, 2009). Yet, the knowledge pieces transferred between task variations depend on task goals, meaning that the transferred knowledge may become inaccurate when the goals change. RTL methods are usually defined in model-free RL frameworks, where the learned policies mapping states to actions depend on their reward system and not the environments' causal structures. However, model-free RL differs from what is needed for autonomous generalization, as when the tasks, goals, or causal structures change, the inaccurate transferred knowledge usually leads to a slow unlearning and relearning of required policies (Eberding et al., 2020).

Which constructivist AI principles do reinforcement learning algorithms not meet?
In a project we conducted, different model-free RL agents, such as Q-learning and DDQNs, undergo evaluations in terms of their autonomous transfer learning capability (Eberding, Sheikhlar, et al., 2022, 2020). The tests are done within ABA settings, where the agents are trained on a classic control task A, i.e., inverted pendulum task, tested on a similar task B, and retested on the original task A again. The evaluation results have demonstrated that all RL agents struggle with generalizing to phase B and relearning in phase A due to the changes in the causal structure of the phases.

- Model-free RL and Deep-RL algorithms are designed to learn policies that maximize expected rewards, implying that their knowledge representation and learning are task-dependent, as when the goals and environments alter, the existing knowledge becomes inaccurate.
- When RLs are utilized in new tasks, they may catastrophically forget their previously acquired knowledge, hindering the ability to transfer knowledge effectively. We will see later in this thesis that autonomous

knowledge transfer calls for tweaking existing knowledge to meet the requirements of a new situation.

As explained in (Bareinboim et al., 2022), model-free reinforcement learning does not support the second and the third rungs of causal understanding (see Section 2.4). However, model-based RLs are more effective in learning and representing the environment's dynamics (Sheikhlar and Fakharian, 2018). Yet, they are based on restrictive assumptions, such as the prerequisite that all the state variables must be at hand and the dynamics of their environment must remain constant, none of which align with the requirement for knowledge defeasibility (detailed in Section 2.5).

3.3 Logic frameworks

When it comes to explicit knowledge representation, logic provides proper ways to represent properties and relations and reasoning over them. First-order logic (FOL) is a widely used form of logic in AI due to its expressiveness and is a basis for various formalisms that enable reasoning, knowledge representation, and planning (Russell and Norvig, 2016). This section analyzes a FOL-based formalism, known as situation calculus, widely used in agent-based AI and cognitive robotics (Van Harmelen et al., 2008) due to its focus on temporal reasoning about actions and plans.
Then, we discuss the Non-Axiomatic Reasoning System (NARS), which performs multi-layered inferences over logic statements (P. Wang, 1995, 2013). We explain how the NARS framework better fulfills the requirements of constructivist AI, providing sufficient justification for choosing it for comparative evaluations with our proposed mechanism.

3.3.1 Situation calculus

Situation calculus, first introduced by McCarthy (1981), consists of three fundamental concepts: situations, actions, and fluents. Situations represent the complete state of the world at a time. Fluents represent transitions describing the action effects, which can be relational (with boolean values) or functional. Situation calculus is based on the assumption that the sets of possible actions and fluents are specified upfront. This framework allows for quantification over actions and situations, enabling the language to abstractly model dynamical properties and define broad rules. The sequences of actions imply the passage of time, as time is tree-like.

Which constructivist AI principles does situation calculus not meet? Despite being a valuable tool in AI, situation calculus has some limitations that must be considered when choosing it as a reasoning framework.

- **Deterministic world assumption**: Situation calculus is based on the assumption of a deterministic world, where the actions have predictable outcomes under relevant situations. Yet, in real-world applications, not all actions lead to the expected outcomes. It is usually non-trivial to extend situation calculus to handle AIKR, detailed in Section 2.5.

- **Lack of induction support**: As situation calculus does not inherently support learning and inductive reasoning, it usually needs to be integrated with domain-specific ML algorithms. This can be considered a hurdle to effective generalization of rules.

It is important to note that FOL has a special focus on deductive reasoning and thus does not inherently support abduction, which is an essential reasoning component for inferring the potential causes of events and the best actions to take (see the requirement detailed in Section 2.6). Also, autonomous generalization requires dealing with complex behaviors and, thus, needs higher-order objects that describe and control other objects, implying that a learning controller needs a representation of knowledge at different levels of detail (see the requirement detailed in Section 2.8). A logic framework that is based on AIKR and performs inductive reasoning on compound symbolic knowledge representation is NARS.

3.3.2 Non-Axiomatic Reasoning System (NARS)

Non-axiomatic reasoning system (NARS) is a term-based logic framework developed to perform reasoning under AIKR (P. Wang, 1995). NARS assumes that the system must be capable of adapting its insufficient knowledge to new environments based on its experience, selecting and applying the pieces of knowledge supported by collected evidence.

The expressive language of NARS, referred to as Narsese, is built around the concept of statements, representing relations between terms. Each term can represent objects, properties, or events. For instance, a Narsese statement can be represented with $A \rightarrow B \quad \$f, c\$$, where $\rightarrow$ shows an inheritance relation between the subject A and the predicate B, stating that A is a type of B. Also, the semantics allow for handling uncertainty through truth values, which are confidence and frequency, c and f, respectively, quantifying the system's belief in the statement's truth. The frequency and confidence values can be calculated via $f = \frac{w^+}{w} \in [0,1], \quad c = \frac{w}{w+1} \in [0,1]$, where

w^+, w^-, and w denote positive, negative, and total evidence for the statement. The truth value combines f and c, providing a summary of the two via $exp(f, c) = \left(c\left(f - \frac{1}{2}\right) + \frac{1}{2}\right)$ (Hammer, 2021). NARS has deductive, inductive, and abductive reasoning capabilities. Deduction requires two statements $A \rightarrow B$ and $B \rightarrow C$, leading to inferring the statement $A \rightarrow C$. Abduction of the statements $A \rightarrow C$ and $B \rightarrow C$ yields $A \rightarrow B$. Induction of $A \rightarrow B$ and $A \rightarrow C$ leads to $B \rightarrow C$.

How does NARS meet the requirements of constructivism?
We chose the NARS framework for a comparative evaluation (see Section 7.4) due to the features of its knowledge representation and reasoning. Here, we briefly describe learning control, ampliative reasoning, and AIKR features in NARS. Later in this chapter, in Sections 3.5 and 3.6, we discuss if and how NARS meets the requirements for causality, temporal grounding, and analogical reasoning.

- **"Self" in NARS**: The concept of Self realizes agency in NARS and generates relations between the agent and the world it interacts with through *operations* performed by the agent. For example, a Narsese statement representing the sentence "grasp the cube" is represented as $< (*, SELF, cube) \rightarrow \hat{}grasp >$, which may lead to receiving a new statement later, implying that the state of something in the environment has changed. This feature meets the requirement of learning controllers, as detailed in Section 2.3.

- **Assumption of Insufficient Knowledge and Resources (P. Wang, 2009b):** Narsese statements have degrees of truth, revised by the accumulation of experience via the revision rule. This implies that the agent has to start learning and reasoning based on insufficient knowledge. The frequency and confidence values in NARS allow the system to choose more salient hypothesized statements when a task is assigned to a NARS-based agent.

- **Ampliative reasoning:** The use of deductive, inductive, and abductive reasoning capabilities enables NARS to perform ampliative reasoning (see Section 2.6), providing the system the capability of generalizing Narsese statements at multiple representation layers. Note that NARS's induction rule is different than our definition of induction in the next Chapter 4.4. However, NARS uses a multi-layered inference mechanism to generate new hypotheses based on the existing knowledge and received inputs.

A NARS agent's reasoning capabilities under AIKR meet three significant principles of constructivism, described in Sections 2.3, 2.5, and 2.6, making

it a suitable candidate to compare it with the presented mechanism in this thesis.

3.4 Automated planning

Autonomous learning agents must be able to infer sequences of actions to test their hypotheses and achieve their goals. Thus, task planning is integral to ampliative reasoning (see Section 2.6). This section explores a few planning methods, such as Stanford Research Institute Problem Solver (STRIPS) and Planning Domain Definition Language (PDDL), which are commonly used frameworks in classic planning, as well as Markov decision processes (MDPs) utilized for planning under uncertainty.

3.4.1 Classic planning

The STRIPS is a formalism for deterministic representations of planning tasks and is a descendant of first-order logic (Fikes and Nilsson, 1971). The STRIPS's representation language is more restricted than the situation calculus. It comprises predicates (states), objects, goals, and actions; every action taken has a precondition set. The objective of classical planning problems is often to minimize the length of the generated plan (Sabbadin et al., 2020). The first limitation of STRIPS is that it assumes actions to be instantaneous, while they can have duration in reality, and thus, the language needs to be revised for temporal planning. Also, the changes in quantities cannot be represented via STRIPS. However, these problems are solved by the Planning Domain Definition Language (PDDL), extending STRIPS by introducing object types, functions, and numeric properties of actions, effects, and goals (Sabbadin et al., 2020).

What are the limitations of classic planning?
In the following, we briefly go over a few assumptions both STRIPS and PDDL are based on,

- **Closed world assumption**: Classic planners do not adequately take knowledge uncertainty into account. For instance, an agent using STRIPS must always know the outcome of actions upfront, and anything unknown is considered false, implying that the AIKR assumption, detailed in Section 2.5, does not fit such frameworks.

- **Boolean logic**: The binary truth values of predicates and rules make the logic of STRIPS and PDDL less effective, especially when the ranges of the truth values become important for a planning process. As explained in 2.5, a learning controller calls for a range of truth

values that dynamically change over time as the agent accumulates experience.

Some of the mentioned limiting assumptions of classic planning have led to task planning methods focusing on uncertainty representation through probabilistic frameworks.

3.4.2 Probabilistic planning

Probabilistic planning involves assigning probability distributions to action outcomes based on a modeler's familiarity with the uncertainties of a specific domain. Probabilistic planners usually choose the solution paths with actions with higher probability outcomes. As a probabilistic framework, Markov decision processes (MDPs) represent the states, actions, transitions, and rewards as probabilistic properties. RL algorithms can learn policies that maximize the expected rewards in an MDP. However, as detailed in Section 3.2, RLs focus more on learning the policies than on learning the transitions between states or between actions and states.

MDPs can extend classical planning so that predicates and rules have probabilities assigned to them to represent the non-deterministic action outcomes. An example is probabilistic PDDL (PPDDL) frameworks, and there are PPDDL-based cognitive architectures that update the outcome probabilities based on the success or failure of a plan (Jiménez et al., 2013), implying that they do some forms of inductive, abductive, and deductive inferences. However, being a powerful tool for planning problems, probabilistic frameworks have some limitations regarding updating the probability of beliefs when contradictory pieces of evidence are accumulated and their overreliance on the initial condition of the probability distributions (P. Wang, 2004).

3.4.3 Artificial Neural Networks

Different architectures of stand-alone ANNs can be utilized in robotic planning tasks (Sung et al., 2021; J. Wang et al., 2021). For instance, Feedforward Neural Networks (FNNs) have been used for reactive control in mobile robots, where the sensory information is directly mapped to a set of control signals (Velagic et al., 2010). Also, Recurrent Neural Networks (RNNs) can be utilized for motion-planning robotic tasks, because they are capable of capturing sequential and temporal data dependencies (Xu et al., 2019). Other ANN types, such as Convolutional Networks and transformer attention systems, combined with traditional planners, have been used for detecting particular regions in the planning spaces (Johnson et al., 2021). However, *autonomous* retraining of these ANNs is far-fetched, as self-guided learning controllers call for output-feedback loops within their architectures

[1] (see Section 2.3). Also, ANN models make correct predictions only when they are fed sufficient data during the training, while a cognitive system must be able to solve tasks where knowledge and resources are limited (P. Wang, 2006). In addition, cognitive systems must be able to contextualize their knowledge and falsify their hypotheses when the models make incorrect predictions.

3.5 Causality

Causal representations of knowledge are more transparent and verifiable than ML-based models, allowing an agent to perform reasoning over given/identified/hypothesized causal relations and learn to deal with uncertainties (Pearl, 2018; Thórisson et al., 2019a). This section is partly adapted from the related work section of our paper (Sheikhlar et al., 2021).

3.5.1 Observation-based causal inference

Starting by discussing observation-based causal discovery is important, as it is the first rung in Pearl's causality ladder (Pearl and Mackenzie, 2018). Observational casual learning (and inference) is predominantly utilized to tackle a well-known issue in ML, referred to as 'covariate shift,' which arises from changes in the input data distribution during the training process (Shuang and Mohd Pozi, 2024; R. Wang et al., 2022). Learning causal structures allows ML models to adapt to the distribution changes resulting from agent interventions (Bengio et al., 2019). Yet, the application of causal discovery is restricted to models where the notion of time is not usually explicitly included. Yet, there exists a handful of approaches that consider temporal causal influences and discovery, one of which is Granger's (1969) causality. In Granger's causality, however, there are restrictive assumptions regarding the types of model classes and uncertainties that influence the dynamical processes, which do not hold in most real-world processes.

3.5.2 Intervention-based causation

As many other papers also have shown the significance of causal discovery through interventions (c.f. Imbens and Rubin, 2015; Pearl, 2009b; Peters et al., 2017a; Spirtes et al., 2000), interventional reasoning as the second rung of Pearl (2018)'s causal ladder must not be disregarded when designing interactive agents. Besides, a more accurate representation of causal behavior

[1] Note that feedback loop in RNNs only feds the hidden state of the "recurrent cell" (h_t) back into itself. In other words, we have $\hat{y_t} = f(x_t, h_{t-1})$, meaning that the output at the current state does not affect the output at the next state.

must explicitly incorporate temporality (Peters et al., 2017a). In dynamical processes, agents can use interventions to identify cause-effect relationships between a finite set of observables and action types using state-space models (Baumann et al., 2020). Our controller presented in (Sheikhlar et al., 2021) which is also presented in Sections 4.2 and 7.2 uses a modified version of (Baumann et al., 2020) where the presented invariant state space model is revised during the controllers' experiments, meaning that the model can be used for solving the task at any time during the learning process. Yet, our method presented in (Sheikhlar et al., 2021) relies on intervening on all available variables, making it impractical when controllers deal with a large number of variables. As we will see in this thesis, we solve this problem by enhancing the knowledge representation based on the concept of causal relational models (Thórisson and Talbot, 2018).

The mathematical model classes introduced in the causality literature focus more on robust prediction-making than other types of reasoning, such as planning; they can be less straightforward for an agent to utilize ampliative reasoning based on them (see Section 2.6). In addition, causal inference in ML is based on the assumption of causal sufficiency; all causal variables are already at hand and can be observed. In many practical applications, however, confounding variables might negatively influence the causal discovery processes. To develop a versatile learning controller, it is essential to establish a conceptual framework, a new model for inferences, and a programming language that facilitates causal knowledge representation and reasoning.

3.5.3 Causality in NARS

As NARS aligns well with the principles of constructivist AI, we chose NARS for the comparative evaluation and investigated the system's procedural implications (P. Wang, 1995), which is the closest notion to the causal representation of events. Procedural inference in NARS occurs at the higher level of non-axiomatic logic when two events are observed at different times. In other words, if event A occurs before event B, then $A \Rightarrow B$ will be inferred. A special case of this is where operations, e.g., a NARS agent's actions, are considered executable events, that is, $(A, Op) \Rightarrow B$. This means that B will occur when operation Op happens immediately after A occurs. The truth values of this implication can be calculated through the pieces of evidence, w^+ and w^-, where w^+ denotes the number of instances where A occurred, Op was executed, and B occurred subsequently, whereas w^- represents the number of instances where A occurred, Op was executed, but B did not occur afterward. As we will see later, in Chapter 7, temporal implications in NARS do not explicitly incorporate the timings of the events A and B, which may lead to imprecise selection of operation sequences.

3.6 Analogy

Analogical reasoning enables agents to identify the overlaps between their existing knowledge and the patterns they observe or imaginatively generate and, subsequently, to create new hypotheses about how to solve new tasks. The hypotheses can be about the behavior of task entities and distinct events, which can be formulated as causal relations and then, if used correctly, can be a foundation for designing effective exploration mechanisms for agents (Sheikhlar et al., 2022). Most of the current AI systems are not able to make *explicit* analogies that systematically and gradually improve the systems' reasoning, preventing them from effectively being used for transferring knowledge from old experiences to newly encountered situations and goals (Sheikhlar et al., 2022). Here, we explore a few approaches concerning analogies in the contexts of artificial neural networks (ANNs) and symbolic AI. The section is partly adapted from the related work sections of our papers (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022).

3.6.1 Analogy by artificial neural networks

As we previously mentioned, stand-alone ANNs are not suitable options to be utilized by interactive agents that aim to adapt and generalize their knowledge to novel tasks and situations whenever required (see Section 3.4). Yet, we briefly explored the literature of analogy via ANNs.

ANNs performing analogies are utilized in different domains, from language and text similarities (Mikolov et al., 2013) to visual analogies between images and Raven's Progressive Matrix problems (Hu et al., 2021; M. Mitchell, 2021). However, they perform task-specific analogies, are based on non-transparent models, and require an enormous amount of data to be fed to them (Sheikhlar et al., 2022), making them far from the principle of the constructivist AI.

3.6.2 Analogy in symbolic AI

In symbolic AI, analogical reasoning methods rely on explicitly defined symbols and relations usually represented in formal languages. They rely on detecting the matches between symbolic statements given to a system, e.g., structure mapping engine (SME), which considers task-related conditions when it maps the source to the target network of statements (Falkenhainer et al., 1989).

Case-based reasoning (CBR) is a method for solving new tasks that are, of course, partly familiar, where the four steps of retrieval, reuse, revise, and retain could enable an agent to solve the tasks by retrieving and applying analogous cases from its memory and storing how the agent's actions change

the world's attributes and revising the agents' subsequent actions if needed (Aamodt and Plaza, 1994). Although successful in particular domains, the approaches do not assume an agent that has active goals and intends to use analogies for dynamic plan generation.

- The underlying theory of the SME system intentionally disregards distinct properties of task elements and only considers a network of symbolic relations, which may or may not be causal (Gentner and Markman, 1997). However, as will be seen in the thesis, distinct properties of entities might have unknown relations to behaviors that may need to be considered.

- A standard CBR does not learn to modify the preconditions of actions based on their successes and failures. This is essentially rooted in the system's axiomatic learning; that is, no learning/forgetting mechanism exists to disregard/remove knowledge that may not be accurate anymore. As a result, when the number of cases increases, the limited resources might not allow for effective search and retrieval of relevant cases.

Proper analogical reasoning must allow an agent to learn the conditions of actions (through identifying matches between situations, goals, and conditions of known actions) via interaction and modify the conditions when predictions about the outcomes of actions fail.

3.6.3 Analogy in NARS

The term-based logic architecture NARS hypothesizes similarity relations based on not only concepts' shared properties and ontology classes but also systems of relations (P. Wang, 2009a). Analogy relations are formed due to the sameness of the predicates or properties of subjects. For example, the statements $< cube \rightarrow [lightweight] > .$ and $< box \rightarrow [lightweight] >$, which means that the cube and the box are lightweight, yield the inference $< cube \leftrightarrow box >$ meaning that the cube is similar to the box. NARS is considered an appropriate framework for a comparative evaluation in relation to the analogy-making mechanism proposed in the thesis. We will, however, see that the analogy relations are inferred but not utilized for hypothesizing new statements in NARS.

3.7 Summary

In the context of generalization, different approaches in machine learning (ML) and symbolic AI were explored. Artificial Neural Net-based transfer

learning (TL) methods have limitations regarding autonomous generalization, as they require large data and human supervision and do not provide explicit knowledge representations. In model-free reinforcement learning, however, TL approaches exist where policies can be transferred between task variations. Yet, we argued these knowledge components depend heavily on the defined reward systems, making the reinforcement TL methods task-dependant and unscalable. We also discussed the pros and cons of First-Order Logic based frameworks, e.g., situation calculus, which, in its standard form, does not support inductive and abductive reasoning, integral components of ampliative reasoning. Additionally, we explored task planning approaches, both classic and probabilistic frameworks, which do not properly model task uncertainties for agents that learn cumulatively under AIKR, preventing the reasoning agents from learning to revise their knowledge properly or falsify their plans when necessary. Then, NARS was studied and selected as a baseline for comparative evaluation in this thesis due to its features' alignment with some of the key constructivist principles: agency, AIKR, and ampliative reasoning. We also explored the literature on causality and NARS's temporal implications, which allow agents to perform operations/interventions for task-solving. Additionally, we study analogical reasoning frameworks in the contexts of ANNs, symbolic AI and, specifically, NARS's analogy-making. We argued why stand-alone ANNs do not learn from inaccurate analogies and explored symbolic AI frameworks' lack of the potential to revise their knowledge accumulation and analogical reasoning processes according to the agents' experience.

Chapter 4

Problem Description

4.1 Introduction

In this chapter, we describe the problem of causal generalization by defining several aspects of it. Our first step involves defining causal relations and how they can be represented. This requires us to analyze the concept of causality from both an agent's and a task designer's perspectives, as overlaps and discrepancies exist between the actual cause-effect relations and the agent's causal understanding. According to Wang's (2009b) Assumption of Insufficient Knowledge and Resources (AIKR), extracting knowledge about causal relations by an agent is usually accompanied by uncertainty; that is, some confounding properties may not allow an agent to accurately create a causal model for an observed behavior (Agrawal et al., 2023). We then proceed to examine different types of knowledge generalization problems that arise in this context.

Disclosure: This chapter is partly adapted from our papers (Sheikhlar et al., 2021; Sheikhlar and Thórisson, 2024). Section 4.2 describes the problem formulation and causal discovery formalization of (Sheikhlar et al., 2021). Sections 4.3 and 4.4 are partly derived from (Sheikhlar and Thórisson, 2024).

4.2 Causal relations: designer's perspective

We start by formulating causal relations from a designer's perspective based on the principles of task theory (Belenchia et al., 2022). According to Pearl (2009a), a causal relation is a physical attribute of the world that controls how some effect properties can change following some cause events or actions. These effects may or may not be directly controllable through the actions of an agent. In other words, the effects are the results of some operation in the

world, not necessarily those of an agent's actions. In case actions cause the changes, the related commands are produced by a controller, transmitted to the agent's actuators, and applied similarly to local modifications (Thórisson and Talevi, 2024). As the task designer is assumed to know all governing rules and their relevant properties, the tasks can be represented in a deterministic manner. From a Platonic point of view, cause-effect relationships are physically invariant mechanisms (i.e., relations that never change) if all relevant properties are considered (i.e., no unknown confounding properties exist), as Pearl (2009a) points out. This section is adapted from our paper (Sheikhlar et al., 2021), describing the causal formulation of processes and interventions.

Given no agent's actions influence the state of a dynamic process, the representation of transitions in the process only depends on the process's initial state. A ***dynamic process*** is a set of sequential (state) transitions that change the state of environments. A (state) ***transition*** refers to the changes in the values of observable properties due to actions/events. Then, assuming this process has a finite set of properties, a representation of it is (Sheikhlar et al., 2021)

$$\mathbf{V}(T) := f(\mathbf{V}(0)), \tag{4.1}$$

where $\mathbf{V}(\mathbf{T})$ and $\mathbf{V}(\mathbf{0})$ represent the set of properties' *values* at time T and their initial conditions, respectively. However, if we add an interactive agent to this process that takes action by issuing internal commands (given the commands are consistently executed by the taken actions), then the vector $\mathbf{V}$ will contain two sets of values: values of observable properties V and values of commands CMD. Therefore, the representation (4.1) can be reformulated as follows (Sheikhlar et al., 2021).

$$V(T) := g(V(0), CMD(0), ..., CMD(T-1)) \tag{4.2}$$

showing that the commands issued at different times $(CMD(0), ..., CMD(T-1))$, together with the initial conditions/values of the observables $V(0)$, cause the values of the observable to be $V(T)$ at T. To represent the (state) **transitions** between times, we can rewrite the equation (4.2) using the following memory-less, discrete-time transition function (Sheikhlar et al., 2021),

$$V(T) := h(V(T-1), CMD(T-1)) \quad \text{with} \quad V(0) := \lambda \tag{4.3}$$

in which λ denotes the initial values of observables.
With the above formulation, we can describe input intervention, which allows an agent to discover the relationships between commands and observables

(Sheikhlar et al., 2021).

The following **input intervention** types can help an agent identify the structure in dynamic processes under the conditions explained later in Definition 4.1.

- **Input Intervention 1**: setting new initial values for the observable properties $V(0)$ for dynamic processes does not alter function h, meaning that the causal structure of the processes remains the same when their initial conditions change, rooted in the "Principle of Independent Mechanisms" (Peters et al., 2017b, Section 2.1).

- **Input Intervention 2**: an agent can set the values of the commands CMD to apply control inputs, which causes the values of observables V to change but does not cause the causal structure h to change.

A systematic application of the two input intervention types enables an agent to identify an invariant causal structure and learn h. The algorithm in Section 7.2 will be an example of how systematic interventions can be applied to discover causal relations between observables and commands. There, we will also demonstrate the generalizability of this representation, Equation (4.3), beyond the agent goals and initial conditions. Causal discovery through intervention is based on the assumption that the agent has learned some correlations between properties' values, allowing it to know about some relevant properties that might be affected by processes (Sheikhlar et al., 2021). In Section 7.2, we will also practically prove that a correlational model is not generalizable, and therefore, having causal models for task-independent generalization is essential. However, as mentioned, some correlations would enable an agent to gain prior knowledge, based on which it can start exploring its surroundings to discover the causal relations and perform the tasks.

Causal discovery in dynamic tasks
We need a formulation that allows an agent to perform distinct experiments with the aim of causal discovery. To formalize causal discovery in this context, we adapt and use the definition and notations from our paper (Sheikhlar et al., 2021) and the paper (Baumann et al., 2020), as follows.

Definition 4.1 (Baumann et al., 2020; Sheikhlar et al., 2021): Based on the two input interventions introduced, two different types of causal discovery methods can be defined as follows.

