THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Explainable and Interpretable Methods for Handling Robot Task Failures

MAXIMILIAN DIEHL

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2025

Explainable and Interpretable Methods for Handling Robot Task Failures

MAXIMILIAN DIEHL
ISBN 978-91-8103-177-5

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5635
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden

Printed by Chalmers Reproservice
Gothenburg, Sweden, March 2025

To Elma. Ljubim te.

Explainable and Interpretable Methods for Handling Robot Task Failures

MAXIMILIAN DIEHL
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Robots are increasingly deployed in dynamic human environments. To avoid failures during task execution, such as setting a table, they must adapt to unexpected changes. This is challenging because robots must proactively *predict* failures and identify controllable factors to *prevent* them. If failures cannot be autonomously prevented, robots should *explain* failure causes, which is challenging because explanations should cater to non-expert users. The first goal of this thesis is therefore to enhance the reliability and explainability of robots by predicting, explaining, and preventing task execution failures using symbolic causal models. We introduce a novel framework for learning causal models from simulated data. To improve the transferability of the causal models between tasks, we propose three parameter transfer methods that leverage the semantic similarities between models. To enhance failure prediction, we propose a novel approach that combines the learned causal models with a breadth-first search procedure for proactive failure prediction and contrastive failure explanation. We validate this approach on object manipulation tasks, such as stacking cubes, achieving a 95% failure prevention rate. We then extend the method to predict human perceptions of a navigation robot's competence and improve its behavior, resulting in a 72% increase in perceived competence.

Another common failure is missing capabilities that hinder a robot from achieving its task goal. The second thesis goal is therefore to enable non-experts to *assist* by *teaching* robots the missing actions intuitively, without coding experience. We propose a novel demonstration system that lets users teach tasks in Virtual Reality. Our system automatically segments and classifies the demonstrations, generating symbolic, robot-agnostic actions that integrate into the robot's existing capabilities. Our approach achieves a 92% success rate in learning task abstractions from a single demonstration in single- and multi-agent tasks. Additionally, our approach enables robots to detect missing actions automatically, allowing users to demonstrate only the missing parts instead of the entire task, reducing demonstration time by 61%.

The presented contributions enable robots to handle dynamic environments more reliably and explainably while continuously expanding their capabilities to adapt to new challenges.

Keywords: Failure Explanations, Causality, Robot Task Planning

List of Publications

This thesis is based on the following publications (in chronological order):

[A] **Diehl Maximilian**, Paxton Chris, Ramirez-Amaro Karinne, “Automated Generation of Robotic Planning Domains from Observations”. Published in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, Online, 2021.

[B] **Diehl Maximilian**, Ramirez-Amaro Karinne, “Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures”. Published in IEEE Robotics and Automation Letters (RAL), vol. 7, no. 4, 2022.

[C] **Diehl Maximilian**, Ramirez-Amaro Karinne, “A causal-based approach to explain, predict and prevent failures in robotic tasks”. Published in Robotics and Autonomous Systems (RAS), vol. 162, 2023.

[D] **Diehl Maximilian**, Ramirez-Amaro Karinne, “Generating and Transferring Priors for Causal Bayesian Network Parameter Estimation in Robotic Tasks”. Published in IEEE Robotics and Automation Letters (RAL), vol. 9, no. 2, 2024.

[E] **Diehl Maximilian**, Zappa Isacco, Zanchettin Andrea Maria and Ramirez-Amaro Karinne, “Learning Robot Skills From Demonstration for Multi-Agent Planning”. Published in 2024 IEEE International Conference on Automation Science and Engineering (CASE), Bari, Italy, 2024.

[F] **Diehl Maximilian**, Chakraborti Tathagata, Ramirez-Amaro Karinne, “Enabling Robots to Identify Missing Steps in Robot Tasks for Guided Learning from Demonstration”. Published in 2025 IEEE/SICE International Symposium on System Integrations (SII), Munich, Germany, 2025.

[G] **Diehl Maximilian**, Tsoi Nathan, Chavez Gustavo, Ramirez-Amaro Karinne, Vázquez Marynel, “A Causal Approach to Predicting and Improving Human Perceptions of Social Navigation Robots”. To be submitted to ACM Transactions on Human-Robot Interaction (THRI), 2025.

Other publications by the author, not included in this thesis, are:

[H] **Diehl Maximilian**, Plopski Alexander, Kato Hirokazu, Ramirez-Amaro Karinne, "Augmented Reality Interface to Verify Robot Learning". Published in 2020 IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Naples, Italy, Online, 2020.

[I] **Diehl Maximilian**, Paxton Chris, Ramirez-Amaro Karinne, "Optimizing robot planning domains to reduce search time for long-horizon planning". 5th Workshop on Semantic Policy and Action Representations for Autonomous Robots (SPAR), at IROS 2021, Prague, Czech Republic, Online, 2021.

[J] **Diehl Maximilian**, Ramirez-Amaro Karinne, "Explaining and Preventing Robot Failures based on Causal Models". 1st Workshop on The Imperfectly Relatable Robot: An interdisciplinary workshop on the role of failure in HRI, at HRI 2023, Stockholm, Sweden, 2023.

[K] Wenhao Lu, **Diehl Maximilian**, Sjöberg Jonas, Ramirez-Amaro Karinne, "Leveraging Symbolic Models in Reinforcement Learning for Multi-Skill Chaining". Published in 2025 IEEE/SICE International Symposium on System Integrations (SII), Munich, Germany, 2025.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Karinne, who has been an exceptional mentor throughout my PhD journey. From day one, she dedicated countless hours to teaching me how to write, guiding me through the ups and downs of research, and always encouraging me to push further. She has opened numerous doors and opportunities, including international research visits and connections with inspiring researchers at conferences. Thanks to her, I have had the opportunity to grow in ways I never imagined. I would like to thank my co-supervisor, Dean, whose passion for research and our never-missed chances to discuss new ideas were both inspiring and energizing. Beyond their supervision, I am grateful for the research environment they have created, surrounded by talented colleagues, Wenhao, Jing, Zhitao, Nanami, Jesper, and our new robotics lab. Working in such an inspiring space has made this journey even more enjoyable, and I will truly miss collaborating with them. I am also grateful to Jonas for welcoming me into the Mechatronics group and for his guidance throughout the PhD journey as my examiner.

I have made many great friends during my PhD. To those who have already graduated, Ahad, David, Rita, Constantin, Sabino, Endre, Yizhou, Stefan, and many others, thank you for the support, great times, and all the conversations that kept me going. To those who are soon to become doctors, Sten, Carl-Johan, Sondre, Lorenzo, Gabriel, Kilian, Alvin, Kristian, Rikard, Nishant, Ze, Martina, Erik, Muhammad, Ektor, Anand, Godwin, Francesco, Albert, Ahmet, and Huang, thank you for the wonderful moments, laughs, and engaging discussions, both during and after work. I wish you all the best of luck in your future endeavors!

Finally, I want to thank my family at home in Munich. Coming back for Christmas or other vacations always provided me with a much-needed mental break from my work and filled me up again with energy for the year ahead. Above all, I want to thank my wife, Elma. Despite the long distance and time apart, you never stopped supporting me, and you are the most important person in my life.

Acronyms

VR:	Virtual Reality
MLE:	Maximum Likelihood Estimation
CBN:	Causal Bayesian Network
LfD:	Learning from Demonstration
BFS:	Breadth-First Search
HRI:	Human-Robot Interaction

Contents

Abstract ii

List of Papers iii

Acknowledgments v

Acronyms vi

I Overview 1

1 Introduction 3

1.1 Background and Research Gaps 5

1.1.1 Failure Prediction and Prevention 6

1.1.2 Failure Explanation 8

1.1.3 Model Transfer and Data Complexity 10

1.1.4 Human Support when Robots Fail 11

1.2 Thesis Goals and Contributions 15

1.3 Thesis Overview 17

2 Our Causal Approach for Explaining, Predicting and Preventing Task Execution Failures 19
2.1 Learning Causal Task Models 21
2.1.1 Variable Definition 23
2.1.2 Variable Preprocessing 28
2.1.3 Learning Cause-Effect Models: Causal Discovery 31
2.1.4 Our Approach for Generating CBN Parameter Priors . 36
2.2 Our Approach to Generate Explanations for Robot Task Execution Failures 41
2.3 Our Approach to Predict and Prevent Future Task Failures . . 46

3 Using Learning from Demonstration to Overcome Task Planning Failures 53
3.1 Automated Planning 54
3.1.1 Planning Domain 54
3.1.2 Planning Problem 57
3.1.3 Formulation of a Planning Problem - PDDL 58
3.1.4 Beyond Classical Planning 63
3.1.5 Planning Algorithms 64
3.2 Planning Failures 66
3.2.1 Challenges 66
3.2.2 Learning Planning Operators from Human Demonstration 68
3.2.3 Guided Demonstrations to Learn Missing Actions 73
3.2.4 Learning Robot Actions from Demonstration for Multi-Agent Planning 75

4 Summary of included papers 81
4.1 Paper A 81
4.2 Paper B 82
4.3 Paper C 83
4.4 Paper D 83
4.5 Paper E 84
4.6 Paper F 85
4.7 Paper G 86

5 Concluding Remarks and Future Work 87
5.1 Future Work 89

References **93**

II Papers **107**

A Automated Generation of Robotic Planning Domains from Observations **A1**

1 Introduction A3
2 Related Work A6
3 Planning problem and environment A7
3.1 Environment A7
3.2 Symbolic Planning A8
4 System Description A11
4.1 Activity segmentation and classification A11
4.2 Automatic Generation of Operators A11
4.3 Plan Quality A13
4.4 Symbolic Planning and execution A15
5 Experiment and results A15
5.1 Data collection and operator generation A15
5.2 Plan generation A17
5.3 Plan execution A18
5.4 Discussion A19
6 Conclusion A20
References A20

B Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures **B1**

1 Introduction B3
2 Related Work B6
2.1 Causality in Robotics B6
2.2 Learning explainable models of cause-effect relations . . B6
2.3 Contrastive Explanations B7
3 Our approach to explaining failures B8
3.1 Variable definitions and assumptions B8
3.2 Our proposed pipeline to learn the causal model B9
3.3 Our proposed method to explain failures B10

4 Experiments B12
4.1 Experiment 1: Stacking Cubes B12
4.2 Experiment 2: Dropping spheres into containers B14
5 Results and Discussion B15
5.1 Analysis of the obtained causal models B15
5.2 sim2real accuracy of the causal models B17
5.3 Explainability capabilities B19
5.4 Comparison of our failure explanation approach with baseline methods B22
6 Conclusion B24
References B24

C A causal-based approach to explain, predict and prevent failures in robotic tasks C1
1 Introduction C4
2 Related work C6
2.1 Causality in robotics C6
2.2 Learning explainable models of cause-effect relations . . C7
2.3 Failure prevention C7
2.4 Contrastive explanations C8
3 Our approach to explaining, predicting and preventing failures C8
3.1 Variable definitions and assumptions C9
3.2 Our proposed pipeline to learn causal models C10
3.3 Our proposed method to explain failures C12
3.4 Our proposed method to predict and prevent failures . . C14
4 Experiments C16
4.1 Experiment 1 setup: Stack-1-Cube scenario C16
4.2 Experiment 2 setup: Stack-3-Cubes scenario C18
5 Results and discussion C19
5.1 Assessing the obtained causal models C20
5.2 Evaluation of prediction and failure prevention capabilities C23
6 Discussion of the obtained results C26
6.1 Impact of the success threshold ϵ on failure prevention . C29
6.2 Data Efficiency and transferability of the causal models C31
7 Conclusion C33
References C33

D Generating and Transferring Priors for Causal Bayesian Network Parameter Estimation in Robotic Tasks D1

1 Introduction D3
2 Related Work D5
3 Our Approach to CBN Parameter Transfer D7
3.1 Preliminaries D8
3.2 Our approach for generating causal BN parameter priors D10
4 Experiment Description D12
4.1 Experiment Environments D12
4.2 Prior Transfer Scenarios D14
4.3 Experiment Evaluation Metrics D17
5 Results and Discussion D18
5.1 2-Stack-Simulation Results D18
5.2 2-Stack-Reality and 2-Stack-SimToReal Results D18
5.3 SphereDropping Results D20
5.4 Stack-To-Drop Results D22
5.5 Discussion D22
6 Conclusion D24
References D24

E Learning Robot Skills From Demonstration for Multi-Agent Planning E1

1 Introduction E3
2 Related Work E6
2.1 Skill-Based Programming for Multi-Agent Systems . . . E6
2.2 Autonomous Planning for Multi-Agent Systems E7
2.3 Learning Skills for Multi-Agent Planning E8
3 Background E8
3.1 Automated/Symbolic Planning E8
3.2 Current PbD and LfD methods E9
3.3 Problems with current PbD and LfD pipeline E12
4 Methodology E13
5 Experimental Validation E16
5.1 Experiment Tasks E16
5.2 User-Study Setup E17
5.3 Evaluation Criteria E18

6 Results E19
6.1 Tower of Hanoi Ablation Study - Plan Length E19
6.2 Tower of Hanoi Kin. Teaching - Space Constraints . . . E20
6.3 Close&Open Bottle VR Teaching - Action Synergies . . E20
6.4 SUS Questionnaire E22
7 Conclusions E23
References E23

F Enabling Robots to Identify Missing Steps in Robot Tasks for Guided Learning from Demonstration F1
1 Introduction F3
2 Related Work F6
3 The Guided Demonstration Task F7
3.1 Measures for Guided Demonstration Effectiveness . . . F9
3.2 Caveats in Using Excuses as Guidance F11
3.3 Used Demonstration and Excuse Generation Methods . F12
4 User Study F13
5 Results F14
5.1 Reduction in Time F14
5.2 Theoretical Savings in Demonstration F14
5.3 Plan solvability after demonstrations F15
5.4 Discussion F16
6 Conclusion F16
References F17

G A Causal Approach to Predicting and Improving Human Perceptions of Social Navigation Robots G1
1 Introduction G3
2 Related Work G6
2.1 Causality in HRI G6
2.2 Failure prediction and prevention G6
3 Navigation Task G8
4 Method G10
4.1 Causal Bayesian Networks G10
4.2 Proposed CBN for the Robot-Following Task G12
4.3 Proposed method to address low perceived performance G17

5 Prediction Performance Results G19
5.1 Clustering Results G19
5.2 Prediction Performance Results G22
6 User Study of Counterfactual Behavior Generation G22
6.1 Hypotheses G23
6.2 Study Phases and Conditions G24
6.3 Experimental Procedure G24
6.4 Participants G25
6.5 Analysis G25
6.6 Results G27
6.7 Discussion G27
7 Conclusion G30
References G31

Part I

Overview

CHAPTER 1

Introduction

Future robots are expected to assist humans across various tasks and environments [1], [2]. At home, they might wash dishes [3], or care for the elderly [4]. In hospitals, robots can transport lab specimen [5], [6], while in industry, they may assist with assembly or repetitive tasks, reducing strain and improving workers' health [7], [8].

However, human environments are often unstructured, dynamic, and subject to variation, even within the same domain [9]. This makes it inherently difficult for robots to navigate and operate effectively. Consequently, triggered by unexpected changes in the environment, robots frequently encounter *failures* during their task execution [10], [11], [12], e.g., while setting the table. Addressing these failures is a complex challenge as it requires robots to proactively *predict* potential issues and identify controllable factors to *prevent* them.

By leveraging robust predictive models and preventive actions, robots have the potential to reduce the likelihood of failure, thereby improving their reliability and autonomy. However, considering the complexity of human environments, preventing every possible failure is unrealistic [13], [14], [15]. This limitation also applies to humans, who, despite their exceptional ability to react and adapt quickly to unexpected changes, still experience failures [16]. To mitigate the impact of such failures, humans can retrospectively reason about potential failure *causes*, enabling them to learn from mistakes and prevent similar issues in the future. This capacity for causal reasoning is also fundamental to our ability to *explain* our actions to others [17], which is crucial for effective collaboration and interaction between humans [18], [19]. Research has shown that robots operating in human environments can similarly benefit from causal reasoning capabilities, particularly in contexts of failure [17], [20], as *explainability* has been shown to enhance trust and acceptance of robots by providing insights into their decision-making processes and the reasons behind their errors [19], [21], [22], [23], [24]. However, providing these explanations is challenging as they must be tailored to the audience's roles and level of experience [17], [19], [25]. This is particularly important for robots operating in human environments, where they are frequently interacting with non-expert users, such as caregivers or elderly individuals.

Proactive failure prediction and retrospective failure explanations are essential for preventing and mitigating the negative effects of failures. Nevertheless, given the complexity of human environments, robots may still encounter failures that they cannot prevent or resolve without external assistance [25], [26], [27], [28], [29], [30]. For instance, robots deployed in homes are likely to face situations they are not capable of handling, requiring them to learn new tasks such as setting the table or cleaning the kitchen. In such cases, a human-in-the-loop can play a supportive role by teaching robots the entire tasks or specific actions robots miss to successfully accomplish their objectives. However, enabling humans to assist robots effectively is challenging. Assistance methods must be intuitive and accessible to non-experts, avoiding the need for technical knowledge or coding [31]. Moreover, a critical capability for robots is the ability to autonomously detect when they are missing specific actions required to achieve a task goal. Instead of requesting demonstrations of an entire task, robots should be able to identify the missing gap and seek targeted human assistance. This ability is essential for reducing the burden on human users while ensuring that robots continuously expand their skill sets in a structured and effective manner.

1.1 Background and Research Gaps

Failures are ubiquitous in robotics, prompting the development of various taxonomies to classify and understand them. For example, Steinbauer [12] analyzed common faults encountered in the RoboCup@Home competition and categorized them into two main types: hardware failures, such as issues with sensors, manipulators, or controllers, and software or algorithmic failures, which affect decision-making, behavior execution, or low-level control. Failures can also arise from human-robot interactions. For example, the robot might not act as expected by a human user [32] or a human blocks the path of a robot [31]. A more comprehensive exploration of these interaction-based failures is provided in the taxonomy proposed by Honig and Oron-Gilad [10]. Beyond these taxonomies, Laprie [33] organized failures by severity, while O'Hare et al. [34] classified them based on recoverability, distinguishing between failures that can be resolved autonomously and those that require external intervention.

In this thesis, we particularly distinguish between *task planning failures* and *task execution failures*. This distinction is based on our adoption of the Automated Planning (AP) framework, which plays a crucial role in robot deliberation [35]. AP enables robots to break down complex, long-horizon tasks, such as setting a table, into sequences of actions (e.g., navigating to the kitchen, opening a drawer, picking up plates, and placing them on the table). AP generates task plans prior to execution, determining which actions must be performed and in what order to achieve a goal. Brooks [36] refers to *failure* as a "degraded state or capability that causes the behavior or service performed by a system to deviate from its ideal, normal, or correct functionality". Consistent with this definition, we define task planning and execution failures as follows:

- **Task Planning Failures:** A task planning failure occurs when the robot is unable to generate a valid plan to achieve its goal. In this case, the planning process deviates from its ideal functionality by failing to produce a sequence of actions that lead to the desired outcome. For example, a robot tasked with setting a table might fail to generate a plan if the drawer containing plates is obstructed by a chair, and the robot lacks the necessary skill to move the chair out of the way.
- **Task Execution Failures:** A task execution failure occurs when the robot successfully generates a plan but encounters difficulties while ex-

ecuting one or more actions within that plan. This type of failure is characterized by deviations from the expected outcomes of individual actions. For instance, a robot attempts to stack two cubes but the top cube falls to the floor instead of staying in place.

In this thesis, we address three aspects of handling failures:

1. Failure prediction and prevention (proactively)
2. Failure explanation (retrospectively)
3. Human-supported failure handling (when robots cannot handle failures autonomously)

The following subsections review relevant work in these three areas, highlighting the challenges and limitations of current approaches, and identifying and motivating the research questions addressed in this thesis.

1.1.1 Failure Prediction and Prevention

One area that has extensively addressed failure detection, prediction, and prevention is fault detection and diagnosis (FDD). Fault detection refers to recognizing when a fault occurs, while fault diagnosis involves identifying its underlying causes [37]. FDD methods [38] monitor a system's state and compare it to known fault patterns or historical observations of normal behavior. Common techniques include particle filters [39], [40] or outlier detection [41]. Once a fault is detected, fault tolerance strategies help maintain system functionality. These include maintenance and repair protocols [37], or using fault-tolerant control strategies [42]. However, many FDD approaches are tightly coupled to specific hardware configurations, sensors, and actuators, making adaptation to new systems data-intensive [43]. Moreover, most FDD methods focus on low-level system states, such as sensor and actuator signals, primarily addressing internal faults [37]. However, as robots increasingly operate in human environments, where explainability and interpretability are crucial, it is equally important to embed fault detection within a higher-level reasoning framework. This allows robots to not only detect failures but also assess their external impact on the overall task objective or the current action being executed, enabling more transparent and effective failure handling. For example, if a robot cleaning a table stacks several plates but drops one due to misalignment, the failure can be explained from two perspectives: an internal perspective might attribute the issue to sensor inaccuracies, motion planning errors, or action policy limitations. In contrast, an external perspective explains the observable effects, such as the plate being misaligned or dropped from too high. This abstraction layer facilitates communication with non-expert users by focusing on observable outcomes rather than internal mechanisms. It also supports building robot-agnostic models by decoupling environmental changes from robot-platform-specific features.

There is also existing work on monitoring robot execution at a symbolic level. Task and Motion Planning (TAMP) frameworks [44], [45] integrate symbolic task planning with physical execution, enabling robots to detect and react to failures at a higher level. These frameworks compare an action's post-execution state to its expected symbolic model, allowing robots to replan when discrepancies arise. That means the robot first fails and then recovery strategies are devised. This is effective for failures that can be resolved by retrying, such as stacking cubes, if a cube falls, the robot can simply attempt the action again. However, failures with irreversible consequences, such as dropping a fragile object, require more sophisticated failure-handling strategies. While such approaches focus on diagnosing and mitigating failures after execution, this thesis aims to predict failures before they occur, enabling proactive intervention rather than reactive replanning.

Other frameworks, such as State Machines [46] and Behavior Trees [47], facilitate reactive failure detection and recovery. However, they require the definition of transitions, fallbacks, or events, which must either be manually specified or learned from experience [48]. An alternative, more implicit approach to handling execution failures is Reinforcement Learning (RL). Methods such as [49] decouple rewards by providing positive reinforcement when the goal is achieved while explicitly penalizing failures, such as colliding with a wall, to discourage them during exploration. However, since failures are implicitly encoded within the learned policy, these approaches lack interpretability and explainability. Inceoglu et al. [50] propose a modular and hierarchical method for safe robot manipulation in multi-objective settings, where multiple failure-prevention policies are learned. Nevertheless, they still rely on hand-crafted state machines to determine when these policies should be triggered. This highlights a key challenge: learning predictive models capable of identifying potential failures and their causes in a timely manner.