> **Causal discovery 1**: Assume i and j are two different observable properties, with v_i and v_j representing their related values,

between which there is a correlation relation. Then, we have (Baumann et al., 2020; Sheikhlar et al., 2021)
($\forall i \;\; v_i \not\perp v_j, \, v_i \rightarrow v_j$ if)

$$\forall\, p \neq i \;\; v_p^k(0) = v_p^{k'}(0) \quad v_i^k(0) \neq v_i^{k'}(0), \\ \forall\, p, T \;\; {cmd^k}_p(T) = {cmd^{k'}}_p(T) \;\Rightarrow\; (v_j^k(T)) \neq ({v^{k'}}_j(T)) \tag{4.4}$$

In simple terms, i causes j if changing the initial state of i in experiment k leads to a change in the value of j in experiment k'. In other words, v_i and v_j are the values of two different properties between which a causal influence exists.

Causal discovery 2: Also, assume a command value cmd_i and the observable j's value v_j are correlated. Then, we have (Baumann et al., 2020; Sheikhlar et al., 2021)

($\forall i \;\; cmd_i \not\perp v_j, \, cmd_i \rightarrow v_j$ if)

$$\forall\, p \;\; {v^k}_p(0) = {v^{k'}}_p(0) \quad \forall\, T, p \neq i \;\; {cmd^k}_i(T) \neq {cmd^{k'}}_i(T) \\ \forall\, T \;\; {cmd^k}_p(T) = {cmd^{k'}}_p(T) \;\Rightarrow\; (v_j^k(T)) \neq ({v^{k'}}_j(T)) \tag{4.5}$$

In simple terms, command i causally influences observable j if changes in cmd_i lead to changes in v_j, assuming the initial conditions are the same in both experiments k and k'. This means that observable property values (a.k.a. variables) and commands can also be correlated, as certain commands may only change the values of specific observable properties while leaving the values of other properties unchanged. We will see an example of this in Section 7.2, wherein in a multi-robot scenario, the command applied to one robot only causally influences the position of that specific robot and not the other robots.

Why does this representation still have generalizability issues?
It is worth noting that causal discovery based on input interventions is based on the assumption of *causal sufficiency*, which refers to the invariance of the causal structure when all observable properties in a dynamic process are at hand. However, when it comes to the agent's modeling perspective, the agent, due to AIKR (see Section 2.5), may not consider all the relevant properties in its modeling process. In high-dimensional observation spaces, an agent must autonomously acquire sufficient knowledge to be able to focus its attention on important properties involved in processes. Also, a single nonlinear forward model h in equation (4.3) takes in all existing observables

and, thus, cannot be easily inverted for backward planning to infer essential commands from observations. Therefore, such centralized large-scale formulation of dynamic processes is not proper for causal modeling in a learning controller dealing with complex tasks due to its lack of scalability (see Section 2.8) and also lack of support for abductive inference (see Section 2.6). Instead, a learning controller calls for a set of fine-grained small causal models relating single causes, e.g., single command value changes, to single effects, i.e., change in the value of single observable properties. If such models also incorporate precondition-postcondition sets, they can be chained or hierarchized if their cause-effect sets match, representing complex processes while allowing the controller to manipulate and learn only the required transitions of the processes. The precondition sets involve the assumed important properties (only a subset of observables and their values) and constraints on them. As some transitions and their preconditions-postconditions can be hypothesized through observations and comparisons, they might prove to be inaccurate in action. In other words, applying input interventions may be needed to validate the generated hypotheses about cause-effect relationships, enabling the agent to know whether its hypotheses/knowledge are accurate. Therefore, the small models must be capable of being falsified and must include confidence values that change via experience. Therefore, we need a new formulation of causal models with the mentioned consideration, as explained in the following section.

4.3 Causal knowledge: a new formulation

Agent input interventions do not change the physical processes but may lead to changes in the knowledge structure, as explained in the previous section. In other words, an agent's causal knowledge, as opposed to physical causal relations, is not invariant because the agent may need to modify or falsify its knowledge as it proves to be inaccurate. The source of inaccuracy might be *confounding properties*, i.e., unconsidered properties, that may lead to inaccurate predictions (Peters et al., 2017b). Prediction failures should make the agent hypothesize the reasons for what might have been the source of failure. We will later see in Section 5.5 how the unconsidered properties can be identified through a new cumulative learning mechanism we introduce. This section provides a detailed description of how causal relational models can be formulated, partly explained in our paper (Sheikhlar and Thórisson, 2024).

Causal relational models

This subsection formulates causal relational models (CRM) introduced by Thórisson and Talbot (2018), a different causal knowledge representation

designed in accordance with the requirements detailed in Chapter 2. The provided definitions are based on the syntax and semantics of the Replicode programming language (Nivel and Thórisson, 2013a).

A CRM is a piece of knowledge that relates a set of cause patterns to the effect patterns. The definition of CRMs calls for precisely defining patterns. We will also define situations and goals, which are significant concepts in forming CRMs. The following definition is adapted from Replicode programming language syntax (Nivel and Thórisson, 2013a) with small changes regarding the naming of the patterns and constraints [1], and categorization of facts.

Definition 4.2: **Patterns** are distinct information structures having two types, facts and constraints, which may contain variables.

> **Facts** are statements with specific time intervals, having two types: observable facts and commands [2] (cf. Nivel and Thórisson, 2013a).
>
> > **Command**: A command is a fact representing an internal operation within the controller leading to taking a physical action and may change the state of the agent's environment. Commands can take one or two arguments as follows.
> >
> > $$(cmd \quad \textbf{CMD}[entity] \quad T1)$$
> >
> > or
> >
> > $$(cmd \quad \textbf{CMD}[entity \quad X] \quad T1)$$
> >
> > where X and $T1$ are variables and can be bound to a numeric value.
> > E.g., $CDM : (cmd \quad \textbf{release}[hand] \quad 100msec)$ means that CMD release the entity *hand* at time $100msec$.
> > E.g., $CDM2 : (cmd \quad \textbf{move}[hand \quad 3] \quad T1)$ means that the command $CMD2$ moves the entity *hand* by 3 units at time $T1$.
>
> > **Observation fact**: An observation fact represents an observed pattern at a time with the following structure:
> >
> > $$(entity \quad property \quad value \quad time)$$

[1] Constraints are called guards in Replicode programming language.

[2] In this study, we focus on these two, as facts may also incorporate other statements, such as instantiated composite states and models, supporting higher-order logic.

where the value can be either numeric or symbolic. Entity and property are symbols, with the difference that entities can be replaced with variables as a result of abstraction while properties cannot. As an example, $F : (ball \quad velocity \quad X \quad 200ms)$ refers to a fact representing a ball with the velocity X at $200ms$. Note that X is a variable.

Constraint: A constraint represents a logical or numerical constraint such as an equation (a.k.a., transition function) or inequality. They specify the ***transition functions***, ***ranges of admissible values***, ***logical conditions*** or ***timing equations*** relating the variables in commands and facts. A constraint on fact F on the above example can be $(X < 10)$, meaning that the ball's velocity cannot be more than 10 units. As another example, $(POS2 = X \cdot \Delta T + POS1)$ represents a transition function pattern showing how positions can change. The instances of timing equations will be shown later in Example 1.

Definition 4.3: A **situation** is all existing observable facts at a time, representing the complete observable state of the process. A situation S can be formally represented as a set of fact patterns with the same occurrence time T:

$$S = \{f_1(T), f_2(T), \ldots, f_k(T)\}$$

where each pattern f_i is an observable fact.

Definition 4.4: A **goal** G is a fact[3] representing the desired state the agent tends to achieve. A goal can either be given or derived by the agent. The derived goals are called ***subgoals*** $SubG$.

Definition 4.5: A **causal relational model** (***CRM***) represents a transition as follows[4]

$$M : [A(T1) \rightarrow B(T2)], \quad 0 \leq cdf \leq 1 \tag{4.6}$$

[3]In Replicode programming language (Nivel and Thórisson, 2013a), goals are objects referring to related goal facts. Here, for simplicity of explanation, we consider them facts.

[4]We will use the notation of propositional logic to communicate the ideas more clearly and not for the actual formalities. There exist fundamental differences between our logic, i.e., ampliative reasoning, and propositional logic. Also, CRMs represent non-boolean physical state transitions and do not necessarily follow the rules of formal logic.

where A is the set of precondition causes (left hand side - LHS) at time T1 and B is the set of postcondition effects at time T2 (right hand side - RHS). cfd denotes the confidence value of M. A special case of CRMs includes a command pattern in its preconditions, with the same timing as the other preconditions, as follows.

$$M : [(P(T1) \wedge CMD(T1)) \rightarrow B(T2)], \quad 0 \leq cdf \leq 1 \tag{4.7}$$

where P represents the preconditions, excluding the command pattern CMD, that hold true at the same time as when *CMD(T1)* is issued. The precondition P is a set of collected facts and constraints representing the conditions of the transition's occurrence. An example of such a CRM is provided as follows.

Example 1: The following CRM represents the moving behavior of a hand. The causal influence occurs via *move* command changing the position of a hand. The CRM is M=[(P $\wedge CMD) \rightarrow B], where$

$$CMD = (cmd \quad \textbf{move}[H \quad \Delta\_POS] \quad T1)$$

representing the command **move** with two variables (a.k.a arguments) H and Δ_POS, meaning that the command moves H by Δ_POS units. The precondition set of patterns P for the above *move* command can be represented as the following set of facts and constraints,

$$P = \begin{pmatrix} (H \quad \text{essence} \quad \text{hand} \quad T1) \wedge \\ (H \quad \text{position} \quad POS1 \quad T1) \wedge \\ (POS2 = POS1 + \Delta\_POS) \wedge \\ (T2 = T1 + 100\text{ms}) \end{pmatrix}$$

where the first two patterns are facts, and the last two patterns are constraints. The facts represent that H has the *essence* of *hand* and is at position $POS1$ at time $T1$. The *essence*, *position*, and *hand* are ontology symbols, whereas H, $POS1$, $POS2$, $T1$, $T2$ and Δ_POS are variables and can be instantiated. The third and fourth patterns are forward equations that calculate numerical values in the postcondition (RHS) patterns based on their values in precondition (LHS) patterns. The constraint ($POS2 = POS1 + \Delta\_POS$) is a *forward transition function* meaning that the next position of the hand $POS2$ depends on the current position $POS1$ and the position change Δ_POS applied by the CMD. The pattern ($T2 = T1 + 100$ms) is a *timing equation*, meaning that the next time frame $T2$ is

calculated by current time $T1$ plus sampling time, which is 100ms here. After the *move* command is applied, we will have the following post-condition B.

$$B = \begin{pmatrix} (H \quad \text{position} \quad POS2 \quad T2) \wedge \\ (POS2 \neq POS1) \wedge \\ (\Delta\_POS = POS2 - POS1) \wedge \\ (T1 = T2 - 100\text{ms}) \end{pmatrix}$$

where the first pattern shows the fact representing the hand's new position $POS2$ at a later time $T2$. The constraint ($POS2 \neq POS1$) shows a condition that the first and second positions of H must not be the same. The third and fourth patterns are equations for representing *backward transition function* and the calculation of initial time $T1$ from $T2$ backward in time. M has a confidence value between 0 and 1.

The following describes how CRMs, as fundamental knowledge representation components, satisfy the principles of constructivism.

- **Learning control**: CRMs enable learning controllers to use their I/O devices to interact with environments by issuing commands (and taking actions), and then learning from the subsequent changes in the values of observable properties. Observations about environments are captured in the form of facts occurring at different times. Interventions occur through commands. CRMs allow an agent to issue a command and learn/test tempro-causal relations between facts occurring at consecutive times.

- **Causality**: A *CRM* is a ***model*** that represents a causal influence from a set of preconditions to postconditions. The preconditions, including a command, trigger a change in a later situation where postconditions are observed. In other words, CRMs represent physical state transitions of properties, in which a command leads to taking an action, which itself causes a change in the value of a property. CRMs can be used for causal reasoning by allowing the agent to predict how the actions influence the state of observables and/or, inversely, inferring the best plausible commands or conditions under which the commands need to be issued to achieve certain goals.

- **Assumption of insufficient knowledge and resources (P. Wang, 2009b)**: The AIKR implies that CRMs are subjective non-axiomatic hypotheses rather than objective axiomatic ground truth, meaning that CRMs's truth degrees can dynamically alter when the agent uses them in action and thus, collects evidence for them (Sheikhlar and Thórisson, 2024). The degrees of truth, referred to as *confidence* in

this thesis, can be updated as evidence is accumulated based on the ratio of the collected success count sc of the CRM to the collected total success plus failure count sfc, that is, $cfd = \frac{sc}{sfc}$, with the constraint that $0 \leq cfd \leq 1$ (Sheikhlar and Thórisson, 2024).

  - Confidence value reflects the non-axiomatic falsifiable nature of knowledge, which can change by experience.
  - Confidence value will later be connected to the notion of familiarity introduced in Section 5.4, which specifies the confidence of the CRMs induced via analogy-making.

  The confidence values of CRMs change as the agent accumulates evidence for them, allowing the agent to choose CRMs/solutions with higher confidence.

- **Ampliative reasoning**: CRMs can be chained in the forward and backward directions, realizing deduction and abduction respectively when CRMs match situations and goals. When a situation matches the preconditions of a CRM, its postconditions are instantiated, meaning that a property's value can be predicted at a later time. If the predicted change is a fact with symbolic value, there will be no transition functions in the pre- and post-condition patterns of the CRM (see an instance of this in Example 2 in this chapter). Otherwise, there will be transition functions such as the ones shown in Example 1, i.e., $(POS2 = POS1 + \Delta\_POS)$. Note that timing equations are always included in both cases, as the change in values of the related properties occurs in time. Generally, the forward transition function patterns have the following structure.

$$\begin{aligned} v_i(T) &:= a_k v_i(T-1) + b_k cmd_j(T-1)) \\ v_i(0) &:= \beta_0 \end{aligned} \tag{4.8}$$

  with $v_i \in \boldsymbol{V} \in \mathbb{R}^n$, $cmd_j \in \boldsymbol{CMD} \in \mathbb{R}^m$, in which the parameters n and m represent the number of all existing observable properties $\boldsymbol{V}$ and agent's commands $\boldsymbol{CMD}$, respectively. Note that v_i is the value of i_{th} observable, and cmd_j is the value of j_{th} command sent through the I/O device. There are also timing equations and variables in the pre- and post-condition patterns, regardless of whether the changed property's value is boolean or not.

  On the other hand, backward chaining occurs for goal achievement. If a goal fact matches the postcondition patterns of a CRM, the preconditions are instantiated, leading to the creation of other goals whose achievement allows for achieving higher-level goals. If the matched goal fact is a boolean relation, there will be no backward transition

functions. Otherwise, there will be transition functions such as the ones shown in Example 1, i.e., $(\Delta\_POS = POS2 - POS1)$, which are the inverse of the forward transition functions mentioned above. The backward transition functions are, therefore, as follows.

$$\begin{aligned} cmd_i(T) &:= (1-a)/b(v_i(T+1) - v_i(T)) \\ v_i(0) &:= \beta_0 \end{aligned} \tag{4.9}$$

Note: The linear modeling of transitions is based on *piecewise linearization*, which approximates complex nonlinear transitions with a set of linear transitions, each with a range of permissible values (Nivel and Thórisson, 2013a).

- **Scalability**: CRMs support hierarchization and chaining.
  - *Hierarchization*: In hierarchical representations, CRMs representing simple behaviors can be reused by other CRMs representing more complex behaviors. For instance, let CRMs be $M_1 : [A \rightarrow B]$ and $M_2 : [C \rightarrow D]$. Then, M_2, as a complex CRM, can reuse M_1 on its preconditions meaning that

    $$M_2 : [[A \rightarrow B] \rightarrow D]$$

    if $A \subset C$, that is, A's patterns are a subset of C's patterns.
  - *Chaining*: Chaining of CRMs allows for generating plans (a.k.a. solutions) and performing forward and backward reasoning. For instance, let models be $M_2 : [C \rightarrow D]$, and $M_3 : [E \rightarrow F]$. Then, in both forward chaining, $M_2 \rightarrow M3$, and backward chaining, $M_2 \leftarrow M3$, we must have $D \subset E$ and not the other way around. This is because $M2$ and $M3$ are initially linked through backward chaining, meaning $M3$ will chain to $M2$ if patterns of E match D.
    Note: Postconditions (RHS) of CRMs have *one single fact pattern* as opposed to preconditions. The reason is that the fact pattern on the RHS represents one single change in the value of an entity's property. Therefore, in the above example, if one of the patterns of E matches D, then the two CRMs match. The following example explains the chaining process.

Example 2: Assume we have two CRMs: M2 and M3, where M2 represents a hand grabbing a cube behavior and M3 represents the cube's releasing behavior. M2 has a command *grab* on its LHS patterns represented as

$$CDM2 = (cmd \quad \mathbf{grab}[H] \quad T1)$$

meaning that *grab* command is applied on H (representing a hand) at time $T1$. The precondition patterns are as follows:

$$P2 = \begin{pmatrix} (H \quad \text{essence} \quad \text{hand} \quad T1) \wedge \\ (H \quad \text{holding} \quad [\ ] \quad T1) \wedge \\ (C \quad \text{essence} \quad \text{cube} \quad T1) \wedge \\ (T2 = T1 + 100\text{ms}) \end{pmatrix}$$

where $(H \quad \text{holding} \quad [\ ] \quad T1)$ means that the hand is holding nothing (the hand is empty). Then, the post-conditions (RHS patterns) of M2 are as follows:

$$B2 = \begin{pmatrix} (H \quad \text{holding} \quad [\text{C}] \quad T2) \wedge \\ (T1 = T2 - 100\text{ms}) \end{pmatrix}$$

meaning that the command $CMD2$ under conditions $P2$ will lead to the hand holding the cube represented as $(H \quad \text{holding} \quad [\text{C}] \quad T2)$ at a later time $T2$. The fact $(H \quad \text{holding} \quad [\text{C}] \quad T2)$ has a symbolic value, and thus, the CRM does not need to have a transition function, whereas the timing equations are present.

$M3$, on the other hand, releases the cube via the following command

$$CDM3 = (cmd \quad \textbf{release}[H] \quad T1)$$

meaning that *release* command is applied on H at time $T1$. The precondition (LHS) and post-condition (RHS) patterns of M3 are as follows

$$P3 = \begin{pmatrix} (H \quad \text{essence} \quad \text{hand} \quad T1) \wedge \\ (H \quad \text{holding} \quad [\text{C}] \quad T1) \wedge \\ (C \quad \text{essence} \quad \text{cube} \quad T1) \wedge \\ (T2 = T1 + 100\text{ms}) \end{pmatrix}$$

$$B3 = \begin{pmatrix} (H \quad \text{holding} \quad [\ ] \quad T2) \wedge \\ (T1 = T2 - 100\text{ms}) \end{pmatrix}$$

meaning that *release* command changes the state of the hand from holding the cube $(H \quad \text{holding} \quad [\text{C}] \quad T1)$ to hand holding nothing $(H \quad \text{holding} \quad [\ ] \quad T2)$. Then, M2 and M3 can be chained via abduction, as the fact on $B2$ matches a fact on $P3$, i.e., $(H \quad \text{holding} \quad [\text{C}] \quad T1)$. In other words, for grabbing and then releasing a cube, the models $M2$ and $M3$ can be chained and used.

The causal-chaining ability and the potential to build hierarchies of causal relations provide an agent with depth and breadth of knowledge regarding a phenomenon. Therefore, CRMs provide scalable knowledge representation, realizing reasoning on multiple levels of detail.

- **Real-time and temporal grounding**: CRMs contain explicit time variables as a part of their LHS and RHS patterns, e.g., in their facts, commands, and timing equations. They also use transition functions (4.8) and (4.9) to represent properties' state transitions. These features allow for real-time control through continuous reality-checking and precisely scheduled planning.

Reasoning mechanisms that use CRMs allow for designing a context-aware agent with a reflective control capability, meaning that the agents can learn, inspect, verify, and modify multiple CRMs according to their needs (Thórisson and Talbot, 2018).

4.4 Causal Generalization

For learning agents performing tasks in partially known physical environments, it is important to generalize behaviors effectively and efficiently, allowing the agent to create reasonable plans and infer correct sequences of actions. Causal generalization happens across multiple levels, progressing from basic abstraction to knowledge pruning and induction (Sheikhlar and Thórisson, 2024). This section is an extended version of the causal generalization section in (Sheikhlar and Thórisson, 2024).

4.4.1 Abstraction

In this thesis, abstraction is considered a basic generalization problem, where an AI system must use the concept of *variables* to represent the observed or stored data in such a way that they match a broad spectrum of tasks and scenarios (Sheikhlar and Thórisson, 2024). According to Mitchell (1997), any instance of a specific hypothesis satisfies a more general hypothesis. With small alterations to Mitchell's definition, the following definition can be derived.

Definition 4.6: If two patterns exist, p_1 and p_2, we can say that p_2 is more general than or equal to p_1 only if any instance that satisfies p_1 also satisfies p_2. But, not all instances of p_2 satisfy p_1. In other words, let $I(p)$ be the set of instances satisfying a pattern p. Then, p_2 is more general than p_1 can be written as [5]:

$$p_2 \geq p_1 \iff I(p_1) \subseteq I(p_2) \tag{4.10}$$

where $\geq$ represents the *more general than* relationship in this context.

[5] This definition is a modified form of *restriction*'s definition presented by Harmelen et al., (2008).

Then, abstraction can be formulated as a function $\alpha(p_1) \rightarrow p_2$, where $p_2 \geq p_1$. To apply the abstraction process, the values of the known properties and/or the known entities having those properties can be replaced with variables in information structures whenever required.

Example 3: Assume the following facts $p_1 = (ball \quad shape \quad sphere \quad T1)$ and $p_2 = (X \quad shape \quad sphere \quad T1)$, where *ball* is an entity, *shape* is the property, and *sphere* is the value. Then, $p_2 \geq p_1$, as p_2 is partly abstracted from p_1 by replacing the entity *ball* with variable X. In other words, p_2 matches to any entity with the shape *sphere*.

Based on the above definitions, the agent must create more general patterns via abstraction as needed. The abstraction of LHS and RHS patterns of CRMs allows the agent to apply the same CRMs for a wider range of values and task entities for other tasks.

4.4.2 Selective attention

In his situation calculus, McCarthy (1986) defines a situation as "the complete state of the universe at an instance of time." It is impossible, however, to consider all observable properties in the universe at once. Humans typically represent situations as abstracted **subsets** of patterns that can be observed simultaneously. A cognitive process known as *selective attention* (Cherry, 2020) helps make this happen, and therefore we call this form of generalization *selective attention*. Selective attention leads to an initial pruning of irrelevant observable facts when modeling an observed transition when no models for the transition exist.

In the context of CRMs, selective attention calls for selecting a subset of the observation facts from a situation and excluding the rest of them to form preconditions for CRMs. One way of creating an initial precondition set is introduced by Nivel et al. (2013a), where the observation facts are selected that match the entities, properties, and values directly involved in transitions. This may, of course, lead to over-specification of CRMs, as not all the properties and values are important in a transition. We will later see in Chapter 7 how our introduced mechanism prunes the irrelevant facts via knowledge pruning (introduced in Section 4.4.4) from this initial set of preconditions and dynamically adjusts the selective attention process. More formally, selective attention can be defined as the following.

Definition 4.8 (Sheikhlar and Thórisson, 2024): Selective attention A is a function that identifies the observation facts in a situation S that need to be considered for modeling transition T. A ***transition*** is a change in the value of a property due to an action taken as a result of an issued command. Formally, $A(S,T) \rightarrow P_{fact}$ in which $P_{fact} \subseteq S$ (Sheikhlar and Thórisson,

2024), meaning that function A takes in situation facts S and transition T and selects a subset of S, which is precondition set P_{fact}. The function A may use the abstraction process to create and include more general patterns in the precondition sets so that they match a wide spectrum of transitions. An example of selective attention is as follows.

Example 4: Assume that there exists a set of observation facts representing the situation S,

$$S = \left\{ \begin{array}{l} (ball \quad \text{shape} \quad \text{sphere} \quad 100ms), \\ (hand \quad \text{position} \quad 5 \quad 100ms), \\ (hand \quad \text{essence} \quad \text{robot} \quad 100ms) \end{array} \right\}$$

meaning that a spherical ball and a robotic hand in position 5 at time 100ms are observed. Assume the following *move* command is issued at the same time $100ms$

$$CMD = (cmd \quad \textbf{move}[hand \quad 3] \quad 100ms)$$

which leads to observing a change in the value of the hand's position later at $200ms$ such that the following fact will be observed.

$$(hand \quad \text{position} \quad 8 \quad 200ms)$$

The selective attention function A selects a subset of observable facts in the situation S by generating the precondition set P'_{fact}:

$$P'_{fact} = \left\{ \begin{array}{l} (hand \quad \text{position} \quad 5 \quad 100ms), \\ (hand \quad \text{essence} \quad \text{robot} \quad 100ms) \end{array} \right\}$$

which only includes the observation facts that are relevant to the transition T from $(hand \quad \text{position} \quad 5 \quad 200ms)$ to $(hand \quad \text{position} \quad 8 \quad 200ms)$ and excludes the observation fact $(ball \quad \text{shape} \quad \text{sphere} \quad 100ms)$ that has nothing to do with the movement of a hand. Transition T only involves *hand* and its property position. Therefore, all properties of hand, i.e., *position* and *essence*, along with their values, are considered by the selective attention function. As mentioned above, selective attention may involve the abstraction of the entities and the numeric value, which leads to having the following *abstracted* precondition set:

$$P_{fact} = \left\{ \begin{array}{l} (H \quad \text{position} \quad POS \quad T), \\ (H \quad \text{essence} \quad \text{hand} \quad T) \end{array} \right\}$$

where H, POS, and T are variables replaced with related entities and values due to abstraction.

4.4.3 Induction

Causality-based agents can create multiple CRMs that model the properties' state transitions via different types of actions. In this work, the generation of new CRMs is called *induction*, which may result from various learning mechanisms. Note that induction may also involve the above-mentioned abstraction and selective attention processes during the generation of pre- and post-conditions for CRMs (Sheikhlar and Thórisson, 2024). Induction generates CRMs due to 1) observed transitions, 2) pattern matching between observations and known CRMs (i.e., via analogy-making), and 3) failure in observing expected patterns. We introduce an instance of induction through analogy-making in this thesis, as shown later in Section 5.2.

4.4.4 Knowledge pruning

Although patterns in CRMs can be abstracted using variables so that they match different task entities and values, a proper generalization mechanism also calls for the ability to modify the patterns in CRMs once the agent finds this modification useful. Modifying CRMs requires identifying the relevant patterns and, subsequently, eliminating irrelevant ones from their pre- and post-conditions, enabling CRMs to be applicable to more encountered tasks and situations (Sheikhlar and Thórisson, 2024).

Definition 4.9: Let $M : [A \rightarrow B] = [(F_A \wedge C_A) \rightarrow (F_B \wedge C_B)]]$ where F_i and C_i with $i = A, B$ represent fact and constraint patterns, respectively. Then a pruning mechanism G takes in M and returns a more general version of it via pruning its patterns. In other words, $G : M \rightarrow M'$, where M' is the generalized CRM. The pruning may involve any of the following

- *Constraint pruning*: The adjusted/pruned set $E' = G(C_A, C_B)$ accommodates a more accurate set of transition functions and conditions where the spurious correlations are pruned. The spurious correlations are constraints that do not affect the actual transitions, do not hold true (e.g., false logical conditions), or are inaccurate (e.g., inaccurate transition functions). We will see an instance of this type of pruning later in the results chapter in Section 7.2 based on an algorithmic set of input interventions through causal discovery introduced in Section 4.2, which eliminates inaccurate transition functions from a model.

- *Fact pruning*: Pruning the facts requires mechanisms for selectively eliminating facts deemed irrelevant for transitions, which leads to fewer conditions on the transitions. Consider a CRM with the original facts $F = F_A \cup F_B$. Then, function G must take in F such that $G(F)$ yields the pruned set of facts, denoted as F', where $F' \subseteq F$ and/or $F' \geq F$

(Sheikhlar and Thórisson, 2024). We will see several instances of this later in Chapter 7.5 based on a mechanism we introduce in Section 5.2.

Such a pruning process is essential when the CRMs' patterns only partially match the encountered situations. This can be done by comparing the situations with the CRMs' patterns. In such cases, knowledge can be pruned via an analogy-making process, detailed later in Section 5.5. However, a pruning mechanism might incorrectly eliminate significant patterns that must not have been eliminated. The agent can find this out by testing the pruned CRMs via making interventions. If the pruned CRMs prove to be wrong in some situations, the CRM must be revised by adding additional patterns as a result of negative analogy, which will be explained later in Section 5.5. The following example shows how induction and knowledge pruning can occur.

Example 5: Assume that there exists a set of observation facts representing the situation S. In this situation, a ball, a box, and a hand, along with their properties, can be observed.

$$S = \left\{ \begin{array}{l} (ball \quad \text{shape} \quad \text{sphere} \quad 100ms), \\ (ball \quad \text{color} \quad \text{red} \quad 100ms), \\ (box \quad \text{shape} \quad \text{cube} \quad 100ms), \\ (box \quad \text{color} \quad \text{blue} \quad 100ms), \\ (hand \quad \text{essence} \quad \text{robot} \quad 100ms), \\ (hand \quad \text{holding} \quad [\ ] \quad 100ms) \end{array} \right\}$$

where the following *grab* command is issued at the same time $100ms$

$$CMD = (cmd \quad \textbf{grab}[hand] \quad 100ms)$$

which leads to observing a state change later at $200ms$ such that

$$(hand \quad \text{holding} \quad [ball] \quad 200ms)$$

meaning that command CMD leads to hand holding the *ball* at 200ms. This state change can be captured by an induction process to generate a new CRM represented as $M1 : [(P1 \wedge CMD) \rightarrow B1]$ with the following preconditions P and postconditions $B1$.

$$P1 = \left(\begin{array}{l} (B \quad \text{shape} \quad \text{sphere} \quad T1) \wedge \\ (B \quad \text{color} \quad \text{red} \quad T1) \wedge \\ (H \quad \text{essence} \quad \text{robot} \quad T1) \wedge \\ (H \quad \text{holding} \quad [\ ] \quad T1) \wedge \\ (T2 = T1 + 100\text{ms}) \end{array} \right)$$

$$B1 = \begin{pmatrix} (H & \text{holding} & [B] & T2)\wedge \\ (T1 = T2 - 100\text{ms}) & & & \end{pmatrix}$$

$P1$ results from a selective attention process that only includes the *relevant* abstracted facts. Selective attention involves the properties, *robot*, *sphere* and *red*, as their related entities (*hand* and *ball*) are involved in the transition. In other words, the patterns related to entity *box* and its properties are disregarded, as it is not involved in the transition. Also, $P1$ itself can be pruned via other mechanisms, e.g., by the analogy mechanism that will be introduced in the next chapter.

Then, assume in another situation, $S2$, the aim is to grab another ball, i.e, *ball*2 that is *blue* and not *red*.

$$S2 = \left\{ \begin{array}{llll} (ball2 & \text{shape} & \text{sphere} & 500ms), \\ (ball2 & \text{color} & \text{blue} & 500ms), \\ (hand & \text{essence} & \text{robot} & 500ms), \\ (hand & \text{holding} & [\ ] & 500ms) \end{array} \right\}$$

with the following goal fact
$G = (hand \quad \text{holding} \quad [ball2] \quad T)$

As $S2$ and G match the learned CRM which is $M1$, a new CRM which is $M2$ can be hypothesized. $M2$ is a pruned version of $M1$, as the knowledge pruning process removes the non-matching property's value, i.e., blue, from preconditions $P1$. Then, we have $M2 : [(P' \wedge CMD) \rightarrow B1]$ where P' is as follows.

$$P' = \begin{pmatrix} (B & \text{shape} & \text{sphere} & T1)\wedge \\ (H & \text{essence} & \text{robot} & T1)\wedge \\ (H & \text{holding} & [\ ] & T1)\wedge \\ (T2 = T1 + 100\text{ms}) & & & \end{pmatrix}$$

where only the matching facts between $P1$ and $S2$ are kept and abstracted, while the difference (i.e., the color-related fact) is pruned. In the next chapter, we present a new mechanism that systematically uses analogy-based knowledge pruning and induction.

4.5 Summary

In this chapter, we first introduced two different causal modeling perspectives: axiomatic representation from a designer's perspective and non-axiomatic agent-based representations which are causal relational models (CRMs). The axiomatic representation includes a large-scale, invariant, centralized representation of value changes of observables through issued commands, which does not meet the requirements of constructivism. However, the second

introduced representation, CRM, is designed based on the principles of constructivist AI, as they represent small, decentralized, falsifiable hypotheses about transitions in environments that can be contextualized, chained, and hierarchized.

Then, we defined different dimensions of causal generalization problems, ranging from abstraction, selective attention, induction, and knowledge pruning. Abstraction involves replacing task entities and values with variables, selective attention selects relevant, abstracted patterns observed in situations, induction generates CRMs, and pruning eliminates patterns deemed irrelevant. Later, we will introduce a mechanism allowing learning agents to use analogy-making to prune their existing CRMs and induce new abstracted CRMs autonomously.

Chapter 5

Autonomous Cumulative Transfer Learning

5.1 Introduction

This chapter outlines the architectural design of a knowledge generalization mechanism that enables learning controllers to grow their knowledge cumulatively and generalize it autonomously. The introduced architecture relies on reasoning based on causal relational models (CRMs) discussed in Section 4.3, allowing an agent to use the familiarities of the task environments at hand to create falsifiable hypotheses about how to deal with situation-goal novelties, choose the most salient hypotheses, and perform input intervention to both achieve the task goals and verify the generated hypotheses (Sheikhlar et al., 2022).

To achieve this, first, we introduce a ***theory for autonomous cumulative transfer learning (ACTL)*** (Sheikhlar et al., 2020), which relates an agent's prediction capabilities to its familiarity with the situation it is in and the goal it tries to achieve, allowing for the transfer of the relevant pieces of knowledge to that situation. Cumulative learning can be described as an agent's ability to incrementally and actively collect and integrate small pieces of knowledge about a phenomenon over its lifetime, leading it to gain a better understanding of the phenomenon gradually (Thórisson, 2021b; Thórisson et al., 2019b). We will also introduce similarity measures that serve as a tool for computing the level of the agent's familiarity with situations in relation to achieving specific goals and, subsequently, applying the relevant knowledge to those situations. Finally, we introduce an ***ACTL mechanism*** based on the ACTL theory, which is a new schema for cumulative learning, allowing agents to generalize their CRMs over their knowledge accumulation process when they try to achieve their goals and, thus, to automatically solve the

causal generalization problems described in Section 4.4.

Disclosure: This chapter is partly adapted from our papers (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2020; Sheikhlar et al., 2022). Sections 5.2 and 5.4 partly explain the autonomous cumulative transfer learning- theory and detailed arguments from similarity sections of (Sheikhlar et al., 2020). Subsection 5.4 provides an extended version of the goal-drive analogy section of (Sheikhlar and Thórisson, 2024).

5.2 A theory for autonomous cumulative transfer learning

Here, we first introduce our theory called autonomous cumulative transfer learning (ACTL) (Sheikhlar et al., 2020), which is the basis of the ACTL mechanism we will introduce later in Section 5.5. The theory is based on several new concepts that are solely introduced in this section and not in other sections. These concepts will be mapped to the concepts defined in Chapter 4 in the next section and then will be formulated so that they can be used for the design of the ACTL mechanism. This section is adapted from our paper (Sheikhlar et al., 2020).

Assume that a cognitive agent is able to evaluate how familiar it is with a certain phenomenon Φ. The ***phenomenon*** Φ is made up of a finite set of parts $\{\varphi_1 \dots \varphi_n\}$ that can be observed and are referred to as ***aspects***, between which various types of relationships can be defined (Sheikhlar et al., 2020; Thórisson, 2021b). Then,

> **"The agent can predict a particular selected aspect $\varphi_i \in \Phi$, $i \in 1, ..., n$, using its prior knowledge, if and only if φ_i is familiar to the agent, and non-novel."** - (Sheikhlar et al., 2020)

To compute familiarity, a ***similarity function*** Ψ is called for that captures the overlap between newly observed aspect and the retrieved relevant piece of ***knowledge*** k_i from the existing knowledge stored in the agent's ***knowledge base*** KB (Sheikhlar et al., 2020). In other words,

$$\Phi_{fam} = \Psi(k_i, \varphi_i) \tag{5.1}$$

where $k_i \in KB$ and $\varphi_i \in \Phi$.

Retrieving k_i from KB calls for identifying its connection to the active goals of the agent, meaning that the familiarity computation must be in line with the agent's goal achievement. As the agent is assumed to know what the goals are, it performs backward reasoning from them to find the pieces

of knowledge in KB that best match the goals of the agent (Sheikhlar et al., 2020).

In the next section, we will map the ACTL theory and its concepts to the notions of CRMs, situations, transitions, and dynamic processes, allowing us to use it in practice for designing the ACTL mechanism.

5.3 Reformulation of ACTL theory: Dynamic processes as phenomena

To relate the ACTL theory to our definitions in Chapter 4, the *phenomenon* Φ is considered a ***dynamic process*** DP. Then, the *aspects* of Φ are the ***situations***, where each situation S represents all existing observable patterns at a time. ***Transitions*** exist between situations that change the state of a dynamic process from one situation to another. The cause of transitions is considered the actions of an agent resulting from internal ***commands*** issued by the agent.

The familiarity of a situation S encountered by the agent must be determined by comparing it with the agent's relevant knowledge in relation to a particular goal. The agent's *knowledge* is formulated as causal relational models (CRMs) that represent hypothesized transitions with pre- and post-conditions. The *knowledge base* is the set of known CRMs. Ampliative reasoning, detailed in Section 2.6, allows for retrieving the relevant CRMs based on abductive reasoning, which identifies the properties related to the agent's goals (Sheikhlar et al., 2020). In other words, CRMs are considered relevant when the goal facts match the postconditions of CRMs, making the patterns of the CRMs both in their pre- and post-conditions relevant. This means that the familiarity computation needs to be done by calculating the overlap between the CRM patterns and situation patterns in relation to the agent's active goals (Sheikhlar and Thórisson, 2024). Now, the ACTL theory can be reformulated in the context of CRMs, as follows.

> **The agent can make a correct prediction in a situation S, using its known CRMs, if and only if S is familiar to the agent.**

As the goals determine the relevance of the CRMs involved in a comparison, the similarity function Ψ of Equation (5.1) must compare CRMs and situation-goal pairs. Therefore, if we assume we have a CRM called M and a situation-goal tuple, τ:(S, G), the familiarity of M is relation to τ can be calculated as follows

$$\Phi_{fam}(M, \tau) = \Psi(M, \tau) \tag{5.2}$$

How the above equation is calculated will be explained later in detail in the analogy section (see Section 5.4). Also, in Section 5.4, different dimen-

sions of comparison and similarity are discussed. According to the ACTL theory, high values of $\Phi_{fam}(M, \tau)$ means M's higher capability in making predictions in situation S.

Familiarity is a continuous concept that depends on how accurate and confident the existing CRMs can represent the transitions between situations. The more accurate and confident CRMs, the more accurate the predictions. Familiarity with situations, therefore, must determine the agent's predictability in those situations. The CRMs and, therefore, familiarity can be tested through agents' actions/commands, which leads to changes in the confidence of the models and subsequent familiarity values. However, the initial familiarity computation in the context of CRMs relies on a similarity/comparison computation explained (in detail in section 5.4) as well as the initial confidence of the CRMs involved in the comparison (explained later in 5.4.1). This comparison captures the overall similarity between CRMs and given situation-goal patterns.

5.4 Similarity dimensions

In this section, we present some of the dimensions of similarity function Ψ in Equations (5.1) and (5.2). The basis of our argument is the comparison between discrete states, where the states are subsets of situations consisting of the patterns the agent can observe or imaginatively generate (Sheikhlar and Thórisson, 2024). Here, we define states in the context of CRMs. This section is adapted from the "Detailed argument from similarity" section of our paper (Sheikhlar et al., 2020) with some modifications in relation to the terminology and naming.

Definition 5.1: State ζ is an arbitrary subset of a situation S. Situation S is a complete set of observable facts $S = \{f_1, f_2, \ldots, f_n\}$, where f_i represents the i-th observable fact. Then, a state ζ is a subset of S, that is, $\zeta \subseteq S$.

Pattern similarity in cardinality
According to (Sheikhlar et al., 2020), two states, ζ_1 and ζ_2, have *pattern similarity in relation to cardinality (SSP_C)* if some of their patterns match while others do not. The value of SSP_C is determined by calculating the following ratio (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2020).

$$SSP_C(\zeta_1, \zeta_2) = \frac{|\zeta_1 \cap \zeta_2|}{|\zeta_1 \cup \zeta_2|} \tag{5.3}$$

where $|\zeta_1 \cap \zeta_2|$ represents the number of matching patterns between the two states and $|\zeta_1 \cup \zeta_2|$ is the number of all patterns including shared and non-shared patterns. Note that $0 \leq SSP_C \leq 1$ must always hold true, where

1 refers to the case where the two states completely match, $\zeta_1 \cap \zeta_2 = \zeta_1 \cup \zeta_2$, while 0 indicates the case where none of the patterns match, $\zeta_1 \cap \zeta_2 = \varnothing$.

Relational similarity

Relational similarity is about comparing the states between which there exist transitions (Sheikhlar et al., 2020). We formulate this in the context of comparing CRMs and situation-goal tuples, introduced in Section 5.3. Assume that we have a CRM called $M : [\zeta_1(T_1) \rightarrow \zeta_2(T_2)]$ and a situation-goal tuple, $\tau : (\zeta_3(T_1), \zeta_4(T_2))$. The states $\zeta_1(T_1)$ and $\zeta_2(T_2)$ are precondition (LHS) and post-condition (RHS) patterns of M, respectively. Also, $\zeta_3(T_1)$ and $\zeta_4(T_2)$ represent the situation and goal patterns, respectively. To compare M and τ, left-hand side (LHS) pairs $\{\zeta_1(T_1)$,$\zeta_3(T_1)\}$ and the right-hand side (RHS) pairs $\{\zeta_2(T_2)$, $\zeta_4(T_2)\}$ must be compared in relation to SSP_C measure. In other words, relational similarity in CRMs calls for an overall similarity Ψ calculation based on the similarity between pairs ψ as follows

$$\psi(\zeta_1(T_1), \zeta_3(T_1)) = SSP_C(\zeta_1(T_1), \zeta_3(T_1)) \tag{5.4}$$

$$\psi(\zeta_2(T_2), \zeta_4(T_2)) = SSP_C(\zeta_2(T_2), \zeta_4(T_2)) \tag{5.5}$$

$$\Psi(M, \tau) = \psi(\zeta_1(T_1), \zeta_3(T_1)) \cdot \psi(\zeta_2(T_2), \zeta_4(T_2)) \tag{5.6}$$

Note that the RHS patterns in CRMs only contain one observable fact, implying that $SSP_C(\zeta_2, \zeta_4)$ must be 1 so that ζ_2 and ζ_4 are similar. If we represent the CRM as $M : [(P \wedge cmd) \rightarrow B]$, the tuple as $\tau : (S, G)$, and remove the time notation for clarity, then the above equation can be rewritten as follows.

$$\Psi(M, \tau) = \psi(P, S) \cdot \psi(B, G) \tag{5.7}$$

As mentioned in Chapter 4, B contains a single fact representing the property value's change. As G is also a single fact, then the value of $\psi(B, G)$ is either 0 or 1. In other words, a goal either matches the RHS of a CRM or not.

Comparison between CRMs and situation-goal pairs can either be

- *top-down*: first comparing the $\psi(B, G)$, meaning that if the similarity between the goal and postcondition pattern is 1 (if the two facts match), then $\psi(P, S)$, the similarity between situation and preconditions, is calculated.

- *bottom-up*: first comparing $\psi(P, S)$ and then $\psi(B, G)$, meaning that if the similarity between the preconditions and the situation is greater than a threshold, γ, then the similarity between the postcondition and the goal is calculated.

Note: The analogy-making mechanism we will introduce later in Section 5.5 uses a top-down comparison.

Implications useful for analogy-making
The theory of autonomous cumulative transfer learning has several implications for autonomous agents intending to use analogical reasoning to generate new models for dealing with novel situations.

- **Familiarity level and prediction confidence:** The similarity function Ψ allows the agent to assess its familiarity with the situation in relation to a goal. The more familiar a situation is to an agent, the better the agent can make predictions in that situation. Predictions are done via CRMs, whose accuracy depends on their confidence values. In partly familiar partly novel situations, the agent has to generate new CRMs (which fully match the new situations) but with confidence values lower than the ones they are derived from. Therefore, we can state that familiarity with situations determines the confidence of newly generated hypotheses about how to achieve certain goals. We will see that this allows our analogy-making mechanism to hypothesize new CRMs with different confidence values determining how plausible it is to reach desired effects from partially known situations.

- **Comparison based on importance**: To identify the importance of patterns, the backward chaining allows the agent to identify CRMs whose conditions match the goals. This leads to inferring the conditions required for achieving the goals, which are the precondition patterns of the CRMs. Those precondition patterns are what we call *important patterns* P_I and need to be involved in their comparison with the situation patterns. The instantiation of those precondition patterns themselves can lead to generating new goals, called subgoals, that have to follow the same matching process to identify other important patterns (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2020).

- **Negative knowledge transfer**: Negative transfer of knowledge is the result of incorrect familiarity estimation where important patterns are not involved in the comparison, due to the lack of precise models, i.e., the CRMs whose preconditions are incorrect. Such preconditions do not include a proper set of patterns and do not accurately describe the conditions of transitions. In such cases, wrong analogies and thus, wrong hypotheses are created that do not hold for the situation-goal patterns and lead to incorrect predictions. This can be identified by the feedback loops when the agent creates and uses new hypotheses based on the comparison/analogy which checks whether the hypotheses lead to expected outcomes. Therefore, the negative transfer of knowledge

can be identified by the agent and modified by creating other more accurate CRMs.

Our ACTL theory considers the above-mentioned dimensions when using similarity calculations to identify how familiar a situation is in relation to a goal.

5.5 Autonomous causal generalization

We introduce a novel cumulative learning mechanism that enables an agent to leverage its existing CRMs to hypothesize new CRMs (called ***models*** in this section) and then integrate them into the agent's model base KB, as illustrated in Figure (5.1). The agent uses an analogy process to compare the patterns of models in KB with situations and generate new models for planning and exploration. During exploration, the agent evaluates generated models and plans by issuing related commands. This way, the agent interacts with its environment and tests its models and plans in practice. From an explanation hypothesis generation perspective (Thórisson, 2021a), the analogy-based models are new explanations for why a decision in a novel situation/task must be made by an agent. If a model generated through analogy proves to be inaccurate, the learning from failure mechanism specializes the analogy-based inaccurate model by adding more models to the KB that control the applicability of the inaccurate model in future situations. In other words, explanations for the agent's decisions can dynamically change as a result of the agent's interventions (Eberding et al., 2024). Note that the goals and subgoals are critical in identifying the important patterns and models that must be analogized, generalized, and generated. The following example shows how the mentioned processes occur.

Example 1: As shown in Figure (2), assume we have a robot hand h that, after successfully learning to grasp a blue cube ($c1$), is assigned to grasp two new cubes ($c2$ and $c3$) with different colors, green and red. Grasping $c1$ means applying the command (cmd **grab** $[h]$ $T1$) in situation $S1$, which causes a transition from (h holding $[\,]$ $T1$) in $S1$ to (h holding $[c1]$ $T2$) in $S2$. This makes the agent learn the model $M1 : [A1 \rightarrow B1]$, where

$$A1 = \begin{pmatrix} (cmd \quad \mathbf{grab}[H] \quad T1)\wedge \\ (C \quad \text{color} \quad \text{blue} \quad T1)\wedge \\ (C \quad \text{essence} \quad \text{cube} \quad T1)\wedge \\ (H \quad \text{essence} \quad \text{hand} \quad T1)\wedge \\ (H \quad \text{holding} \quad [\ ] \quad T1)\wedge \\ (T2 = T1 + 100\text{ms}) \end{pmatrix}$$

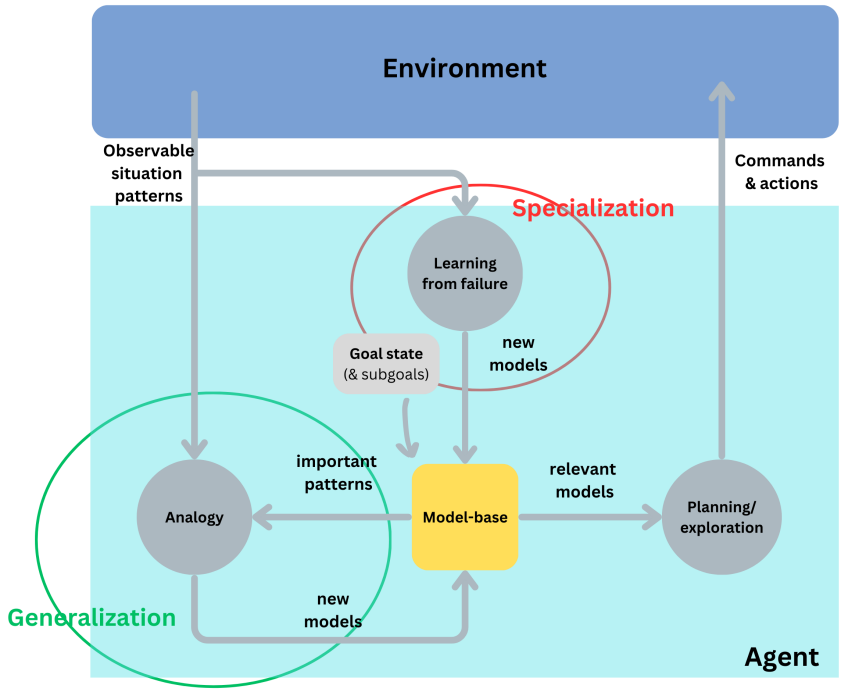

Figure 5.1: Autonomous cumulative transfer learning (ACTL) mechanism controls the interactions of a cognitive agent with its environment. The mechanism involves both generalization and specialization. Generalization prunes non-matching conditions (through analogizing important patterns with observable situations) from models and accordingly generates new models, while specialization adds new constraints to overly/incorrectly generalized models when they fail in practice (learning from failure). Important patterns are the components of precondition sets of models that are obtained from backward chaining from (sub) goals. The relevant models, in this context, are the ones that the planner selects based on the priorities it calculates.

$$B1 = \begin{pmatrix} (H & \text{holding} & [C] & T2)\wedge \\ (T1 = T2 - 100\text{ms}) & & & \end{pmatrix}$$

The model $M1$ can be generalized when needed. Assume that in situation $S3$, the goal fact is

$$G1 : \begin{pmatrix} H & \text{holding} & [c2] & T2 \end{pmatrix}$$

meaning that the hand is desired to be holding the green cube ($c2$). $G1$ matches the postconditions (RHS) of $M1$. In other words, when $G1$ matches

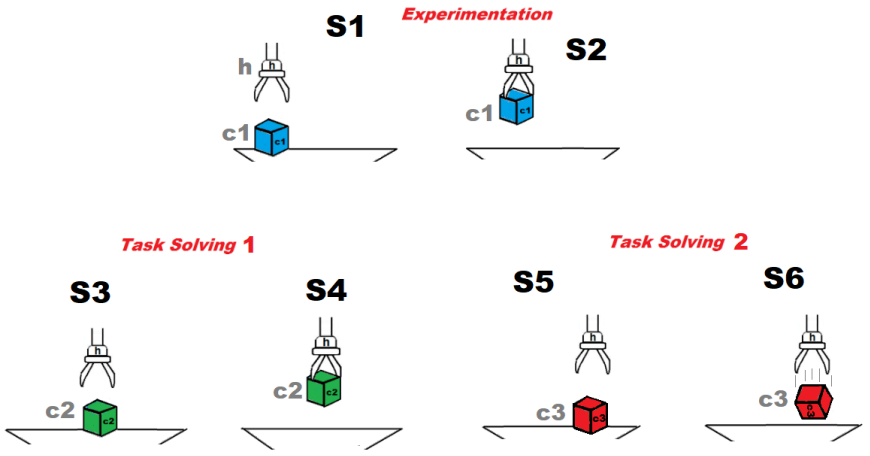

Figure 5.2: The experimentation has to do with learning $M1$ after grabbing $c1$, which is then used for deriving $M2$ in Task Solving 1 for grabbing $c2$. Yet, h fails to grab $c3$ using $M2$, leading to the induction of $M3$.