Learning predictive models is particularly difficult when the error does not manifest immediately but instead has delayed consequences. For instance, in the task of stacking four cubes, a slight misalignment between the first and second cube may not cause an immediate failure. However, this misalignment could compromise the stability of the entire structure, leading to a collapse when the fourth cube is added. Predictive models must be able to capture such causal chains, spanning multiple actions, to anticipate failures that may emerge over extended horizons. In this thesis, we therefore propose to learn causal models that capture if and how certain environment factors (e.g., the offset between two cube centers or the cube color) impact the outcome of a task (e.g., the ability to stack two cubes on top of each other).

Research Question 1

How can we detect and learn cause-effect relationships in robot tasks involving timely shifted and erroneous action effects?

Cognitive scientists emphasize that constructing explanatory causal models is fundamental to human decision-making [51], [52]. Similarly, causal models can empower robots to identify failure causes and reason about interventive actions that enable robots to prevent future failures.

Research Question 2

How can a robot use the previously obtained causal models to predict and prevent future failures?

1.1.2 Failure Explanation

With robots increasingly being deployed in human environments, a key consideration is enabling humans to understand the decision-making processes of robots. This ability becomes particularly crucial when robots encounter failures, as it plays a critical role in fostering trust, acceptance, and motivation to engage with robotic systems [22], [23], [24]. This consideration reflects a broader trend in Artificial Intelligence (AI), which has led to the emergence of the subfield of Explainable Artificial Intelligence (XAI). XAI aims to make the underlying models of an AI or robot's decision-making system understandable to stakeholders [53], and it has become an active area of research in recent years [54]. Two key concepts in XAI are interpretability and explainability. Interpretability refers to systems that generate decisions based on humanly understandable rules, while explainability involves providing explicit explanations and justifications for those decisions [17].

Historically, much of XAI research has focused on addressing the interpretability and explainability of black-box deep learning systems [55], [56]. For instance, Zhang et al. [57] explains the rationales behind a pre-trained convolutional neural network's predictions using a decision tree. Similarly, LIME [58] (Local Interpretable Model-agnostic Explanation) creates an interpretable representation to explain decisions made by any classifier or regressor. However, as robots increasingly interact with humans in shared environments and collaborate on tasks, there is a growing demand for explainable agency in robotics. Most existing XAI methods are designed for technical experts with expertise in AI and Machine Learning (ML), often remaining inaccessible to end-users who lack this specialized background [25]. This gap highlights the need for explanations that are understandable to non-expert users, which we address in this thesis.

However, providing explanations for non-expert users is a difficult challenge, as social sciences suggest that humans tend to anthropomorphize machines and expect explanations to resemble human-to-human communication [17]. Human-centered explanations typically exhibit the following characteristics:

- Contrastiveness: Humans often prefer counterfactual explanations, such as, "I would have arrived on time if I had taken a different route."
- Selectiveness: Explanations highlight key causes rather than the entire causal chain. For example, explaining why someone was late to work might involve mentioning a traffic jam but not the car's color or unrelated events from the previous week.
- Abstraction over Probabilities: Humans rarely use probabilities in everyday explanations. For instance, rather than stating that the success probability of stacking a tower of cubes was 12%, people tend to refer to abstract features, such as the cubes being misaligned.

Causality is another critical component in creating effective human-centered explanations [54]. Causal models enable robots to identify variables relevant to task outcomes and predict the success or failure of actions. However, causal models alone are insufficient for generating human-centered explanations for failures.

Research Question 3

How can a robot use previously obtained causal models to generate contrastive and selective explanations for task failures?

1.1.3 Model Transfer and Data Complexity

One significant challenge of using causal models in robotics is the often substantial data requirements needed to obtain them. While sufficient time for offline training enables robots to leverage simulations to precompute these causal models, real-world environments frequently introduce changes or variations that necessitate dynamic adaptation. Such changes may involve entirely new tasks or familiar tasks applied to novel objects. For example, a robot may transition from stacking one cube to stacking two cubes or adapt from dropping a sphere into a bowl to dropping it into a glass. Successfully handling these variations is especially critical when the robot must operate in a zero-shot fashion [59]. This challenge is further compounded in scenarios like human-robot interaction, where data collection is inherently expensive and difficult, as these applications often involve real human participants[1].

Priors have been proposed as a solution to mitigate the substantial data requirements in learning (causal) Bayesian Networks. Previous research has demonstrated that transferring the BN variable structure can help generalize across tasks [60], [61]. The work in this thesis specifically focuses on transferring BN probability distribution parameters. Our approach aligns with methods that transfer parameters from multiple prior BNs to a new target network [62], [63]. These methods use a relatedness measure to identify prior BN variables that closely match target BN variables, transferring a weighted combination of their conditional probability distribution parameters. However, a key limitation of these approaches is their assumption that the prior and target variables have the same number of probability distribution parameters, a condition that often does not hold. For instance, stacking two cubes

[1]The following review of prior work in this area is adapted from Paper D.

has more causally relevant parameters than stacking only a single cube, because the success of the second stacking action also depends on the previous stack.

Some works focus on transferring robot task executions to novel situations using ontologies. Bauer et al. [64] learn probabilistic action effects for dropping objects into various containers and generalize success probability predictions to different object types, such as bowls and bread boxes. Another approach [65] combines an ontology with edge weights that represent the generalization strength between object classes for tasks like grasping objects and stowing them in drawers. While these studies aim to learn generalizable prediction models, we emphasize identifying the object properties and environmental preconditions that are critical for transferring tasks more reliably.

In addition to ontologies, data-driven models such as neural networks have demonstrated success in transfer tasks [66], [67]. For instance, Xu et al. [66] learn reward functions from related tasks in an inverse reinforcement learning setting. Another study [67] highlights domain adaptation and a vision-based grasping approach to transfer grasps from simulated to real-world objects. However, while effective, these methods face limitations in human-centered scenarios, where generalization failures are difficult to analyze due to the lack of semantically meaningful features in neural networks.

Research Question 4

How can we increase the data efficiency for learning causal models? How can we transfer causal models between tasks?

1.1.4 Human Support when Robots Fail

The literature identifies two primary roles for human support in addressing robot failures: directly solving (sub-)tasks on behalf of the robot or teaching the robot how to perform them independently. The first role is particularly relevant when the robot lacks the physical capabilities required to continue task execution. For instance, in [26], a robot requests users to call an elevator because it is physically incapable of doing so. Similarly, in [27], a robot seeks assistance in picking up IKEA furniture pieces when mechanical problems such as grasping failures or perceptual issues such as a part not being visible prevent it from completing the task. In [25], a framework is introduced that leverages failure explanations to guide users in resolving problems, such as moving objects to eliminate occlusions or repositioning items to facilitate manipulation. These approaches primarily address recovery from failures during task execution.

In this thesis, we focus on the second role, where a human acts as a teacher. In particular, we investigate situations where no feasible plan can be generated because the robot lacks necessary capabilities to accomplish the task goal. In such cases, we would like the human to provide instructions or demonstrations to enable the robot to learn the missing actions or whole tasks. An important consideration is that many envisioned human teachers are non-experts which require intuitive teaching modalities that do not require any coding skills or in-depth experience in robotics. Early approaches include modular skill-based programming and block-based graphical interfaces, which allow users to define logic without coding [68], [69], or abstracting action sequences from human demonstrations [70]. Van Waveren et al. [28] introduced a block-based interface for crowdsourcing simple robot programs, while another study [29] explored using non-experts to repair reinforcement learning (RL) policies. A widely studied approach is Learning from Demonstration (LfD), also known as Programming by Demonstration [30]. LfD encompasses various teaching methodologies [71], including:

- Kinesthetic teaching, where a human physically guides the robot through desired motions.
- Teleoperation, where the human controls the robot using external tools like joysticks or graphical interfaces.
- Passive observation, where the robot observes the human performing a task to learn from the demonstration.

LfD also varies in its outcomes, which may include policies (mapping states to actions), cost/reward functions, or task abstractions [71].

In this thesis, we utilize *passive observations* through a Virtual Reality (VR) environment to collect human demonstrations because VR allows users to perform demonstrations naturally with their own “hands and body”, eliminating the need for direct interaction with the robot, as required in kinesthetic teaching or teleoperation. Additionally, VR offers full access to the simulated environment, making it possible to capture variables that might be challenging to measure in real-world settings. Furthermore, we focus on *task abstractions* as the outcome of demonstrations for three key reasons:

- High-level task descriptions abstract low-level execution details (e.g., joint trajectories) into meaningful actions that emphasize intended effects, which aids in planning long-horizon tasks [72]. Also, due to their symbolic nature, action models can be collected and reused across different tasks.
- Task abstractions can be designed to be robot-agnostic, allowing the reuse of action models across different robots.
- High-level, symbolic action descriptions enhance the interpretability of plans, making it easier for humans to understand and debug the robot's behavior.

Extracting high-level task abstractions from human demonstrations in Virtual Reality (VR) is challenging because demonstrations typically consist of a continuous flow of states. To effectively learn from demonstrations, robots must segment this continuous data stream into meaningful actions while identifying the relevant preconditions and effects for each action.

Beyond extracting individual actions, robots should also be capable of reusing and integrating previous experiences with newly obtained demonstrations. This knowledge should be transferable across robots and flexibly adaptable to various situations, ensuring broader applicability and efficiency in learning.

However, even with the ability to transfer and reuse learned capabilities, a robot's existing set of capabilities may sometimes be insufficient to complete a new task without additional demonstrations. A critical limitation is that robots often struggle to reason about which specific sub-tasks are missing, forcing humans to either demonstrate the entire task sequence, including actions the robot already knows, or manually identify and teach only the missing components. This process can be tedious, repetitive, and discouraging for users. Additionally, expecting users to have a deep understanding of the robot's capabilities contradicts the core goal of learning from demonstrations, as non-experts often lack this knowledge. A more practical solution would empower the robot to identify and communicate the specific parts of the task it is missing, allowing human demonstrators to focus solely on teaching those necessary components.

While users should not be required to have a deep understanding of the robot's capabilities, some may still want a deeper understanding of what the robot has learned from the demonstrations, which motivates the use of interpretable learning methods.

Research Question 5

How can a robot learn tasks in an efficient, agnostic, and interpretable way by observing human demonstrations?

Most Learning from Demonstration (LfD) methods focus on single-robot scenarios, making them unsuitable for multi-robot tasks that require coordination, shared space management, and action synergies[2]. Stenmark et al. [73] developed a programming interface for dual-arm robots based on PbD, enabling the modular reuse of skills. A subsequent study enriched skill descriptions with high-level semantics, incorporating preconditions and postconditions via a skill ontology [74]. However, skill sequences still needed manual definition by the operator. Mayr et al. [75] introduced a skill-based framework for multi-agent tasks that enables autonomous task planning. Nevertheless, the semantic skill descriptions were manually encoded, not learned from demonstrations. Similarly, Wang et al. [76] tackled task sequence planning for dual-arm robots sharing workspace, modeling geometrical constraints and semantic skill relations. While the planner computed feasible plans, it relied on a predefined skill library and did not address tasks that require concurrent actions. The Planning community proposed learning multi-agent domains from plan traces [77], [78]. However, these approaches often require hundreds of demonstrations, limiting their practicality for non-experts. When learning multi-agent skills from demonstrations the following problems must be addressed:

- Current planning systems typically generate sequential plans, limiting their ability to handle concurrent multi-agent tasks.
- Shared spaces and resource constraints in shared work areas are often overlooked which can lead to plans with collisions.
- Action synergies, where tasks (e.g., closing a bottle) require parallel actions by multiple robots, are not addressed.

[2]The following review of prior work in this area is adapted from Paper E.

Research Question 6

How can we extend current Learning from Demonstration methods from teaching single- to multi-agent tasks?

1.2 Thesis Goals and Contributions

This thesis is structured around two primary goals, each tackling a subset of the six research questions identified before.

Thesis Goal 1

Explainable Handling of Robot Task Execution Failures based on Causal Task Models

The first goal of this thesis is to enhance a robot's ability to operate effectively in human environments. Specifically, we aim to increase the autonomy and reliability of robots by proactively predicting and preventing failures caused by unexpected changes in the environment. Additionally, we aim to equip robots with the ability to explain failures retrospectively, a crucial factor in fostering trust and increasing user acceptance. In particular, we focus on providing explanations tailored for non-expert users.

To achieve this, we leverage causal models, which play a key role in human decision-making and can similarly enable robots to identify actionable causes of failures. Our research explores several key questions: how to learn causal models from simulated data (RQ1), how to use them for failure prediction and prevention (RQ2), how to generate causal, contrastive, and selective failure explanations (RQ3), and how to transfer these models to new tasks or acquire them efficiently from limited data (RQ4). By integrating causal models into failure prediction and explanation, this thesis aims to enhance both the reliability and explainability of robots in human environments.

Contributions towards Goal 1:

- In Papers B and C, we propose a pipeline for learning symbolic, robot-agnostic causal models, specifically, Causal Bayesian Networks (CBNs), from simulation (RQ1). This pipeline involves three steps: (1) task modeling (variable definition), (2) CBN structure learning, and (3) CBN parameter estimation.

- Paper B proposes a new method for enabling robots to provide contrastive and selective explanations for task execution failures based on causal models. When a task fails, the robot contrasts the failure state with the nearest state that would have enabled success, identified through a breadth-first search guided by the learned causal model (RQ3).
- Paper C paper presents a novel approach that leverages causal models to proactively predict and prevent task execution failures (RQ2).
- Paper D enhances the transferability of CBNs by proposing three strategies for generating and transferring informed distribution priors based on the semantic similarity between two CBNs (RQ4).
- Paper G presents a CBN for predicting human perceptions of robot competence and intent during navigation, addressing challenges in learning from limited data. Additionally, it introduces a method for improving perceived robot performance through counterfactual navigation behaviors. Paper G also proposes a framework for integrating different variable types, continuous, discrete, and time-series, into a single CBN, expanding the range of tasks that can be modeled (RQ4).

Thesis Goal 2

Human Support for Robot Planning Failures through Learning from Demonstration

Our second goal is to equip robots with the ability to continuously learn and adapt to novel situations they may not have been initially trained to handle, which is a common challenge in dynamic human environments. Specifically, we aim to enable humans to assist robots by teaching them new tasks or missing actions.

To achieve this, we aim to develop new Learning from Demonstration (LfD) methods that can segment human demonstrations into meaningful actions along with their relevant preconditions and effects, allowing robots to reuse and integrate previous experiences with new demonstrations for broader applicability. To streamline the demonstration process, robots should be able to identify missing actions, enabling targeted demonstrations so that humans

do not have to teach the entire task each time. Our approach will also utilize interpretable learning methods to enhance user understanding of the robot's learning process (RQ5). Additionally, these methods should accommodate dual-arm tasks that require coordination between multiple arms and may be restricted in their use of shared resources (RQ6).
Contributions towards Goal 2:

- In Paper A, we propose a system that learns tasks from human demonstrations in a Virtual Reality environment. It automatically segments and classifies the demonstration into high-level actions using a pre-learned decision tree, then generates symbolic action descriptions by linking hand activities to corresponding environment changes (RQ5).

- Paper F introduces a method that uses combinatorial search to identify the smallest necessary change in the initial task conditions to enable the robot to solve the task with its current knowledge. This allows the human to demonstrate only the missing subtask rather than the entire task, streamlining the teaching process (RQ5).

- Paper E extends the LfD approach from Paper A to multi-agent environments, introducing constraints to prevent resource conflicts and schedule parallel actions effectively (RQ6).

1.3 Thesis Overview

Chapter 2: Begins with a background section on the role of causality in robotics. We then describe how we model robot tasks using causal models, specifically Causal Bayesian Networks (CBNs), and outline our approaches for acquiring these models (RQ1) and transferring them to different tasks (RQ4). Following this, we present methods for explaining failures retrospectively (RQ3) and for predicting and preventing future failures (RQ2).
Chapter 3: Presents our human-centered approach to addressing task planning failures, starting with an introduction to key concepts such as automated planning, the formalization of planning problems, and task planning failures. Finally, we summarize the challenges and proposed methods to enable robots to mitigate planning failures through flexible, robot-agnostic, and interpretable task learning from human demonstrations (RQ5, RQ6).

Chapter 4: Provides a brief summary of all the papers that are included in the thesis.
Chapter 5: Concluding discussion and future work.

CHAPTER 2

Our Causal Approach for Explaining, Predicting and Preventing Task Execution Failures

Causality plays a central role in human cognition [79]. By building causal models, humans can effectively perform few-shot concept learning, generalize across different situations, and fill in missing information, such as inferring unobserved features, making predictions, or determining the functionality of an object [51], [80]. Causal reasoning also helps distinguish core features from irrelevant ones, thereby enhancing decision-making and reasoning abilities [79].

For similar reasons, causality is gaining attention in Machine Learning (ML) and Reinforcement Learning (RL). In ML, causal representations are expected to improve generalization, especially in non-stationary environments or under interventions [81], and make models more interpretable [82]. In the field of robotics, we aim to leverage these causal insights to enhance understanding, explanation, prediction, and the prevention of failures[1].

While the application of causality in robotics has been relatively nascent [20], its importance is increasingly being recognized. In robotics, causal reasoning has been applied to identify task-relevant variables, enabling robust and in-

[1]The following review of work on causality in robotics is adapted from Papers B, C, D, and G.

terpretable decision-making [83]. For instance, CREST [84] employs causal interventions to identify environmental variables that influence reinforcement learning (RL) policies, while [85] builds on CREST to discover and learn a diverse set of interpretable robot skills from limited data. Similarly, CAR-DESPOT [86] integrates causal modeling into an anytime online POMDP (Partially Observable Markov Decision Process) planner for more effective planning. They address biases caused by unmeasured confounders (shared causes of multiple variables, that introduce spurious correlations that causal models aim to disentangle from true causal relationships). Furthermore, partial parameterizations of causal models are learned offline. In [87], the authors propose a probabilistic reasoning method, combining CBN-based modeling, physics simulation, and probabilistic programming to predict manipulation outcomes and guide next-best action selection under uncertainty, as demonstrated in cube stacking tasks. To support further research, benchmarks like CausalWorld [88] provide simulated environments for evaluating policies under distribution shifts, enabling interventions and structured learning curricula. These tools continue to foster advancements in causal structure learning and transfer learning across diverse robotic tasks. Many RL approaches represent robot dynamics through state-action descriptions, whereas our work focuses on high-level causal features by leveraging a symbolic, object-centered environment representation to enhance explainability, improve decision-making, and make causal reasoning more accessible in robotic systems.

In [89], a humanoid iCub robot learns causal rules to distinguish between relevant and irrelevant features in physical interactions and affordances. For example, it learns from cumulative experiences that dropping heavy objects into a jar increases the water level, while variables like color are irrelevant. Brawer et al. [90] propose a causal approach to tool affordance learning, focusing on identifying how tools interact with objects. Other works use Bayesian Networks to uncover statistical dependencies between object attributes, grasping actions, and task constraints [91]. While these approaches aim to generalize task execution through graphical models, they do not address how these models can explain failures. Similarly, probabilistic methods have been explored to generalize action effects. For example, Bauer et al. [64] model the effects of dropping objects into containers, generalizing predictions for objects like bowls and bread boxes. However, their method does not investigate why success probabilities differ across objects. Uhde et al. [92] focus on learning causal relationships between sequential actions in household tasks, such as identifying the causal link between opening a drawer and retrieving plates. Their approach is based on data collected from expert demonstrations in virtual reality environments. While they extract causal links between actions, our focus differs by examining causal relationships between environment variables, such as object features and action outcomes, to provide deeper explanations of task dynamics.

Due to its promise to increase explainability and interpretability, researchers have started to use causal models in the context of Human-Robot Interaction (HRI) [93]. Some early work applied causal time series analysis to model human and robot motion behaviors based on variables like distance to a goal, angle, and velocity [94]. Similarly, Edström et al. [95] investigated enhancing a robot's causal understanding by allowing it to ask humans about causal relationships. Their algorithm selects direct causal effects to query, based on partial causal graphs (PDAGs) learned from observations. Unlike prior approaches, our goal is to leverage causality for timely prediction and prevention of failures and for providing retrospective explanations when failures occur.

2.1 Learning Causal Task Models

In our work (Papers B, C, D and G), we model a robot task T, such as stacking cubes, as a Causal Bayesian Network (CBN). Formally, CBNs are defined as *Directed Acyclic Graphs* (DAG)

$$\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A}), \tag{2.1}$$

where the nodes $\boldsymbol{X} = \{X_1, X_2, ..., X_N\}$ are a set of N random variables X_i and $\boldsymbol{A}$ is the set of arcs [96] that describe the causal connections between the variables. An exemplarly graph with five nodes is shown in Fig. 2.1. Based on the dependency structure of the DAG and the *Markov property*, the *joint probability distribution* of a CBN can be factorized into a set of *local probability distributions*, where each random variable X_i only depends on its direct parents $\mathbf{Pa}_{X_i}$:

$$P(X_1, X_2, ..., X_N) = \prod_{i=1}^{N} P(X_i|\mathbf{Pa}_{X_i}) \tag{2.2}$$

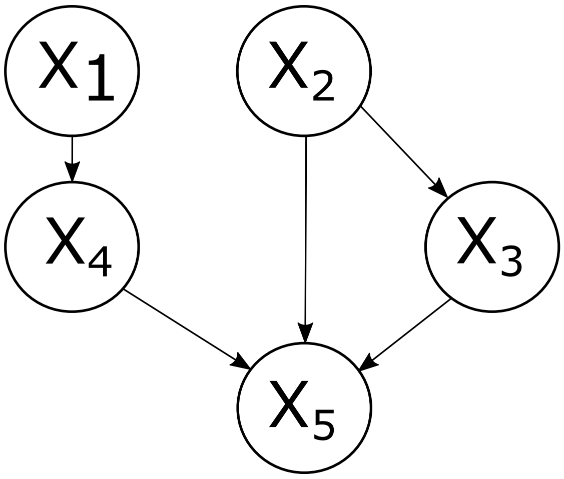


Figure 2.1: Shows an exemplary Causal Bayesian Network (CBN) graph with five nodes $\boldsymbol{X} = \{X_1, X_2, X_3, X_4, X_5\}$.

The set of random variables $\boldsymbol{X}$ represents potentially task-relevant features of the task execution or environment (causes) and their effect on the task outcome (effects). In Papers B and C, we propose a multi-step approach to detect causal relationships in the form of a Causal Bayesian Network from task simulations (RQ1):

Step 1 **Variable Definition**: Identify and define the set of variables $\boldsymbol{X}$ that are relevant to the task or system domain being modeled (currently manually performed by the experiment designer). These variables represent both the potential causes and effects within the domain.

Step 2 **Variable Preprocessing**: Depending on the variable type, preprocessing steps are required, particularly the discretization of the data.

Step 3 **Causal Model Learning**: Using the defined and preprocessed variables, this step involves:

- **Step 3.1** **Structure Learning:** Obtain the graphical structure of the CBN, which represents the causal relationships among the variables in the form of a Directed Acyclic Graph (DAG).
- **Step 3.2** **Parameter Learning:** Estimate the conditional probability distributions associated with each variable, quantifying the dependencies and interactions captured by the structure.

2.1.1 Variable Definition

We conceptually divide $\boldsymbol{X}$ into two subsets: treatment variables (causes) $\boldsymbol{C} \subset \boldsymbol{X}$ and outcome variables (effects) $\boldsymbol{E} \subset \boldsymbol{X}$. Treatment variables $\boldsymbol{C}$ are those that can be actively decided and set, whereas outcome variables $\boldsymbol{E}$ are passively measured at the end or during the task. We denote a specific parametrization of $\boldsymbol{X}$ as $\boldsymbol{x} = \{X_1 = x_1, X_2 = x_2, ..., X_N = x_N\}$. We define another set $\boldsymbol{X}_{\text{goal}}$ which contains all possible variable parameterizations that denote a successful action execution. A task is succesfull *iff* its parameterization $\boldsymbol{x} \in \boldsymbol{X}_{\text{goal}}$. In this thesis, we assume $\boldsymbol{X}_{\text{goal}}$ is provided a priori. In other words, we assume the robot can identify unsuccessful task executions by comparing the outcomes of its actions (as defined by the outcome variables $\boldsymbol{E}$) with $\boldsymbol{X}_{\text{goal}}$. However, the robot has no prior knowledge about which variables in $\mathbf{X} = X_1, X_2, ..., X_N$ belong to $\boldsymbol{C}$ or $\boldsymbol{E}$, nor how these variables are related. This information is generated by learning the CBN.

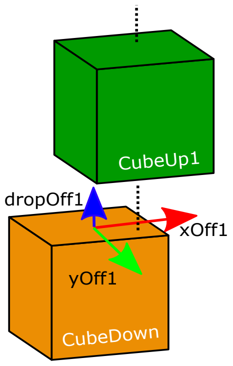
Currently, variable selection is a manual process conducted by the person responsible for setting up the simulation experiment and collecting the data. Below, we present several examples drawn from the included papers. In Examples 1 (Papers B, C, D) and 2 (Papers B, D), we developed our own simulation environments, which offered greater flexibility in selecting the task-specific variables[2]. In contrast, in Example 3 (Paper G), we applied our method to an existing HRI dataset (the SEAN Together dataset [97]), requiring us to use the available data and carefully define the variables based on the dataset variables.

Example 1. ***Stacking-Cubes Task:*** *In the cube stacking scenario, the environment contains two cubes: CubeUp and CubeDown (see Fig. 2.2). The goal of the stacking task is to place CubeUp on top of CubeDown.*

We define six variables as $\mathbf{X} = \{$`xOff`, `yOff`, `dropOff`,`colorDown`, `colorUp`, `onTop`$\}$ (see Fig. 2.2-b) for their definitions). In this task, `onTop` is the goal variable, and any variable parametrization where `onTop` $= 1$ corresponds to a successful stacking task.

In Papers C and D, we also utilized tasks where the robot had to stack two or three cubes. The variables for these examples were defined similarly as in Fig. 2.2, with subscripts to distinguish between the different cubes (e.g.,

[2]The datasets and implementations for Paper B and C are publicly available under `https://gitlab.com/craft_lab/causality-robotics/explainandpreventrobotfailures`

a)


b)

Variable	Meaning
$\texttt{xOff}_1$	$\text{cubeUp}_1.x - \text{cubeDown}.x$ (distance between cube centers)
$\texttt{yOff}_1$	$\text{cubeUp}_1.y - \text{cubeDown}.y$
$\texttt{dropOff}_1$	$(\text{cubeUp}_1.z - \text{cubeDown}.z) - \text{cubeLength}$ (distance between cube surfaces)
$\texttt{onTop}_1$	indicates if cubeUp_1 is on top of CubeDown after the stacking process

Figure 2.2: a) visualizes the used variables $\mathbf{X}$ of Example 1 and b) describes their meaning. Please refer to the Papers B, C and D for more information about the respective variable distributions and ranges used during the data generation process in our Unity3d simulation environment. Source: Paper C. © 2023 Elsevier (RAS).

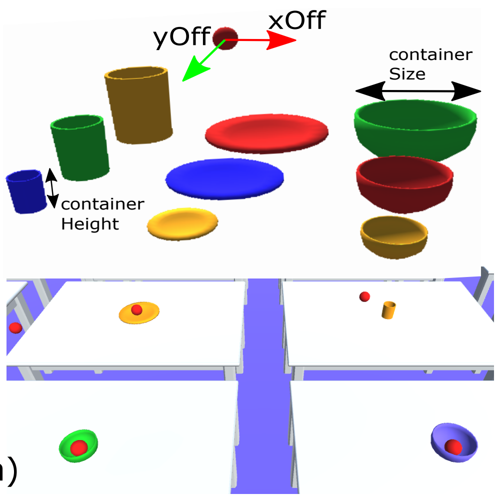
$\texttt{xOff}_1$ denoting the x-offset of the first stacked cube, or $\texttt{onTop}_2$ describing the outcome of the second stack).

Example 2. ***Sphere-Dropping Task:*** *The robot needs to drop spheres into different containers. The environment contains a Sphere and one of several possible Containers, which are shaped like a plate, bowl or glass (see Fig. 2.3).*

We define eight variables as follows: $\mathbf{X} = \{\texttt{xOff}, \texttt{yOff}, \texttt{inCont}, \texttt{containerHeight}, \texttt{containerSize}, \texttt{containerType}, \texttt{containerCurvature}, \texttt{containerColor}\}$. In this task, `inCont` is the goal variable, and any variable parametrization where $\texttt{inCont} = 1$ corresponds to a successful stacking task.

Example 3. ***Robot-Following Task:*** *This is a dynamic HRI task where a robot guides a person (the human follower) to a pre-specified goal, as introduced in the SEAN Together dataset [97]. The task takes place in a Virtual Reality (VR) simulation of a warehouse, which includes obstacles such as shelves and other autonomous agents (e.g., pedestrians) navigating the environment (see Fig. 2.4a). The human follower is controlled by a participant wearing a VR headset.*

For Example 3, we used the CBN to predict how the human follower is rating the performance of the robot based on variables such as the robot's trajectory or map of the environment (see Fig. 2.5). Furthermore, if the robot

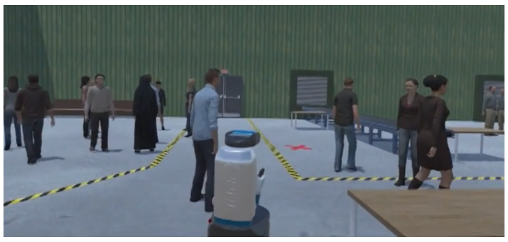


a)

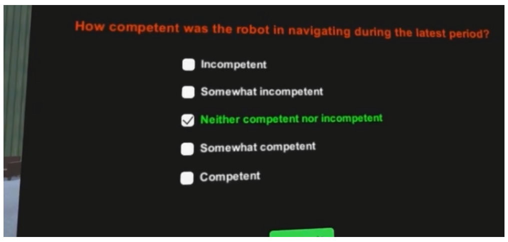
b)

Variable	Meaning
`xOff`	sphere.x − container.x (distance between sphere and container center)
`yOff`	sphere.y − container.y
`containerType`	denotes if container is a glass, bowl or plate
`containerSize`	denotes the diameter of the container
`containerHeight`	denotes the height of the container
`containerCurvature`	denotes if the container has a curvature
`containerColor`	denotes the color of the container
`inCont`	denotes if the sphere ended up in the container

Figure 2.3: a) visualizes some of the variables **X** from Example 2, b) describes the variable meanings. Please refer to Paper B for more information about the respective variable distributions and ranges used during the data generation process in our Unity3d simulation environment. Source: Adapted from Paper B. © 2022 IEEE (RAL).



(a) Shows an example where the person is following a robot inside a virtual warehouse.



How competent was the robot in navigating during the latest period?

- Incompetent
- Somewhat incompetent
- [✓ Neither competent nor incompetent]
- Somewhat competent
- Competent

(b) Shows the survey view where the human follower rates the robot's competence.

Figure 2.4: In 2.4a, the robot is guiding a person to the goal location, which is marked with a red X on the ground. This X was visible to people in the VR simulation when they were in the line-of-sight of the goal location. At certain intervals, the simulation is paused and the person will rate the robot's competence by answering survey questions within the VR simulation (see Fig. 2.4b). Source: Paper G.

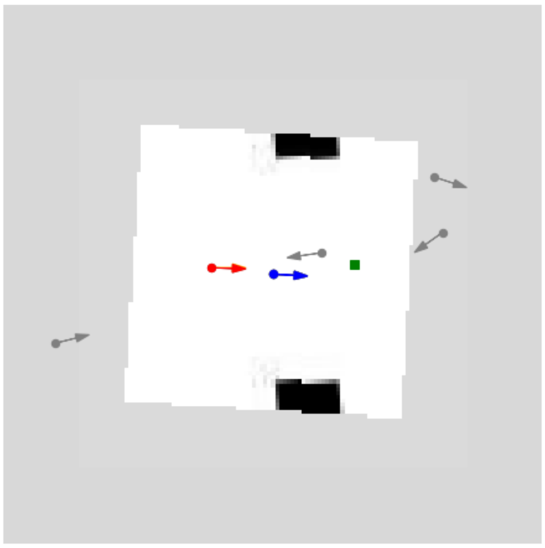


Figure 2.5: Shows a top-down visualization of the robot-follower task environment: The blue arrow represents the robot's position and orientation. The red arrow indicates the body orientation of the human follower. Other people in the environment are indicated by grey arrows. The destination, located at the top of the images, is a green rectangle. The black areas are static obstacles and the white areas is navigable space around the robot. The surrounding grey region is beyond the 7.2m public space where the 2D map was recorded.

is predicted to have a low perceived competence, we use the CBN to find an alternative trajectory that will lead to higher perceived competence.

The original set of variables in the SEAN Together dataset include:

- **Agent poses** were calculated relative to the robot. Only the agents within a distance of 7.2m from the robot are considered in the set, as this constitutes the robot's public space, defined by Hall's proxemics [98]. Each feature was composed of (x, y, θ), where x, y represent the position, and θ denotes the yaw angle of the agent's body.

- **The goal position**, to which the robot guided the person, describes the robot's proximity and relative orientation to the desired destination.

- **A 2D map**, cropped around the robot (7.2m $\times$ 7.2m), was used to describe the occupancy of nearby space by static objects, also known as Region of Interest (ROI).

During the navigation, the interactive simulation was occasionally paused, and participants were asked to provide their impressions of the robot's performance through an interface within the simulation.

The participants provided ratings along several dimensions (see Fig. 2.4b):

1. How *competent* was the robot at navigating?

2. How clear was the robot's *intention* during navigation?

Participants provided ratings using a 5-point Likert responding format. For example in the case of the competence dimension, (1 point) corresponded to "incompetent," (2 points) "somewhat incompetent," (3 points) "neither competent nor incompetent," (4 points) "somewhat competent," and (5 points) "competent". Since in our work, we want to modify the robot's behavior when the person perceives the robot as performing poorly (low competence), we transformed the 5-point format to a binary rating where a value in the range $[1, 3]$ indicated *low=0* and $[4, 5]$ indicated *high=1.*

The dataset consists of $2,964$ examples, each consisting of time-series data for the aforementioned variables of an 8-second window, along with a single competence rating recorded at the end of the interaction. Formally, consider a dataset of observations and performance labels, $\mathcal{D} = (o1_i : T, y_i)$, where $o1 : T$ is an observation sequence of length T (in our case fixed to 8 seconds), y is a performance rating given by a person interacting with the robot at the end of the sequence, and i identifies a given data sample.

Although 2,964 examples constitute a relatively large dataset for the field of HRI, the dataset size still presents a significant challenge for training predictive models. We, therefore, approached the variable selection process for learning the causal model with several objectives in mind:

1. The number of parent variables for each node in the graph should be limited, to keep the parameter estimation feasible.

2. Each node should have a clear semantic meaning, making the model more understandable to human users.

3. Wherever possible, nodes (in particular the parent variables of competence and intention) should be actionable by the robot.

To achieve these goals, we manually implemented three key modifications to the original SEAN Together variables. First, we combined variables that express distances, originally measured in two variables (distance in (x, y) direction), into a single new variable representing the L^2 norm. This transformation was applied to both the robot-goal distance and the human-robot distance. Second, we converted all distances, originally measured as absolute values, into relative changes with respect to the first value of their respective 8-second time series. In this formulation, each distance time-series trajectory begins at 0 (representing the current location) and indicates how the distance changes over the 8-second interval. This adjustment was made with the executability of failure prevention actions in mind. For example, if our method were to recommend an alternative robot trajectory with a different initial distance to the goal, executing such a trajectory would be infeasible without "teleporting" the robot to a new initial state, an operation that is physically impossible. By ensuring all distance trajectories start at 0, every robot-goal trajectory becomes executable. Finally, we eliminated all variables related to autonomous (non-human) pedestrians and map information. This variable selection process remains a manual and task-dependent effort that requires domain expertise. However, these modifications make the model learning process more tractable by reducing complexity and improving the interpretability of the resulting model.

As a result, we defined a set of seven variables, as outlined in Table 2.1. This set includes a mix of time-series data variables (robot_rotation_change, robot_pos_change, human_pos_change), single-valued continuous variables (initial_robot_rotation, total_robot_rotation), and naturally discrete variables (competence, intention).

2.1.2 Variable Preprocessing

Before beginning the CBN learning step, the data must undergo a preprocessing step to discretize the variables. This is essential for two main reasons.

First, many structure learning algorithms impose specific assumptions when handling continuous data [99]. In particular, some algorithms cannot construct CBN graphs where a continuous variable acts as a parent of a discrete one. However, such relationships are common in our CBNs. For example, a continuous variable like `xOffset` (representing the horizontal displacement of a cube) might influence a binary task success variable such as `onTop`, which indicates whether a cube was successfully stacked. To address this, all continuous random variables in $\mathbf{X}$ are discretized into intervals ($\boldsymbol{X}_{\text{int}}$).

Variable Name	**Formula**	**Interpretation & Type**
robot_rotation_change	$\left\{\begin{array}{l}\theta_0^{\text{robot\_goal}} - \theta_0^{\text{robot\_goal}}, \\ \ldots, \\ \theta_8^{\text{robot\_goal}} - \theta_0^{\text{robot\_goal}}\end{array}\right\}$	Is the robot rotating towards or away from the goal? (time-series)
total_robot_rotation	$\sum_{t=0}^{t=8} \lvert\theta_t^{\text{robot\_goal}}\rvert$	Total robot rotation over 8-second period (continuous)
initial_robot_rotation	$\theta_0^{\text{robot\_goal}}$	Initial robot-goal angle (continuous)
robot_pos_change	$\left\{\begin{array}{l}\text{dist}_0^{\text{robot\_goal}} - \text{dist}_0^{\text{robot\_goal}}, \\ \ldots, \\ \text{dist}_8^{\text{robot\_goal}} - \text{dist}_0^{\text{robot\_goal}}\end{array}\right\}$	Is the robot moving towards or away from the goal? (time-series)
competence	$\{0,1\}_{t=8}$	Perceived competence at the end of an observation (categorical)
intention	$\{0,1\}_{t=8}$	Perceived intention at the end of an observation (categorical)
human_pos_change	$\left\{\begin{array}{l}\text{dist}_0^{\text{human\_robot}} - \text{dist}_0^{\text{human\_robot}}, \\ \ldots, \\ \text{dist}_8^{\text{human\_robot}} - \text{dist}_0^{\text{human\_robot}}\end{array}\right\}$	Is the human moving towards or away from the robot? (time-series)

Table 2.1: CBN variables $\boldsymbol{X}$. $\theta_t^{\text{robot\_goal}}$ denotes the angle between robot and goal and dist_t denotes the Euclidean distance at time t. Source: Adapted from Paper G.

Algorithm 1 GetVariableIntervals function

```
1:  function GetVariableIntervals($\mathcal{D}$, $\boldsymbol{\Lambda}$)
2:      $\boldsymbol{X}_{\text{int}} \leftarrow []$
3:      for all $i \in |\boldsymbol{X}|$ do
4:          if $\mathcal{D}_i$ is not categorical then
5:              if $\mathcal{D}_i$ is time-series then
6:                  $\boldsymbol{X}_{\text{int}_i} \leftarrow \text{Add}(\boldsymbol{X}_{\text{int}_i}, \text{Cluster}(\mathcal{D}_i, \Lambda_i))$
7:              else
8:                  $\boldsymbol{X}_{\text{int}_i} \leftarrow \text{Add}(\boldsymbol{X}_{\text{int}_i}, \text{Discretize}(\mathcal{D}_i, \Lambda_i))$
9:          else
10:             $\boldsymbol{X}_{\text{int}_i} \leftarrow \text{Add}(\boldsymbol{X}_{\text{int}_i}, \mathit{Val}(\mathcal{D}_i))$
11:     return $\boldsymbol{X}_{\text{int}}$
```

Second, for the chosen variable set from Example 3, our model needs to integrate both single-valued variables and time-dependent (time-series) data. To handle this complexity, we proposed a generalized discretization algorithm in Paper G, which effectively discretizes time-varying and static variables in a unified Bayesian network framework, increasing the flexibility of existing models.

The inputs to Alg. 1 include training data-set $\mathcal{D}$, which consists of K Independent and Identically Distributed (IID) samples $\boldsymbol{\xi}$. Each sample ξ is a fully observed instance of all network variables $\boldsymbol{X}$. $\mathcal{D}_i$ are all training values in $\mathcal{D}$ of the i-th variable X_i and $\boldsymbol{\Lambda}$ is a vector with the number of discretization intervals for each X_i. The output of Alg. 1 is a list of discretization intervals $\boldsymbol{X}_{\text{int}}$ for each variable in $\boldsymbol{X}$. If $\mathcal{D}_i$ represents time-series data, we apply K-means clustering, as implemented in scikit-learn [100] (Line 6 of Alg. 1). Specifically, we treat each time-series as a sequence of data points and use Euclidean distance to group similar time-series into clusters. Let the Euclidean distance between two time-series instances of the i-th variable of our dataset $y, z \in \mathcal{D}_i$ as:

$$\text{dist}(y, z) = \sqrt{\sum_{t=1}^{T} (y(t) - z(t))^2} \tag{2.3}$$

where $y(t)$ and $z(t)$ are the values of the time-series at time t, and T is the length of the time-series. The K-means algorithm then partitions all instances of each $\mathcal{D}_i$ that is a time-series into Λ_i clusters $\{\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_{\Lambda_i}\}$ such that each cluster $\boldsymbol{I}_j$ is a set that contains time-series that are similar to each other in terms of their Euclidean distance. The centroids of these clusters, $\{c_1, c_2, \ldots, c_{\Lambda_i}\}$, are used as the discretization intervals, where each centroid c_j represents the average pattern of the time-series within that cluster.

For variables $\mathcal{D}_i$ that are single-valued but continuously distributed, we perform quantile discretization [96] (Line 8 of Alg. 1). In this method, $\mathcal{D}_i$ is divided into Λ_i intervals such that each interval contains approximately the same number of data points. The intervals $\boldsymbol{I}_j$ are determined by sorting the data and splitting it into equal-sized groups based on the quantiles of the data distribution. Then, each interval contains approximately the same number of data points.

Finally, if $\mathcal{D}_i$ consists of categorical data, we directly assign the unique values found in $\mathcal{D}_i$ as the discretization intervals (Line 10 of Alg. 1).

In Papers B, C, and D, we chose the number of discretization intervals heuristically. In Paper G, we performed a hyperparameter search to identify Λ_i that performs optimally w.r.t. the task success prediction. Please refer to [101] for more automated discretization methods.

2.1.3 Learning Cause-Effect Models: Causal Discovery

Given the preprocessed dataset, we learn the Causal Bayesian Network (CBN) via the two steps of structure learning and parameter estimation.

2.1.3.1 Structure Learning

The purpose of this step is to learn the graphical representation $\mathcal{G}$ (as defined in Eq. 2.1) of the CBN.

There are different approaches to learn the graph: Constraint-based methods, such as the Grow-and-Shrink (GS) algorithm [102] and the PC[3] algorithm [96], identify conditional independencies within the data to construct a graph that reflects these relationships. The GS algorithm adopts a local approach by iteratively building a Markov blanket for each variable, adding or removing variables based on statistical tests. Conversely, the PC algorithm begins with a fully connected graph and systematically removes edges based on independence tests before trying to orient the remaining edges to form a Directed Acyclic Graph (DAG) [103]. Score-based methods evaluate

[3]Named after its authors Peter Spirtes and Clark Glymour

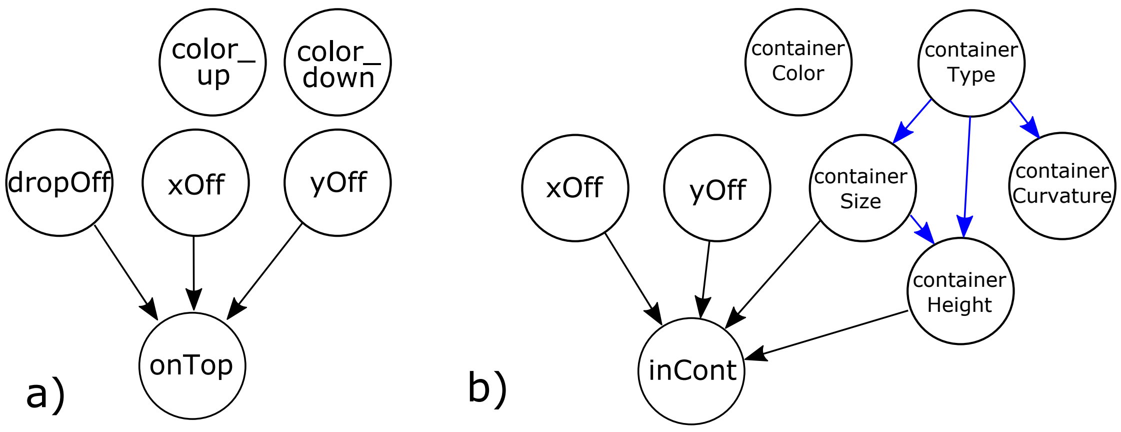


Figure 2.6: Obtained BN for a) the Cube-Stacking and b) the Sphere-Dropping task. The blue edges have been detected by the structure learning algorithm but had to be directed manually. Source: Paper B. © 2022 IEEE (RAL).

candidate graphs by using a scoring function S that measures how well a graph fits the observed data. The goal is to find the graph that maximizes this score, using measures like the Bayesian Information Criterion (BIC) or the Bayesian Dirichlet equivalence (BDe) [103]. Another approach to structure learning involves continuous optimization-based methods, which reformulate the problem as a differentiable optimization task. For example, the NO TEARS (Non-combinatoric Optimization via Trace Exponential Augmented lagRangian Structure) learning algorithm introduces a differentiable acyclic constraint, enabling gradient-based optimization to search for the graph structure [104]. For a comprehensive discussion of these methodologies, their strengths, and their limitations, see [103].