$(H \quad \text{holding} \quad [C] \quad T2)$ in B, backward chaining from $G1$ leads to the instantiation of fact patterns in $A1$, selecting the important patterns in situation $S3$. Then, the instantiated important patterns become

$$P_I = \begin{pmatrix} \mathbf{(c2 \quad color \quad blue \quad T1)}\wedge \\ (c2 \quad \text{essence} \quad \text{cube} \quad T1)\wedge \\ (h \quad \text{essence} \quad \text{hand} \quad T1)\wedge \\ (h \quad \text{holding} \quad [\ ] \quad T1) \end{pmatrix}$$

The analogy mechanism has to compare the patterns in P_I with the current observation facts of situation $S3$. In $S3$, the cube $c2$ is observed to be green, which does not match the color value (blue) in P_I, bolded in the set. The pruning mechanism in the analogy process removes the non-matching fact $(c2 \quad \text{color} \quad \text{blue} \quad T1)$ and keeps the rest. Then, the induction mechanism in analogy generates a new model $M2$ similar to $M1$ but with the difference that $M2$ does not include the pattern $(C \quad \text{color} \quad \text{blue} \quad T1)$, meaning that the color property of the cubes is disregarded in the new model $M2$. In other words, the newly induced $M2 : [A2 \rightarrow B2]$ injected into the model-base is as follows.

$$A2 = \begin{pmatrix} (cmd \quad \mathbf{grab}[H] \quad T1)\wedge \\ (C \quad \text{essence} \quad \text{cube} \quad T1)\wedge \\ (H \quad \text{essence} \quad \text{hand} \quad T1)\wedge \\ (H \quad \text{holding} \quad [\ ] \quad T1)\wedge \\ (T2 = T1 + 100\text{ms}) \end{pmatrix}$$

$$B2 = \begin{pmatrix} (H \quad \text{holding} \quad [C] \quad T2)\wedge \\ (T1 = T2 - 100\text{ms}) \end{pmatrix}$$

$M2$ is a generalized version of $M1$ with fewer precondition patterns on its LHS, which will immediately be used by the planner, leading to issuing the command of (cmd **grab** [h] T1). The issued command successfully grabs and leads to observing the fact (h *holding* $[c2]$ $T2$), which increases the confidence of $M2$. However, such generalized models may not always be accurate and thus useful. Situations $S5$ and $S6$ (Task solving 2) is an instance of such scenarios, where the goal is to grab and hold a *red* cube ($c3$) in situation $S5$. Based on a process similar to what was shown above, the goal fact $G2 = (H \quad \text{holding} \quad [c3] \quad T2)$ matching the RHS of the analogy-based induced model $M2$, leading to issuing the *grab* command. After the *grab* command is issued in $S5$, the grabbing action fails in situation $S6$, leading to the failure of $M2$. This makes the learning-from-failure mechanism induce a new model $M3$ that restricts the use of $M2$ when the cube is red. In other words, we $M3 : [A3 \rightarrow B3]$ is learned, where

$$A3 = \begin{pmatrix} (C \quad \text{essence} \quad \text{cube} \quad T1)\wedge \\ \mathbf{(C \quad color \quad red \quad T1)}\wedge \\ (H \quad \text{essence} \quad \text{hand} \quad T1)\wedge \\ (H \quad \text{holding} \quad [\ ] \quad T1) \end{pmatrix}$$

$$B3 = \begin{pmatrix} mdl \quad \neg\mathbf{M2} \quad T1 \end{pmatrix}$$

where $\neg$ indicates negation. $M3$ prevents $M2$ from being used in future encounters with red cubes. Incorporating the bolded fact **(C color red T1)** results from the *negative analogy* where all the differences between the current situation and known preconditions must be included in the newly induced precondition set $A3$ for grabbing action. Such a mechanism does not allow overgeneralization of models learned during goal achievement. Now, the models $M1$, $M2$, and $M3$ together mean that the ***grab*** command works for cubes with any color except *red*.

The following section explains how each component of the introduced cumulative learning mechanism works.

5.5.1 Goal-Driven Analogy

The analogy process starts with a top-down comparison between models and situation-goal pairs, computes the familiarity, performs knowledge pruning, and induces new models (detailed in Chapter 4), guiding the agent to learn to incorporate fewer yet more relevant patterns in the new models it creates over time (Sheikhlar and Thórisson, 2024). The goal-driven analogy also allows for the generation of new hypotheses as explanations of why certain properties of processes must be considered when performing a task (Thórisson, 2021a). This subsection is adapted from *Goal-Driven Analogy: A Mechanism* section of our paper (Sheikhlar and Thórisson, 2024).

Top-down comparison[1]
The first step to performing analogical reasoning is comparing the known with the new pieces of information. Assume the agent knows a model $M : [LHS \rightarrow RHS]$. To compare M with encountered situation-goal pairs, discussed in Section 5.4, the top-down method must be chosen, as the comparison must involve the important patterns identified by backward chaining from the goals. This means that as shown in Figure 5.3 (Sheikhlar and Thórisson, 2024), the comparison is first initiated by pattern matching between the model's post-conditions and the goal patterns, and then between the preconditions and situation patterns (Sheikhlar and Thórisson, 2024).

Knowledge pruning[2]
Analogy solves the knowledge pruning problem, detailed in Section 4.4, by eliminating unmatched patterns between LHS and S and retaining the matched ones. In a top-down comparison, as detailed in Section 5.4, if $\psi(RHS, G) = 1$, then the pruned M will have the following pre- and post-conditions.

$$LHS^* = LHS \cap S \tag{5.8}$$

$$RHS^* = RHS \tag{5.9}$$

Such knowledge pruning generalizes the preconditions by creating the new precondition set LHS^*, which has fewer patterns and, thus, has the potential to match a wider range of situations in comparison to the original precondition set LHS.

[1] This part extends the description of Figure 5.3 (Sheikhlar and Thórisson, 2024).

[2] Here we explain the details of knowledge pruning introduced in "Analogy, induction, and knowledge pruning" subsection of our paper (Sheikhlar and Thórisson, 2024).

Induction[3]
The pruned preconditions and postconditions (LHS^* and RHS^*) form the the new model M^*, represented as follows (Sheikhlar and Thórisson, 2024)

$$M^* = [LHS^* \rightarrow RHS^*], \qquad cfd^* < cfd \tag{5.10}$$

where cfd^* must be lower than cfd, because M^* is a hypothesis inferred from M in a novel situation-goal. In the next subsection, we will show how cdf^* is computed based on the notion of familiarity and the confidence of old models.

As shown in Figure 5.3 (Sheikhlar and Thórisson, 2024), the pattern matching of M^* is tested in forward chaining from LHS to the goal fact. Note that there is no guarantee that the generalized model M^* makes a correct prediction in practice. Therefore, interventions from the agent are needed to verify the accuracy of the new model M^* generated via induction. Also, models like M^* must allow the agent to come up with solutions when facing new situation-goals, which requires analogy to be based on a top-down abductive process.

Familiarity computation
The analogy process constantly computes the agent's familiarity with situations to guide the agent's planning and exploration. The relational similarity equation (5.2) can be used to estimate how familiar the situation-goal is in relation to a model, implying that every model has an accompanying familiarity degree.

According to equation (5.2) familiarity of a situation in relation to a model depends on how similar the situation is to the preconditions of the model. In other words,

$$\Phi_{fam}(M, \tau) = \psi(LHS, S) \tag{5.11}$$

The above equation will be useful for causality-based frameworks whose RHSs of cause-effect models are abstracted, single patterns, and can fully match the (sub)goals.

Confidence computation[4]
According to our ACTL theory (see Section 5.2), familiarity value gives clues to the agent about how predictable the situations are. An important

[3]This part describes the details of the induced models via goal-driven analogy, briefly explained in the "Analogy, induction, and knowledge pruning" subsection of our paper (Sheikhlar and Thórisson, 2024).

[4]This part explains the computation details of the confidence of newly induced models via goal-driven analogy, briefly described in "Confidence Computation" subsection of our paper (Sheikhlar and Thórisson, 2024).

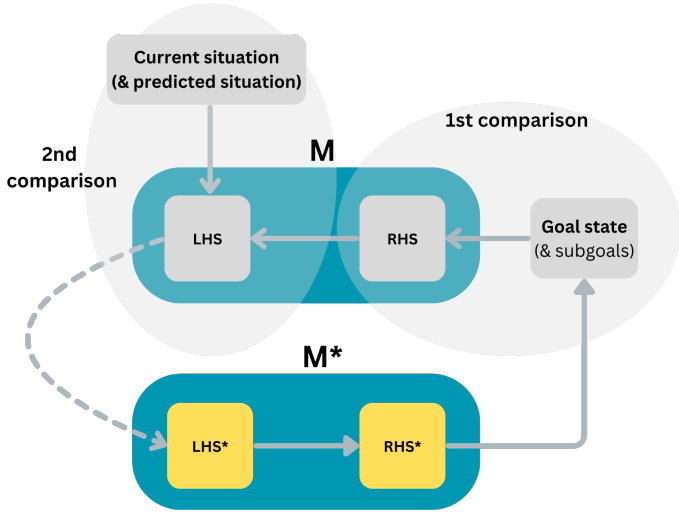

Figure 5.3: The first step in the analogy process is to check if the RHS of M matches (sub) goals (1^{st} comparison). If so, the patterns of LHS will be compared to the current or predicted situations (2^{nd} comparison), leading to the induction of M^*, generated over the abductive backward chaining from the goal state. M^*'s matching will be verified over the deductive forward chaining from the LHS^* towards the RHS^* and the goal state.- *figure from (Sheikhlar and Thórisson, 2024)*

implication of the theory, as discussed in the previous section, is that the level of familiarity, which has to do with to what extent the situation-goal pairs match the known models' patterns, specifies how confident the predictions can be in relation to the situation-goal (Sheikhlar and Thórisson, 2024). As the predictions are made via models, familiarity value is considered a factor in calculating the confidence values of the models induced via the analogy process.

The analogy process induces the new model M^* with confidence cfd^*. The familiarity value $\Phi_{fam}(M, \tau)$ computed by equation (5.11) and the confidence of the original model M, i.e., cfd, together determine the induced model's confidence cfd^* (Sheikhlar and Thórisson, 2024). Therefore, as in (Sheikhlar and Thórisson, 2024), we have

$$cfd^* = \Phi_{fam}(M, \tau) \cdot cfd = \psi(LHS, S) \cdot cfd \tag{5.12}$$

where $0 \leq cfd^* \leq cfd \leq 1$ holds true as long as it has not been tested through intervention.

Note that cfd^* will change once M^* is tested through input interventions that is, applying the related command on the LHS of M^*. The prediction success of M^* collects positive evidence, which in turn increases cfd^*, whereas its failure and, thus, negative evidence does the opposite.

Generalizing CRM chains [5]

As shown in figure (5.4) (Sheikhlar and Thórisson, 2024), models can be chained both in the forward and backward directions. Yet, as the analogy process is a top-down mechanism, the induction of new models must happen during the backward chaining. This leads to the creation of multiple variations of the existing models. The induction of multiple models enables the planning system (explained later in this chapter) to chain those models (and the old models) in different ways and to generate extra solutions to the same task, which the agent will test based on their priority, partly specified by the confidence of their models.

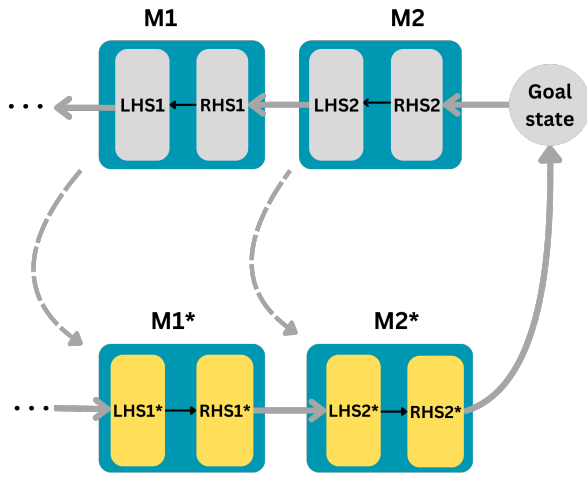

Figure 5.4: If $LHS2$ matches $RHS1$ and partially matches the relevant predicted situation, then $M2^*$ will be induced. Similarly, when $LHS1$ matches the RHS of another model and partially matches the relevant predicted situation, then $M1^*$ will also be induced. The induction of $M2^*$ and then $M1^*$ follows the same process shown in figure (5.3).- *figure from* (Sheikhlar and Thórisson, 2024)

[5] This part is adapted from the "Generalizing chains of models" subsection of our paper (Sheikhlar and Thórisson, 2024).

5.5.2 Planning/exploration

The newly generated models through analogy, can chain and create new solutions (a.k.a. plans for goal achievement) when no solutions exist to achieve the agent's active goals. Therefore, the analogy-based model creation must occur while the agent performs planning. A solutions' dynamic confidence computation partly specifies the priority of choosing and deciding which solutions to explore. The analogy process creates multiple models in each predicted situation, by which the agent can create a set of solutions with different confidence values and durations, leading to a systematic goal-driven exploration that evaluates each generated solution and the models it uses in light of the goal achievement. The solutions are tested individually via the agent's command sequences (i.e., input interventions) based on their priority. The priority partly depends on the confidence of a solution, which can be computed via the product of confidence of each model involved in the chain from the first command towards the final goal fact. This means that if there exist n models in a chain where m models are induced via analogy ($m < n$), then the solution confidence can be calculated as follows.

$$cfd_{\text{solution}} = \prod_{i=1}^{m} cfd_i^* \cdot \prod_{j=m+1}^{n} cfd_j \tag{5.13}$$

The solution's priority is dependent on both the solution's confidence and the length of its chain (a.k.a. duration). According to AERA architecture (Nivel, Thórisson, Dindo, et al., 2013), the longer it takes to reach the goal state, the less inclined the planner is to choose that solution. Therefore, when there exist multiple valid solutions to a task, the priority of the k^{th} solution, p_k, is calculated with its confidence $cfd_{solution,k}$ over the solution's duration d_k, as follows (Nivel, Thórisson, Dindo, et al., 2013).

$$p_k = \alpha . \frac{cfd_{\text{solution},k}}{d_k} \tag{5.14}$$

where higher α values make the exploration more conservative, as α's value depends on the assigned deadline to reach the goal state, with smaller values for shorter deadlines.

While performing its tasks, the agent evaluates the solutions in practice by issuing the relevant sequences of commands using newly hypothesized models. The initial confidence computation for each model results from the analogy mechanism explained earlier, guiding the goal-driven exploration process based on the priority order. Note that the confidence values of the solutions are updated as the agent issues the commands and takes the

relevant sequences of actions, which updates the confidence of the involved models. In other words, the verification of plans generated based on analogy-based induced models occurs once the related commands are issued, which, if they lead to expected outcomes, increases the confidence value of the models, solidifying the newly induced models via increasing their confidence values. On the other hand, if the outcomes of commands prove the induced models are incorrect, their confidence decreases, and anti-models for them will be hypothesized, preventing those models from being used in similar situations later. In the next section, we will see how the anti-models are induced via learning from failure mechanism.

5.5.3 Learning from failure

While analogy removes some conditions on the use of models, learning from failure adds new conditions. Some models induced by the analogy process may lead to unexpected outcomes due to over-generalization or incorrect generalization. An instance of a learning-from-failure mechanism has already been implemented in the AERA system through an algorithm called prediction targeted pattern extractor (PTPX) (Nivel and Thórisson, 2013a; Nivel, Thórisson, Dindo, et al., 2013). In the next chapter, we will see how we integrate our goal-driven analogy mechanism to effectively use PTPX to prevent incorrect- or over-generalization.

Learning from failure is a specialization mechanism that falsifies or restricts the incorrectly generalized models. When the agent's intervention proves some models' inaccuracy, the mechanism revises the misbehaved models by learning new models that control the failed models' use in future situations (Nivel and Thórisson, 2013a; Nivel, Thórisson, Dindo, et al., 2013). In our work, the learning-from-failure mechanism induces new models that include all non-matching patterns disregarded when generalizing the models via analogy. A ***negative analogy***, therefore, creates anti-models M_{anti}, that is, models that incorporate non-matching patterns between model preconditions LHS^* and situation S in its preconditions V. The following anti-model restricts the situations in which M^* holds.

$$M_{anti} : [V \quad \neg M^*] \qquad cfd_V \tag{5.15}$$

where $\neg$ indicates negation, and V is a new set of preconditions, including the differences (non-matching patterns) between LHS and S, specifying under which circumstances M^* does not hold true. In other words, $(S \cup LHS) - (S \cap LHS) \subseteq V$. The confidence value cfd_V is considered 1, as the M_{anti} is generated as result of the intervention of the agent.

The next chapter describes the mechanism's implementation details in a general machine intelligence (GMI) aspiring system called AERA (Nivel,

Thórisson, Dindo, et al., 2013). We will also see how we expect the AERA system to extend its models through the introduced ACTL mechanism.

5.5.4 ACTL mechanism

Figure (5.1) shows that the ACTL mechanism is a goal-oriented process. When the goals and subgoals match the RHS of models in the model-base *MB* (a.k.a. knowledge base), important patterns on the LHS of matched models are extracted. The important patterns (P_I) are compared with situations via the analogy process, leading to the induction of new models with generalized patterns and various confidence values. The analogy-based models are sent to and stored in the agent's *MB* and employed during the task-solving process, where model chains with higher confidence and shorter lengths are chosen. The selected solutions and, thus, the chosen commands are applied, leading to the evaluation of the generated models. If the utilized analogy-based models fail to make correct predictions in practice, the learning-from-failure process induces new anti-models injected into *MB*, which restricts the use of models in similar situations in future encounters. In other words, learning-from-failure specializes the incorrectly generalized models.

5.6 Summary

In the chapter, we introduced a new schema for cumulative learning that uses an analogy-based generalization mechanism to build new models based on existing models, learns from the failure of the analogy-based models, performs planning by generating different hypothesized solutions using the analogy-based, failure-based, and existing models, and does an exploration process that tries each generated solution (a.k.a. plan) based on the assigned priority to it. Analogy induces new models that only contain the overlaps of the old models with encountered situation-goal pairs during the planning process, allowing the agent to immediately verify the accuracy of the new models. If they are wrong, new models, called anti-models, will be produced, preventing the analogy models from being used later in similar situations. The models are tested by applying the generated solutions for goal achievement. If a solution fails, another solution with a lower initial priority is chosen. Solutions are explored based on a priority order determined via the confidence of the models used in the solution and the length of the causal chain. The confidence of the analogy-based models results from familiarity computations. These are the elements of an explicit cumulative learning mechanism working together to allow an agent to learn from correct or incorrect analogies made to achieve goals. The mechanism is integrated

into and tested by a running causality-based system described in the next chapter.

Chapter 6

Design & Implementation

6.1 Introduction

The following chapter explains how the autonomous cumulative transfer learning (ACTL) mechanism is implemented and integrated into an existing cognitive architecture autocatalytic endogenous reflective architecture (AERA) (Nivel, Thórisson, Dindo, et al., 2013), a GMI-aspiring, causality-based control architecture designed based on the principles of constructivism. The integration enables an AERA agent to generate different falsifiable hypotheses about how to achieve goals and, subsequently, steer its exploration toward goal achievement. The hypotheses are in the form of new models that carry uncertainty by their assigned confidence values, initially calculated based on the confidence of the original models they are derived from and the amount of the original models' overlap with the experienced situations (see Section 5.4). The interventions of the agent verify whether the hypothesized models are correct/accurate. The explicit models in the AERA system enable an AERA-based agent to inspect the significance of models' patterns used for planning and revise the models when needed.

The introduced ACTL mechanism extends AERA's learning and planning capabilities, allowing it to autonomously perform explicit analogies and efficiently generalize its models while achieving its goals. The current implementation of the AERA framework, called OpenAERA,[1] is chosen to integrate the ACTL mechanism. First, we introduce the knowledge representation and reasoning components of the OpenAERA system. Then, we describe how the ACTL mechanism is integrated with this system.

Disclosure: This chapter is partly adapted from our papers (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022). Section 6.2 describes the *knowledge*

[1] See http://www.openaera.org — accessed Apr. 2, 2024.

representation and reasoning in AERA section of (Sheikhlar et al., 2022). Section 6.4 provides an extensive description of *extension via anaogy* subsection of (Sheikhlar and Thórisson, 2024).

6.2 Knowledge representation and reasoning in AERA

We provide a high-level description of the relevant syntax and semantics of replicode, OpenAERA's programming language (Nivel and Thórisson, 2013a). Some related concepts, such as facts, entities, goals, and commands, were introduced in previous chapters. Here, we focus on describing command models, requirement models, and composite states in OpenAERA, which together represent CRMs introduced in previous chapters. We also describe how these components allow for ampliative reasoning. The section is adapted from the knowledge representation and reasoning sections of (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022).

- A *Command models (CoM)* represents an abstracted command pattern at a time causing a value change of a property at a later time.
  E.g. M_1: [(*cmd* release[H] T_1) → (H holding[ ] T_2)] means that issuing the *release* command through variable H (abstracted entity *hand*) at time T_1 will cause H to be *holding* nothing, i.e., [ ], at time T_2. Note that H, T_1, and T_2 are variables that can be present in AERA facts and be instantiated via pattern matching. Every CoM has a success rate that specifies the confidence of the model and can change through evidence collection.

- *Composite states (CSTs)* represent abstracted precondition patterns that need to be instantiated all at the same time so that CoMs can be used for prediction-making or planning (Sheikhlar and Thórisson, 2024). E.g. CST_1: [(H *position* P T_0) ∧(C position P T_0) ∧(H holding [C] T_0)] refers to a composite state representing a subset of a situation where both C and H are at the same position P at the same time T_0, and H is holding C at that time. If H represents a hand and C represents a cube, then CST_1 can represent the preconditions of releasing a hand holding a cube via M_1.

- *Requirement models (M_{req})* are models that specify the requirements for using CoMs under specific CSTs (Sheikhlar and Thórisson, 2024). E.g. M_{req}:[(*icst* $CST_1(H,C,P, T_0)$) → (*imdl* $M_1(H,T_0)$)], is a requirement model, where *icst* and *imdl* represent instantiation operators on composite state CST_1 and M_1, respectively. The requirement models states that when the positions of H and C are identical (P), then *release* command will lead to (H holding [ ]). Every M_{req} has a

success rate that specifies the confidence of the model and can change through evidence collection.

OpenAERA's planning is based on two mechanisms: forward chaining, which concerns deductive reasoning, and backward chaining, which realizes abductive reasoning (Nivel and Thórisson, 2013a; Nivel, Thórisson, Dindo, et al., 2013; Sheikhlar and Thórisson, 2024), described as follows.

- *Forward chaining (deduction)* is the chaining of known M_{req}s in the forward direction, which occurs if situation patterns match all the CST components in a M_{req} of a CoM, causing the CoM to predict the effect of the issued command on its LHS (Sheikhlar and Thórisson, 2024).

- *Backward chaining (abduction)* is the set of inference steps starting from the top-level goal, moving backward in time towards the commands, and generating subgoals on the way. The subgoals are instantiated components of the CSTs instantiated on LHS of the M_{req}s during backward chaining. Each subgoal must match the RHS of another CoM such that backward chaining continues (Sheikhlar and Thórisson, 2024).

- *Induction*: Standard OpenAERA induces new knowledge pieces via two algorithms[2]: 1) *change targeted pattern extractor* (CTPX) which creates a CST, a CoM and their M_{req} when a command issued by the AERA agent alters the value of a property in the environment, and 2) *prediction targeted pattern extractor* (PTPX), which is triggered when a known CoM fails to make a correct prediction in a particular situation causing the AERA agent to generate a new CST-M_{req} specifying the conditions under which the CoM can not predict accurately (Nivel, Thórisson, Dindo, et al., 2013; Sheikhlar et al., 2022).

6.3 Frame problem and AERA

Introduced by McCarthy and Hayes (1981), the frame problem points to the insufficiency of effect descriptions in describing what remains unchanged after an action. The frame problem is also about efficiently updating an agent's knowledge about the world (Dennett, 1990). In AERA, command models and their preconditions are used for continuous prediction monitoring and reasoning at every time frame (Nivel, Thórisson, Dindo, et al., 2013).

[2]In the current implementation, there also exists a third mechanism called goal targeted pattern extractor (GTPX), which is not involved in any part of this thesis. The GTPX induces a new triad of CST, M_{req}, and CoM when the AERA agent reaches a goal state accidentally.

If any prediction fails in reality, the system is notified, and the models with incorrect predictions are revised and updated immediately. The predictions are made via temporal models, allowing the AERA agents to constantly and precisely predict what outcome will be observed at what time. Models in AERA have preconditions that focus only on a simultaneous subset of patterns that can be observed in a situation. In other words, AERA selectively pays attention to specific patterns and their values when making predictions. To achieve this, it uses fine-grained small models and pattern-matching principles (Thórisson, 2012), which are also used in learning and reasoning processes through backward and forward chaining. The ACTL mechanism provides a way to improve selective attention in AERA by enabling the system to learn to modify the models' preconditions and consider relevant properties when the models are created and used.

6.4 ACTL in AERA

The ACTL mechanism, explained in Chapter 5), cumulatively enhances the ampliative reasoning in OpenAERA and gradually improves an AERA agent's flexibility in dealing with novel tasks and situations.

In the standard OpenAERA, backward chaining from a M_{req} happens only if every component of instantiated CST on its LHS fully matches the current or predicted situations. Yet, this is a restrictive design decision, as some CSTs may incorporate irrelevant facts that do not fully match situations, limiting the applicability of learned CoMs during planning. Our ACTL mechanism can solve this over-specification issue in OpenAERA by inducing new, generalized pairs of composite state-requirement models (CST-M_{req}) for known CoMs (during planning) whose patterns fully match the newly encountered situations, allowing OpenAERA to learn to incorporate significant properties and facts in its model preconditions (Sheikhlar and Thórisson, 2024). The induced CST-M_{req}s are generated during backward chaining and used within the same planning process according to their priority. In other words, with ACTL integrated into OpenAERA, if some of the models' preconditions do not match the situations, an AERA agent can still commit to reasonable solutions to achieve its goals. The implemented C++ code of OpenAERA integrated with the ACTL mechanism can be found in [3]. This section is partly adapted from (Sheikhlar and Thórisson, 2024).

[3] See https://github.com/IIIM-IS/AERA/tree/analogy-during-backward-chaining - Accessed Jul. 14, 2024

6.4.1 Integrated mechanism for goal-driven learning and planning

Figure (6.1) illustrates the general procedure of how the integrated mechanism works in the OpenAERA system. The system activates the mechanism during backward chaining, which starts its chaining process from the top-level goal G_{top} towards commands backward in time (Sheikhlar and Thórisson, 2024). G_{top} is defined by the human task designer. The ACTL mechanism runs in parallel with the backward chaining in OpenAERA, starting when the G_{top} matches the RHS of existing CoMs, triggering the instantiation of CSTs on the LHS of their M_{req}s (Sheikhlar and Thórisson, 2024). Note that a CoM might have multiple M_{req}s, each with its own CST. Instantiation of CSTs leads to the creation of subgoals G_{sub}s, which must either match the observation *fact*s (current situation) or the RHS of other CoMs (predicted situations).

If the G_{sub}s generated from the instantiation of a M_{req} *fully match* the current or predicted situations, there will be no need for analogy and generalization, as in this case, regular planning can occur without having analogy models and generalization. In other words, if accurate CST-M_{req}s for goal achievement exist, the system will use those without creating new ones. However, if the CST-M_{req}s of a CoM do not fully match the situations, the ACTL mechanism creates new CST-M_{req}s for that CoM, pruning the non-matching components of the old CST-M_{req}s and keeping the matched components. The new CST-M_{req}s are then immediately used within the same backward chaining process, allowing the system to continue planning based on the existing models plus the new induced CST-M_{req}s.

Therefore, the new CST-M_{req}s generate new G_{sub}s during the backward chaining process, allowing the system to add additional plans for solving the task at hand. Plans are the solutions introduced in Chapter 5. Analogy-based generalization and backward chaining occur simultaneously, meaning that the analogies are made and CST-M_{req}s are induced while OpenAERA's planner chains backward from the G_{sub}s toward the initial situation S_{init}. Multiple CST-M_{req}s for the existing command models are created during backward chaining. As soon as the backward chaining process reaches S_{init}, all the analogy-based CST-M_{req}s (which must fully match the situations as only the matching components of their original CST-M_{req}s were kept) have already been induced. The CST-M_{req}s allow the planner to generate new plans for goal achievement with different priorities to be employed by the planner.

The planner explores the generated plans, the order of which is based on the plans' priorities (see Section 5.5.2). The first plan to be executed is the one with the highest priority. If the plan does not lead to the goal state, the specialization mechanism via learning from failure generates anti-CST-

M_{req}s (see Section 5.5.3), which restrict the use of analogy-based induced CST-M_{req}s. The plan is then removed from the list of generated plans, and the next plan is explored based on its priority. This procedure goes on until the goal state is reached, leading to the accumulation of positive evidence for the plan and the utilized CST-M_{req}s, thereby increasing the confidence in the CST-M_{req}s and their related CoMs. The algorithm of this mechanism is provided in Algorithm 1[4].

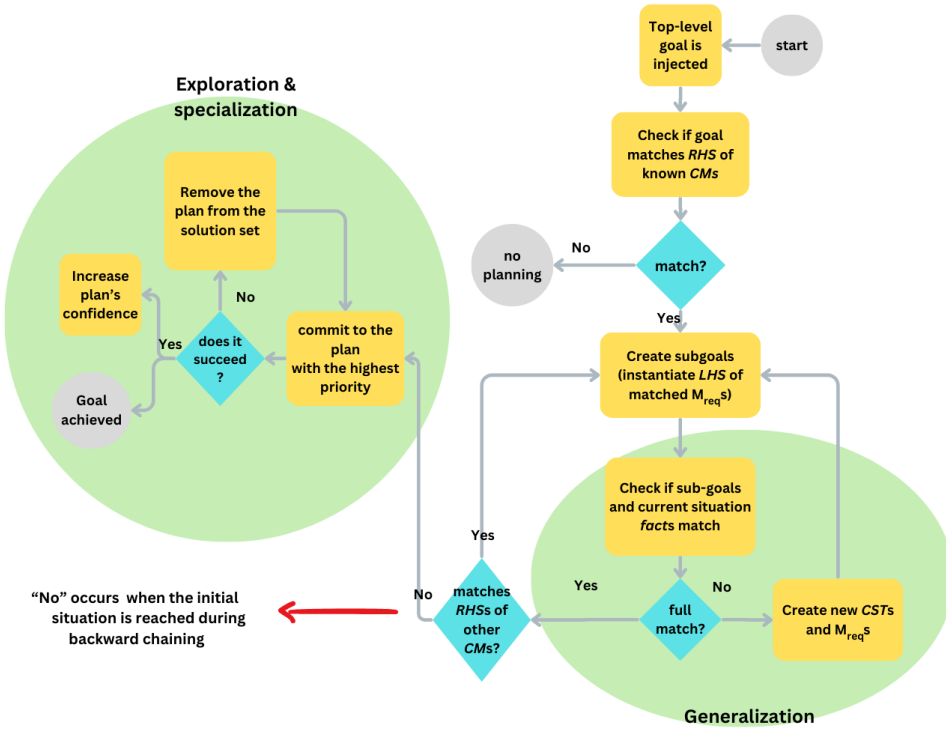

Figure 6.1: The integrated ACTL generalizes the preconditions of known models during backward chaining, allowing it to generate new solutions during the planning process. The created solutions are explored based on a confidence-based priority order.

[4]Note that the algorithm only shows a pseudocode of one method of implementing the ACTL mechanism, which is not limited to the generating preconditions of CoMs theoretically. Also, the algorithm does not entail all the mechanisms involved in planning and the parallel processing of information in the OpenAERA system.

Algorithm 1: ACTL-Based Planning in OpenAERA

Input: 1) The top level goal G_{top},
2) Initial causal and requirement models, $CoMs$ and M_{req}s, and composite states CSTs,
3) The current situation S's observation facts.
Output: Return the successful plan among other generated plans.

G_{top} is injected, setting G_{sub} to G_{top}
for *each* G_{sub} *that match RHS of* $CoMs$ **do**
  for *each matched* CoM **do**
    for *each* CST-M_{req} **do**
      if G'_{sub}*s fully match* S **then**
        Initial situation reached and plan generated
        Break
      if G'_{sub}*s partly match* S *and partly match RHS of* CoM' **then**
        Generate G''_{sub}s and setting G_{sub}s to G''_{sub}s
      if *Some* G'_{sub}*s neither matches* S *nor RHS of* CoM' **then**
        Generate new CST-M_{req} via analogy
        Generate new G''_{sub}s and setting G_{sub}s to G''_{sub}s
Assign priorities to each generated plan
for *each plan in order of priority* **do**
  if *the plan achieves* G_{top} **then**
    Involved CoMs and M_{req}s confidence update
    Break
  else
    Generate anti-CST-M_{req}s when the plan fails, removing the failed plan from the list of considered plans
    Continue with the next highest priority plan

6.4.2 Analogy in AERA

The analogy process happens by comparing the situation-goal pairs with the instantiated LHS and RHS of M_{req} (instantiated CSTs and CoMs) during backward chaining and identifying the matching and non-matching patterns. Other than the comparison, the analogy process prunes $CST-M_{req}$s by leaving out the non-matching patterns and subsequently generating new $CST-M_{req}$s with matching patterns, where fewer patterns in new $CST-M_{req}$s mean more general $CST-M_{req}$s that can be applied to more situations (Sheikhlar and Thórisson, 2024).

As illustrated in Figure 6.1, the ACTL involves reasoning backward from the G_{top} towards the initial situation (Sheikhlar and Thórisson, 2024). Chaining backward makes the *generalization* process shown in Figure 6.1 generates multiple $CST-M_{req}$ pairs for the existing CoMs. The new CSTs are the pruned versions of the original CSTs. The knowledge pruning occurs via the *comparison process* (see Figure 5.3) and then generalizing the preconditions where the non-matching patterns are left out. Analogy happens concurrently with OpenAERA's planning for goal achievement, leading to generating new models that can be employed in the same planning, allowing the agent to create new plans if necessary and immediately utilize them to infer new plans (a.k.a. solutions).

Comparison and knowledge pruning
First, it is checked whether G_{top} matches RHS of a known CoM. If so, the G_{sub}s (i.e., instantiated components of CSTs) must be compared with the current or predicted situation facts. Note that the predicted facts are instantiated RHS patterns of other CoMs. When a G_{sub} does not match a situation, the analogy process prevents the agent from adding the non-matching G_{sub}'s pattern to the newly created CST and M_{req}. This way, it creates a pruned and generalized version of the original CST and M_{req} from which G_{sub}s are initially created. The newly created CSTs resulting from analogy are the abstracted intersections between instantiated CSTs (i.e., G_{sub}s) and observed/predicted facts (similarities are kept), where the non-matching patterns are pruned (see Section 5.4 for details). In other words, a generalized CST consists only of similarities between model preconditions and the situation patterns.

Planning, exploration, and confidence computation
In the old version of OpenAERA, for a plan to be valid, each G_{sub} must either be achievable in predicted situations through a (set of) command(s) or match the observed facts in the current situation. However, our analogy mechanism releases OpenAERA from such a restrictive pattern-matching requirement. Creating new CSTs and M_{req}s via the analogy process creates

new plans when no valid plans exist. The mechanism permits committing to plans that only ***partially*** match the CSTs of known CoMs. This becomes possible when the analogy process creates new CST-M_{req} that fully match when planning and thus can be used right away. Different CST-M_{req} might have different confidence values. The confidence values of models are called ***success rate*** in the AERA framework (Nivel, Thórisson, Dindo, et al., 2013). The initial success rate of a CST-M_{req} is specified based on the familiarity ratio calculation described in Section 5.4 in detail. The ratio determines the familiarity of a CST-M_{req} in relation to the G_{top}-situation. This will help the AERA agent decide which plan to choose and explore considering the confidence of CST-M_{req}s involved in a generated plan (detailed in Section 5.5.2).

Analogizing chains of models
The G_{sub}s generated based on analogy-based induced $CST - M_{req}$s of a CoM can match the RHS of other CoMs. If some $CST - M_{req}$s of the other CoMs lead to generating non-matching G_{sub}s, the goal-driven analogy mechanism steps in again and induces new $CST - M_{req}$s whose all derived G_{sub}s match the situations. This means that as the backward chaining occurs, the analogy mechanism generates new $CST - M_{req}$s for different CoMs, allowing them to be employed for the agent's planning right away. The idea of pruning and then inducing multiple preconditions, each for a different CoM makes the agent's planning more adaptive and allows it to create more plans for the same task (See Section 5.4).

6.4.3 Learning from failure

The analogy process may generate overly generalized preconditions leading to failed predictions. The process of learning from failure occurs when the newly generated preconditions result in the failure of the plans. The failures trigger OpenAERA's $PTPX$ mechanism that creates new sets of preconditions (CST-M_{req}s), as described in detail in (Nivel, Thórisson, Steunebrink, et al., 2013). This will help our mechanism to prevent analogy-based, overgeneralized preconditions from being used in similar situations in the future. The generated CST-M_{req} via $PTPX$ are called ***anti-requirement***s $anti-M_{req}$s, as they state that under the new CST conditions M_{req} created via analogy does **not** hold anymore, according to the Equation (5.15). In other words, this mechanism helps to refine and specialize the incorrectly generalized analogy-based CST-M_{req}s by adding new conditions to the use of their referenced CoMs. Those new conditions are the non-matching patterns that were pruned during the analogy process, as detailed in Section 5.5.3.

6.4.4 Planning/exploration in AERA

In its planning, OpenAERA generates multiple plans (i.e., solution paths) to G_{top}s via backward and forward chaining. The backward and forward chaining leads to inferring the correct sequences of commands that need to be taken to achieve the G_{sub}s and eventually the G_{top}. In the standard OpenAERA system, no model creation, i.e., induction, exists within the backward and forward chaining process. Our ACTL mechanism, however, allows the OpenAERA agent to generate new CST-M_{req}s during its planning via backward chaining and test them in forward chaining. As mentioned in the previous sections, generating new CST-M_{req}s leads to the creation of a set of valid solutions the OpenAERA planner can commit to, each with confidence values between 0 and 1. The agent calculates these values via Equation (5.13), selects the most confidence while shorter path based on Equation (5.14), and issues the inferred sequence of commands. Suppose the plan does not lead to the G_{top}. In that case, the agent removes the plan from the solution set by reducing its confidence and learning from the failure based on anti-M_{req}s (see Equation (5.15), and tries an alternative plan it had created before, as illustrated in Figure 6.1. This way, the agent explores different hypotheses created by the generalization (i.e., analogy) part of the ACTL mechanism.

6.5 Summary

In this chapter, we discussed the integration of the introduced Autonomous Cumulative Transfer Learning (ACTL) mechanism into the AERA cognitive architecture to enhance its learning and reasoning capabilities. We choose OpenAERA, the main implementation of AERA framework, to evaluate our proposal. The ACTL mechanism introduces a way to create defeasible hypotheses by making analogies between the situation-goals and cause-effect patterns of known models. The chapter explained the details of implementation, starting with introducing the OpenAERA system, its knowledge representation, and its reasoning processes. The knowledge representation relies on components called command models (CoMs), composite states (CSTs), requirement models (M_{reqs}), and mechanisms for forward and backward chaining, are what the ACTL mechanism relies on initially. The ACTL mechanism extends OpenAERA's capabilities by deriving new preconditions, i.e., $CST - M_{req}$s, for CoMs from their old preconditions so that goal achievement, planning, and exploration of different plans become facilitated.

The analogy process involves creating new preconditions by pruning non-matching patterns, which enables an agent to generate new plans for the same task that can be explored based on their priority. The ACTL also allows

the agent to learn from failures by refining overgeneralized models and improving their predictions in future tasks. In other words, if the analogy-based models are generalized incorrectly, learning from failure will fix them by specializing them via learning other models (called anti-requirement models) that constrain their use and prevent wrong generalizations. By integrating ACTL, OpenAERA can make improved goal-driven analogies, thereby more effectively generalizing its known models beyond strict pattern-matching assumptions, allowing the system to perform more flexible and efficient planning and exploration in novel tasks and situations.

In summary, the chapter details the design and implementation of ACTL within the AERA framework, focusing on how this integration enhances the system's learning, reasoning, and planning capabilities through analogy, model generalization, and a systematic approach to learning from failures. This makes AERA more capable of handling novel, unpredictable environments and tasks.

Chapter 7

Results & Evaluation

7.1 Introduction

This chapter starts with demonstrating the effectiveness of causal models in providing task-independent knowledge generalization. Then, it describes a motor-skills learning task presenting a framework for evaluating the performance of the autonomous cumulative transfer learning (ACTL) (Sheikhlar et al., 2020) in the AERA system (Nivel, Thórisson, Dindo, et al., 2013). We will also compare the ACTL-based extended OpenAERA[1] with another general machine intelligence (GMI) aspiring architecture, the non-axiomatic logic system (NARS) (P. Wang, 1995, 2006). The experiments and results achieved with the extended OpenAERA system and a NARS-based implementation, OpenNARS for Applications (ONA) (Hammer and Lofthouse, 2020) are analyzed.

Disclosure: This chapter is partly adapted from our papers (Sheikhlar et al., 2021; Sheikhlar and Thórisson, 2024). Section 7.2 describes the presented algorithm and evaluation results of our paper (Sheikhlar et al., 2021). A part of the section 7.5 explains the *results and evaluation* section of our paper (Sheikhlar and Thórisson, 2024).

7.2 Invariant causal learning

Causality provides task-independent representations, which are information structures that do not readily change across task variations. In this section, a controller is presented that allows learning agents to identify the causal relations between observable properties and commands via systematic experiments and interventions. The controller uses the two types of causal

[1] See http://www.openaera.org — accessed Apr. 2, 2024.

discovery methods via input interventions, introduced in Definition 4.1 in Section 4.2, enabling the agents to remove spurious correlations from an inaccurate correlational model and eventually learn an invariant causal model. The robustness of the learned invariant causal model is tested through two dynamic tasks: Randesvous task and circular path following with mobile robots. The model generalizes beyond goals and initial conditions, even though the learned model's generalizability is limited to the arguments we provided in section 4.2. This section is partly adapted from the paper (Sheikhlar et al., 2021).

Causal discovery based on invariant causal models

The presented controller's algorithm relies on similar definitions of ***causal discoveries 1 and 2*** (See Definition 4.1 in Section 4.2). The difference is that the causal discoveries 1 and 2 in (Sheikhlar et al., 2021) call for inspecting the changes in probability distributions of the property values, as noises are considered on observables properties and commands to formulate uncertainty in the related task representation.

According to the definition, a causal influence exists between two different properties if changing the initial conditions of an observable property's value leads to a change in the distribution of another property's value, with the assumption that the trajectories of the rest of the values and command sequences remain the same, called *causal discovery 1.* Additionally, a causal influence exists between a command and an observable if changing a command sequence alters the distribution of that observable, assuming that the initial conditions of all observables and the trajectory of the rest of commands remain the same, called *causal discovery 2.* The algorithm 2 (Sheikhlar et al., 2021) provides the necessary conditions to perform causal discovery experiments, where CMD represents the command vector, V denotes the vector of observable properties' values, k represents the experiment, and A and B are the matrices in a suggested linear model class as $V(T) = A \cdot V(T-1) + B \cdot CMD(T-1)$ with T representing the time step (Sheikhlar et al., 2021). Also, p and q represent the number of observables and commands. Additionally, v_i and cmd_i denote the chosen observable and command for performing causal discoveries 1 and 2, respectively.

The model class $V(t) = A \cdot V(T-1) + B \cdot CMD(T-1)$ is a single state space equation. The aim of Algorithm 2 is to learn invariant dynamics A and B matrices through the causal discovery experiments mentioned earlier. Initially, the controller uses a correlational modeling method (i.e., via a standard system identification-based approach using least squares) to learn some initial A and B matrices (Sheikhlar et al., 2021)[2]. Over time, the algorithm

[2]To calculate the initial correlational model, we first excite the system with some control inputs to receive the outputs, which generates data to be fed to the black-box

Algorithm 2: Pseudocode of learning the invariant causal model (Sheikhlar et al., 2021)

Input: Correlation-based initial A and B as well as values of V and CMD from different initial conditions $(CMD^k, V^k|V^k(0))$

Output: Matrices A^* and B^* estimated through causal discovery experiments

```
while 1 do
    for i = 1:p do
        Causal discovery 1 by changing the initial conditions of v_i;
        Identify the properties whose distributions do not change;
        Eliminate spurious correlations between property values;
        Update A and B;
    end
    for j = 1:q do
        Causal discovery 2 by changing the commands on cmd_j;
        Identify the properties whose distributions do not change;
        Eliminate spurious correlations between commands and
         properties;
        Update A and B;
    end
    if mean squared prediction error ≤ γ then
        Break;
    end
end
```

refines this model by identifying the properties whose distributions do not change, leading to detecting and thus removing spurious correlations from A and B and estimating the actual dynamics A^* and B^*. Spurious correlations are the learned correlations between observables and/or commands, which do not causally influence each other. The spurious correlations can be detected by the two different causal discovery experiments. Causal discovery 1 requires the causal experiments to be performed from different initial conditions while keeping the same command trajectories; the algorithms move the task to different initial conditions for different experiments. However, causal discovery 2 requires the causal experiments to be performed from the same initial conditions with different command trajectories. The state space equation uses the linear quadratic tracking controller to solve the tracking control problems for the required trajectories. The model becomes updated as experiments are performed.

Discussion of results

For evaluating the presented causal discovery algorithm and invariant model, two tasks are designed: 1) Rendezvous task, where four mobile robots with eight distinct commands and eight positions (as the robots are simulated in an x-y plane, each robot's position and command has x and y dimensions) use a centralized controller to learn an invariant robot movement model containing the causal relations between the control inputs and positions, which eventually makes the robots perform the task of all robots meeting each other at the goal position; 2) Path following, where single robots use the same invariant causal model learned in Rendezvous task to follow a circular path. After performing the causal discovery experiments, the robots were tested with novel command sequences and initial conditions not used during the training process (i.e., during the causal discovery experiments). This was considered for both tasks. The tasks are depicted in figure (7.1).

The comparisons are made between a correlational model learned by a system-identification-based method (for further details, please see (Sheikhlar et al., 2021)) and the invariant causal models learned through Algorithm 2. Assuming that the state space equation $V(T) = A \cdot V(T-1) + B \cdot CMD(T-1)$ represents the relations between robots' command inputs and position values, the invariant model the robots eventually learn through Algorithm 2 is $V(T) := I \cdot V(T-1) + I \cdot CMD(T-1)$ where $CMD = [{cmd_1}^x, {cmd_1}^y, ..., {cmd_4}^x, {cmd_4}^y]$ represents the command value vector; $V = [x_1, y_1, ..., x_4, y_4]$ is the position value vector containing the robot's location on x-y plane; I is the identity matrix, meaning that the position of each robot is influenced solely on the its previous position and its command in-

system identification method. The system identification method provides us with the matrices A and B that describe the behavior of data

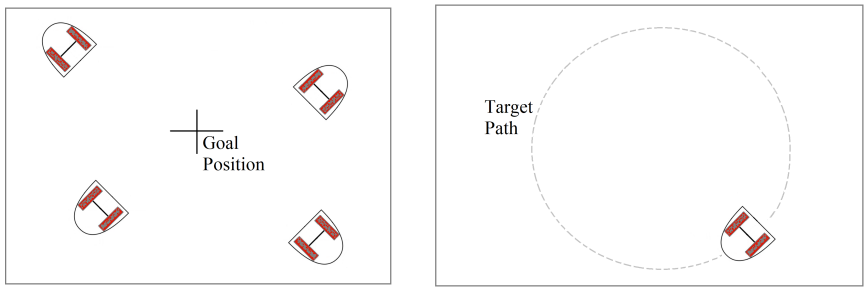

Figure 7.1: **Left**: The Rendezvous task includes four robots that use the learned causal model with removed spurious correlations to meet each other at the goal position. Before performing the tasks, the robots learn the causal model via the experiments done by the proposed algorithm. **Right**: A single robot uses the same model learned in the first task (Rendezvous task) for performing the new task of following a circular trajectory.

put. Simply put, Algorithm 2 will eventually learn that a robot's position and control inputs do not affect the positions of other robots, assuming that no collisions exist between them, which is the knowledge learned through systematic input interventions. It is important to note that the algorithm starts performing causal discovery experiments based on a correlational model (as its initial knowledge) it learns through a system identification method. The correlational model incorporates spurious correlations in A and B matrices, which Algorithm 2 removes through causal discovery experiments (replacing non-zero with 0 values) and keeps the causal relations between robot commands and positions. The causal influences between control input and position properties are inspected by the maximum mean discrepancy (MMD) method, which captures the changes in the distribution of the property values in two different experiments. The results of comparisons, as shown in the figure (7.2) (Sheikhlar et al., 2021), demonstrate that the learned causal model has a significantly lower prediction error than the correlational model.

The results of (Sheikhlar et al., 2021) empirically prove our *first research hypothesis* (see Section 1.3) that a causal knowledge representation and discovery is a proper foundation for agents solving different variations of learned tasks with different goals and initial conditions. In other words, causal models improve the adaptability of agents when facing novel tasks, unlike representations based on correlation. The results also demonstrate how pruning irrelevant constraints (in this particular case, the spurious transition functions) from models improves their accuracy and performance, solving the knowledge-pruning problem detailed in Section 4.4.

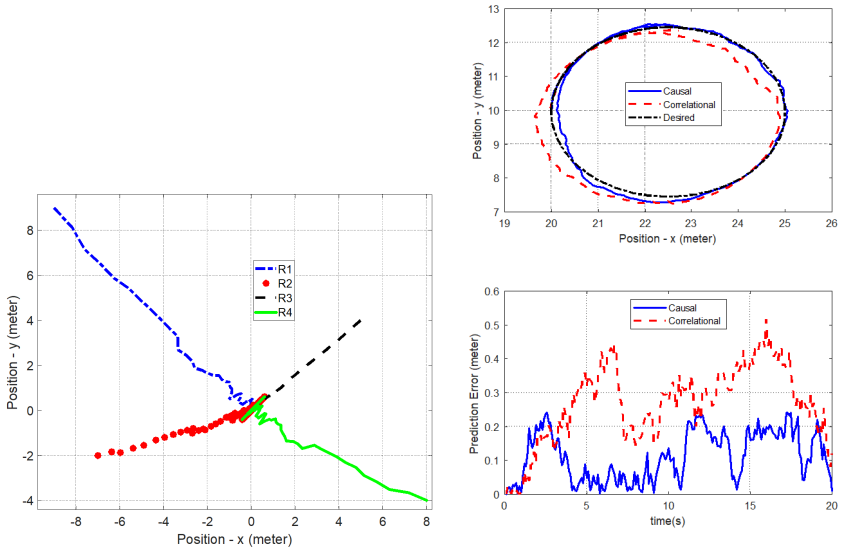

Figure 7.2: **Left**: Once the robots learn the invariant causal model, they perform the Rendezvous task based on the learned model. The initial positions from which the robots begin moving towards each other (to meet at $(x, y) = (0, 0)$) are novel initial conditions for the robots that did not exist in their training, showing the model's generalizability beyond initial conditions. The oscillations in the robots' movements are due to the measurement noise of the positions. **Right**: The invariant model learned in the Randevsous task is compared with the correlational model. The comparison is made by testing the model's performance in the new task of the circular path following. The results show that the invariant causal model is more effective in terms of predictability in a new task. - *figure from* (Sheikhlar et al., 2021)

Yet, the task representation in this work is a single dynamic state space equation that connects the observable properties to related commands and is assumed to be invariant. As we argued in Section 4.2, this representation is limited due to the assumptions it is based on. However, it allows us to assess the effectiveness of causal model classes in providing task-independent knowledge generalization in dynamic tasks. Note this algorithm has yet to be implemented in the OpenAERA architecture but was tested separately.