Note that learning plausible assumptions about causal relations is one of the biggest challenges in the whole process of causal inference [105]. For example, in some cases, it is challenging to determine the direction of causal relations purely from the joint distribution of the observational data without additional interventional experiments, additional domain knowledge, or certain assumptions about the data distribution [106]. Structure learning is an active field of research [105], and we will use the learned structure to generate causal-based explanations of failures.

For Examples 1 and 2, we obtained causal graphs as visualized in Fig. 2.6. However for Example 3 from Paper G, we opted to manually design the causal model, due to the limited data size. The proposed Causal Bayesian Network

(CBN) structure $\mathcal{G}$ for the Robot-Following task is illustrated in Fig. 2.7. The set of variables $\boldsymbol{X}$ = {robot_rotation_change, total_robot_rotation, initial_robot_rotation, robot_pos_change, competence, intention, human_pos_change} is defined as in Table 2.1 and discretized according to the procedure outlined in Alg. 1.

The variables robot_rotation_change, total_robot_rotation, initial_robot_rotation capture different aspects of the robot's orientation. The key variable used to describe the robot's rotational trajectory is robot_rotation_change. However, this alone is insufficient for estimating the competence reliably. Since each rotational trajectory is measured relative to the starting point of an 8-second time interval, robot_rotation_change does not capture the initial rotational difference toward the goal. However, the robot's initial orientation is crucial in determining whether its rotational behavior leads to competent robot behavior. For example, if the robot is already facing the goal, it should maintain its orientation, whereas if it starts misaligned, it should rotate toward the goal. To account for this, we introduce initial_robot_rotation, which captures the robot's initial orientation at the beginning of each interval. Additionally, clustering methods struggle to distinguish between trajectories that maintain their direction and those that rotate around their own axis as, on average, these trajectories appear to keep the same rotational difference toward the goal. This distinction is important because a robot spinning in place may appear less competent than one maintaining its orientation toward the goal. We, therefore, introduce the total_robot_rotation variable, which captures the total accumulated rotation over the 8-second interval. All three rotation variables, robot_rotation_change, initial_robot_rotation, and total_robot_rotation, contribute to predicting perceived competence. These variables are also interdependent: the robot's initial orientation influences its rotational trajectory, as robots tend to rotate toward the goal when misaligned and maintain their orientation when already facing it. Furthermore, the rotational trajectory directly affects the total accumulated rotation, as each change contributes to the overall rotation over time.

Another key variable that significantly impacts the robot's perceived competence and intention is robot_pos_change, which indicates the robot's movement relative to the goal (whether it is moving toward or away from it). The robot's movement is partially influenced by its rotational behavior, as all three

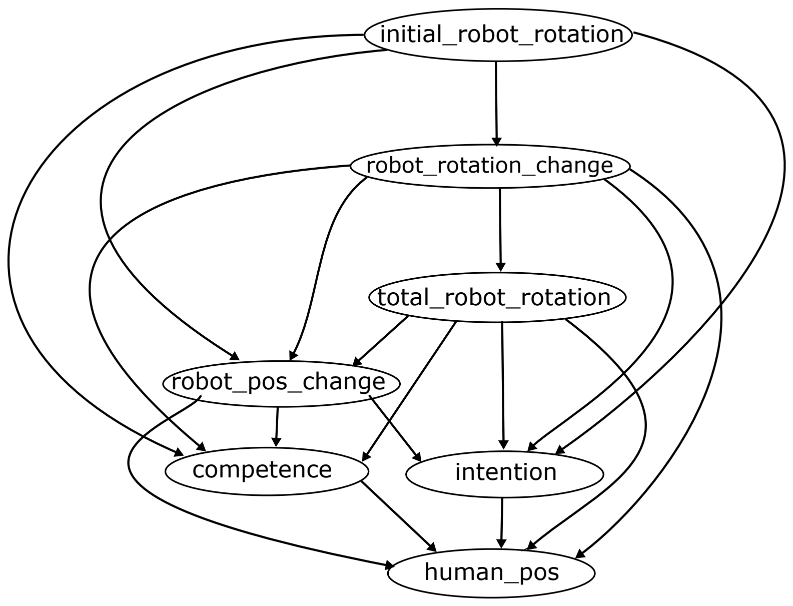


Figure 2.7: Proposed CBN graph for the Robot-Follower task. Source: Paper G.

rotational variables affect its ability to advance. For example, if the robot is rotating in place, it cannot move toward the goal. Conversely, if it is oriented toward the goal, it can reduce the distance based on its forward movement speed.

Finally, human movement (human_pos_change) is modeled as a direct consequence of perceived competence, perceived intention, and the robot's behavior (robot_pos_change, robot_rotation_change, total_robot_rotation). We assume that initial robot rotation only affects the human movement indirectly, via its influence on the robot's rotational trajectory.

Since CBN learning is divided into structure learning and parameter estimation, the structure learning phase allows for the integration of expert input. For example, in Paper G, we handcrafted the causal model by explicitly defining the relationships between variables, and then learned the CBN parameters using the available data.

2.1.3.2 Parameter Learning

Once $\mathcal{G}$ is obtained, the local probability distributions $P(X_i|\mathbf{Pa}_{X_i})$ (Eq. 2.2) can be represented using parameters $\boldsymbol{\theta}$, with their interpretation depending on the probability distribution type of X_i. In our case, we model each X_i as a multinomial random variable due to variable discretization[4].. Thus, $\boldsymbol{\theta}$ takes the form of a table. In this table, each $\theta_{x|\boldsymbol{u}} \in \boldsymbol{\theta}$ represents the probability that $\theta_{x|\boldsymbol{u}} = P(X_i = x|\mathbf{Pa}_{X_i} = \boldsymbol{u})$ for every $x \in \mathit{Val}(X_i)$ and $\boldsymbol{u} \in \mathit{Val}(\mathbf{Pa}_{X_i})$, where $\mathit{Val}(X_i)$ signifies all possible values of X_i, and $\mathit{Val}(\mathbf{Pa}_{X_i})$ encompasses all potential value combinations of $\mathbf{Pa}_{X_i}$. The cardinality of $|\boldsymbol{\theta}| = |x||\boldsymbol{u}|$ increases with the number of discretization intervals for each $\boldsymbol{X}$ and the number of parent variables, directly affecting the number of data samples needed to estimate the distribution of θ. For instance, if we double the number of parent variables and use 5 intervals for each variable, the number of intervals grows from $|\boldsymbol{u}| = 5 \times 5$ to $|\boldsymbol{u}| = 5 \times 5 \times 5 \times 5$ quadratically, leading to a quadratic increase in the required data samples.

There are two main parameter estimation approaches: *Maximum Likelihood Estimation* (MLE), and *Bayesian estimation* [107]. For both, we assume that we have obtained a training data set $\mathcal{D}$, which consists of K IID samples $\boldsymbol{\xi}$. Each sample ξ is a fully observed instance of all network variables $\boldsymbol{X}$.

Maximum Likelihood Estimation aims to find the optimal parameters $\hat{\boldsymbol{\theta}} \in \Theta$ of the parametric model $P(\mathcal{D} : \boldsymbol{\theta})$ that represent the unknown CBN distribution $P(X_1, X_2, ..., X_N)$. Due to the global likelihood decomposition [108], the optimal CBN parameters can be obtained for each $\hat{\theta}_{x|\boldsymbol{u}}$ individually by maximizing the likelihood function

$$L_{X_i}(\theta_{X_i|\mathbf{Pa}_{X_i}} : \mathcal{D}) = \prod_{\boldsymbol{u} \in \mathit{Val}(\mathbf{Pa}_{X_i})} \left[\prod_{x \in \mathit{Val}(X_i)} \theta_{x|\boldsymbol{u}}^{M[\boldsymbol{u},x]} \right], \tag{2.4}$$

where $M[\boldsymbol{u}, x]$ is the number of times $X_i[k] = x$ and its parent variables $\boldsymbol{U}[k] = \boldsymbol{u}$ for $k \in K$ in $\boldsymbol{\xi}$. Then, L_{X_i} is computed by maximizing each of the $\theta_{x|\boldsymbol{u}}^{M[\boldsymbol{u},x]}$, individually:

$$\hat{\theta}_{x|\boldsymbol{u}} = \frac{M[\boldsymbol{u}, x]}{M[\boldsymbol{u}]}. \tag{2.5}$$

Bayesian Parameter Estimation treats each θ as a random variable and, unlike MLE, encodes prior information about θ, leading to the formulation of the posterior distribution

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})}. \tag{2.6}$$

[4]This subsection was adapted from Paper D.

With the assumption that parameter priors for each of the local conditional probability distributions of Eq. 2.2 are independent, we can show that the posterior distribution of the CBN parameters $\boldsymbol{\theta}$ can be decomposed into

$$P(\boldsymbol{\theta}|\mathcal{D}) = \prod_{i} \prod_{\boldsymbol{u} \in \mathit{Val}(\mathbf{Pa}_{X_i})} P(\boldsymbol{\theta}_{X_i|\boldsymbol{u}}|\mathcal{D}). \tag{2.7}$$

Furthermore, in the case of a single multinomial variable X_i with Λ_i possible outcomes, we can derive a predictive model for observing a new data sample $\xi[K+1]$ where $X_i[K+1] = x_j$ with $j \in \Lambda_i$, given that the parent variables $\boldsymbol{U}[K+1] = \boldsymbol{u}$ and the previously observed data $\mathcal{D} = \{\xi_1, \xi_2, ..., \xi_K\}$, as

$$P(X[K+1] = x_j|\boldsymbol{U}[K+1] = \boldsymbol{u}, \mathcal{D}) = \frac{\alpha_{x_j|\boldsymbol{u}} + M[x_j, \boldsymbol{u}]}{\alpha_{\boldsymbol{u}} + M[\boldsymbol{u}]}, \tag{2.8}$$

by modeling the prior distribution $P(\boldsymbol{\theta}_{X_i})$ as a Dirichlet distribution with hyperparameters $\alpha_{x_1|\boldsymbol{u}}, \alpha_{x_2|\boldsymbol{u}}, ..., \alpha_{x_J|\boldsymbol{u}}$. The advantage of using the Dirichlet distribution is that the hyperparameters can be interpreted as an imaginary count [108] of successful task executions in a prior data set $\mathcal{D}'$. More formally, we can set $\alpha_{x_j|\boldsymbol{u}} = \alpha[x_j, \boldsymbol{u}]$ where $\alpha[x_j, \boldsymbol{u}]$ is the number of times $X_i = x_j$ and $\mathit{Val}(\mathbf{Pa}_{X_i}) = \boldsymbol{u}$ in $\mathcal{D}'$. Analogously, $\alpha_{\boldsymbol{u}} = \alpha[\boldsymbol{u}]$ where $\alpha[\boldsymbol{u}]$ is the number of times that $\mathit{Val}(\mathbf{Pa}_{X_i}) = \boldsymbol{u}$ in $\mathcal{D}'$.

2.1.4 Our Approach for Generating CBN Parameter Priors

In real-world environments, robots frequently encounter changes or variations that require them to adapt dynamically. These changes can manifest as entirely new tasks or as familiar tasks applied to new objects. For instance, a robot may transition from stacking one cube to stacking two cubes, or it might need to perform a task like dropping a sphere into a glass instead of a bowl. Adapting to such variations is crucial, especially when the robot must handle these new scenarios in a zero-shot fashion [59].

To enable robots generalize from one task to another, we explore the use of CBN structures to detect similarities across different tasks or scenarios in Paper D. For example, as illustrated in Fig. 2.8, the two stacking tasks share variables such as `xOff1` and `xOff2`, representing the stacking offset of the first and the second cube that is stacked. Similarly, the Sphere-Dropping tasks include `xOff`, which measures the offset between the sphere and its container

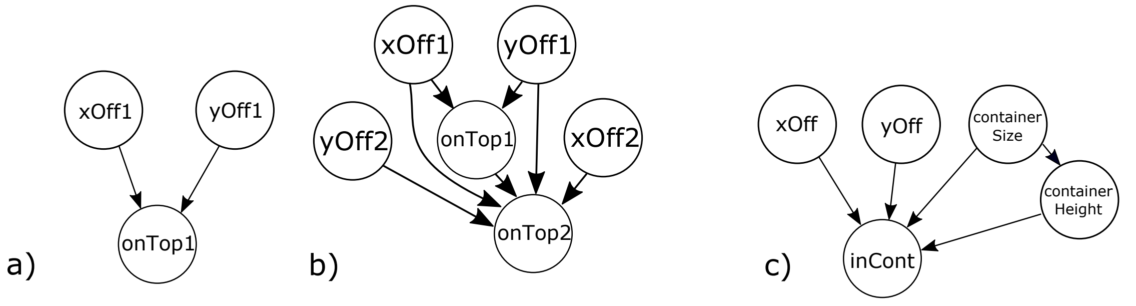


Figure 2.8: Shows semantic similarity between different tasks a) Cube-Stacking (single cube), b) Cube-Stacking (two cubes), c) Sphere-Dropping. Source: Adapted from Paper D. © 2024 IEEE (RAL).

and is semantically related to the offsets in cube-stacking tasks. By identifying such overlaps in causal structures, the robot can transfer knowledge from prior tasks to new ones, enabling more effective adaptation and robust performance in dynamic environments.

Our approach addresses the challenge of deriving the imaginary prior count (hyperparameters from the Dirichlet distribution) $\alpha[x_j, \boldsymbol{u}]$ and $\alpha[\boldsymbol{u}]$ in Eq. 2.8 from a prior CBN task model, which is already known or can be obtained with significantly less effort (fewer samples or less time) due to a smaller cardinality $|\boldsymbol{\theta}|$. To formalize the generation of parameter priors, we define a prior task T' as a causal BN structure $\mathcal{G}'$, with a set of network variables $\boldsymbol{X}'$, a set of parameters $\boldsymbol{\theta}'$ and a dataset $\mathcal{D}'$. Furthermore, we define the target task as T, with its own associated CBN structure $\mathcal{G}$, network variables $\boldsymbol{X}$, parameters $\boldsymbol{\theta}$ and dataset $\mathcal{D}$, where generally $\mathcal{G}' \neq \mathcal{G}$ and $\mathcal{D}' \neq \mathcal{D}$ but $\boldsymbol{X}' \cap \boldsymbol{X} \neq \varnothing$. We denote $X' \in \boldsymbol{X}'$ as the variable whose parameters $\boldsymbol{\theta}'_{X'}$ we want to transfer to $X \in \boldsymbol{X}$. In this paper, we then propose to use T' as prior information for learning the BN parameters of T.

Challenge 1: Large number of pseudo counts. A naive parameter transfer based on the notion of pseudo counts in Eq. 2.8 is problematic because $\alpha[\boldsymbol{u}]$ might be much larger than $M[\boldsymbol{u}]$, due to the easier availability of prior data. Thus the new data $\mathcal{D}$ would barely play a role. We therefore reformulate Eq. 2.8 as

$$\frac{\alpha_{x_j|\boldsymbol{u}} + M[x_j, \boldsymbol{u}]}{\alpha_{\boldsymbol{u}} + M[\boldsymbol{u}]} = \frac{w_{\text{prior}}\theta'_{x_j|\boldsymbol{u}} + M[x_j, \boldsymbol{u}]}{w_{\text{prior}} + M[\boldsymbol{u}]}, \tag{2.9}$$

where $\theta'_{x_j|\boldsymbol{u}}$ either is the Maximum Likelihood estimate $\frac{M'[x_j,\boldsymbol{u}]}{M'[\boldsymbol{u}]}$ in the prior data $\mathcal{D}'$ or the Baysian estimate $P(X[K+1] = x_j|\boldsymbol{U}[K+1] = \boldsymbol{u}, \mathcal{D}')$. Eq. 2.9 allows us to deliberately control the importance of the prior. By choosing $w_{\text{prior}} = M[\boldsymbol{u}]$, we would weigh the prior and the new data equally. For example, with $\theta'_{x_j|\boldsymbol{u}} = \frac{1}{2}$, $M[x_j, \boldsymbol{u}] = 6$ and $M[\boldsymbol{u}] = 24$, we would obtain

$$\frac{24 * 0.5 + 6}{24 + 24} = \frac{18}{48} = \frac{3}{8} = 0.5 \times (\frac{1}{2} + \frac{6}{24}) = 0.5 \times (\frac{1}{2} + \frac{1}{4}).$$

In other words, w_{prior} allows us to express Eq. 2.8 as a weighted sum between the prior ratio, e.g.,

$$\frac{\alpha_{x_j|\boldsymbol{u}}}{\alpha_{\boldsymbol{u}}} = \frac{1}{2}$$

and the Maximum Likelihood estimate of $\theta_{x_j|\boldsymbol{u}}$ in $\mathcal{D}$, e.g.,

$$\frac{M[x_j, \boldsymbol{u}]}{M[\boldsymbol{u}]} = \frac{6}{24} = \frac{1}{4}.$$

Challenge 2: Difference in number of parameters. So far in Eq. 2.9 we assume that we can find a corresponding prior parameter $\theta'_{x_j|\boldsymbol{u}}$ for each $M[x_j, \boldsymbol{u}]$. However, if either the number of parent variables is not consistent

$$|\mathbf{Pa}_{X'}| \neq |\mathbf{Pa}_X|,$$

where $|.|$ denotes the cardinality, or the number of values is different

$$|\mathit{Val}(\mathbf{Pa}_X)| \neq |\mathit{Val}(\mathbf{Pa}_{X'})|,$$

this could lead to a condition where $|\boldsymbol{u}'| \neq |\boldsymbol{u}|$, which means we won't find a $\theta'_{x_j|\boldsymbol{u}}$ for every $M[x_j, \boldsymbol{u}]$. To illustrate this problem let's consider Ex. 4.

Example 4. *Assume the robot already knows how to stack a single cube, which means it has learned the distribution $P(\texttt{onTop1}|\texttt{xOff1}, \texttt{yOff1})$ from prior data $\mathcal{D}'$. In a different situation, the robot has to stack two cubes and thus needs to predict*

$$P(\texttt{onTop2} = 1|\texttt{xOff1} = 0.5cm, \texttt{yOff1} = 1cm,$$
$$\texttt{xOff2} = 1.5cm, \texttt{yOff2} = 2cm).$$

That means we would like to transfer the distribution of $\texttt{onTop1}$ *(X') to* $\texttt{onTop2}$ *(X). The issue here is that there is no full correspondence between* $\boldsymbol{u}$ *in* $\mathcal{D}$ *and in* $\mathcal{D}'$ *because* $\texttt{xOff2}$ *and* $\texttt{yOff2}$ *do not exist in* $\mathcal{D}'$*. Therefore, we cannot query* $P(\texttt{onTop1}|\texttt{xOff}, \texttt{yOff})$ *for the exact same stacking configuration. However, we can query* $P(\texttt{onTop1}|\texttt{xOff}, \texttt{yOff})$ *based on the offset of the first cube* $\texttt{xOff1} = 0.5cm, \texttt{yOff1} = 1cm$*, thus we can estimate the chance of succeeding in stacking both cubes as*

$$\begin{aligned}P(\texttt{onTop2} = 1|&\texttt{xOff1} = 0.5cm, \texttt{yOff1} = 1cm,\\ &\texttt{xOff2} = 1.5cm, \texttt{yOff2} = 2cm)\\ =&P(\texttt{onTop1} = 1|\texttt{xOff} = 0.5cm, \texttt{yOff} = 1cm).\end{aligned} \tag{2.10}$$

Alternatively, we could query $P(\texttt{onTop1}|\texttt{xOff}, \texttt{yOff})$ *based on the offset of the second cube* $\texttt{xOff2} = 1.5cm, \texttt{yOff2} = 2cm$*, thus we can estimate the chance of succeeding in stacking both cubes as*

$$\begin{aligned}P(\texttt{onTop2} = 1|&\texttt{xOff1} = 0.5cm, \texttt{yOff1} = 1cm,\\ &\texttt{xOff2} = 1.5cm, \texttt{yOff2} = 2cm)\\ =&P(\texttt{onTop1} = 1|\texttt{xOff} = 1.5cm, \texttt{yOff} = 2cm).\end{aligned} \tag{2.11}$$

To formalize this idea, we proposed a generalization of Eq. 2.9 as

$$\frac{\alpha_{x_j|\boldsymbol{u}} + M[x_j, \boldsymbol{u}]}{\alpha_{\boldsymbol{u}} + M[\boldsymbol{u}]} = \frac{w_{\text{prior}} f(\theta'_{x_j|\boldsymbol{u}'}) + M[x_j, \boldsymbol{u}]}{w_{\text{prior}} + M[\boldsymbol{u}]}, \tag{2.12}$$

where $f(\theta'_{x_j|\boldsymbol{u}'}) = \theta'_{x_j|\boldsymbol{u}}$ and $\boldsymbol{u}'$ is defined based on one of the following three proposed prior generation and transfer approaches:

1) Direct-Replacement:

$$\boldsymbol{u}' = \boldsymbol{u}_{\mathbf{Pa}_{X'} \cap \mathbf{Pa}_X} \in \mathit{Val}(\mathbf{Pa}_{X'} \cap \mathbf{Pa}_X). \tag{2.13}$$

That means the discretization intervals of the non-empty variable intersection set $\mathbf{Pa}_{X'} \cap \mathbf{Pa}_X \neq \varnothing$ in $\boldsymbol{u}'$ are set to the discretization intervals of the similar subset in $\boldsymbol{u}$. In Ex. 4, the set of intersecting variables between `onTop1` and `onTop2` would be `xOff1`, `yOff1`. Thus to obtain the prior probability parameters $f(\theta'_{x_j|\boldsymbol{u}'})$, the direct-replacement strategy would query the prior distribution model based on the respective interval values for `xOff1`, `yOff1` as exemplified in Eq. 2.10.

2) Cross-Replacement:

$$\boldsymbol{u}' = \boldsymbol{u}_{rel(\mathbf{Pa}_{X'},\mathbf{Pa}_X)} \in \mathit{Val}(rel(\mathbf{Pa}_{X'}, \mathbf{Pa}_X)), \tag{2.14}$$

where $rel()$ returns a subset of variables $X'_{\text{rel}} \subset \mathbf{Pa}_{X'}$ that have a related variable in $\mathbf{Pa}_X$. We denote variables as related if they are semantically similar but not the same. In Ex. 4, the variable $\texttt{xOff1} \in \mathbf{Pa}_{X'}$ is related to $\texttt{xOff2}$ which is one of the parent variables $\texttt{xOff2} \in \mathbf{Pa}_X$ of the new task T. Thus to obtain the prior probability parameters $f(\theta'_{x_j|\boldsymbol{u}'})$, the direct-replacement strategy would query the prior distribution model based on the respective interval values for $\texttt{xOff2}, \texttt{yOff2}$ as exemplified in Eq. 2.11. The automatic definition of related variables is left as future work. Possible solutions could involve partially matching variable names and matching the variables based on their distributions or value ranges.

3) Minimal-Replacement:

$$\boldsymbol{u}' = \begin{cases} \boldsymbol{u}'_{\text{direct}} & \text{if } \theta'_{x_j|\boldsymbol{u}'_{\text{direct}}} < \theta'_{x_j|\boldsymbol{u}'_{\text{cross}}} \\ \boldsymbol{u}'_{\text{cross}} & \text{else} \end{cases}, \tag{2.15}$$

where $\boldsymbol{u}'_{\text{direct}} = \boldsymbol{u}_{\mathbf{Pa}_{X'} \cap \mathbf{Pa}_X}$ and $\boldsymbol{u}'_{\text{cross}} = \boldsymbol{u}_{rel(\mathbf{Pa}_{X'},\mathbf{Pa}_X)}$. This strategy regards both Direct- and Cross-Replacement parameter candidates and keeps the smaller probability parameter value $f(\theta'_{x_j|\boldsymbol{u}'})$ as prior for Eq. 2.12.

To validate our approach, we conducted several transfer experiments, including sim-to-real transfer (for the Cube-Stacking task), transferring parameters to more complex tasks with a larger number of parameters (e.g., stacking one cube to two cubes), and even transferring parameters between entirely different tasks (Cube-Stacking to Sphere-Dropping). For parameter estimation, we employed a Bayesian Estimation approach that integrates priors derived from three proposed transfer strategies. These estimates were compared to Maximum Likelihood Estimation (MLE), which relies solely on data from the new task.

Our results showed that MLE estimates generally converge faster toward the ground truth compared to Bayesian Estimates, as measured by the mean parameter difference. This result is expected since priors in Bayesian Estimation induce an initial error. However, this error diminishes with more data, and depends on the weight assigned to the priors (ω_{prior}). Intuitively, a higher emphasis on the priors (i.e., a larger ω_{prior}) slows convergence but provides stability in data-scarce scenarios.

However, we are particularly interested in cases where little to no data is available for the new task. In such situations ($\mathcal{D} = 0$) we assigned a default value of 0.5 to all parameters for the Maximum Likelihood Estimation (MLE), whereas Bayesian Estimation leverages the priors we obtained. These priors were subsequently applied in a decision-making task, specifically as input to our failure prevention method (described in detail in Sec. 2.3). By incorporating priors, we observed a significant reduction in execution failures: a 50% reduction in failures for stacking two cubes, and a 20% reduction when transferring knowledge between the Cube-Stacking and Sphere-Dropping tasks. This demonstrates that utilizing priors allows robots to better handle variations or novel situations, improving their decision-making abilities in new environments.

2.2 Our Approach to Generate Explanations for Robot Task Execution Failures

With the obtained causal model, the robot is able to make predictions about task success. However, without an additional layer, it cannot provide explanations for why failures occur. To address this, Paper B introduces a method that identifies contrastive and selective explanations [17] for task failures (RQ3), as outlined in Algorithm 2. This method enables the robot to not only predict task outcomes but also to pinpoint and explain the specific factors that contribute to failures, enhancing its ability to understand and adapt in dynamic environments[5].

In (L-2 Alg. 2)), a matrix is generated which defines transitions for every single-variable change for all possible variable parametrizations. For example, if we have two variables X and Y, each with five intervals ($X = \{x_1, x_2, x_3, x_4, x_5\}$ and $Y = \{y_1, y_2, y_3, y_4, y_5\}$), the possible valid transitions from $node = (X = x_1, Y = y_4)$ would be $child_1 = (X = x_2, Y = y_4)$, or $child_2 = (X = x_1, Y = y_5)$, or $child_3 = (X = x_1, Y = y_3)$. In contrast, $child_4 = (X = x_2, X_2 = x_3)$ would not be a valid transition from *node* because it involves changes to both variables simultaneously. Furthermore, transitions

[5]This subsection was adapted from Paper B.

Algorithm 2 Failure Explanation

Input: failure variable parameterization x_{failure}, graphical model $\mathcal{G}$, structural equations $P(X_i|\mathbf{Pa}_{X_i})$, discretization intervals of all model variables $\boldsymbol{X}_{\text{int}}$, success threshold ϵ, goal parametrizations $\boldsymbol{X}_{\text{goal}}$

Output: solution variable parameterization $x_{\text{solution}_{\text{int}}}$, solution success probability prediction p_{solution}

```
 1: $x_{\text{current}_{\text{int}}} \leftarrow \textsc{getIntervalFromValues}(x_{\text{failure}}, \boldsymbol{X}_{\text{int}})$
 2: $P \leftarrow \textsc{generateTransitionMatrix}(\boldsymbol{X}_{\text{int}})$
 3: $q \leftarrow [x_{\text{current}_{\text{int}}}]$
 4: $v \leftarrow []$
 5: while $q \neq \varnothing$ do
 6:     $node \leftarrow \textsc{Pop}(q)$
 7:     $v \leftarrow \textsc{Append}(v, node)$
 8:     for all transition $t \in P(node)$ do
 9:         $child \leftarrow \textsc{Child}(P, node)$
10:         if $child \notin q, v$ then
11:             $p_{\text{solution}} = P(child \in \boldsymbol{X}_{\text{goal}}|\mathbf{Pa}_{child})$
12:             if $p_{\text{solution}} > \epsilon$ then
13:                 $x_{\text{solution}_{\text{int}}} \leftarrow child$
14:                 $\textsc{return}(p_{\text{solution}}, x_{\text{solution}_{\text{int}}})$
15:             $q \leftarrow \textsc{Append}(q, x_{\text{current}_{\text{int}}})$
```

are restricted to neighboring intervals only, for example, from x_1 to x_2, but not directly from x_1 to x_3. Whether this ordering has any meaning depends on the concept represented by the variable. For instance, if we consider a continuous variable like `xOff` from Example 1, the transition from x_1 to x_2 represents two neighboring offset intervals, making x_1 closer to x_2 than to x_3. However, for a variable such as `cubeColor`, there may not necessarily be a physical notion of distance between different parameterizations.

Lines 5-15 (Alg. 2) describe the adapted BFS procedure, which searches for the closest variable parametrization that fulfills the goal criteria of $P(child \in \boldsymbol{X}_{\text{goal}}|\mathbf{Pa}_{child}) > \epsilon$, where ϵ is the success threshold, which can be heuristically set. The concept of our proposed method is to generate contrastive explanations that compare the current variable parametrization associated with the execution failure $x_{\text{current}_{\text{int}}}$ with the closest parametrization that would have allowed for a successful task execution $x_{\text{solution}_{\text{int}}}$. Consider Figure 2.9 for a visualization of the explanation generation, exemplified on two previously introduced variables $X = \{x_1, x_2, x_3, x_4, x_5\}$ and $Y = \{y_1, y_2, y_3, y_4, y_5\}$, which are both causally influencing the variable X_{out}. Furthermore, it is known that $x_{\text{out}} = 1 \in \boldsymbol{X}_{\text{goal}}$. In this example, the resulting explanation would be that the task failed because $X = x_1$ instead of $X = x_2$ and $Y = y_4$ instead of $Y = y_3$.

Table 2.2 provides several examples that showcase how our method finds explanations for robot failures.

Our method generates contrastive explanations by comparing the current variable values with the closest solution that results in the minimal number of interval changes. This approach adheres to Occam's razor principle [17]. The advantage of using an uninformed Breadth-First Search (BFS) approach is that it ensures this principle is always applied without requiring any human domain knowledge. Additionally, the explanations are selective: among the subset of causally relevant variables, only those that undergo changes are included, as determined by the BFS procedure.

We compare our method of finding explanations of robot task failures with the two closely related methods of Context-Based History (CB-H) [25] explanations, and the ranked Semantic Scene Graph method (SSG-R) [109], based on the criteria that are summarized in Table 2.3.

For CB-H all failures and their causes need to be manually defined in the form of Fault Trees. In SSG-R failures are not modeled, but explained in

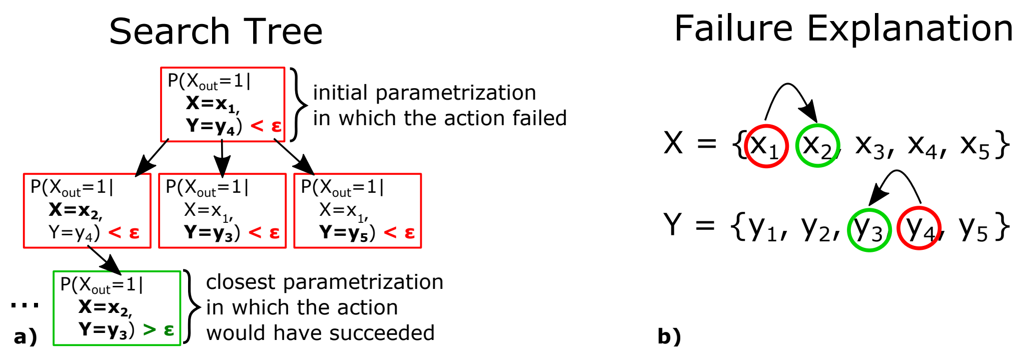

Search Tree
P(X_out=1| X=x1, Y=y4) < ε
initial parametrization in which the action failed
P(X_out=1| X=x2, Y=y4) < ε
P(X_out=1| X=x1, Y=y3) < ε
P(X_out=1| X=x1, Y=y5) < ε
P(X_out=1| X=x2, Y=y3) > ε
closest parametrization in which the action would have succeeded
...
a)
Failure Explanation
X = {x1, x2, x3, x4, x5}
Y = {y1, y2, y3, y4, y5}
b)

Figure 2.9: Exemplifies how contrastive explanations are generated from the BFS search tree with two variables $X = \{x_1, x_2, x_3, x_4, x_5\}$ and $Y = \{y_1, y_2, y_3, y_4, y_5\}$. The search procedure starts (see a-Search Tree - top box) with the initial parametrization in which the robot task execution has failed $P(X_{\text{Out}}|X = x_1, Y = y_4) < \epsilon$, where X_{Out} is exemplary for the outcome variable that is used to measure the success for the task. The search proceeds in the second row by exploring all single-variable interval changes as determined by the previously obtained transition matrix (L-2 Alg. 2). In this example a single interval change is not sufficient and, given the causal model, would also have led to unsuccessful task executions $P(X_{\text{out}} = 1|...) < \epsilon$ (indicated by the red boxes). The search terminates, once it has found variable intervals which, based on the causal model, have a high success chance greater than ϵ (see green box in third row). Because we are using BFS, we are guaranteed to find the solution with the least number of interval changes. Thus, the robot failed because $X = x_1$ instead of $X = x_2$ and $Y = y_4$ instead of $Y = y_3$ (see b-Failure Explanation). Source: Adapted from Paper B. © 2022 IEEE (RAL).

Input	Input Interval	Current Success Prob.	Closest Solution Interval	Expected Success Prob.
Cube Stacking - Example 1:				

xOff = 1.5
yOff = 0.0
dropOff = 5.0
	x_4: [1.2, 1.8]
	y_3: [-0.6, 0.6]
	$\boldsymbol{z_4}$**: [4.5, 5.5]**
		0.58	x_4: [1.2, 1.8]
			y_3: [-0.6, 0.6]
			$\boldsymbol{z_3}$**: [3.2, 4.5]**
				0.91

Explanation: The upper cube was stacked too high.				
Cube Stacking - Example 2:				

xOff = -1.5
yOff = -1.5
dropOff = 2.0
	$\boldsymbol{x_1}$**: [-1.8, -1.2]**
	$\boldsymbol{y_1}$**: [-1.8, -1.2]**
	z_2: [1.8, 2.2]
		0.017	$\boldsymbol{x_2}$: [-1.2, -0.6]
			$\boldsymbol{y_2}$: [-1.2, -0.6]
			z_2: [1.8, 2.2]
				1.0

Explanation: The upper cube was stacked too far to the left and too far to the back of the lower cube.				
Sphere Dropping - Example 1:				

xOff = 5.9
yOff = 5.9
ContSize = 21
ContHeight = 9
	$\boldsymbol{x_5}$**: [5.4, 6.4]**
	y_5: [5.4, 6.4]
	s_4: [17, 22]
	h_3: [6.3, 9.5]
		0.727	$\boldsymbol{x_4}$**: [4.5, 5.4]**
			y_5: [5.4, 6.4]
			s_4: [17, 22]
			h_3: [6.3, 9.5]
				0.98

Explanation: The sphere was dropped too far to the right.				
Sphere Dropping - Example 2:				

xOff = -3.0
yOff = -3.0
ContSize = 15
ContHeight = 1.7
	x_2: [-3.6, -1.2]
	y_2: [-3.6, -1.2]
	$\boldsymbol{s_3}$**: [14, 17]**
	h_1: [1.5, 2.9]
		0.58	x_2: [-3.6, -1.2]
			y_2: [-3.6, -1.2]
			$\boldsymbol{s_4}$**: [17, 22]**
			h_1: [1.5, 2.9]
				0.933

Explanation: The container (plate) was too small.				

Table 2.2: Examples of failure explanations for Cube Stacking and Sphere Dropping tasks. Intervals (in cm) subject to changes in the closest solution are highlighted in bold. More examples can be found in Paper B. Source: Paper B. © 2022 IEEE (RAL).

form of a list of spatial relations (like *close to* or *occluded*) and object features (like *fragile* or *heavy*), automatically detected through the semantic scene graph model MOTIFNET [110]. We explain failures via contrastive variable parametrizations. Due to these differences in failure representation, all three methods have different requirements during the learning phase. For learning the encoder-decoder network that generates language failure explanations for CB-H, simulations must be annotated with the respective failure cause. In [25], 2100 annotated time-steps were used to train for six different failure causes. However, the number of required samples will drastically increase for the two discussed examples of cube stacking and sphere dropping due to the increased number of failure possibilities. Additionally, samples are more expensive than in our method since it is required to label the failure cause instead of a simple binary action success label.

In SSG-R, pairwise ranking distinguishes between relevant and irrelevant relations. Pairwise relation preferences must be provided via domain knowledge of the failure scenario and which are more expensive than the automatically retrievable binary action success labels from our method. Another difficulty in terms of applicability to the presented scenarios of cube stacking and sphere dropping provide the continuous variables (e.g., `contSize` or `xOff`), which are discretized into more than two categories (as opposed to binary object relations). For these variables, MOTFNET is not applicable. While, in principle, a range of variables was detected to influence the action outcome causally, it is due to a specific variable parametrization that they lead to the action failure. Our method automatically discerns between relevant and irrelevant relations. Last but not least, neither CB-H nor SSG-R learn an action success model, which can be useful for other tasks beyond failure explanation, e.g., failure prediction and prevention.

2.3 Our Approach to Predict and Prevent Future Task Failures

Papers C and G extend the contrastive Breadth-First-Search approach from Paper B, to utilize the CBN to predict and prevent future failures proactively (RQ2). When the probability of a failure is high given the current state, our method identifies an alternative execution state expected to lead to a successful action, allowing the agent to both prevent failures and provide

	Method	**Output**	**Detect. of caus. relevant variables**	**Learning Prerequisites**	**Task Succ. Predict.**
CB-H [25]	Fault trees + encoder-decoder network	language model (spoken failure explanation)	no differentiation between causally relevant and irrelevant variables	failure-cause annotated simulations	no
SSG-R [109]	MOTIFNET [110] + pairwise ranking	list of relevant spatial/object relations	informally, through pairwise ranking	relationship ranking labels	no
ours	causal BNs + contrastive BFS	contrastive failure explanation	formally, through BN structure learning	task simulations (incl. action outcome)	MLE (or similar like Bayesian est.)

Table 2.3: Comparison of our explanation generation pipeline with other approaches. Source: Paper B. © 2022 IEEE (RAL).

explanations for its corrective actions[6].

Predicting and preventing errors becomes particularly challenging when the consequences of an action are not immediately apparent but manifest in future actions [111]. These cases, which we term *timely shifted action errors*, require models to account for the history of preceding actions. For instance, consider the task of constructing a tower with four cubes. If the second cube is not stacked perfectly centered on the first cube, this specific action might still be deemed successful. However, the misalignment compromises the overall stability of the tower, potentially leading to failure after subsequent stacking actions. In such scenarios, the system must incorporate past actions into its reasoning to anticipate and mitigate cascading errors.

Alg. 3 describes the proposed failure prevention approach. In particular, we first retrieve the discretization intervals for the current variable parametrization in (GetIntervalFromVal: L-2, Alg. 3)) and query the causal model to predict the success probability for the current state (L-3, Alg. 3)). In case the predicted probability is above a chosen threshold of ϵ, we continue with the execution based on the current parameters (L-4, Alg. 3)). If, however, the probability is below the threshold (L-5, Alg. 3)), we retrieve the closest success parametrization through Alg. 2 (L-7, Alg. 3)). Finally, we use the middle values of the corrected intervals as concrete parameters to retrieve a corrected variable parametrization x_{success} (MiddleValFromIntervals:

[6]The first part of this subsection, including the description of Alg. 3 and the results from the cube stacking failure prediction and prevention, is adapted from Paper C. The second part, covering competence prediction and the prevention of low-competence robot behavior, is adapted from Paper G.

Algorithm 3 Predict and prevent failures

```
    Input: current variable parameterization $x_{\text{current}}$, structural equations $P(X_i|\mathbf{Pa}_{X_i})$, discretization intervals of all model variables $\boldsymbol{X}_{\text{int}}$, success threshold $\epsilon$, goal parametrizations $\mathbf{X}_{goal}$
    Output: Concrete success variable parametrization $x_{\text{success}}$
1: procedure PREVENTFAILURES($x_{\text{current}}, P(X_i|\mathbf{Pa}_{X_i}), \boldsymbol{X}_{\text{int}}, \epsilon, \mathbf{X}_{goal}$)
2:     $x_{\text{solution}_{\text{int}}} \leftarrow$ GETINTERVALFROMVAL($x_{\text{current}}, \boldsymbol{X}_{\text{int}}$)
3:     $p_{\text{solution}} = P(x_{\text{solution}_{\text{int}}} \in \mathbf{X}_{goal}|\mathbf{Pa}_{x_{\text{solution}_{\text{int}}}})$
4:     $x_{\text{success}} \leftarrow x_{\text{current}}$
5:     if $p_{\text{solution}} < \epsilon$ then
6:         $x_{\text{failure}} = x_{\text{current}}$
7:         $p_{\text{solution}}, x_{\text{solution}_{\text{int}}} \leftarrow$
               GETCLOSESTSUCCINTERVALS($x_{\text{failure}}, P(X_i|\mathbf{Pa}_{X_i}),$
               $\boldsymbol{X}_{\text{int}}, \epsilon, \mathbf{X}_{goal}$)
8:         $x_{\text{success}} \leftarrow$
               MIDDLEVALFROMINTERVALS($x_{\text{solution}_{\text{int}}}, x_{\text{current}}, \boldsymbol{X}_{\text{int}}$)
9:     RETURN($x_{\text{success}}$)
```

L-8, Alg. 3)). The output parametrization x_{success} can then be used to manipulate the environment to ensure the action will succeed, e.g., by moving the robot gripper into a different position.

To validate our approach, we applied the proposed failure prevention method to the tasks of stacking a single cube and stacking three cubes, as described in Paper C. Our method successfully prevented 97% and 95% of failures, respectively. Figure 2.10 illustrates an example of a stacking execution performed on the real robot. In the first row, the first cube (green) is stacked too far to the right, leading to a failure in the third stack. In the corrected sequence, the green cube is stacked more to the left, which allows the robot to successfully stack all three cubes. This demonstrates the effectiveness of our method in preventing task failures by adjusting actions based on the robot's predictions.

In Paper G, we applied the proposed failure prevention method to the Robot-Follower task from Example 3. The goal was to predict the human follower's competence rating of the robot. In cases where the predicted competence was low, our method aimed to find an alternative robot trajectory that would lead to a higher perceived competence. We first compared the prediction performance of the causal model with a Random Forest (RF), which was identified as the best-performing prediction method in the related literature [112] (Table 2.4). Using accuracy as a metric, our model outper-

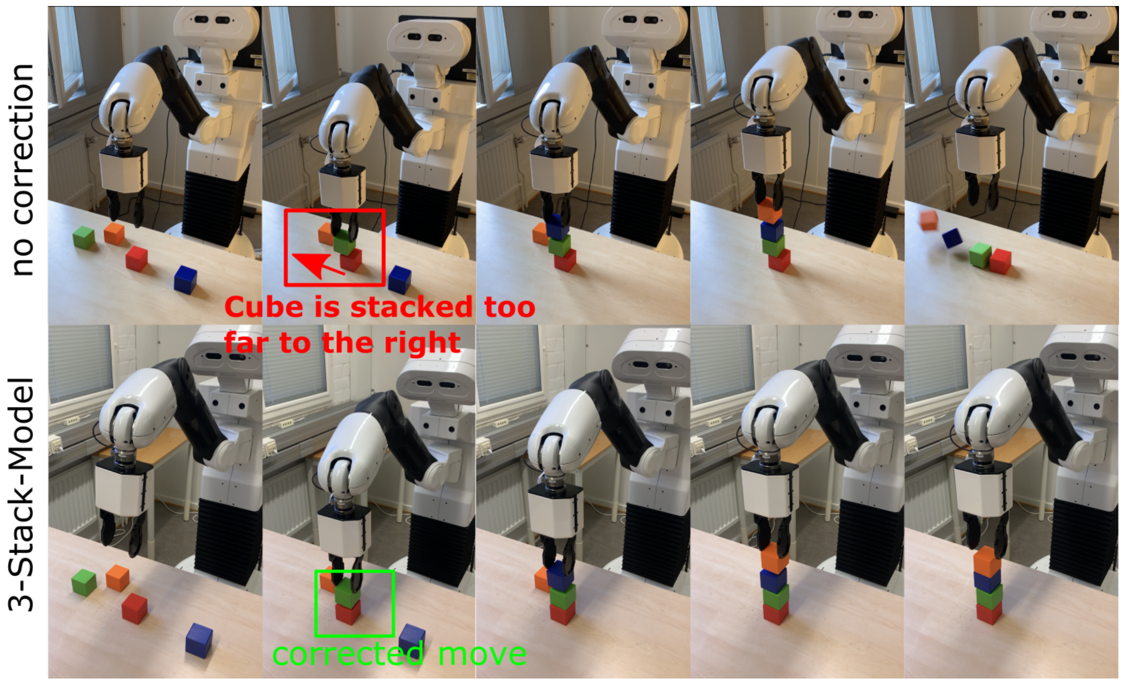


Figure 2.10: Cube stacking failure prevention example. Source: Paper C. © 2023 Elsevier (RAS).

formed the RF by 2.1% for Competence and 2.9% for Intention. Similarly, measured by F1-Score, our method outperformed the baseline by 0.047 and 0.044 for Competence and Intention, respectively. In conclusion, on average, our method outperforms prior black-box machine learning approaches. Our causal model has the additional benefit that it is interpretable and simpler, using only a subset of the full feature set from the SEAN Together dataset, where we exclude factors like the map or the behavior of other agents. Moreover, the model encodes causal information, which is crucial for generating counterfactual behaviors for improving the robot's perceived competence.