7.3 Motor skills learning

In Section 7.2, we investigated the importance of causal representation for task-independent generalization. To assess the causal generalization capabilities of general-purpose AI agents, we design object manipulation tasks that require motor skills learning, as engaging in activities that require motor skills can help agents observe, learn, and generalize cause-and-effect relationships. The candidates for the tasks, OpenAERA[3] and ONA (Hammer and Lofthouse, 2020) agents, must generalize basic motor movements and object manipulation skills, such as grasping items with different properties and under different circumstances. We assess each agent's ability to generate general causal hypotheses and use them in its learning and task execution process. Moreover, we evaluate the agents' ability to adapt to varying circumstances and their capacity to create and use their hypotheses. The chapter also investigates how AERA (Nivel, Thórisson, Dindo, et al., 2013) and NARS's (P. Wang, 1995, 2006) generalization mechanisms allow their architecture to comprehend the relevance of contextual factors, such as properties and environmental conditions, in determining the appropriate motor actions to employ.

7.4 NARS

In this section, we evaluate the generalization capabilities of Non-Axiomatic Reasoning System (NARS) a simple motor skills learning task. The NARS theory, developed by Pei Wang (1995), concerns the ability to reason under AIKR. NARS is based on non-axiomatic logic, meaning that the rules a NARS-based system learns or bootstraps its reasoning from are falsifiable, allowing the system to autonomously adapt its knowledge and reasoning in new contexts. NARS has different layers of inference, ranging from inheritance to variables, events, and temporal implications. It uses several inference rules, including analogy, comparison, deduction, abduction, induction, and revision, enabling the system to make hypothetical inferences and reflect on its internal operations (P. Wang, 1995, 2006).

We choose OpenNars for Applications (ONA) (Hammer and Lofthouse, 2020), a well-known implementation of NARS theory, for performing our motor skills learning experiments. Although there are differences in knowledge representation, OpenAERA and ONA share similar principles, such as ampliative reasoning and the AIKR detailed in Section 2.5. Therefore, ONA is considered suitable for a comparative evaluation, as its design has also been inspired by the AERA architecture (Hammer and Lofthouse, 2020).

[3]See http://www.openaera.org — accessed Apr. 2, 2024.

ONA solving motor skills learning task

We assign the ONA agent to solve a robot arm object-picking task. In this task, two items exist, each with two properties. The items are a cube and a sphere, with the cube being small and lightweight and the sphere being large and lightweight. So, the items' weight is identical, but their sizes are different.

Let us assume that first, ONA receives inputs for the cube. The *grasp* operation on a lightweight, small cube leads to the cube being *picked*. The required Narsese predicates that represent this are as follows

$$< cube \rightarrow [lightweight \quad small] > . : | :$$
$$< (SELF * cube) \rightarrow \hat{} grasp > . : | :$$
$$< cube \rightarrow [picked] > . : | :$$

The above inputs are given to ONA in order. The initial frequency and confidence values of each statement are 1 and 0.5. The terms inside the brackets, i.e., *lightweight*, *small*, and *picked*, represent the properties. The sign : | : at the end of each statement denotes a tense and signifies that the statement is about the current time. Also, signifies an operation. The first statement is a declarative assertion about the properties of the term *cube* with a set denoting that the cube is both *lightweight* and *small*. The second statement describes a *grasp* operation. The $(SELF * cube)$ is a compound term representing the relationship between $SELF$ and *cube*, where $SELF$ represents the agent taking the grasping operation. The third statement means that the *cube* is picked. Then, ONA induces multiple hypotheses from these events, three of which are the following temporal implications

$$< (< \$1 \rightarrow [lightweight \quad small] > \&/ < (SELF * \$1) \rightarrow \hat{} grasp >) =/> < \$1 \rightarrow [picked] >> .\%1.0, 0.24\% \quad (1)$$

,

$$< (< \$1 \rightarrow [lightweight] > \&/ < (SELF * \$1) \rightarrow \hat{} grasp >) =/> < \$1 \rightarrow [picked] >> .\%1.0, 0.2\% \quad (2)$$

, and

$$< (< \$1 \rightarrow [small] > \&/ < (SELF * \$1) \rightarrow \hat{} grasp >) =/> < \$1 \rightarrow [picked] >> .\%1.0, 0.2\% \quad (3)$$

which all state that grasping some variable \$1 will lead to appearance of a new property for the variable, i.e., the variable being picked. Such ***abstraction***, which is the replacement of the subject *cube* with variable \$1, is the

result of the sixth NARS's inference layer (NAL-6), making the statements more abstract while keeping the same structure (Hammer, 2021).

The generation of temporal implication hypothesis results from the observation of events in chronological order. As the event $< cube \rightarrow [lightweight\ small] > . : | :$ occurs before $< (SELF * cube) \rightarrow \hat{}grasp > . : | :$, which itself occurs before $< cube \rightarrow [picked] > . : | :$ ", then NARS uses its procedural inference on its eighth inference layer to generate the temporal implications (1-3) that can be used by the ONA agent to achieve its goals if needed.

Numbers 1.0 and 0.24 in the first statement and 1.0 and 0.2 in the second and third statements are the frequency and confidence values, respectively. According to the hypothesized temporal implications (1-3), the precondition for the grasp operation is that the object must be lightweight and/or small, where the **and** case has higher confidence than the **or** case. These temporal relations are general statements based on a "weak inference" as defined by Wang (1995), which means that their confidence value is considerably small due to being generated by NARS's induction rule. The revision rule, however, allows ONA to make strong conclusions due to accumulated weak conclusions. Therefore, if ONA had received the first three Narsese statements again in the same order, the confidence, frequency, and, thus, the truth expectation (a combination of confidence and frequency values) of the hypothesized temporal relations would have increased significantly.

Let us assume that, later, the ONA agent receives inputs for another term, a *sphere* that is also *lightweight*. But its difference from the *cube* is that it is *large*. After giving input statements about the properties of the *sphere*, the goal of the *sphere* to be *picked* is given as another input to ONA. All the given inputs are a set of NARS events, given as the following Narsese statements to ONA in order:

$$< sphere \rightarrow [lightweight \quad large] > . : | :$$
$$< sphere \rightarrow [picked] >! : | :$$

The sign ! signifies a *task* given to ONA. The task leads to the inference of the following operation under the precondition $[lightweight]$:

$$< (SELF * sphere) \rightarrow \hat{}grasp > . : | : \%1.0, 0.9\%$$

resulting from the following hypothesized temporal implication:
$< (< sphere \rightarrow [lightweight] > \&/ < (SELF * sphere) \rightarrow \hat{}grasp >) =/>$
$< sphere \rightarrow [picked] >> .\%1.0, 0.2\%$

As both *sphere* and the *cube* have the same *lightweight* property, ONA infers the following analogy relation indicated by $\leftrightarrow$ that means the cube and the sphere are alike,

$$< sphere \leftrightarrow cube > .\%1.0, 0.24\%$$

This is a derived hypothesis based on the evidence. However, it is not utilized to infer that the *grasp* operation must occur. The assigned goal achievement task makes ONA infer the *grasp* operation given a piece of evidence (i.e., lightweight-ness and smallness of the sphere) that matches the preconditions of the temporal implication (2) it has already induced. It takes the $\langle$**sphere** $\rightarrow$ [**lightweight large**]$\rangle$. : | : event and decomposes it to $\langle$**sphere** $\rightarrow$ [**lightweight**]$\rangle$. : | :, which now matches $\langle \$1 \rightarrow$ [**lightweight**]$\rangle$ in the general temporal relation (2). Note that the lightweight-ness property has received two pieces of positive evidence while the small-ness has received one; thus, smallness is not considered in the preconditions. More precisely, in the inference process, multiple competing temporal implication hypotheses exist, e.g., hypotheses (1) and (2), some of which have preconditions that match the input events received, while others do not. The evidence and, therefore, their truth expectation (combined confidence and frequency values) will help them to fight the others out, depending on the positive and negative evidence they receive. In other words, NARS identifies the hypothesis that does not depend on smallness to be higher in truth expectation than the competing options.

To have a more complete analysis, we consider two scenarios for evaluating the cumulative reasoning of ONA, as follows.

- In the ***first scenario***, the hypothesized precondition [*lightweight*] and operation *grasp* turn out to be correct as ONA receives positive evidence for them by getting the following observation input event right after the *grasp* operation is applied

  $$< sphere \rightarrow [picked] > . : | :$$

  through which the ONA agent verifies the temporal implication (2), whose frequency and confidence values will be updated with the above input and therefore gains a higher truth expectation due to collecting positive evidence.

- In the ***opposite scenario***, if grasping the *sphere* does not lead to the *sphere* being *picked*, ONA collects negative evidence for the temporal implication (2). This occurs when ONA does not receive the above input event. This, of course, may not immediately change the lightweight-ness as the only precondition for the temporal implication (2). In fact, ONA needs multiple negative pieces of evidence to revise the hypothesis and infer that **both *lightwieght* and *small*** are necessary preconditions for the *grasp* operation. Note that ONA

does not add smallness to the hypothesis of having lightweightness. ONA had already created the hypothesis of having both smallness and lightweightness. But it had a lower priority, as it had not received enough positive evidence. On the other hand, the hypothesis of only having lightweightness was already created and received positive evidence since a sphere is lightweight. However, as it received multiple negative evidence, its frequency was reduced over time, and its priority decreased until it went below the priority of the hypothesis that had both lightweight and smallness. This shows that when multiple failures happen, and thus enough negative evidence is collected, ONA considers both *lightweight*-ness and *small*-ness properties as preconditions, both properties of the *cube* it received once.

Evaluation and discussion
This experiment shows how NARS extends its hypothesized temporal implications when it receives tasks. Here, we provide an analysis of the similarities and differences between NARS's representation and inference axioms and the ACTL-based AERA's.

- **Generalization**: Generalization in NARS partly happens through its higher-order inference. For instance, it performs abstraction over subjects of statements by replacing terms with variables, similar to \$1 variable in the temporal implications (1-3). It also uses its procedural inference to hypothesize temporal implications such as (1-3). These two features are similar to how AERA performs abstraction and induction. AERA replaces task entities and values with variables when creating model patterns and generates command models with their preconditions. Abstraction is done via the two existing learning mechanisms in standard OpenAERA, *CTPX* and *PTPX*. However, the integrated analogy mechanism via ACTL also involves abstraction when creating new $CST\text{-}M_{req}$s. Additionally, NARS may decompose the sets of preconditions (e.g., *lightweight* and *small*) correlated with terms to generate separate temporal implications, similar to the temporal implications (2) and (3). The integrated AERA generating multiple separate $CST\text{-}M_{req}$s based on distinct properties is the subject of future work.

- **Planning**: In NARS, temporal implications are not considered causal relations but rather the closest notion to them. In the task above, the *grasp* operation in NARS may not necessarily cause an object to be *picked*. This is because the object may be picked at any time after the *grasp* operation. The longer the time gap between the *grasp* operation and the *picked* event, the lower the confidence the hypotheses (1) and (2) will have. This would be a sensible inference when

dealing with queries about single operations but not about a sequence of events/operations that require precise timing. However, the lack of an explicit time representation in Narsese limits a NARS-based agent's ability to precisely predict when and at what time an event, e.g., $\langle \text{sphere} \rightarrow [\text{picked}] \rangle$. $: | :$, will be observed, which is required for planning command sequences with precise timing. Furthermore, the induced preconditions for OpenAERA's command models are assumed to be a set of patterns observed simultaneously. This design decision was also considered when integrating the ACTL mechanism with the OpenAERA. The ACTL-based AERA generates new CST-M_{req}s for different CoMs representing different behaviors capable of being chained effectively, as the patterns of the induced CSTs are simultaneous precondition facts, allowing the planner to use the related CST-M_{req}s for the right CoMs during the planning as soon as they are induced. Yet, the preconditions in NARS can be a set of input events received at any time before an operation occurs. This may lead to inaccurate selective attention (see Section 4.4), as the precise timings of precondition patterns are significant for creating subgoals over abduction, which must be achieved in the proper order and at the right time.

- **Analogy**: In 7.4, it was noted that the similarity relationship between a *sphere* and a *cube*, i.e., $sphere \leftrightarrow cube$, is a derived relation and not considered in the ONA's generalization of temporal implications. Wang (2009a) highlights that NARS is capable of handling various types of analogical reasoning, and anything that can be substituted for anything else is considered analogous. However, unlike AERA, NARS does not necessarily involve an analogy rule in generalizing its known temporal implications. As demonstrated in the example, generalization in NARS results from a set of competing hypotheses, among which the one that receives more positive pieces of evidence is selected. NARS generates analogy relations but does not use them to drive other hypotheses. It is a limitation in that it needs to take advantage of the potential of analogy-making to infer new temporal implications as it accumulates evidence while trying to achieve the task's goals. The induced temporal implications (1-3) are generated irrespective of whether they will be useful for an agent's goal achievement. On the other hand, ACTL-based AERA takes a more efficient approach by using goal-driven analogy to make new inferences and induce multiple preconditions for command models when they are needed for goal achievement. Some of these preconditions are tried immediately to check if the new hypotheses hold. If not, it would im-

mediately remove the plan that relies on them and try another plan that initially had lower priority.

The experiments we did with ONA show how NARS can generate hypotheses and refine its understanding based on evidence received during the task. Our comparative analysis between AERA and NARS shows the different approaches to causality, planning, and analogy, while it also shows similarities in abstraction processes. As we will see in the next section, the ACTL-based AERA does not need to gather multiple negative evidence to refute a hypothesis, as NARS does. Instead, it uses analogy-based hypotheses to immediately use a plan and learn from failure to falsify a hypothesis. When these plans/hypotheses fail, they can be quickly restricted or removed from the solution/plan sets, and other hypotheses are then tested, which results in faster and more efficient planning.

7.5 AERA

In this section, we test and analyze OpenAERA's[4] capability to generalize its model preconditions using the introduced ACTL mechanism. The evaluations are done through two robotic experiments in the context of motor skills learning, both of which have been tested in a realistic robot simulation environment called Webots[5] (Michel, 2004), as illustrated in Figure 7.3. In the first experiment, called *SizeMattersGrab*, the OpenAERA agent, after generalizing its models, learns that an important property that should be considered when choosing the proper strategy to grasp the items is their size. This learning occurs through pruning irrelevant patterns from the precondition sets of the related models via the implemented analogy mechanism. In the second experiment, called *AlignGrab* experiment, during the object grasping, the robot arm fails to successfully pick up a long rectangular item, after which it learns from this experience that the right strategy to grasp rectangular items is first to adjust the hand's orientation and align it with the items before attempting to grasp them. The links to demos of both experiments can be found in [6] and [7]. The figures in this section are snapshots of models, facts, commands, and inferences in OpenAERA's Visualizer[8], a software designed to demonstrate the details of knowledge components and reasoning processes of OpenAERA in an experiment. The snapshots show the results of the performed experiments (*SizeMattersGrab* and *AlignGrab*) with OpenAERA. For better reproducibility of the results, the links to the

[4] See http://www.openaera.org — accessed Apr. 2, 2024.
[5] https://cyberbotics.com/ available on date 3/29/2024.
[6] https://youtu.be/JXgdSjU-7OI available on date 3/29/2024.
[7] https://youtu.be/ceYjyiyBBsU available on date 3/29/2024.
[8] https://github.com/IIIM-IS/AERA_Visualizer – available on date 3/29/2024.

Replicode codes of these two experiments [9], along with the C++ codes of the Webots controller design [10] are provided in the footnotes. This section is partly adapted from our paper (Sheikhlar and Thórisson, 2024).

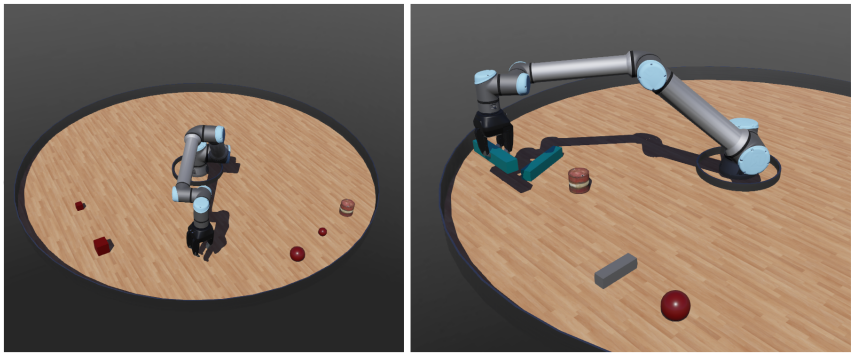

Figure 7.3: **Left** shows the SizeMatterGrab experiment where the OpenAERA agent uses the goal-driven analogy to come up with the right plans for grabbing items with the correct grab types it learns through initial training. The autonomous hypothesis generation enables the AERA agent to learn to open the arm's gripper's finger proportionate to the size of items when grabbing them. **Right** shows the AlignGrab experiment where the OpenAERA agent, after failing in a grasping experience, learns that the correct way to grasp long rectangular items is to adjust the hand's orientation first and then align the hand with the item before grasping it.

7.5.1 SizeMatterGrab experiment

The details of the"SizeMatterGrasp" experiment are illustrated in figure (7.4) (Sheikhlar and Thórisson, 2024). First, we teach a robot hand (h) to move towards a *small* cube ($c1$) and a *large* cube ($c2$) by a *move* command and issue *grab_type_*1 and *grab_type_*2 commands to pick up the items. This training happens in the guided experimentation phase. The *grab_type_*1 command opens the robot arm gripper slightly and grasps $c1$, whereas *grab_type_*2 opens the gripper widely and grasps $c2$. The observable properties are as the following facts ($c1$ *shape cube* T), ($c1$ *size small* T), ($c2$ *shape cube* T), and ($c2$ *size large* T). During the Task Solving phase, the goal is to grasp and then release new items, sphere $s1$, sphere $s2$, and cylinder *cyl*, one after the other. We will see that the only known property of these items, their *size*, will guide the

[9] https://rb.gy/csn554 and https://rb.gy/pite58 available on date 3/29/2024.
[10] https://rb.gy/h6c0rw and https://rb.gy/fpnm5x available on date 3/29/2024.

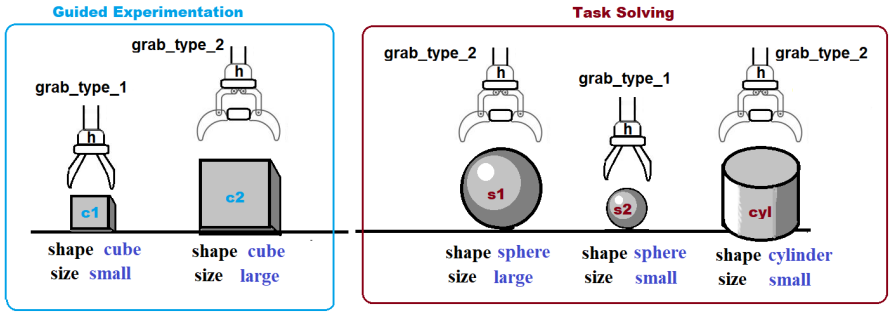

Figure 7.4: In SizeMatterGrab experiment, the significance of the property value of *size* is learned by pruning the shape-related property from the preconditions for grabbing while keeping the size within the preconditions, as the size's values (small and large) match the properties of the items in the Task Solving phase. - *figure from* (Sheikhlar and Thórisson, 2024)

OpenAERA agent in selecting the appropriate grabbing type. The section is adapted from the Results & Evaluation section of our paper (Sheikhlar and Thórisson, 2024).

Guided experimentation[11]

During the first phase of the experiment, called *Guided Experimentation*, we teach h to perform specific motor actions, including moving the hand using the *move* command, releasing cubes $c1$ and $c2$ via a *release* command, and grasping the cubes using the "$grab\_type\_1$" and $grab\_type\_2$ commands when the h is in the same position as the cubes. This leads to the induction of the two sets of CoMs, CSTs, and M_{req}s by the CTPX mechanism of OpenAERA due to the change of the value of the hand h's boolean property *holding* from the fact (h *holding* *[ ]* $T1$) to the hand holding something, e.g., (h *holding* *[c1]* *T2*) by the grab commands. For illustration simplicity, Figure 7.5 only shows one of the triads that belong to $grab\_type\_2$ command, learned after h applying the $grab\_type\_2$ command.

Note. The ACTL mechanism is not involved in the guided experimentation phase. However, it must use the triads created in the Guided Experimentation phase to learn additional CSTs and M_{req}s for the same CoMs in the Task-Solving phase.

The command model mdl_514 (shown in Figure 7.5) is a CoM that can be used for reasoning under the preconditions represented by the shown

[11] This part provides a detailed description of the Results and Evaluation section of our paper (Sheikhlar and Thórisson, 2024).

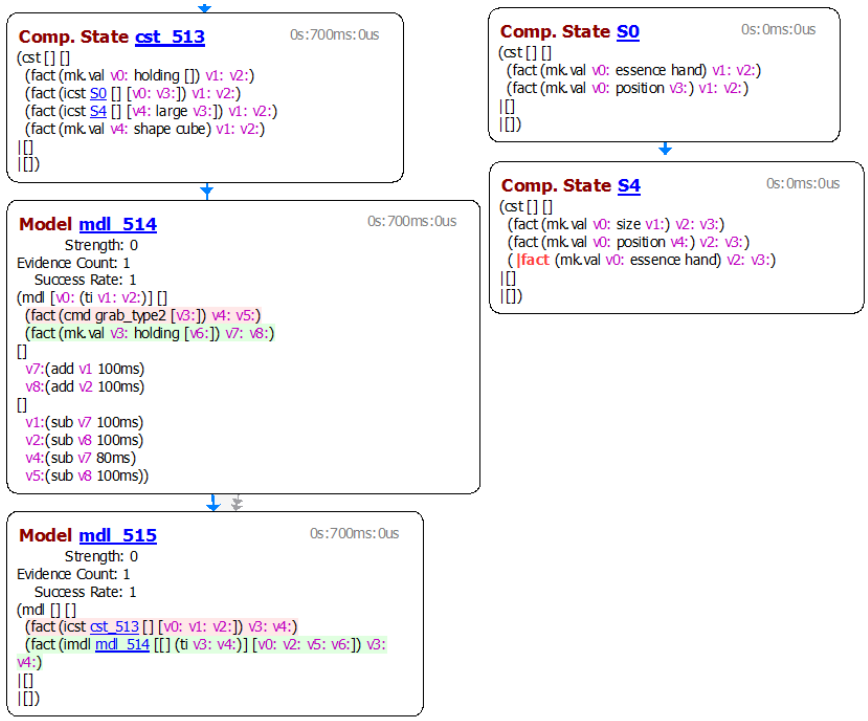

Figure 7.5: The CST (cst_513) and M_{req} (mdl_514) determine the circumstance of using the CoM (mdl_514). The models' strength increases when the positive evidence count surpasses a threshold.

CST in the same figure, cst_513, once both mdl_514 and cst_513 are instantiated through mdl_515, which is an M_{req}. The cst_513 is a nested CST involving other CSTs, $S0$ and $S4$. $S0$ represents some variable $v0$ with the *essence* of *hand* and position $v3$. The variables $v1$ and $v2$ in $S0$ are timing variables representing a time interval. $S4$ represents the state of some variable $v0$ having the size $v1$, position $v4$ and NOT having the essence of *hand*. Therefore, $S0$ and $S4$ together in cst_513 mean that there is a hand (indicated by $v0$) at the same position (indicated by $v3$) as something *large* with *cubic* shape that is not a hand (indicated by $v4$). The variables $v1$ and $v2$ represent the time interval, which, as can be seen, is the same for all facts in a composite state. An instance of the abstraction process here is the entity $c2$, which is replaced with $v4$.

Also, cst_513 is a result of an initial selective attention process in OpenAERA that aggregates all the properties of entities involved in the tran-

sition represented by the command model mdl_514. The model mdl_514 shows a transition, stating that issuing $grab\_type\_2$ command at a time interval $[v4, v5]$ leads to the new value of a property of h (abstracted with $v3$) holding something $v6$ at a later time interval $[v7, v8]$. The numerical operations in mdl_514, such as $v7$:(add $v1$ 100ms), represent the forward and backward timing equations from one time interval to another, where the sampling time is $100msec$. The observation facts based on which the mentioned CSTs and M_{req}s are created are demonstrated in Figure 7.6, which shows that the $grab\_type\_2$ command alters the value of property *holding*, enabling the induction process by the $CTPX$ mechanism.

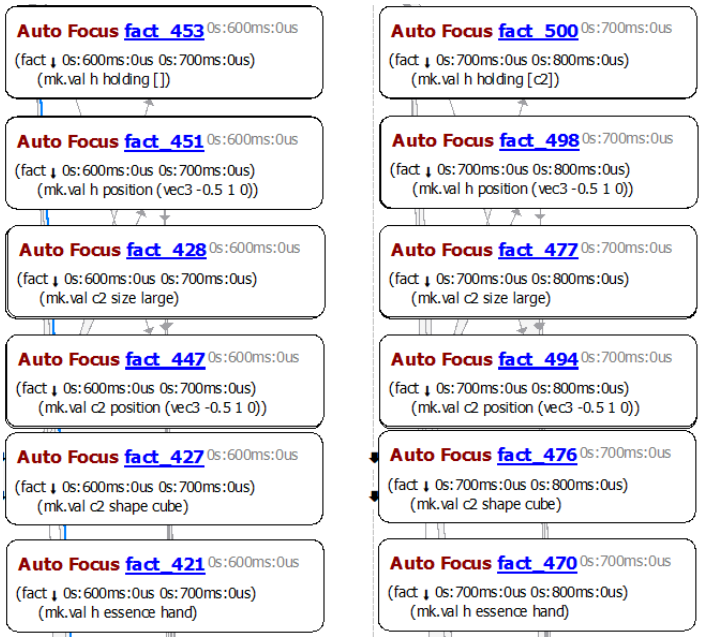

Figure 7.6: The facts in the right column are the observations that come after those in the left column. The positions of h and $c2$ are $vec3$ properties that represent the 3D positions of all objects in the Webots simulation environment. The size, shape, and essence are the other defined properties for the entities, h and $c2$ that have ontological symbolic values.