To evaluate whether our model successfully prevents incompetent robot behaviors by generating alternatives perceived as more competent, we conducted an online user study using Prolific. We hypothesize that our method can improve the perceived competence of robot navigation behavior, specifically, that:

H1) When our causal model predicts the perceived competence *correctly as low*, our approach generates navigation behaviors that are perceived as more competent than the original robot behavior.

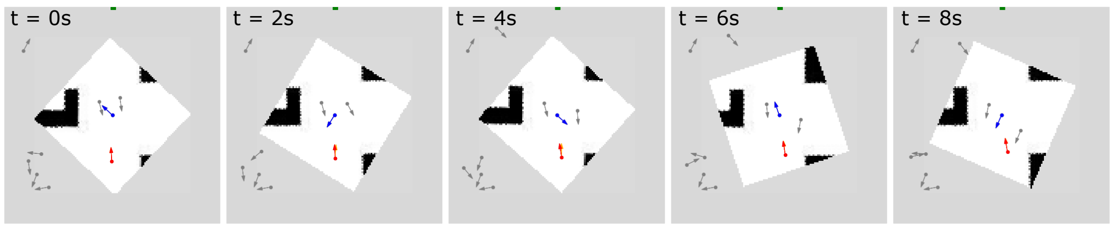
Dimension	F1	Accuracy	Precision	Recall
Causal Model (ours)				
Competence	$\mathbf{0.777 \pm 0.09}$	$\mathbf{0.835 \pm 0.08}$	0.811 ± 0.12	$\mathbf{0.768 \pm 0.14}$
Intention	$\mathbf{0.751 \pm 0.1}$	$\mathbf{0.788 \pm 0.10}$	$\mathbf{0.823 \pm 0.11}$	$\mathbf{0.713 \pm 0.15}$
Random Forest (eye_follower_gaze_goal_lip_map_resnet18)				
Competence	0.73 ± 0.12	0.814 ± 0.09	$\mathbf{0.816 \pm 0.13}$	0.686 ± 0.16
Intention	0.707 ± 0.12	0.759 ± 0.12	0.817 ± 0.11	0.654 ± 0.18

Table 2.4: Leave-One-Out Cross-Validation (LOOCV) ($\mu \pm \sigma$) on binary F1-Score, Accuracy, Precision, and Recall for 2 classifiers and 2 dimensions (Competence or Intention). The proposed Causal Model (ours) is compared to the best performant baseline RF classifier. Bold indicates the highest performance by metric and dimension. Source: Adapted from Paper G.

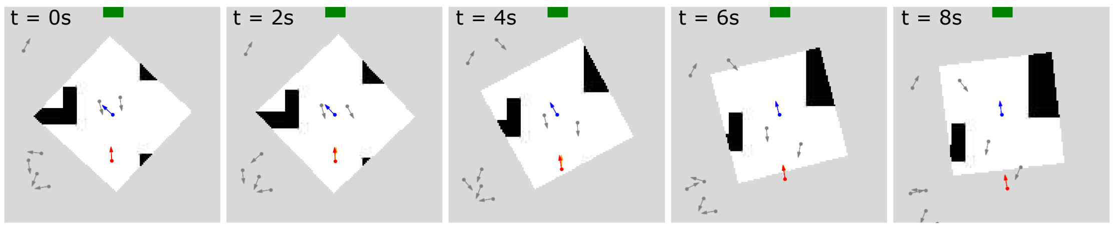
H2) When our causal model predicts the perceived competence *erroneously low*, our approach still generates navigation behaviors that are perceived as more competent than the original robot behavior.

To test both of these hypotheses, we implemented two distinct study phases: One phase studied scenarios (10 total) in which our model correctly predicted low robot competence. The other phase studied scenarios (10 total) where the model incorrectly classified the robot as low competent. We did not sample examples from the dataset where our model predicted high perceived competence, because our model only generates counterfactual behaviors in cases where the predicted competence is low. We recruited a different set 20 of participants for each study phase.

In both studies, the participants watched 20 videos where 10 showed the original robot behavior and another 10 videos showed the counterfactual behavior. Both pairs came from the same scenario, so a participant would always see the original and counterfactual video of each scenario (see Fig. 2.11 for one example of original and counterfactual behavior for the same scenario). After each video, participants rated how they believed the human follower perceived the robot's competence using a 5-point Likert responding format where 1 corresponds to "very incompetent" and 5 corresponds to "very competent". **(H1) Competence of the Counterfactuals in cases where the model correctly predicted the Original as low competent.** We found a significant difference in participants' ratings between the original trajectories and our model's proposed counterfactual trajectories when the model made a correct prediction, $F(1, 199) = 153.28, p < 0.0001$. Overall, participants rated



(a) Original robot behavior



(b) Counterfactual robot behavior

Figure 2.11: The image series, extracted from two videos of a navigation task, illustrate maps of the environment. The blue arrow represents the robot's position and orientation. The red arrow depicts the follower. Other people in the environment are indicated by grey arrows. The goal, located at the top of the images, is a green rectangle. The black areas are static obstacles and the white area is navigable space around the robot. The surrounding grey region is beyond the 7.2m public space where the 2D map was recorded. The upper image series shows the original robot behavior that our model classified as low competent. The lower series visualizes the counterfactual robot behavior that our method generated to address the low competence behavior. The images are centered around the new robot position and orientation. The behavior of the human follower and the other pedestrians is unchanged and displayed at its original location. Source: Paper G.

the competence of counterfactual trajectories ($3.60 \pm .09$) higher than the original trajectories (2.09 ± 0.09) using the 5-point responding format (where higher is better). This means that, on average, counterfactual trajectories improved the perceived competence of the robot by 72%.

These results supported H1, demonstrating our method's potential to improve robot trajectories and enhance the perceived competence of a robot.

(H2) Competence of the Counterfactuals in cases where the model erroneously predicted the Original as low competent. We found a significant difference in participants' ratings between the original trajectories and our model's proposed counterfactual trajectories when the model made an incorrect prediction, $F(1, 199) = 30.02, p < 0.0001$. Overall, participants rated the competence of counterfactual trajectories ($3.50 \pm .10$) higher than the original trajectories (2.80 ± 0.11). On average, this meant that counterfactual trajectories improved the perceived competence of the robot by 25%.

These results supported H2, demonstrating our method's potential to improve robot trajectories and enhance the perceived competence of the robot, even when the model makes an incorrect initial prediction regarding the robot's perceived competence.

Our results show that, across both study phases, the counterfactual trajectories were rated significantly more competent than the original robot behaviors. Generally, we found that the counterfactuals help prevent low-competence behaviors by guiding the robot to rotate and move toward the goal.

CHAPTER 3

Using Learning from Demonstration to Overcome Task Planning Failures

Task planning failures occur when a robot's existing set of actions is insufficient to generate a plan to achieve its goal because the robot lacks knowledge of the complete task or specific subtasks required for its completion. These failures are particularly prevalent in dynamic human environments, where conditions and task requirements frequently diverge from those the robot was originally trained or programmed to handle. For example, a robot is already capable of storing a plate inside a kitchen drawer, but doesn't know yet how to remove a chair that obstructs the drawer.

To overcome such task planning failures, this thesis proposes leveraging human demonstrations to teach robots tasks or individual capabilities they cannot yet perform. Specifically, our work focuses on extracting symbolic, robot-agnostic action descriptions from these demonstrations, which can then be integrated into the robot's existing action library. This approach enables robots to continuously expand their capabilities and reapply learned actions to future tasks. Such action models are commonly used in the well-established field of Automated Planning (AP) [35]. This chapter introduces the foundational principles of AP, particularly how action models are defined and utilized to plan robot tasks. We then formalize task planning failures and outline our approach to learning tasks from human demonstrations (RQ5) and how we extended this framework to multi-agent task planning (RQ6).

3.1 Automated Planning

Since the early days of AI, Automated Planning (AP) has played a crucial role in enabling autonomous robot decision-making [113]. The STRIPS planning system, developed in the 1970s for the Shakey robot [114], laid the groundwork for numerous robotic applications, including autonomous spacecraft [115], exploration and rescue robots [116], and autonomous aerial vehicles (AUVs)[117]. More recently, planning has been used for collaborative robots in assembly lines [118], benefiting from its ability to replan in response to unexpected situations or changes. Furthermore, the ROSPlan framework [119] integrates planning capabilities into the widely used Robot Operating System (ROS).

3.1.1 Planning Domain

The core principle of AP is to abstract lower-level execution details, such as joint trajectories or sensor readings, into high-level, symbolic actions. By leveraging such descriptive models, robots can plan more effectively long-horizon tasks. This approach has been proven especially important for robot decision-making in complex scenarios, such as setting a table [113]. In AP, the environment and the agent's ability to interact with it are defined in the so-called planning domain $\Sigma = (S, A, \delta)$ or $\Sigma = (S, A, \delta, c)$, where S is a finite set of states in which the system can be, A is a finite set of actions that the agent can perform, δ is a state transition function that describes how the system state changes when an action is applied, and c is a function that represents the costs of the possible state transitions [35]. The cost is a metric which can be monetary, time, energy, or something else depending on the domain of the environment.

3.1.1.1 Object and State Representation

In AP we need to define a finite set of objects that the planner can reason about, which typically contains all task-relevant objects, thus, all the objects an agent might interact with while executing its task. This set is task-dependent and usually needs to be provided manually. Formally, we define a set B that contains all object names, and is typically further divided into object-type-specific subsets [35]. For instance, for a robot tasked with cleaning a kitchen, the environment could be described using the following objects:

$$\begin{aligned}
\mathit{Robot} &= \{\mathrm{r}_1\};\\
\mathit{Table} &= \{\mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3\};\\
\mathit{Fork} &= \{\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3, \mathrm{f}_4\};\\
\mathit{Knife} &= \{\mathrm{k}_1, \mathrm{k}_2, \mathrm{k}_3, \mathrm{k}_4\};\\
\mathit{Spoon} &= \{\mathrm{sp}_1, \mathrm{sp}_2\};\\
\mathit{Plate} &= \{\mathrm{p}_1, \mathrm{p}_2\};\\
\mathit{Bowl} &= \{\mathrm{b}_1\};\\
\mathit{Glass} &= \{\mathrm{g}_1, \mathrm{g}_2, \mathrm{g}_3\};\\
\mathit{Drawer} &= \{\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3\};\\
B &= \mathit{Robot} \cup \mathit{Table} \cup \mathit{Fork} \cup \mathit{Knife} \cup \mathit{Spoon}\\
&\quad \cup \mathit{Plate} \cup \mathit{Bowl} \cup \mathit{Glass} \cup \mathit{Drawer},
\end{aligned}$$

where *Robot* is a the subset of B that contains one robot called r_1, and Table is a set of three tables named t_1, t_2, and t_3.

To describe relations and properties for the previously defined set of objects, First-Order-Logic predicates are used. Each predicate has a *predicate symbol*, such as `onTable` or `handClear`, and comes with an *arity* that defines the number of arguments. Furthermore, each predicate symbol has an interpretation that specifies what it means [120]. Typically relations refer to predicates with arity $>= 2$ and properties refer to predicates with only one argument. When we instantiate predicate symbols with objects, we call them ground atoms $g \in G$. Ground atoms are boolean facts, such as:

- `insideDrawer`$(\mathrm{f}_1, \mathrm{d}_2)$: True if fork f_1 is inside drawer d_2,
- `onTable`$(\mathrm{b}_1, \mathrm{t}_1)$: True if bowl b_1 is on top of table t_1,
- `inHand`$(\mathrm{p}_2, \mathrm{r}_1)$: True if robot r_1 is holding the plate p_2,
- `handClear`(r_1): True if the gripper of robot r_1 is free,

- $\texttt{graspable}(\mathrm{sp}_2, \mathrm{r}_1)$: True if spoon sp_2 is graspable by a robot gripper r_1.

G is the set that contains all possible predicate instantiations:

$$\begin{aligned}G = \{&\texttt{insideDrawer}(\mathrm{f}_1, \mathrm{d}_1), \texttt{insideDrawer}(\mathrm{f}_1, \mathrm{d}_2), ..., \texttt{insideDrawer}(\mathrm{g}_3, \mathrm{d}_3),\\ &\texttt{onTable}(\mathrm{f}_1, \mathrm{t}_1), \texttt{onTable}(\mathrm{f}_1, \mathrm{t}_2), ..., \texttt{onTable}(\mathrm{g}_3, \mathrm{t}_3),\\ &\texttt{inHand}(\mathrm{f}_1, \mathrm{r}_1), \texttt{inHand}(\mathrm{f}_2, \mathrm{r}_1), ..., \texttt{inHand}(\mathrm{g}_3, \mathrm{r}_1),\\ &\texttt{handClear}(\mathrm{r}_1),\\ &\texttt{graspable}(\mathrm{f}_1, \mathrm{r}_1), \texttt{graspable}(\mathrm{f}_2, \mathrm{r}_1), ..., \texttt{graspable}(\mathrm{g}_3, \mathrm{r}_1)\}.\end{aligned}$$

The world state $s \subseteq G$ is a set of ground atoms that are true at a given moment. For example:

$$s = \{\texttt{insideDrawer}(\mathrm{f}_1, \mathrm{d}_1), \texttt{onTable}(\mathrm{b}_1, \mathrm{t}_1)\},$$

describes a state where $\texttt{insideDrawer}(\mathrm{f}_1, \mathrm{d}_1)$ and $\texttt{onTable}(\mathrm{b}_1, \mathrm{t}_1)$ is true and all other ground atoms in G evaluate to false. The state space S has $2^{|\mathrm{G}|}$ states s, which can be large but is always finite.

3.1.1.2 Action Representation

An agent interacts with the world through a set of symbolic actions A, which are descriptive models that specify the preconditions and effects of those actions. Formally, each action $a \in A$ is defined by a 4-tuple:

$$a = \langle c(a), \mathit{pre}(a), \mathit{eff}^+(a), \mathit{eff}^-(a)\rangle,$$

where

- $c(a)$ is the *cost* of the action,
- $\mathit{pre}(a)$ is the set of *preconditions*, which are ground atoms that must be true in a state for the action to be applicable,
- $\mathit{eff}^+(a)$ and $\mathit{eff}^-(a)$ are the *add effects* and *delete effects*, respectively, which define how the action changes the state. *Add effects* are ground atoms that become true after applying the action and *delete effects* ground atoms that become false after applying the action.

Consider for example the `pick` action, where a spoon is picked by a robot from a table:

- $\text{Pre}(\texttt{pick}) = \{\texttt{onTable}(\text{sp}_2, \text{t}_3), \texttt{handClear}(\text{r}_1)\}$
- $\textit{eff}^+(\texttt{pick}) = \{\texttt{inHand}(\text{sp}_2, \text{r}_1)\}$
- $\textit{eff}^-(\text{pick}) = \{\texttt{onTable}(\text{sp}_2, \text{t}_3), \texttt{handClear}(\text{r}_1)\}$

The action `pick` can be executed if the spoon sp_2 is on the table t_3 and the robot's gripper r_1 is clear. Upon completion, the spoon sp_2 is now in the gripper of the robot r_1, no longer on the table t_3, and the robot's gripper is not clear anymore.

The *transition function* $\delta(s, a)$ describes how the system state s changes when an action a is applied:

$$\delta(s, a) = \begin{cases} \perp & \text{if } s \not\models \textit{pre}(a) \\ s \cup \textit{eff}^+(a) \setminus \textit{eff}^-(a) & \text{otherwise} \end{cases}$$

If the preconditions $\textit{pre}(a)$ are not met in s, the action cannot be applied, therefore $\delta(s, a) \models \perp$ (indicating an invalid transition). Otherwise, applying a updates s by adding the ground atoms in $\textit{eff}^+(a)$ and removing those in $\textit{eff}^-(a)$.

The cumulative transition function applies a sequence of actions $\langle a_1, a_2, \ldots, a_n \rangle$ starting from s, effectively chaining actions:

$$\delta(s, \langle a_1, a_2, \ldots, a_n \rangle) = \delta(\delta(s, a_1), \langle a_2, \ldots, a_n \rangle)$$

These symbolic action representations provide a high-level abstraction of the behavior, focusing on the desired outcomes of an action rather than the specifics of its implementation. Translating these actions into real-world execution typically requires lower-level processes, such as motion planning or sensor processing, tailored to the robot hardware that embodies the agent.

3.1.2 Planning Problem

The goal of AP is to solve planning problems by determining a sequence of actions that move a system from an initial state to a desired goal state.

A **Classical Planning Problem** [121] is represented by the 3-tuple

$$\mathcal{M} = \langle \Sigma, I, G \rangle,$$

where:

- Σ is the state transition system (or classical planning domain), as previously defined in Sec. 3.1.1.
- I is the initial state, describing the system's starting conditions. An initial state can, similarly to any other state s, be defined as a set of ground atoms that are true at the initial state.
- G is the goal state, which the plan aims to achieve. It is defined as a set of ground literals. A literal is either
  - A ground atom (e.g., `onTable`$(\mathrm{f}_1, \mathrm{t}_2)$), or
  - A negated ground atom ($\neg$`(onTable`$(\mathrm{f}_1, \mathrm{t}_2)$`)`).

  Any ground atom not explicitly mentioned in the goal set, whether as a positive or negated literal, is left unconstrained. This means that the planner is not required to satisfy or avoid such atoms; their truth values in the final state are irrelevant to achieving the goal.

A *solution* to a planning problem $\mathcal{M}$ is a sequence of actions, typically called a *plan*, $\pi = \langle a_1, a_2, \ldots, a_n \rangle$, that transitions the initial state I to a goal state G, i.e.,

$$\delta(I, \pi) \models G.$$

The *cost of a plan* π is calculated as the sum of the costs of all actions in π:

$$C(\pi) = \sum_{a \in \pi} c_a.$$

If a plan cannot achieve G from I, we set $C(\pi) = \infty$. If a plan achieves the goal with the minimum cost, then it is an optimal solution. That is, no other plan exists that has a smaller cost.

3.1.3 Formulation of a Planning Problem - PDDL

To standardize the formulation of planning problems, the planning community developed the Planning Domain Definition Language (PDDL) [122]. To formulate the previously discussed robot example in PDDL, two separate files are needed:

- Domain Specification (domain.pddl): This file encodes the planning domain Σ. It encapsulates the general rules, actions, predicates, and object types that apply across various problem instances.
- Problem Specification (problem.pddl): This file specifies the initial state I, the goal state G, and the specific set of objects B relevant to the particular planning problem.

The separation into two files is advantageous because if the robot needs to solve a new planning problem (e.g., the initial or goal state changes), only the problem file needs to be adapted. The domain file remains unchanged. Below, we provide an example domain.pddl file.

```
(define (domain kitchen)
(:requirements :typing :strips)
(:types robot object table drawer
       fork knife spoon plate bowl glass - object)

(:predicates
  (onTable ?obj - object ?tab - table)  ; Object is on top of a table
  (insideDrawer ?obj - object ?draw - drawer) ; Object is inside a drawer
  (inHand ?obj - object ?rob - robot) ; Object is in robot hand
  (handClear ?rob - robot) ; Robot hand is free
  (graspable ?obj - object ?rob - robot) ; Object is within reach of the robot
)

; Action to reach an object
(:action reach
  :parameters (?rob - robot ?obj - object ?tab - table)
  :precondition (and
    (handClear ?rob)
    (not (inHand ?obj ?rob))
    (onTable ?obj ?tab))
  :effect (and
22        (graspable ?obj ?rob)
23        (forall (?o - object) ; makes all other objects
            non-graspable
24          (when (not (= ?o ?obj))
25            (not (graspable ?o ?rob)))))
26    )
27
28    ; Action to pick an object from a table
29    (:action pick
30      :parameters (?rob - robot ?obj - object ?tab - table)
31      :precondition (and
32        (handClear ?rob)
33        (graspable ?obj ?rob)
34        (onTable ?obj ?tab))
35      :effect (and
36        (not (handClear ?rob))
37        (inHand ?obj ?rob)
38        (not (graspable ?obj ?rob)))
39    )
40
41    ; Action to put an object into a drawer
42    (:action put-in
43      :parameters (?rob - robot ?obj - object ?draw - drawer)
44      :precondition (and
45        (not (handClear ?rob))
46        (inHand ?obj ?rob)
47        (not (insideDrawer ?obj ?draw)))
48      :effect (and
49        (handClear ?rob)
50        (not (inHand ?obj ?rob))
51        (insideDrawer ?obj ?draw))
52    )
53  )
```

The domain specification in PDDL begins with the domain name (line 1), which serves as a reference in the problem specification. Line 2 specifies the requirements or features used to define the planning domain. In this example, we utilize :typing, which allows predicates to apply to groups of objects rather than individual objects alone. In this example, we defined the object types `robot`, `object`, `table`, `drawer`, as well as `fork`, `knife`, `spoon`, `plate`, `bowl` and `glass` which are subtypes (as indicated by the hyphen) of `object`. PDDL also supports advanced features such as cost metrics (e.g., minimizing distance or time), temporal constraints (e.g., scheduling actions with durations), and numeric constraints (e.g., managing resources or quantities). However, not all planning algorithms are equipped to handle these advanced features, and the planning algorithm must evaluate whether it can support the domain's specified requirements.

Lines 5–11 define predicates. Note that the predicate definition in PDDL allows to specify the range of its arguments. E.g., we can define that `onTable` relates an object of type *object* with an object of type *table*. Starting at line 14, action templates or operators O, generalized blueprints for actions, are defined. Each operator includes a name, parameters (object types), preconditions, and effects (add and delete effects). The three operators in this domain, reach, pick, put-in, represent the robot's actions and their effects. For instance, the reach action requires the robot's hand to be free and the target object to be unheld. Upon execution, the target object becomes graspable, while any object previously graspable by the hand becomes non-graspable.

The domain specification does not include details about the environments's initial state, or the goal conditions. It also omits particular object names. Instead, these details are defined in the problem specification. Below is the PDDL code for a potential kitchen problem:

```
(define (problem pickandplaceproblem)
(:domain kitchen)

;; Object definitions
(:objects
  r1 - robot
  d1 d2 d3 - drawer
  t1 t2 t3 - table
  f1 f2 f3 f4 - fork
  k1 k2 k3 k4 - knife
  sp1 sp2 - spoon
  p1 p2 - plate
  b1 - bowl
  g1 g2 g3 - glass
)

;; Initial state
```

```
  (:init
    ;; Objects initially on tables
    (onTable f1 t1)
    (onTable f2 t2)
    (onTable k1 t3)
    (onTable sp1 t1)
    (onTable p1 t2)
    (onTable g1 t3)

    ;; Robot hand state
    (handClear r1)
  )

  ;; Goal state
  (:goal
    (and
      (handClear r1)
      (not (inhand f1 r1))
      (not (inhand f2 r1))
      (not (inhand k1 r1))
      (not (inhand sp1 r1))
      (not (graspable f1 r1))
      (not (graspable f2 r1))
      (not (graspable k1 r1))
      (not (graspable sp1 r1))

      ;; Objects that are moved from the tables into the
         drawers
      (insideDrawer f1 d1)
      (insideDrawer f2 d1)
      (insideDrawer k1 d2)
      (insideDrawer sp1 d3)
      )
    )
  )
```

The :objects section lists all objects found in the environment. In this example, these include the robot (r1), drawers (d1-d3), tables (t1-t3), forks (f1–f4), knives (k1–k4), spoons (sp1, sp2), plates (p1, p2), a bowl (b1), and glasses (g1–g3).