Task solving phase[12]

In the Task-Solving phase, the OpenAERA agent must grab and then release three novel items one after another based on the goal facts it receives.

[12]This part is adapted from our paper (Sheikhlar and Thórisson, 2024).

The items of this phase, $s1$, $s2$, and *cyl*, are novel to the agent, as the learned CoMs in the guided experimentation phase have *cube* shape values in their CSTs (see cst_513 in Figure 7.5). However, $s1$, $s2$, and *cyl* have *sphere* and *cylinder* shapes, which are non-matching property values. In other words, their related observation facts ($s1$ *shape sphere* T), ($s2$ *shape sphere* T), and (*cyl shape cylinder* T) do not match cst_513 and the CST related to $grab\_type\_1$. However, the analogy mechanism integrated with OpenAERA allows it to prune the non-matching shape-related fact from the cst_513 and induce a new $CST\text{-}M_{req}$ that matches the property values of the novel items in the Task Solving phase. Also, the OpenAERA agent will be able to infer the correct grabbing type for each item based on its familiarity, which is the size-related matching value, e.g., being *large*. This inference happens when the top-level goal facts are injected. The first goal fact is (h *holding* $s1$ T), which, when injected into the system, triggers the planning process through backward and forward chaining processes, as shown in Figure 7.7 (the yellow objects).

Improved selective attention[13]
As discussed above, cst_513 does not allow mdl_514 and the $grab\_type\_2$ command to be utilized for grabbing items with non-cubic shapes, because a non-cubic shape, e.g., *sphere* or *cylinder*, does not match the shape value considered in the composite state cst_513 (the shape must be *cube* according to cst_513). Our integrated analogy process releases the standard planning system of OpenAERA from this strict pattern-matching requirement via inducing new $CST\text{-}M_{req}$s with pruned patterns during the planning process.

As shown in Figure 7.7 at the bottom, the current observation facts (referred to as ***autofocus*** facts) at 800ms show that $s1$, which is targeted to be held by the hand, is a *sphere*. The *sphere*, however, does not match the shape value considered in cst_513. On the other hand, $s1$'s *large size* matches the size value in cst_513. Due to the analogy process, the ACTL mechanism prunes the fact of cst_513 that does not match the goal-situation properties, i.e., the *shape* in this case. Yet, it keeps the matched fact, which is related to the *size* property. Then, after the matching facts are identified and kept while the non-matching facts are pruned, the analogy process induces a new $CST\text{-}M_{req}$, which are cst_581 and mdl_582, shown at the top of Figure (5.4) with pruned patterns, where the shape-related fact is eliminated. This happens during the planning process shown by the yellow objects in Figure 7.7 at the bottom. Now, all facts of cst_581 and mdl_582 match the current observation facts. To put it simply, the newly created $CST\text{-}M_{req}$, which are cst_581 and mdl_582, mean that to grab an object using $grab\_type\_2$, the item must be large, regardless of any other properties it may have. Then,

[13]This part is also adapted from our paper (Sheikhlar and Thórisson, 2024).

*cst*_581 and *mdl*_582 are tested in practice over the planning process. Their related confidence (called *success rate* in AERA) increases as they succeed. This means that the ACTL mechanism, via analogy-based knowledge pruning and direct experience, helps the agent learn to disregard the properties that are insignificant for goal achievement by generalizing the preconditions of known CoMs and pruning the patterns of their CSTs to only include the *familiar* properties. In this case, the *size* is a familiar property that *cst*_581 selectively keeps out of other non-matching properties like *shape*. Note that the value of size (i.e., *large*) in the original CST, *cst*_513, is kept in the derived CST, *cst*_581.

Flexible planning in task solving[14]

The *mdl*_582, which is generated through the ACTL mechanism after knowledge pruning, allows the old command model *mdl*_514 to be used (i.e., instantiated) under the generalized conditions specified by *cst*_581. The OpenAERA agent utilizes the new preconditions right away during planning and issues the proper sequence of commands, first moving towards $s1$ and then applying *grab_type*_2. This sequence of commands is shown in Figure 7.7 at the bottom.

The goal state/fact is where planning (backward chaining) begins, matching the *mdl*_514. This instantiates the components of *cst*_513 through *mdl*_515, which generates subgoals, such as *(mk.val h position (vec3 -1.2 -0.1 0))*[15], meaning the goal position of the hand (h) must be *(vec3 -1.2 -0.1 0)* to achieve the top-level goal of *(h holding [s1])*, as h and s are not at the same position (see *fact*_554 and *fact*_550). The top 5 yellow objects in the figure pointing backward represent backward chaining processes, while the remaining objects pointing forward represent forward chaining processes. OpenAERA performs forward chaining to validate the commands inferred during backward chaining. As shown, it verifies two commands: *(cmd move [h (vec3 -0.7 -1.1 0)])* and *(cmd grab_type2 [h])*, indicating that the hand (h) must first move toward the sphere (*s1*) and then grab it to achieve the goal of the h holding *s1*.

Note that the analogy process uses similarity in the cardinality of patterns (SSP_C) criteria, detailed in Section 5.4, for familiarity computation before inducing *cst*_581 and *mdl*_582. The familiarity computation is based on calculating the ratio of matching components of *cst*_513 to all components of *cst*_513. In this case, as 3 out of 4 components match, it equals 0.75. This number, which in fact shows the familiarity of the model with the situation-goal, will become the confidence value (referred to as *success rate* here) of *mdl*_582 (instantiating *cst*_581 on its LHS) induced via analogy, indicating

[14] This part is adapted from our paper (Sheikhlar and Thórisson, 2024).

[15] mk.val is marker class, in OpenAERA, representing the class of relationships of the entities, e.g., h in this example.

how successful it can be for prediction-making and goal achievement as an initial guess. Once the grasping action based on the *grab_type_2* command leads to (*h holding [s]*), *mdl_582*'s success rate goes up. If *grab_type_2* had failed in grasping *s*, then the learning-from-failure mechanism would have created a composite state-anti-requirement model that would explain the failure.

The agent uses the same line of reasoning to grasp the second item in the Task Solving phase, which is the sphere *s*2. It infers that it should employ *grab_type_1*, as the preconditions of the relevant command model it learned during the experimentation phase better match the properties of *s*2. In other words, when applying *grab_type_1* in the guided experimentation phase, it learned that the *c*2's size was small and added that to its related *CST*. Therefore, as the sphere *s*2 is also small, a more confident solution to choose is *grab_type_1*. After successfully picking up the second sphere, it learns that the only important factor in choosing a grabbing type is the object's size, not its shape. This learning is in the form of increasing the success rates of newly generated *CST*s and M_{req}s, such as the ones shown in Figure 5.4 at the top. Therefore, there is no need to create new composite state-requirement models when attempting to pick up an object with a new shape, i.e., the cylinder *cyl*. Since *cyl* is large, the agent chooses *grab_type_2* as *cyl*'s has *large size*, which matches composite state *cst_581* that had already been induced when grabbing *s*1. This way, the OpenAERA agent cumulatively learns the significance of properties through both analogy-making and performing experiments.

7.5.2 AlignGrab experiment

Our second motor skills learning experiment, called AlignGrab, has to do with the robot arm learning the importance of the orientation of long, thin objects like rectangles when grasping. The details of the AlignGrab experiment are shown in Figure 7.8. During the *guided experimentation* phase, we teach the robot hand *h* to grab two items, *rec*1 and *s*, each with multiple properties shown in the figure. For grasping *rec*1, first *h* applies the *rotate* command to adjust its orientation of *h* and then issues the *grab* command. However, grabbing *s* does not require any orientation adjustment due to the sphere's symmetrical shape, so the hand is taught to apply only the *grab* command directly. In the *task-solving* phase, the goal is to *grab* and *release* three novel but partly familiar items *rec*2, *rec*3, and *cyl*, one after the other. As *rec*2 shares more similarities with *s* than *rec*1 (due to their size and color), the analogy process makes the hand *h* try the shorter and more confident solution, which is to grab *rec*2 directly without any rotation. This results in failure, as the *h*'s orientation does not match that of *rec*2. After the first plan/solution fails, the second plan is chosen and applied, which

is to first rotate h to align its orientation with $rec2$ and then apply a grab command, leading to successful grasping. From this, the hand learns the importance of orientation adjustment when grabbing an item with *rectangle shape*. Therefore, to grab $rec3$, the hand adjusts its orientation first and then grabs the rectangle. However, *cyl* does not have a *rectangle shape*, so it is grabbed directly without hand rotation. The AlignGrab experiment shows how the preconditions of grasping are generalized incorrectly first and then constrained through the learning-from-failure mechanism.

Guided Experimentation
During the guided experimentation phase, we teach the robot hand h to learn sequences of motor actions to grasp and release a rectangle $rec1$ and then a sphere s. The used commands for applying the actions are the *move*, *rotate*, *release*, and $grab\_type\_1$ and $grab\_type\_2$ commands. For grasping $rec1$ by the hand (h), their position and orientation must be identical. Under the conditions shown in Figure 7.8, applying the grab commands leads to learning the CoMs, CSTs, M_{req}s for $grab\_type\_1$ and $grab\_type\_2$ commands. For illustration simplicity, Figure 7.9 only shows the two triads for grasping $rec1$ and s, which the CTPX mechanism induces during this phase.

The CoMs mdl_500 and mdl_709 hold under the conditions represented in the composite state cst_499 and cst_708. The composite states aggregate all correlated properties of entities at the times the grab commands ($grab\_type\_1$ and $grab\_type\_2$) are issued. However, some of these properties referred to in cst_499, e.g., color, are not indeed relevant in the grasping behavior. The next section demonstrates how the ACTL mechanism learns to prune cst_499 by removing such irrelevant properties and generating a new composite state-requirement model. We will see that some important correlations must be learned to keep, such as the hand's orientation and the rectangle's orientation in cst_499, which are identical. In cst_499, the variable that represents this correlation is $v4$ which is significant for determining how a rectangular cuboid should be grasped (the hand must be aligned with it before grasping). Therefore, a *rotate* command must first set the orientation of h equal to that of a rectangle. On the other hand, the composite state cst_708 represents the conditions for grasping a sphere and does not require the same correlation between the orientations. In cst_708, the orientations of the hand and the sphere are represented by different variables ($v4$ and $v6$), meaning they do not have to be the same. Again, note that the ACTL mechanism is not involved during the guided experimentation phase, as learning is done through teaching the OpenAERA agent the necessary actions to take. In the task-solving phase, the ACTL mechanism enables the OpenAERA agent to create new generalized composite states and requirement models for the command models mdl_500 and mdl_709 and verify them in practice by taking relevant actions.

Task Solving
During the task-solving phase, the OpenAERA agent must generalize its existing models by pruning them and then using them to grasp new but partly familiar items. In this phase, the new items are *rec*2, *rec*3, and *cyl*, which have multiple properties that are similar to and different from the properties of the items in the guided experimentation phase, *rec*1, and *s*. The OpenAERA agent analogizes the items in relation to their properties before grasping them. Then, it identifies the most familiar item (*s* due to its color and size) and its related object-grasping strategy. Then, it creates new pruned $CST - M_{req}$s for applying the same CoM to solving the new task of grabbing *rec*2, leading to the failure of grasping. Then, it chose another plan it had created, which initially had a lower confidence. The second plan, which was to rotate and then grab, succeeds, enabling the OpenAERA agent to learn to infer the correct sequence of commands for grabbing the other items *rec*3 and *cyl* later. This planning process starts once the goal facts are injected into the OpenAERA agent. The first goal state is (*h* holding *rec*2), which triggers the planning process through backward chaining.

Generalization in task solving phase
Figure 7.10 shows the CST-M_{req} pair generated by the ACTL mechanism during the task-solving phase, where *cst*_798 and *mdl*_799 on the left column are the generalized version of *cst*_499 and *mdl*_501, shown in Figure 7.9, as two components (out of eight) are pruned in *cst*_798. The size and color of objects are pruned as those of *rec*2, which is supposed to be grasped, do not match the size and shape in the original composite state *cst*_499. In *cst*_499, the size and color must be small and grey whereas *rec*2 is large and blue. Therefore, in the newly created *cst*_798, facts related to the size and color are pruned via the analogy process. Therefore, the ACTL mechanism assigns a success rate of 0.75 (derived from 6/8) to *mdl*_799. On the other hand, *cst*_791 and *mdl*_792 on the right column are the generalized version of *cst*_708 and *mdl*_710 shown in Figure 7.9, where one (out of eight) property, the shape, is pruned, as the shape of *rec*2 does not match *cst*_708. The analogy process, therefore, assigns the success rate of 0.875 (derived by 7/8) to *mdl*_710.
Planning in task solving
The success rates of M_{req}s resulting from the familiarity computation, shown in Figure 7.10, specify the priority of plans built based on the related CST-M_{req}s, as the agent is inclined to choose plans with higher confidence, as detailed in 5.5.2. As the goal state (*h holding rec*2) matches the RHS of both command models *mdl*_500 and *mdl*_709, both newly generated

requirement models, *mdl_*799 and *mdl_*792, can be involved in the creation of two different plans for achieving the goal fact. The requirement model *mdl_*792 has a higher success rate; thus, the plan that uses it has higher confidence and, since it also provides a shorter solution (no need to rotate the hand before grabbing), has a higher priority to be selected.

When the goal fact (*h holding rec*2) is injected, the state of the hand is empty (i.e., *h holding* []), and its orientation has a different value from *rec*2's orientation. Thus, it will create two different plans for acheiving the goal. The plans are based on the two M_{req}s of *mdl_*799 and *mdl_*792:

- **1. AlignGrab plan** makes the agent first rotate *h* and then grab *rec*2 using *mdl_*799 and *cst_*798 (as the *h*'s and *rec*2's orientation must be the same, as can be seen, the variable *v*4 in *cst_*798 is identical in the two). This plan, however, has a lower priority, as it takes longer than the DirectGrab solution to reach the goal state. Note that *mdl_*799 also has a lower success rate than *mdl_*792, making the AlignGrab plan have a much lower priority than that of *DirectGrab* plan.
- **2. DirectGrab plan** is to grab *rec*2 directly without rotating *h*. The plan allows *mdl_*792 and *cst_*791 to be utilized, where no prior rotation is required before grabbing. This is because *v*4 and *v*6, which represent the orientations, are different in *cst_*791. So, this plan has a higher initial priority than the AlignGrab plan as it is shorter and has a higher initial confidence.

Due to *mdl_*792 having a higher success rate, the planner commits to the DirectGrab solution offered by *mdl_*792, grabbing *rec*2 directly.

Learning from failure in task solving
Commitment to the DirectGrab solution is faster and more confident. However, it leads to grasping failure due to improper alignment of the hand with the rectangle *rec*2. The grasping failure triggers the learning-from-failure mechanism, creating *cst_*1430 and its *anti-requirement* which is *mdl_*1431, depicted in Figure 7.11. The *cst_*1430 and *mdl_*1431 together means that if the *shape* is *rectangle*, and the hand's and the rectangle's orientations *v*5 and *v*6 are different, then the action of grabbing does not lead to the hand holding that object. The anti-fact indicated by |fact in red on the RHS of *mdl_*1431 means negation. In other words, the learning from failure mechanism specializes (adds more constraints to) the incorrectly generalized hypotheses *cst_*791 and *mdl_*792, which prevents them from being used in future encounters with rectangle shapes.

Exploration in task solving
Exploration in the ACTL tests hypothesized solutions based on their priority

order. The exploration continues as long as the goal is not achieved. In the task-solving phase, after the DirectGrab solution fails, the planner commits to the AlignGrab solution, which is to rotate the hand and then grab. Once the AlignGrab solution succeeds (leads to the goal state (h *holding* $rec2$)), the success rate of mdl_799 increases, making AlignGrab a confident solution for the next grasping experience. In other words, an increase in a success rate solidifies the hypothesized preconditions that state for grabbing a **rectangle**, the hand must rotate first and then grab. Therefore, before grasping the third rectangle $rec3$, it has already learned from its mistake and its success in grasping. So it does not make the same mistake of grasping $rec3$ directly again. This time, it performs the process of rotating h before grabbing the $rec3$.

The same reasoning process happens for grasping the cylinder *cyl*. The analogy mechanism creates two pairs of $CST\text{-}M_{req}$s, shown in Figure 7.12, each providing a plan for grabbing *cyl*. The first plan in the first column has faster goal achievement (no need for initial rotation). Therefore, the planner commits to that one first, leading to success, and thus, the agent does not have to explore the other solution.

7.5.3 Discussion of results

The two experiments practically prove our second research hypothesis, which states that utilizing analogical reasoning enables causality-based agents to generate new hypotheses, integrate those hypotheses with their knowledge base through testing, and explore the plans generated based on those hypotheses.

In the SizeMatterGrab experiment, the ACTL-based OpenAERA agent generalizes the models learned from grabbing cubes (in the guided experimentation phase) to the new tasks of grabbing the spheres and a cylinder (task-solving phase). Generalization occurs by generating new preconditions (composite states and requirement models) for the learned command models (one for *grab_ type_ 1* and the other for *grab_ type_ 2*) in the guided experimentation phase. The agent becomes familiar with the properties *size* and *shape* and their values (being small, large, and/or cube) by incorporating them into its models' preconditions.

Once new tasks are assigned, the ACTL allows the OpenAERA agent to use analogy to prune the irrelevant properties (e.g., the shape in the SizeMatterGrab example) from the old model precondition, keep the assumed significant properties (e.g., the size), and induce new preconditions accordingly. When the preconditions do not fully match the goal entities' properties (the sphere and cylinder are non-matching properties, as they do not match the old preconditions that have cube in them), the analogy-based models will only keep the matching property (e.g., size, which is either small or large,

both of which match the new objects in the task-solving phase), which leads to issuing the right commands for grabbing those new objects.

In other words, it learns to apply the right actions based on the familiar properties by using its familiarity regarding how objects must be manipulated based on their properties. So, the agent hypothesizes that the right grasping strategy could be grabbing large objects with *grab_type_2* and small objects with *grab_type_1*, regardless of their shape. Once the grab commands are issued and the grasping experience succeeds, the hypotheses with relatively low success rates/confidences are solidified, increasing their confidence.

Additionally, as we saw in the AlignGrab experiment, the OpenAERA agent hypothesizes multiple preconditions based on the similarity of the rectangles in the guided experimentation phase to the ones in the task-solving phase. The first analogy-making, which was based on color and size, had a higher familiarity level and, thus, higher success rate/confidence. Yet, it was a misleading analogy-making, as it led to the failure of grasping experience due to the wrong inferred plan of grasping directly. Once failure occurs, the agent explores the second plan it had created, which was only based on the shape (being a rectangle), which, although it had a low initial confidence/success rate, led to the right plan and grasping experience success. The success of this plan (move-rotate-grab) led the agent to learn to apply the right plan for the second encounter with a rectangle, meaning that the agent cumulatively learned the significance of the size properties after failures during the exploration process.

Learning from misleading analogies in the AlignGrab experiments also proves our third research hypothesis that a cumulative learning mechanism that learns from wrong analogies and improves itself calls for a unified reasoning framework involving analogy, induction, abduction, and deduction. These four reasoning components are involved in the ACTL process, i.e.:

- Backward abductive reasoning allows for both making top-down analogies while inferring the right commands and plans to commit.
- Analogies lead to inducing new generalized preconditions that match new situations.
- And deductive inference for validating the analogy-based model preconditions once backward chaining ends.

7.6 Summary

In this chapter, we evaluated the causal generalization capabilities of general-purpose AI agents, concentrating on tasks that require motor skills learning

and extending the understanding of cause-and-effect relationships through object manipulation. The tasks involved the OpenAERA and ONA agents and evaluated their ability to generalize basic motor movements and object manipulation skills in novel situations. NARS-based ONA generated general falsifiable hypotheses and refined its knowledge and reasoning based on accumulated evidence. Additionally, we compared the generalization features of AERA and NARS and analyzed how each system generates new hypotheses for how to deal with novel tasks. We compared their similarities and differences in their abstraction, planning, and analogy approaches. We also evaluated the OpenAERA agent in robotic experiments conducted in a simulation environment, illustrating and analyzing the system's generalization processes. We designed two different experiences for evaluating the extended OpenAER system: SizeMattersGrab and AlignGrab. The experiments showcase the OpenAERA agent's learning to adjust its plans and identify the significance of environment properties, showing the effectiveness of the introduced autonomous cumulative learning (ACTL) mechanism.

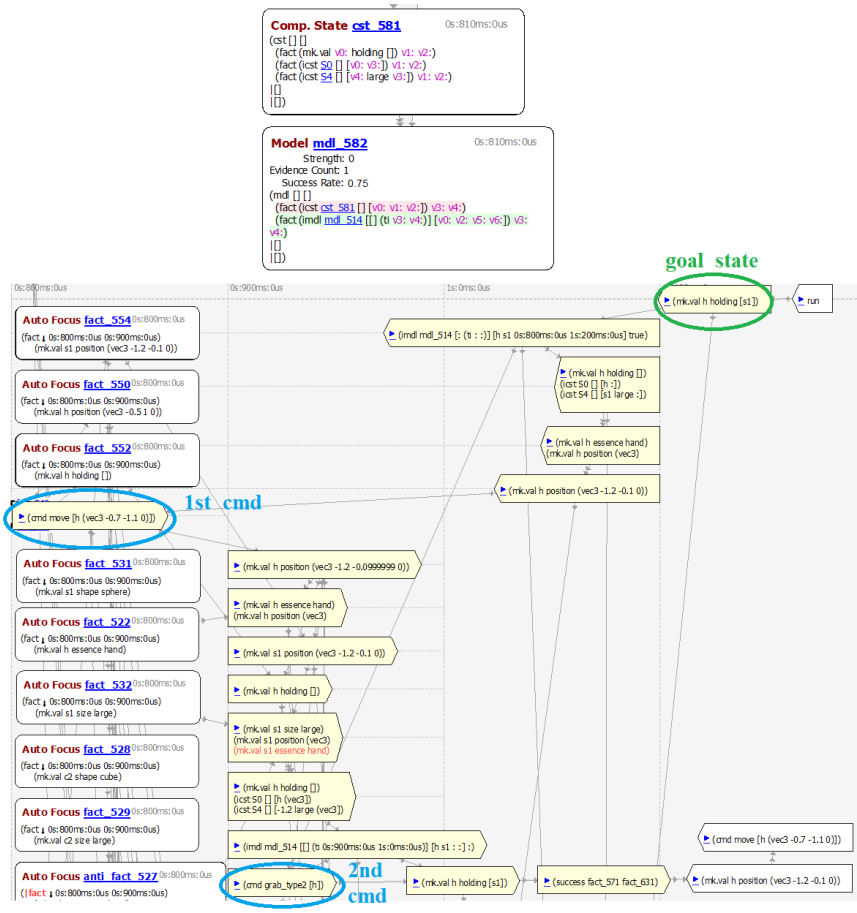

Figure 7.7: **Top:** During the backward chaining, the analogy process induces *cst_581* and *mdl_582* after pruning the shape property from *cst_513*, which is the original CST *cst_581* is derived from. They are induced during the planning process, shown at the bottom via yellow objects when there is no *CST* matching the sphere property. **Bottom:** The planning is based on backward chaining from *(h holding [s1])* towards the commands, and forward chaining, which happens afterward and moves in the opposite direction to validate the inferred process by backward chaining. The column of Auto Focus facts represents the current situation. In the situation at $800ms$, the hand h is empty and not at the same position as sphere s. Therefore, the system first infers that it must issue the *move* command and then *grab_type_2*. The model *mdl_514* is instantiated via the newly created preconditions shown at the top.

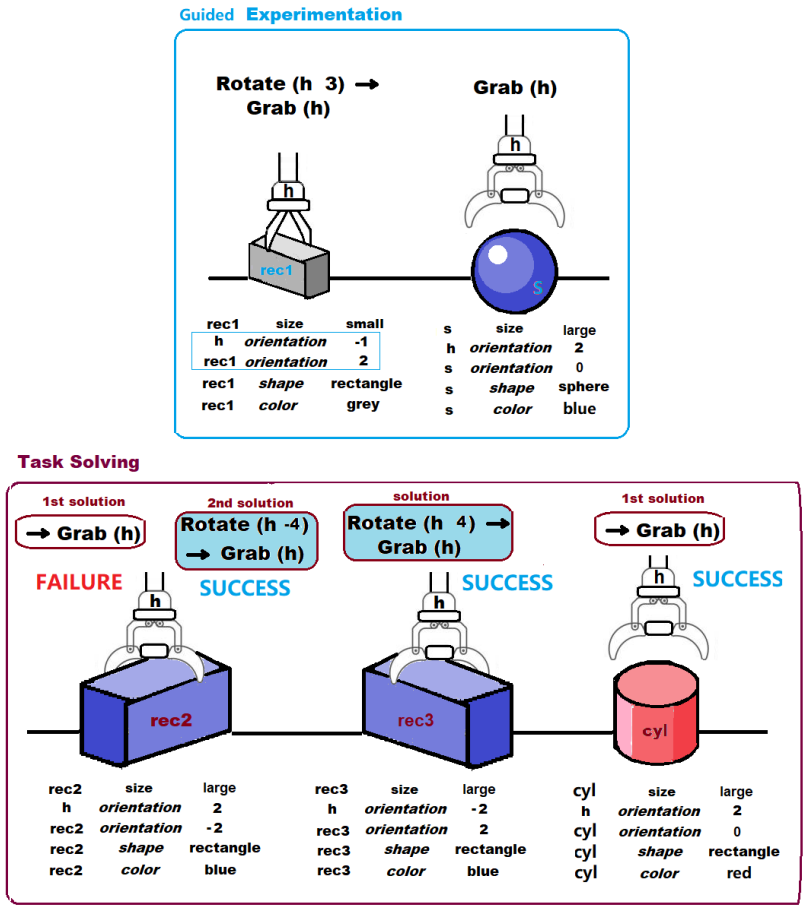

Figure 7.8: In the AlignGrab experiment, the agent learns the significance of the rectangle's orientation after pruning the irrelevant patterns, allowing it to consider it when choosing the correct sequence of actions needed to grasp them, i.e., the hand must align the rectangle before grabbing. The items of this experiment have different properties, e.g., shape, size, orientation, color, and position (represented in vec3 format). For simplicity, the items' positions are not shown here. Also, note that the orientation is measured as the Euler angle around the z-axis.