The :init section describes the initial state using predicates associated with these objects. For instance, objects like forks, knives, spoons, and plates might initially be on specific tables (t1, t2, t3). The robot (r1) starts with its gripper clear, holding no objects.

The :goal section defines the desired final state. For example, the goal might require placing part of the cutlery from the tables into specific drawers with the robot's hand ending up clear.

3.1.4 Beyond Classical Planning

The previously introduced formulation of a planning domain and problem is referred to as a classical planning domain, which assumes a discrete, fully observable world with deterministic actions. Extensions to the classical PDDL formulation support more complex scenarios, including temporal planning (accounting for time-dependent actions), probabilistic models, and multi-agent systems [1]. Planning tasks can also be formalized using alternative frameworks, such as Markov Decision Processes (MDPs) or Hierarchical Task Networks (HTNs).

In this work, we approach operator generation from human demonstrations as a classical planning problem. While the associated assumptions (discrete, fully observable, deterministic) may seem restrictive, they are necessary due to the limited observation size compared to other data collection methods, such as simulated environments. Moreover, these action models are utilized only during the initial planning phase, prior to robot execution. They define what needs to be done, while leaving the specifics of execution to lower-level processes. For example, as in Paper K or [123], a Reinforcement Learning policy can implement specific robot operators as low-level execution skills, thereby managing environmental uncertainty at the execution level.

In Paper E, we leverage a multi-agent version of PDDL to model scenarios involving multiple actors. This enables effective representation of complex, collaborative behaviors, as illustrated in the following example:

```
(:action unscrew
  :agent ?a - agent
  :parameters (?l - lid ?bot - bottle)
  :precondition (and
```

[1] A comprehensive overview of PDDL extensions is available at: https://planning.wiki/guide/whatis/pddl.

```
      (not (inHand ?l ?a))
      (closed ?bot ?l) ; bottle is closed by a lid
      (exists (hold ?bot ?l)))
    :effect (and
      (inHand ?l ?a)
      (not (closed ?bot ?l)))
  )

  (:action hold
    :agent ?a - agent
    :parameters (?bot - bottle ?l - lid)
    :precondition (and
      (inHand ?bot ?a)
      (closed ?bot ?l)
      (exists (unscrew ?bot ?l)))
    :effect (and
      (inHand ?bot ?a)
      (not (closed ?bot ?l)))
  )
```

This example defines two interdependent actions, `unscrew` and `hold`, involving a robotic hand, a bottle, and its lid. These actions must be scheduled in parallel to ensure the common goal of opening the bottle is achieved. The Multi-Agent PDDL formulation captures this requirement using two additional features:

- Agent Specification (:agent): This feature allows the planner to assign specific agents to execute each action. For example, with a dual-arm robot, two agents could be defined, one for each arm.

- Existential Quantifiers (:exists): By adding an exists quantifier, we can encode interdependent actions that must be scheduled in parallel. In this example, the hold action references the unscrew action, and conversely, the unscrew action references the hold action.

3.1.5 Planning Algorithms

To solve a planning problem, we aim to identify an (optimal) sequence of actions, $\pi = \langle a_1, a_2, \ldots, a_n \rangle$, that is a transition from the initial state I to a goal state G, satisfying

$$\delta(I, \pi) \models G.$$

The literature typically distinguishes between two main approaches: search-based methods and satisfiability-based (SAT) methods.

Search-based approaches use algorithms such as A* or Greedy Best-First Search [120] combined with heuristic functions to compute π. Heuristic functions $h(s)$ guide the search process by estimating the cost from the current state to the goal. In modern planning, heuristics are a cornerstone of efficient plan search. Historically, heuristics were domain-specific (e.g., using Euclidean distance for pathfinding) [120]. However, recent advances have enabled the development of domain-independent heuristics [35]. Planning systems typically use these heuristics in two phases: first, computing heuristic values for states in a simplified search; then, using these values to guide a full search to find π. A good heuristic balances computational efficiency with accurate cost estimation, often relying on relaxed problem formulations that simplify computations while preserving the problem's structure. For example, the Delete Relaxation Heuristic [35] simplifies the problem by ignoring the negative effects of actions, making it easier to solve. Another popular approach is the Landmark Heuristic [35], which identifies critical subgoals (landmarks) that must be achieved to reach the goal. These landmarks guide the search process by structuring the sequence of actions. For instance, in a block-stacking task, stacking the first cube is a prerequisite for placing the second cube.

SAT-based approaches [124] encode the planning problem as a propositional formula in Conjunctive Normal Form (CNF), where each state, action, and transition is represented by a Boolean variable. A general SAT solver is then used to check the satisfiability of the formula, determining whether there exists a sequence of actions that transitions the system from the initial state to the goal. If the formula is satisfiable, the solver provides a solution corresponding to a valid plan.

While both search-based and SAT-based methods are effective for solving planning problems, we chose search-based frameworks for our work. These frameworks, particularly those utilizing PDDL, offer a human-readable and modular representation of planning tasks, making debugging, modification, and maintenance easier. For our work, we integrated the Fast-Downward planning system [125], one of the most widely used tools due to its efficiency in solving large-scale planning problems, support for numerous features, and flexibility in search methods and heuristics.

3.2 Planning Failures

Planning failures occur when no sequence of actions can lead the system from the current state to the goal, i.e.,

$$\nexists \pi \text{ such that } \delta_R(I_R, \pi) \models G_R.$$

This is referred to as an unsolvable planning task. In this context, we use the subscript R to denote plans related to a robot, and H to refer to human plans. This distinction is consistently applied across all relevant domain artifacts (e.g., δ_R, A_R, δ_H, A_H, etc.). To address planning failures, we propose a human-in-the-loop approach, where a human demonstrates how to achieve the robot's goal, and the robot learns from that demonstration. In this scenario, the human demonstrator is not pursuing their own goal but instead shows how to accomplish the robot's goal (which is assumed to be known to both the robot and the human). In Paper F, we defined a human demonstration as follows:

Definition 1. An (Unguided) Demonstration for an unsolvable task, is a human plan π_H that achieves the goal, wherein the robot can replicate parts of the demonstration to achieve the goal itself from the original initial state:

$$\delta_H(I_R, \pi) \models G_R \text{ such that } \delta_R(I_R, \mathbb{M}(\pi)) \models G_R,$$

where $\mathbb{M}$ is a mapping function that translates the demonstrated human actions into robot-executable actions, a process known as embodiment mapping [126]. From the obtained demonstration $\delta_H(I_R, \pi)$, we need to extract a set of actions A_H which is then added to the robot's current action set $A'_P = A_P \cup \mathbb{M}(A_H)$. In this thesis, we assume $\mathbb{M}$ is the identity mapping, thus $A'_P = A_P \cup A_H$, which can be ensured by designing a state abstraction such that actions are directly applicable to both human and robot domains.

3.2.1 Challenges

C1: Extracting A_H from a human demonstration is not straightforward because demonstrations typically involve a continuous flow of states. Therefore, we need a method that can segment this continuous data stream into individual actions (C1.1), and identify the relevant preconditions and effects of each action (C1.2). Additionally, the method must translate the observed actions, along with their precondition and effect predicates, into general action templates, known as operators O (C1.3). These challenges are addressed in Paper A.

C2: When planning failures occur, the exact cause is often unclear. It could be that the robot doesn't know how to complete the entire task, for instance, if $A_P = \varnothing$, or because a single action is missing. In such cases, a human must either demonstrate the entire task sequence, even if parts of it are already known to the robot, or manually identify and demonstrate the specific actions the robot is lacking. Re-teaching the entire task sequence can become tedious and repetitive, which may diminish the user's willingness to continue teaching the robot. Moreover, requiring the user to gain a deeper understanding of the robot's action set contradicts the purpose of learning from demonstration, as such insight may not be readily available to the user, even if human-readable action models provide some assistance. The underlying issue is that robots typically lack the ability to reason about which sub-tasks are necessary to complete the overall task, an issue we address in Paper F.

By addressing C1 and C2 this thesis contributes towards an efficient, robot-agnostic, and interpretable way of learning new tasks from human demonstrations (RQ5).

C3: Finally, most Learning from Demonstration (LfD) methods, including the one presented in Paper A, focus on single-robot scenarios, despite the fact that humans have two hands and often perform tasks in parallel or require synergies to accomplish a task successfully. Moreover, many industrial settings use multi-robot systems. However, applying current LfD methods to multi-robot systems is not straightforward, as they do not explicitly account for constraints related to shared spaces and action synergies between robots. The key challenges[2] in this context include:

C3.1 **Sequential Plans:** To solve tasks with newly taught robot actions, current PbD and LfD pipelines [30] integrate planning systems, such as FastDownward [125], for automatic action sequenc-

[2]Adapted from Paper E.

ing. However, the resulting plans consist only of sequential actions, making them unsuitable for concurrent multi-agent (MA) tasks.

C3.2 **Space and Resource Constraints:** Although transitioning from single-agent to multi-agent PDDL (MA-PDDL) can generate multi-agent plans, ensuring the feasibility of plan execution requires additional precautions. A common issue arises when multiple robots can reach a space, but only one robot can occupy it at a time. Without properly modeling these shared spaces, where both agents cannot be present simultaneously, the planner may generate a plan that results in collisions as the robots attempt to place objects onto the tray concurrently. Such resource constraints are common in various industrial tasks and must be explicitly incorporated into the operator-generation process.

C3.3 **Action synergies:** Actions requiring synergies between agents are not effectively learned in current operator-learning methods, as concurrency does not apply to single-agent scenarios [30]. For example, a robot can only close the bottle if the `Screw` and `Hold` actions are scheduled in parallel. Single-agent teaching methods typically do not account for such synergies. Therefore, to successfully close the bottle, we must explicitly model the parallel actions required to achieve the desired outcome.

We address challenges C3.1 - C3.3 in Paper E, thus contributing towards RQ6.

3.2.2 Learning Planning Operators from Human Demonstration

In Paper A, we proposed a demonstration tool that enables a robot to learn semantic operator descriptions from human demonstrations in Virtual Reality[3].

C1.1: To extract A_H from a human demonstration, our system first automatically segments and recognizes the demonstrated human activities using a state-of-the-art learning method that extracts semantic representations from the demonstrations [127]. This method segments continuous hand motions

[3]This subsection is adapted from Paper A.

features	*Granular Activity*	*Idle*	*Reach*	*Put*	*Take*
`handMove`	T	T ∨ F	T	T	F
`actedOn`	¬nil	nil	¬nil	nil	nil
`inHand`	¬nil	nil	nil	¬nil	¬nil

Table 3.1: Hand activity classification rules. T and F stand for true and false respectively, and ¬nil means an object as opposed to no-object (nil). Source: Adapted from Paper A. © 2021 IEEE (IROS).

based on a minimalistic subset of hand-specific state variables such as `inHand`, `actedOn`, and `handMove` (more details can be found in Table 2 of Paper A). Each segmented motion is then labeled with a specific symbolic meaning representing one of the recognized activities, such as `IdleMotion`, `Reach`, `Put`, `Take`, and `Granular`. Granular activities refer to actions in which one object held in the hand interacts with another (e.g., stacking two cubes or cutting bread). The mapping between hand state variables and the detected activity is realized through a decision tree that is learned from human annotated demonstrations in prior work [127]. The learned rules are presented in Table 3.1.

One of the main advantages of this semantic-based recognition method is its ability to segment and recognize continuous data without requiring additional training. This means that for the scenarios analyzed in Paper A, Paper E, and Paper F, we used the same set of rules from prior work [127]. Therefore, no new training was necessary for activity segmentation and recognition. Another advantage is that both the activity labels and the learned rules are human-readable.

C1.2 To identify relevant preconditions/effects and generates operators, we propose Alg. 4. The algorithm takes two inputs: a demonstration D, represented as a sequence of symbolic states s_t at time t, which is further subdivided into hand state $s_{t,h}$ and environment state $s_{t,e}$, along with the hand activity classification $a_{t,h}$ for each hand $h \in H$ and time point t. We differentiate between hand variables, which describe the state of the hand or its interactions with objects in the environment (see features column in Table 3.1), and environment variables, which describe the relationships between objects in the environment. The second input is a list of existing operators O, which corresponds to the existing set of robot capabilities A_R.

C1.3: A new operator is generated when the segmentation and classification system detects a transition from one activity to another, i.e., $a_{t,h} \neq a_{t-1,h}$ (L-4, Alg. 4). For example, when transitioning from *Reach* to *Take*, the `Take` operator would be generated, as changes occur in the `actedOn` and `inHand` state variables. It's important to note that not all state transitions, especially those involving environment state variables like `onTop`, result in a new classification, and therefore do not trigger the creation of a new operator. This is because classification is based on a subset of the state variables.

The advantage of our activity recognition method is that newly generated planning operators can be automatically named with human-understandable labels (L-9, Alg. 4), providing a semantic description of the underlying functionality. The preconditions of an operator are derived from the effects of the last state before the activity transition, while the effects are based on the last state of the current activity. For example, in the transition from *Reach* to *Take*, the preconditions of the `Take` operator are based on the last state of *Reach*, and its effects are based on the last state of *Take*. To account for hand-state changes that do not lead to a new activity classification, operators are not directly added to O. Instead, they are stored in a buffer list O_{Buffer} (L-10, Alg. 4), which is continuously updated (L-13, Alg. 4).

For selecting relevant environment predicates, we focus only on those that change their value during the operator's application. For hand state variables, we additionally consider predicates that remain true throughout the activity. The next step involves generalizing from specific objects, such as `Cube_green1`, to the type of object, such as `Wooden_cube` (L-8, Alg. 4). This generalization is based on the assumption that any activity applied to a specific cube can be generalized to all objects of the same type.

The last step is updating the operator list O with the newly observed operators from the buffer O_{Buffer}. Each newly observed operator is either added, in case it is different from any other operator (L-19, Alg. 4), or the count is incremented for operators who have been already observed before (L-21, Alg. 4). The output list represents the new set, combining the existing operators with the newly demonstrated ones: $A'_R = A_R \cup A_H$.

To test the functionality of the system, three participants were asked to demonstrate how to stack cubes within our VR environment. Before the first demonstration, each participant had an opportunity to familiarize themselves with the environment. Following this introductory phase, four different demonstrations were collected from each participant, based on instructions to

Algorithm 4 Operator generation & collection

Input: demonstration $D = [s_1, a_1, s_2, a_2, ...s_n, a_n]$, current list of operators O

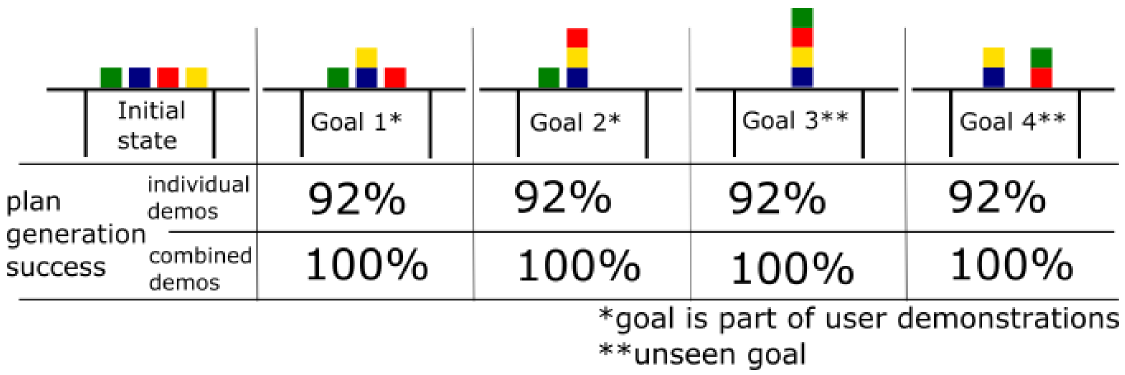
Output: updated list of operators O

```
1:  $O_{\text{buffer}} \leftarrow \{\}$
2:  for $t \leftarrow 1, n$ do                                              ▷ $n = |D|$
3:      for all $h \in H$ do
4:          if $a_{t,h} \neq a_{t-1,h}$ then
5:              $pre \leftarrow$ StateVarToPredicate($s_{t-1,h}$)
6:              $eff \leftarrow$ StateVarToPredicate($s_{t,h}$)
7:              RemoveIrreleventPredicates($pre$, $eff$)
8:              GeneralisePredicates($pre$, $eff$)
9:              $o \leftarrow$ Operator($a_{t,h}$, $pre$, $eff$)
10:             Append($O_{\text{buffer}}$, $o$)
11:         else
12:             if $s_{t,h} \neq s_{t-1,h}$ then
13:                 UpdateOperator($O_{\text{buffer}_h}[-1]$, $s_{t,h}$)
14:     if $s_{t,e} \neq s_{t-1,e}$ then
15:         $h \leftarrow$ ConnectEnvChangeToHand()
16:         AddEnvPredicates($O_{\text{buffer}_h}[-1]$, $s_{t,e}$, $s_{t-1,e}$)
17: for all $o \in O_{\text{buffer}}$ do
18:     if $o \in O$ then
19:         IncrementCount($O$, $o$)
20:     else
21:         Append($O$, $o$)
```

stack either one or two cubes using either the left or right hand. No further requirements were imposed, such as the speed of execution or the specific cubes to be stacked.

To evaluate the operator generation process, four different stacking goals were used (see Fig. 3.1). The first two goals were part of the experimental instructions, though participants were free to select their own cubes. Goals 3 and 4, however, introduced new cube configurations to further assess the system's capability. For all goals, plans were generated by calling the Fast Downward planner, which utilized the operators extracted from the human demonstrations.

Two experiments were conducted: Plans were generated using the domains from a single demonstration at a time (individual demonstrations). Plans were generated using the operators from a domain that combined all 12 demonstrations (combined demonstrations). The results showed that, with a single demonstration, plan generation was successful in 11 out of 12 cases (92%). When operators from all 12 demonstrations were combined, plan generation was successful in all cases. These results suggest that while a single demonstration is typically sufficient to generate a plan that satisfies the goals, incorporating more demonstrations increases the likelihood of success. It is also important to note that the plans generated were not simple repetitions of the sequences observed in the demonstrations. The planner was able to discern which actions from the demonstrations were essential for achieving the goal, scheduling only those actions, and omitting extraneous ones.



		Initial state	Goal 1*	Goal 2*	Goal 3**	Goal 4**
plan generation success	individual demos		92%	92%	92%	92%
	combined demos		100%	100%	100%	100%

*goal is part of user demonstrations
**unseen goal

Figure 3.1: Plan goals with corresponding plan generation success ratio. Source: Paper A. © 2021 IEEE (IROS).

3.2.3 Guided Demonstrations to Learn Missing Actions

In Paper A, we demonstrated that a single demonstration of the full task was (in 11 out of 12 cases) sufficient to extract the necessary information for generating a plan that the robot could execute. That approach required a complete demonstration of the entire task. However, what if the robot already possesses prior knowledge and only needs to learn a single missing action? In Paper F, we introduce the concept of a guided demonstration, which seeks to reduce the demonstration burden by automatically identifying missing gaps in a plan that led to planning failures. The robot can then request specific demonstrations to fill these gaps, thus addressing challenge C2. To formalize the missing gap we define the concept of an excuse[4].

The concept of an excuse was first introduced in the seminal work of [128] as a method for providing open-ended feedback to address issues in planning tasks. Since then, this concept has been broadened to include general revisions of planning tasks [129] and has been applied to debug planning models across various domain authoring tasks. These applications include goal-oriented conversational agents, decision support systems, and intelligent tutoring systems [130], [131], [132].

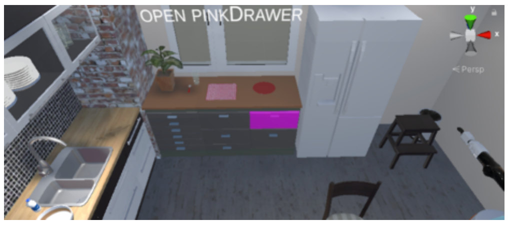
Definition 2. An Excuse $\mathcal{E} = I \Delta I'$ is a change (symmetric difference) in the state of the world with a new state I' that does not model the goal, such that the unsolvable problem becomes solvable with the new state as the initial state:

$$I_R \mapsto I'_R, \text{ such that } I'_R \not\models G_R \text{ and } \exists \pi \; \delta_R(I'_R, \pi) \models G_R.$$

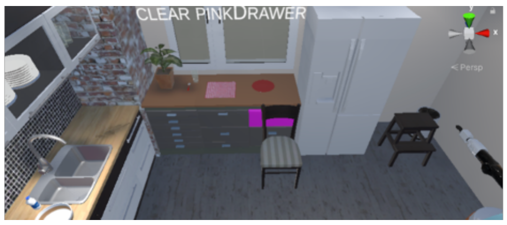
Definition 3. A minimal excuse $\mathcal{E}_{min}$ is the smallest excuse that satisfies Definition 2: $\mathcal{E}_{min} = \min_{I'} ||I \Delta I'||$. Unless otherwise mentioned, we will use an excuse to mean a minimal excuse.

To identify the missing subtask(s), we use the approach in [133], originally built for explanations in the form of model reconciliation. It mimics the excuse generation algorithm in [128] but without a user mental model. The approach performs combinatorial search in the space of possible models $\mathcal{M}$ to find a model where the task is solvable. For Paper F, we only perform model edits to the initial state I_R of the unsolvable task, changing one ground atom at a time until the task is solvable. Then the excuse is the set of model edits.

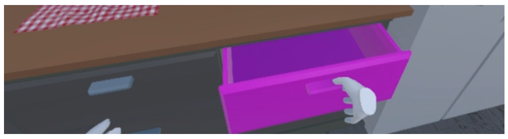
[4]This subsection is adapted from Paper F.



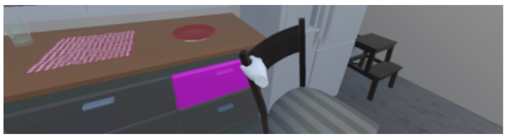
(a) Kitchen - I (showing excuse: `Open PinkDrawer`)



(c) Kitchen - II (showing excuse: `Clear PinkDrawer`)



(b) Kitchen - I (ego perspective)



(d) Kitchen - II (ego perspective)

Figure 3.2: Kitchen Domain scenarios. Source: Adapted from Paper F. © 2025 IEEE/SICE (SII).

Since there are usually multiple alternative model edits that make the task solvable, we stop the search procedure when we find the first solution.