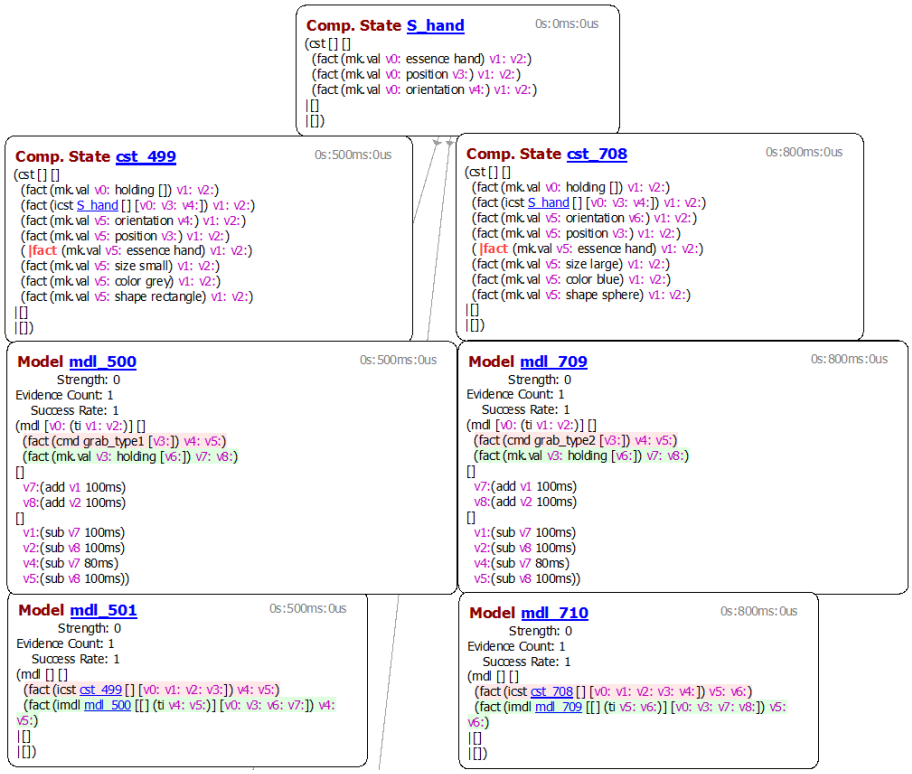

Figure 7.9: *cst_499* and *cst_708* show the circumstances of using *mdl_500* and *mdl_709* based on *mdl_501*, and *mdl_710*, respectively.

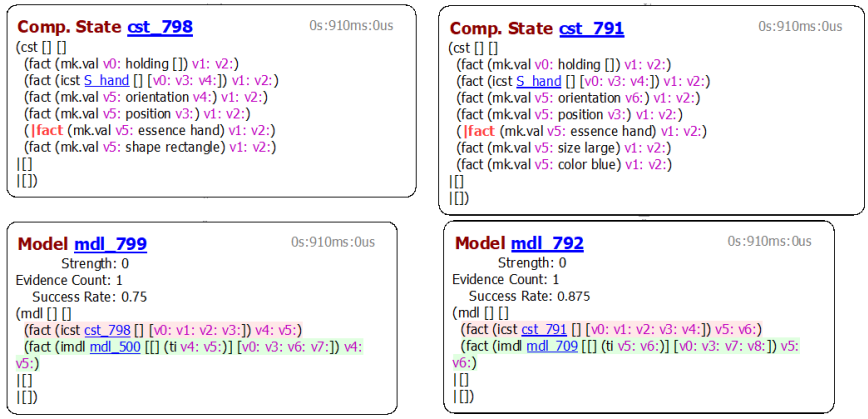

Figure 7.10: *cst*_798 and *cst*_791, as well as *mdl*_799 and *mdl*_792, are the new preconditions for the known model shown in Figure 7.9 created as a result of the analogy process. The *CST*s are pruned versions of *cst*_499 and *cst*_708. The above *CST*-M_{req}s can be immediately used in the planning and exploration process.

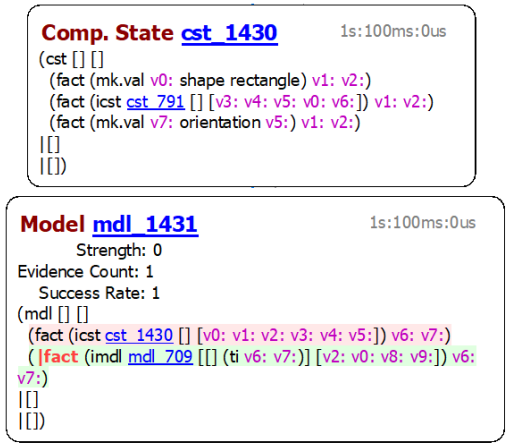

Figure 7.11: *cst*_1430 and *mdl*_1431 are created after the grasping failure by *mdl*_709, an incorrectly generalized hypothesis. The *cst*_1430 and *mdl*_1431 specialize *mdl*_709 by adding constraints to its use, preventing it from being utilized for objects with rectangle shapes and non-aligned orientations.

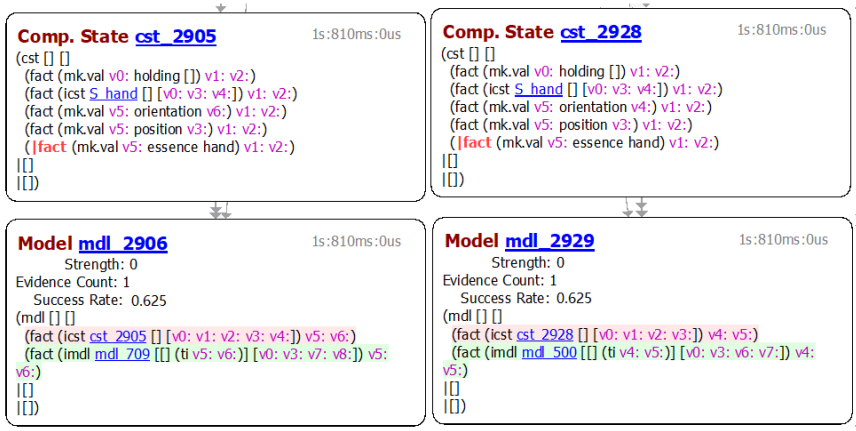

Figure 7.12: Composite states *cst*_2905 and *cst*_2928 are pruned versions of *cst*_499 and *cst*_708 shown in Figure 7.9. The properties of size, color, and shape are pruned, as none of these properties in the cylinder *cyl* match *cst*_499 and *cst*_708. Note that the subtle difference between *cst*_2905 and *cst*_2928 is that in *cst*_2905 the hand's and the object's orientations are different ($v4$ and $v6$), while in *cst*_2928 they are the same ($v4$).

Chapter 8

Conclusions and Future Work

This thesis presented a new approach to autonomous causal generalization within the framework of constructivist AI, causality, and General Machine Intelligence (GMI). The focus was on introducing and integrating a novel mechanism for extending causal knowledge to new situations in line with the agents' goal achievement and direct experimentation.

This work's contributions are related to the presented mechanism's ability to utilize the known causal knowledge to hypothesize new pieces of knowledge while performing the task. This allows an agent to immediately verify the generated hypotheses in action, enabling effective and efficient generalization. Our mechanism is designed and integrated within the Autocatalytic Endogenous Reflective Architecture (AERA) (Nivel, Thórisson, Dindo, et al., 2013) and validated by a few robotic experiments as proof of concept. The comparative and analytical evaluation highlights the effectiveness and efficiency of the proposed approach in autonomously extending causal knowledge and reasoning.

We proved our research hypotheses that 1) causal models are more proper foundations for task-independent generalization of knowledge compared to correlational representations (Sheikhlar et al., 2021), 2) analogical reasoning enables causality-based agents to generate new hypotheses in line with their goal-achievement, integrate them into their knowledge base through testing, and explore plans based on these hypotheses (Sheikhlar and Thórisson, 2024; Sheikhlar et al., 2022), and 3) agents with unified reasoning can learn from incorrect analogies and improve their learning/planning strategies, which calls for a cumulative learning mechanism that learns to improves itself as experience accumulates.

This work is a practical contribution to the fields of causality-based and agent-based AI systems by enabling the relevant systems to use ampliative reasoning for improved knowledge pruning and induction of knowledge when

devising solutions for goal achievement, similar to how human children develop cognitive abilities. Our research also investigates the importance of having causal representations for effective generation, involving both intervention and observation-based learning, and validating their knowledge in practice. The work and the extensions to this work will lead to the design and creation of more adaptive and autonomous AI agents that have the potential to handle unforeseen scenarios and tasks where no prior training can be provided.

This thesis is a significant step towards autonomous causal generalization, but several areas still call for further exploration.

- **Expanding the analogy process's implementation:** We will expand the OpenAERA implementation of the ACTL mechanism by incorporating *Similarity in value proximity* (SP_V) computation in our introduced analogy process in Section 5.4. This idea is already presented in our paper (Sheikhlar et al., 2020), which can be defined for two patterns representing the same task properties having non-identical values, where similarity is about the closeness of their values. Incorporating SP_V leads to having more comprehensive relational similarity calculations in the context of causal relational models (CRMs).

  These calculations, once implemented, will allow the ACTL mechanism to compare the values of properties/variables and generalize causal models in relation to their values.

- **Expanding the ACTL's theoretical foundation**: A future focus will be expanding the theoretical foundation of the proposed ACTL and analogy-making mechanism by incorporating argumentation theory. This work will be in line with the extension of the AERA (Nivel, Thórisson, Dindo, et al., 2013) framework's planning architecture using the assumption-based argumentation theory, which has already been started via the AERA team (Eberding et al., 2024), enabling AERA to explain its plans more systematically. This enhancement will also enable planning with multiple simultaneous commands, addressing known unknowns, and integrating meta-knowledge for interactive agents. This addition is intended as an expansion of the theoretical foundation of both AERA and the integrated ACTL mechanism.

- **Precise formalization**: The future work will also involve a more precise formalization of the components of the introduced analogy mechanism and its integration with the ACTL. The formalization will clearly connect the described components of the ACTL in Chapters 4 and 5 of this thesis with the aim of developing a more precise, rigorous mathematical framework, ensuring that the reasoning processes

are well-defined. Such formalization will need exact definitions of the conditions under which the analogy processes in chains of causal networks could prune the existing knowledge and induce new pieces of casual models, as well as the logical structures governing the cumulative learning process where the learning from failure is included. Building a clear, formal foundation will improve our approach's transparency and robustness, which will subsequently facilitate its use and further development by the AI research community.

- **Scalability and practial applications**. Future work will also focus on improving the scalability of the introduced mechanism, which, of course, depends on further formalization and more extensive empirical evaluation. We will design additional experiments to test whether the mechanism can be applied in more complex and dynamic environments with higher-dimensional sets of properties with complex and longer chains of causal relationships. Additionally, extending the mechanism's scope to real-world scenarios, e.g., using them for physical robots' task learning/planning, will expand our awareness of the practical challenges and benefits of implementing our approach.

Bibliography

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, *7*(1), 39–59.

Agrawal, R., Squires, C., Prasad, N., & Uhler, C. (2023). The decamfounder: Nonlinear causal discovery in the presence of hidden variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *85*(5), 1639–1658.

Bareinboim, E., Correa, J. D., Ibeling, D., & Icard, T. (2022). On pearl's hierarchy and the foundations of causal inference. In *Probabilistic and causal inference: The works of judea pearl* (pp. 507–556).

Baumann, D., Solowjow, F., Johansson, K. H., & Trimpe, S. (2020). Identifying causal structure in dynamical systems. *arXiv preprint arXiv:2006.03906*.

Belenchia, M., et al. (n.d.). *Towards a theory of causally grounded tasks* (Doctoral dissertation).

Belenchia, M., Thórisson, K. R., Eberding, L. M., & Sheikhlar, A. (2022). Elements of task theory. *Artificial General Intelligence: 14th International Conference, AGI 2021, Palo Alto, CA, USA, October 15–18, 2021, Proceedings 14*, 19–29.

Bengio, Y., Deleu, T., Rahaman, N., Ke, R., Lachapelle, S., Bilaniuk, O., Goyal, A., & Pal, C. (2019). A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*.

Cherry, K. (2020). How we use selective attention to filter information and focus.

Conant, R. C., & Ross Ashby, W. (1970). Every good regulator of a system must be a model of that system. *International journal of systems science*, *1*(2), 89–97.

Dennett, D. C. (1990). Cognitive wheels: The frame problem of ai. *The philosophy of artificial intelligence*, *147*, 170.

Drescher, G. L. (1991). *Made-up minds: A constructivist approach to artificial intelligence*. MIT press.

Eberding, L. M. (2022). Comparison of machine learners on an aba experiment format of the cart-pole task. *International Workshop on Self-Supervised Learning*, 49–63.

Eberding, L. M., Sheikhlar, A., & Thórisson, K. R. (2020). Sage: Task-environment platform for autonomy and generality evaluation. *International Conference on Artificial General Intelligence. Springer, submitted in*.

Eberding, L. M., Thompson, J., & Thórisson, K. R. (2024). Argument-driven planning and autonomous explanation generation. *International Conference on Artificial General Intelligence*, 73–83.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial intelligence*, *41*(1), 1–63.

Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, *2*(3-4), 189–208.

Gentner, D., & Markman, A. B. (1997). Structure mapping in analogy and similarity. *American psychologist*, *52*(1), 45.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, 424–438.

Hammer, P. (2021). *Autonomy through real-time learning and opennars for applications*. Temple University.

Hammer, P., & Lofthouse, T. (2020). 'opennars for applications': Architecture and control. *Artificial General Intelligence: 13th International Conference, AGI 2020, St. Petersburg, Russia, September 16–19, 2020, Proceedings 13*, 193–204.

Hu, S., Ma, Y., Liu, X., Wei, Y., & Bai, S. (2021). Stratified rule-aware network for abstract visual reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(2), 1567–1574.

Imbens, G. W., & Rubin, D. B. (2015). *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.

Jiménez, S., Fernández, F., & Borrajo, D. (2013). Integrating planning, execution, and learning to improve plan execution. *Computational Intelligence*, *29*(1), 1–36.

Johnson, J. J., Li, L., Qureshi, A., & Yip, M. C. (2021). Motion planning transformers: One model to plan them all.

McCarthy, J., & Hayes, P. J. (1981). Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence* (pp. 431–450). Elsevier.

McCarthy, J. C. (1986). Mental situation calculus. *Theoretical Aspects of Reasoning About Knowledge*, 307.

Michel, O. (2004). Cyberbotics ltd. webots™: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, *1*(1), 5.

Mikolov, T., Yih, W.-t., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 746–751.

Mitchell, M. (2021). Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, *1505*(1), 79–101.

Mitchell, T. M. (1997). Machine learning.

Nivel, E., & Thórisson, K. R. (2013a). Replicode: A constructivist programming paradigm and language. *Technical RUTR-SCS13001, Reykjavik University School of Computer Science*.

Nivel, E., & Thórisson, K. R. (2013b). Towards a programming paradigm for control systems with high levels of existential autonomy. *International Conference on Artificial General Intelligence*, 78–87.

Nivel, E., Thórisson, K. R., Dindo, H., Pezzulo, G., Rodriguez, M., Corbato, C., Steunebrink, B., Ognibene, D., Chella, A., et al. (2013). Autocatalytic endogenous reflective architecture.

Nivel, E., Thórisson, K. R., Steunebrink, B. R., Dindo, H., Pezzulo, G., Rodriguez, M., Hernández, C., Ognibene, D., Schmidhuber, J., Sanz, R., et al. (2013). Bounded recursive self-improvement. *arXiv preprint arXiv:1312.6764*.

Pearl, J. (2009a). Causal inference in statistics: An overview. *Statistics surveys*, *3*, 96–146.

Pearl, J. (2009b). Causality. Cambridge University Press.

Pearl, J. (2018). Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*.

Pearl, J., & Mackenzie, D. (2018). *The book of why: The new science of cause and effect*. Basic books.

Peters, J., Janzing, D., & Schölkopf, B. (2017a). Elements of causal inference: Foundations and learning algorithms. The MIT Press.

Peters, J., Janzing, D., & Schölkopf, B. (2017b). *Elements of causal inference: Foundations and learning algorithms*. The MIT Press.

Piaget, J., Piercy, M., & Berlyne, D. (1951). The psychology of intelligence.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A modern approach*. Pearson.

Sabbadin, R., Teichteil-Königsbuch, F., & Vidal, V. (2020). Planning in artificial intelligence. *A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms*, 285–312.

Sheikhlar, A., Eberding, L. M., & Thórisson, K. R. (2021). Causal generalization in autonomous learning controllers. *International Conference on Artificial General Intelligence*, 228–238.

Sheikhlar, A., & Fakharian, A. (2018). Online policy iteration-based tracking control of four wheeled omni-directional robots. *Journal of Dynamic Systems, Measurement, and Control*, *140*(8), 081017.

Sheikhlar, A., & Thórisson, K. R. (2024). Causal generalization via goal-driven analogy. *International Conference on Artificial General Intelligence*, 163–172.

Sheikhlar, A., Thórisson, K. R., & Eberding, L. M. (2020). Autonomous cumulative transfer learning. *International Conference on Artificial General Intelligence*, 306–316.

Sheikhlar, A., Thórisson, K. R., & Thompson, J. (2022). Explicit general analogy for autonomous transversal learning. *International Workshop on Self-Supervised Learning*, 48–62.

Shuang, S., & Mohd Pozi, M. S. B. (2024). Using causal inference to solve uncertainty issues in dataset shift. *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 1155–1157.

Spirtes, P., Glymour, C. N., Scheines, R., & Heckerman, D. (2000). *Causation, prediction, and search*. MIT press.

Sung, I., Choi, B., & Nielsen, P. (2021). On the training of a neural network for online path planning with offline path planning algorithms. *International Journal of Information Management*, *57*, 102142.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. *International conference on artificial neural networks*, 270–279.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, *10*(Jul), 1633–1685.

Thórisson, K. R. (2009). From constructionist to constructivist AI. *2009 AAAI Fall Symposium Series Tech Report FS-09-01*, 175–183.

Thórisson, K. R. (2012). A new constructivist AI: from manual methods to self-constructive systems. In *Theoretical foundations of artificial general intelligence* (pp. 145–171). Springer.

Thórisson, K. R. (2021a). The explanation hypothesis in autonomous general learning. *Proceedings of Machine Learning Research*, *159*, 5–27.

Thórisson, K. R. (2021b). Seed-programmed autonomous general learning. *Proceedings of Machine Learning Research*, *131*, 32–70.

Thórisson, K. R., Bieger, J., Li, X., & Wang, P. (2019a). Cumulative learning. *Proceedings of the 12th International Conference on Artificial General Intelligence*, 198–208.

Thórisson, K. R., Bieger, J., Li, X., & Wang, P. (2019b). Cumulative learning. *International Conference on Artificial General Intelligence*, 198–208.

Thórisson, K. R., Kremelberg, D., Steunebrink, B. R., & Nivel, E. (2016). About understanding. *International Conference on Artificial General Intelligence*, 106–117.

Thórisson, K. R., & Talbot, A. (2018). Cumulative learning with causal-relational models. *International Conference on Artificial General Intelligence*, 227–237.

Thórisson, K. R., & Talevi, G. (2024). A theory of foundational meaning generation in autonomous systems, natural and artificial. *International Conference on Artificial General Intelligence*, 188–198.

Turing, A. M. (1950). *Computing machinery and intelligence*. Springer.

Van Harmelen, F., Lifschitz, V., & Porter, B. (2008). *Handbook of knowledge representation*. Elsevier.

Velagic, J., Osmic, N., & Lacevic, B. (2010). Design of neural network mobile robot motion controller. In *New trends in technologies*. IntechOpen.

Wang, J., Liu, J., Chen, W., Chi, W., & Meng, M. Q.-H. (2021). Robot path planning via neural-network-driven prediction. *IEEE transactions on artificial intelligence*, *3*(3), 451–460.

Wang, P. (1995). *Non-axiomatic reasoning system: Exploring the essence of intelligence*. Citeseer.

Wang, P. (2004). The limitation of bayesianism. *Artificial Intelligence*, *158*(1), 97–106.

Wang, P. (2006). *Rigid flexibility: The logic of intelligence* (Vol. 34). Springer Science & Business Media.

Wang, P. (2009a). Analogy in a general-purpose reasoning system. *Cognitive Systems Research*, *10*(3), 286–296.

Wang, P. (2009b). Insufficient knowledge and resources—a biological constraint and its functional implications. *2009 AAAI Fall Symposium Series*.

Wang, P. (2013). *Non-axiomatic logic: A model of intelligent reasoning*. World Scientific.

Wang, P. (2019). On defining artificial intelligence. *Journal of Artificial General Intelligence*, *10*(2), 1–37.

Wang, P. (2020). On defining artificial intelligence. *Journal of Artificial General Intelligence*, *11*(2), 73–86.

Wang, R., Yi, M., Chen, Z., & Zhu, S. (2022). Out-of-distribution generalization with causal invariant transformations. *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 375–385.

Xu, Z., Zhou, X., & Li, S. (2019). Deep recurrent neural networks based obstacle avoidance control for redundant manipulators. *Frontiers in neurorobotics*, *13*, 47.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 3320–3328.

Appendix A

Definitions

Definitions in Section 4.2

Dynamic process: A dynamic process is a set of sequential (state) transitions that change the state of environments.

Transition: A (state) transition refers to the changes in the values of observable properties due to commands/actions/events.

Transition function: a function that represents a transition taking in the values of commands/events/causes and observable properties at a time and returns the changed values of the observables at a later time.

Input interventions: Input interventions have two different types: 1. setting the initial values (a.k.a. initial conditions) of observable properties in a dynamic process and 2. setting the values of commands.

Casual discovery: Casual discovery is identifying causal relations between the values of properties and commands through input interventions.

Definitions in Section 4.3

Causal relational model (CRM): A CRM is a knowledge construct representing a transition that relates a set of cause patterns or preconditions on its left-hand side (LHS) to the effect patterns or postconditions on its right-hand side (RHS). The cause patterns involve commands or events. In this work, we focus on the case where LHS includes a command leading to a transition. CRMs are falsifiable pieces of knowledge and have degrees of truth specified by their confidence values.

Patterns: Patterns are distinct information structures having two types, facts and constraints, which may contain variables.

Facts: Facts are statements with specific time intervals, having two types: observable facts and commands. In this study, we focus on these two, as facts may also incorporate other statements, such as instantiated composite states and models, supporting higher-order logic.

Command: A command is a fact representing an internal operation within the controller leading to taking a physical action and may change the state of the agent's environment.

Observation fact: An observation fact represents an observed pattern at a time with the structure (entity property value time). Entity and property are symbols, with the difference that entities can be replaced with variables while properties cannot.

Constraint: A constraint represents a logical or numerical constraint such as an equation (a.k.a., transition function) or inequality. They specify the transition functions, ranges of admissible values, logical conditions or timing equations relating the variables in commands and facts.

Situation: A situation is all existing observable facts at a time, representing the complete observable state of a process.

Goal: A goal is a fact representing the desired state the agent tends to achieve. A goal can either be given or derived by the agent. The derived goals are called subgoals. In Replicode programming language, goals are objects referring to related goal facts. Here, for simplicity of explanation, we consider them facts.

Definitions in Section 4.4

Asbtraction: Abstraction occurs when the values of the known properties and/or the known entities having those properties can be replaced with variables in patterns and facts.

Selective attention: Selective attention is the ability to choose a subset of the observation facts from a situation and exclude the rest of them to form *initial* preconditions for CRMs.

Induction: The creation of CRMs is referred to as induction in this work, which may result from various learning mechanisms. Induction may also involve the above-mentioned abstraction and selective attention processes to include relevant, generalized sets of patterns in CRMs. In this work, induction generates CRMs due to 1) observed transitions via guided experimentation, 2) pattern matching between observations and known CRMs (i.e., via analogy-making), and 3) failure in observing expected patterns (i.e., learning from failure).

Knowledge pruning: Eliminating patterns from CRMs' preconditions to match a wider range of situations, leading to having fewer precondition patterns, making the CRMs more general. Knowledge pruning has two types: constraints pruning and fact pruning. *Constraint pruning* is about identifying a more accurate set of transition functions and conditions where the spurious correlations are pruned. The spurious correlations are constraints that do not affect the transitions, do not hold true, or are inaccurate. *Fact pruning* is selectively eliminating facts from patterns of existing CRMs that are deemed irrelevant for transitions.

Definitions in Section 5.2

Phenomenon: The ***phenomenon*** Φ is made up of a finite set of parts $\{\varphi_1 \dots \varphi_n\}$ that can be observed. The parts are referred to as ***aspects***, between which various types of relationships can be defined (Thórisson, 2021b).

Knowledge base (*KB*): Knowledge base is the existing set of pieces of knowledge. Knowledge is also referred to as model-base in the text.

Definitions in Section 5.4

State: State is an arbitrary subset of a situation.

Definitions in Section 6.2

Command models (CoMs): In OpenAERA (Nivel and Thórisson, 2013a), CoMs represents an abstracted command pattern at a time causing a value change of a property at a later time.

Composite states (CSTs): In OpenAERA (Nivel and Thórisson, 2013a), CSTs are precondition patterns that need to be instantiated all at the same time so that CoMs can be used for prediction-making or planning.

Requirement models (M_{req}):In OpenAERA (Nivel and Thórisson, 2013a),

M_{req}s are models that enable the use of CoMs under specific CSTs via instantiating CoMs on their RHS and CSTs on their LHS.