Definition 4. A Guided Demonstration for an unsolvable task, is a human plan π_H that demonstrates how to negate the excuse from the current state, wherein the robot can replicate parts of the demonstration to achieve the goal itself from the original state: $\delta_H(I_R, \pi_H) \models I_R + \mathcal{E}$ such that $\exists \pi\ \delta_R(I_R, \pi) \models G_R$ with $\exists a \in \pi$ and $a \in \mathbb{M}(\pi_H)$.

We tested our method in the kitchen domain (3.2a-3.2d) which represents the class of HomeWorld domains [134] where the robot has to store a plate in a drawer, but initially does not know how to open a drawer (3.2a-3.2b) and how to unblock the drawer from a chair (3.2c-3.2d). Instead of having to demonstrate the full tasks, our proposed approach of guided demonstrations facilitates the teaching process, by instructing the human to only teach how to reach the automatically generated excuse states where the drawer is open (3.2a) and the drawer is clear (3.2c). We directly communicate the obtained excuse states by displaying its symbolic state description in the VR environment used for the teaching process (`Open PinkDrawer` - Fig. 3.2a and `Clear PinkDrawer` - Fig. 3.2c) as seen at the top of the respective images.

To measure the impact of guided demonstrations, we conducted an experiment with 5 participants. The participants were informed that the goal of the study was to demonstrate missing actions (referred to as *excuses*) for tasks where the robot already had partial knowledge. However, the robot's existing knowledge was not disclosed to participants, in order to assess whether the method works without the users being familiar with the robot's capabilities. The experiment started with a training phase that allowed participants to interact freely with blocks. Then, each participant performed three demonstrations per scenario, as shown in Fig. 3.2 (excuse - full task - excuse). The robot's prior domain knowledge was pre-generated based on demonstrations by one of the authors of Paper F, which enabled the robot to perform actions like placing a plate in an open drawer. In the second scenario, this knowledge also included the action of opening the drawer. During the experiment, participants demonstrated minimal excuses, such as *open PinkDrawer* and *clear PinkDrawer*, as well as the overall goal, *RedPlate inside PinkDrawer*. Both the excuses and the goal were visually displayed in the VR environment.

The results showed that our method was effective, achieving a 61% and 72% reduction in time and demonstration size. It is important to note that time savings may vary depending on the total length of the plan and the size of the knowledge gap. If the robot is missing a single action in a lengthy task (e.g., setting a table), we would expect even greater time savings. However, if the robot is missing multiple sub-tasks, the resulting savings might be smaller. In the post-study interview, participants reported that the excuses were generally easy to comprehend. They rated their confidence in responding appropriately to the excuses with a demonstration, as expected by the robot, very positively, with a mean confidence score of 6.2 out of 7 (where 7 represents very high confidence). Furthermore, although participants had the opportunity to reconsider how to address the excuse after learning the full task goal, none of the participants changed their excuse demonstration. All participants also reported that they did not require prior knowledge of the overall task goal or the robot's existing knowledge in order to understand how to perform the demonstration.

3.2.4 Learning Robot Actions from Demonstration for Multi-Agent Planning

To address the challenge of extending state-of-the-art Programming by Demonstration (PbD) and Learning from Demonstration (LfD) methods to multi-agent tasks, Paper E proposes guidelines with three key enhancements to the existing action learning and planning pipeline. As illustrated in Fig. 3.3, the gray portion represents the standard PbD and LfD approach, while the blue

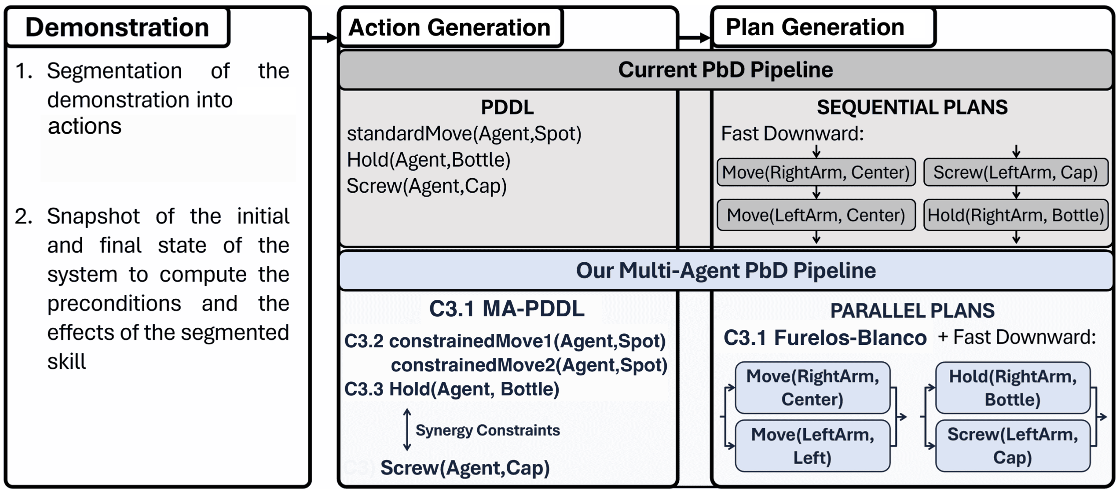


Figure 3.3: State-of-the-art pipeline from action demonstration to plan generation (gray) compared to the modified pipeline following our guidelines (blue). Source: Adapted from Paper E. © 2024 IEEE (CASE).

part highlights the modified pipeline incorporating our three contributions[5].

Single to Multi-Agent planning

To address C3.1, our guideline proposes an adaption of the action model representation from PDDL to MA-PDDL [135] (see C3.1 in the Action Generation step in Fig. 3.3). Typically LfD methods like [30] or the one presented in Paper A contain a step where the action, learned from a demonstration, are parsed into a PDDL domain. We changed this step such that the resulting output is a MA-PDDL domain. Our method furthermore introduces an additional step during the Plan Generation: As few planning systems are capable of solving MA-PDDL problems, we integrated Furelos-Blanco's method to recompile MA-PDDL back to classical planning [136] (see C3.1 in the Plan Generation step in Fig. 3.3), which then allows the usage of traditional off-the-shelf planning systems (e.g. FastDownward [125]) to solve the original multi-agent problem and produce plans where actions can be scheduled concurrently.

Constrained Movement Actions To address C3.2 our guidelines propose to manually replace any existing `standardMove` action with two constrained movement actions during the Action Generation Step of the PbD pipeline (see C3.2 in Fig. 3.3). A `standardMove` action requires only that the robot is currently at some spot Spot1 (isAgentAtSpot) and can reach another spot

[5]This subsection is adapted from Paper E.

Spot2 (reachable). The `constrainedMove1` checks that Spot2 is not being occupied at the moment and that no other robot is moving towards Spot2. The `constrainedMove2` would allow a robot to move to Spot2 if any other robot that is currently at Spot2 is moving away (Spot3). Adding these two actions makes the planner regulate the occupancy of the shared areas of the workspace, preventing any robot from moving into an area already occupied by another robot and consequently acting on any shared resources.

```
(:action standardMove
  :agent ?a - agent
  :parameters (?spot1 - spot  ?spot2 - spot)
  :precondition (and
    (isAgentAtSpot ?a ?spot1) ; agent ?a is at ?spot1
    (reachable ?spot2 ?a)) ; ?spot2 is reachable by agent ?a
  :effect (and
    not(isAgentAtSpot ?a ?spot1)
    (isAgentAtSpot ?a ?spot2)))

(:action constrainedMove1
  :agent ?a - agent
  :parameters (?spot1 - spot  ?spot2 - spot)
  :precondition (and
  (isAgentAtSpot ?a ?spot1)
  (reachable ?spot2 ?a))
  (forall ?a2 - agent (and
    not(isAgentAtSpot ?a2 ?spot2) ; no other agents are at
        ?spot2
  (forall ?spot3 - spot ; no other agents are moving towards
      ?spot2
    not(constrainedMove1(?a2 ?spot3 ?spot2)))))
  :effect (and
    not(isAgentAtSpot ?a ?spot1)
    (isAgentAtSpot ?a ?spot2)))

(:action constrainedMove2
  :agent ?a - agent
  :precondition (and
    (isAgentAtSpot ?a ?spot1)
    (reachable ?spot2 ?a)
    (exists ?a2 - agent (and
      (isAgentAtSpot ?a2 ?spot2)
      (exists ?spot3 - spot
        (constrainedMove1(?a2 ?spot2 ?spot3)))))) ; there is
            an agent ?a2 at ?spot2 that moves away from ?spot2
  :effect (and
        not(isAgentAtSpot ?a ?spot1)
        (isAgentAtSpot ?a ?spot2)))
```

Listing 3.1: PDDL description of the move actions.

Our Approach to Detect Action Synergies

To address C3.3, our guidelines propose an adaptation of the Action Generation step (see C3.3 in Fig. 3.3) to explicitly encode action synergies. MA-PDDL provides the required formalism, as it allows to define actions as preconditions. Concretely during the parsing step, any actions that have to be performed concurrently are therefore added to each other's preconditions. Our guidelines also propose automated methods for detecting, during the demonstration, when two actions indeed have to be performed in parallel to achieve a common goal. For the kinesthetic teaching method [30], this can be detected via the existing user interactions. For example, if the user wants to perform a demonstration on both arms of a YuMi robot (as shown in Fig. 3.4-a), the user has to enable the teaching mode for both arms by pushing a button. We then used this signal as a flag during the Action-Generation step to add the demonstrated actions of both arms as preconditions of the other in the MA-PDDL action description. We modified the VR LfD method in Paper A such that it automatically detects the necessity for concurrency, based on the obtained activity classifications. Activities are classified as *Idle* if a hand does not act on any object or holds an object in its hand. Thus, any action observed in parallel with an idle action (e.g., Left Hand: *Reach*, Right Hand: *Idle Motion*) is saved as a single-agent action. However, when two parallel non-idle actions are identified, we incorporate them by adding each action to the other's precondition in the MA-PDDL action description.

In the experimental validation, we aim to demonstrate that our proposed guidelines can successfully extend state-of-the-art Programming by Demonstration (PbD) and Learning from Demonstration (LfD) methods, enabling non-experts to teach actions for Multi-Agent (MA) tasks, even when these tasks involve shared space constraints and action synergies between two robots. To this end, we applied our guidelines to two single-robot teaching methods: the kinesthetic teaching method presented in [30] and our VR teaching system presented in Paper A. Subsequently, we conducted a user study where non-experts used the extended version of [30] to teach the robot new actions for an adapted version of the Tower-of-Hanoi task, using cubes instead of discs (see Fig. 3.4-a), and the MA version of Paper A for opening and closing a bottle

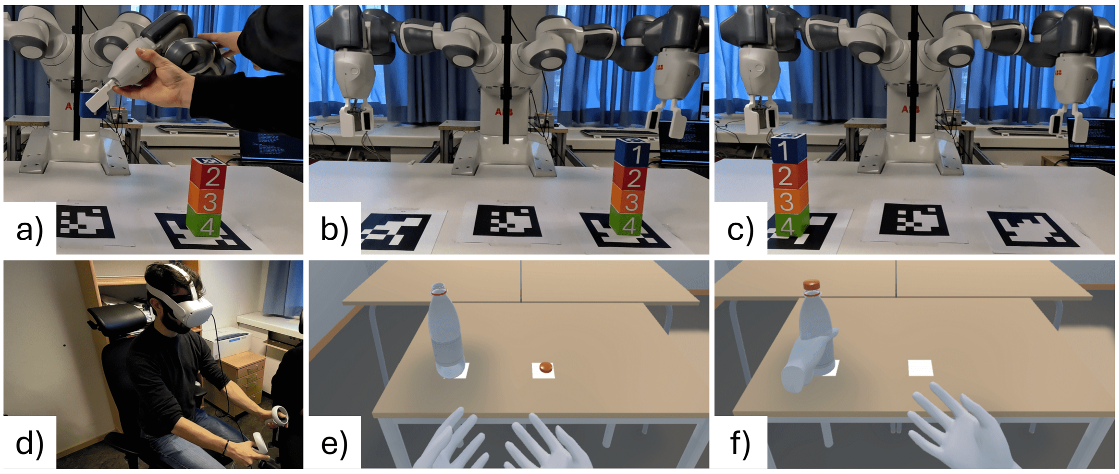


Figure 3.4: Shows the user study environments: a) Kinesthetic teaching of the Tower-of-Hanoi task: b) INIT & c) GOAL. d) VR teaching of the Close-Bottle task: e) INIT & f) GOAL. Source: Paper E. © 2024 IEEE (CASE).

(see Fig. 3.4-d). Results showed that our method can correctly encode the MA action models leading to 90% and 95% plan feasibility in the two tasks respectively. In contrast, current non-MA methods result in plans that are unfeasible or are up to 40% less efficient in plan length.

CHAPTER 4

Summary of included papers

This chapter provides a summary of the included papers.

4.1 Paper A

Diehl Maximilian, Paxton Chris, Ramirez-Amaro Karinne
Automated Generation of Robotic Planning Domains from Observations
Published in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, pp. 6732-6738, Sep. 2021.
©2021 IEEE DOI: 10.1109/IROS51168.2021.9636781.

This paper presents a method for automatically generating planning domains for robots based on human demonstrations, eliminating the need for manual domain definition. The method involves segmenting and recognizing actions from demonstrations, identifying relevant preconditions and effects, and generating planning operators. A symbolic planner then uses these operators to create action sequences that achieve user-defined goals. The approach was tested with the TIAGo robot in a simulated environment, achieving a 92%

success rate with a single demonstration and 100% success when using multiple demonstrations, even for new tasks.

Maximilian Diehl: Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation.
Chris Paxton: Supervision, Writing- Reviewing and Editing.
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing.

4.2 Paper B

Diehl Maximilian, Ramirez-Amaro Karinne
Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures
Published in IEEE Robotics and Automation Letters (RAL), vol. 7, no. 4, pp. 8925-8932, Oct. 2022.
©2022 IEEE DOI: 10.1109/LRA.2022.3188889.

This paper presents a method for enabling robots to explain task failures in human-centered environments, which is essential for enhancing trust and transparency. We tackle two main challenges: obtaining sufficient data to learn a cause-effect model of the environment and generating causal explanations based on that model. We address the first challenge by training a causal Bayesian network with simulation data. For the second, we introduce a novel approach that allows robots to provide contrastive explanations by comparing the failure state with the nearest state that would have led to success, identified through breadth-first search. This method was tested on tasks like stacking cubes and dropping spheres, achieving sim-to-real accuracy rates of 70% and 72%, respectively. The approach also scales across tasks and enables robots to offer specific explanations, such as “the upper cube was stacked too high and too far to the right of the lower cube.”

Maximilian Diehl: Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation.
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing.

4.3 Paper C

Diehl Maximilian, Ramirez-Amaro Karinne
A causal-based approach to explain, predict and prevent failures in robotic tasks
Published in Robotics and Autonomous Systems (RAS), vol. 162, Apr. 2023.
©2023 Elsevier DOI: 10.1016/j.robot.2023.104376.

This paper introduces a causal-based approach to help robots working in human environments adapt to unexpected changes and prevent failures by predicting and addressing errors. The method enables robots to foresee immediate failures and also future *timely-shifted action failures*, where current actions may compromise future success. This is achieved by detecting cause-effect relationships between tasks and their outcomes through a causal Bayesian network (BN) trained on simulation data, which is then transferred to real-world scenarios. The BN allows the robot to predict whether a current action will succeed and, if a failure is likely, identify the closest success state using contrastive Breadth-First-Search. The method was tested in stacking tasks, single and multi-cube stacks, and showed a 97% error reduction in single stacks, with around 95% of stacking errors prevented in more complex cases. These results indicate the method's effectiveness in predicting, explaining, and preventing execution failures, even in complex situations where the impact of previous actions on future tasks must be understood.

Maximilian Diehl: Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation.
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing.

4.4 Paper D

Diehl Maximilian, Ramirez-Amaro Karinne
Generating and Transferring Priors for Causal Bayesian Network Parameter Estimation in Robotic Tasks
Published in IEEE Robotics and Automation Letters (RAL), vol. 9, no. 2, pp. 1011-1018, Feb. 2024.
©2024 IEEE DOI: 10.1109/LRA.2023.3339062.

Robots in human environments often face new situations and can benefit from transferring prior experience to handle tasks zero-shot and prevent costly failures. Causal Bayesian Networks (CBNs) are widely used for modeling cause-effect relations, but while their structure transfers well, their probability distributions often require data-intensive relearning. This paper proposes three strategies leveraging semantic similarity between CBN variables to generate and transfer informed distribution priors. We evaluate their accuracy across five transfer scenarios, including sim-to-real and tasks with increased complexity. Results show our priors improve distribution estimates and enhance failure prediction by up to 50%.

Maximilian Diehl: Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation.
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing.

4.5 Paper E

Diehl Maximilian, Zappa Isacco, Zanchettin Andrea Maria and Ramirez-Amaro Karinne
Learning Robot Skills From Demonstration for Multi-Agent Planning
Published in 2024 International Conference on Automation Science and Engineering (CASE), Bari, Italy, pp. 2348-2355, Aug. 2024.
©2024 IEEE DOI: 10.1109/CASE59546.2024.10711439.

This paper addresses the limitations of Programming by Demonstration (PbD) and Learning from Demonstrations (LfD) methods, which are commonly used to simplify robot skill programming but traditionally focus on single-robot tasks. In multi-robot systems, direct application of PbD/LfD methods is challenging due to constraints related to shared spaces and action synergies between robots. To overcome this, we propose guidelines for extending PbD to Multi-Agent (MA) systems. We incorporate constrained movement skills to prevent simultaneous actions on shared resources and use action classification to identify and encode parallel actions for concurrent scheduling. The guidelines were tested with kinesthetic teaching and Virtual Reality demonstrations in a user study, focusing on tasks like a multi-agent Tower of Hanoi and Opening/Closing-Bottle task. The method achieved 90% to 95% plan feasibility, outperforming non-MA approaches that often result in unfeasible plans or plans up to 40% less efficient in length.

Maximilian Diehl: Conceptualization, Methodology, Software (VR LfD system), Data curation, Writing- Original draft preparation.
Isacco Zappa: Conceptualization, Methodology, Software (Kinesthetic teaching system), Data curation, Writing- Original draft preparation.
Zanchettin Andrea Maria: Supervision, Writing- Reviewing and Editing.
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing, Project administration.

4.6 Paper F

Diehl Maximilian, Chakraborti Tathagata, Ramirez-Amaro Karinne
Enabling Robots to Identify Missing Steps in Robot Tasks for Guided Learning from Demonstration
Published in 2025 IEEE/SICE International Symposium on System Integrations (SII), Munich, Germany, pp. 43-48, Jan. 2025.
©2024 IEEE DOI: 10.1109/SII59315.2025.10870986.

Learning from Demonstration (LfD) systems typically require users to provide full demonstrations, even when robots already know parts of a task. This redundancy makes teaching inefficient. We propose a guided demonstration method that reduces user effort by identifying which sub-task the robot still needs to learn. Using a combinatorial search, our approach determines the smallest necessary adjustment (a state we refer to as *excuse state*), that enables the robot to complete the task with its existing skills. Users then only demonstrate the missing sub-task. Experiments show that our method reduces demonstration time by 61% and decreases the size of demonstrations by 72%.

Maximilian Diehl: Conceptualization, Methodology, Software (VR LfD system), Data curation, Writing- Original draft preparation.
Tathagata Chakraborti: Conceptualization, Methodology, Software (Excuse State Generation), Data curation, Writing- Original draft preparation.
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing.

4.7 Paper G

Diehl Maximilian, Tsoi Nathan, Chavez Gustavo, Ramirez-Amaro Karinne, Vázquez Marynel
A Causal Approach to Predicting and Improving Human Perceptions of Social Navigation Robots
To be submitted to ACM Transactions on Human-Robot Interaction (THRI), 2025.

This paper addresses the challenge of enabling mobile robots to predict human perceptions of their competence and intent during navigation, which is essential for socially adaptable interactions in human environments. The two main issues are i) the need for prediction methods that can learn from limited data and ii) the importance of interpretability in these models to ensure safer interactions. Interpretable models allow robots to explain their reasoning, particularly in situations where they may be perceived as incompetent. To tackle these challenges, we propose a Causal Bayesian Network (CBN) for predicting human perceptions and interpreting robot intent. Additionally, they introduce a novel method that uses combinatorial search guided by the causal model to identify alternative navigation behaviors, thereby improving perceived robot performance when low performance is anticipated. Our approach is designed to be more interpretable and capable of generating counterfactual robot motions, achieving predictive performance comparable to or better than state-of-the-art methods, with F1-Scores of 0.79 for competence and 0.75 for intention on a binary scale. User evaluations in simulation demonstrated that their model produced motion trajectories that resulted in 72% increased competence.

Maximilian Diehl: Conceptualization, Methodology (Prediction and Counterfactual Trajectory Generation), Software (Prediction and Counterfactual Trajectory Generation), Data curation (Study design), Writing- Original draft preparation.
Nathan Tsoi Conceptualization, Data curation (Study design + study setup), Writing- Original draft preparation.
Gustavo Chavez: Data curation (Study setup).
Karinne Ramirez-Amaro: Supervision, Writing- Reviewing and Editing.
Marynel Vázquez: Supervision, Writing- Reviewing and Editing.

CHAPTER 5

Concluding Remarks and Future Work

Future robots are expected to take on challenging tasks such as cleaning, washing dishes, and doing laundry, requiring them to operate in human environments. However, to function effectively in these dynamic settings, robots must flexibly adapt to unexpected changes and proactively predict and prevent failures. Importantly, as robots will work alongside humans, who are typically non-experts, they must foster trust and acceptance. Therefore, the first goal addressed in this thesis was to develop new decision-making methods that not only enable robots to reason about and prevent failures but also explain why they occurred.

Building on the central role of causality in human reasoning, decision-making, and failure explanation, this thesis proposed using symbolic causal models as a foundation for failure prediction, prevention, and explanation. We introduced an approach to learning causal robot task models from simulation data (RQ1). Subsequently, we proposed a new method that enables robots to generate contrastive failure explanations using a breadth-first search procedure. Guided by the causal model, this method identifies the closest variable parametrization in which the robot would have successfully completed the task (RQ3). Our explanations are selective, considering only causally relevant variables and incorporating only those that change during the breadth-first search procedure. Moreover, the chosen abstraction layer of our causal models includes variables such as the x and y offset between two cubes that the robot is tasked with stacking. This allows the robot to explain failures based on external effects (e.g., "the cube was stacked too far to the right") rather than internal mechanisms (e.g., motion planning errors). Additionally, our explanations do not rely on probabilities (e.g., "I failed because my success chance in this configuration was only 20%"), making them more intuitive for non-expert users.

Building on this foundation, we extended our method to allow robots to proactively predict and prevent failures. Using causal models, the robot can search for the closest configuration in which it is likely to succeed. Our results show that this approach achieves a failure prevention rate of 95% (RQ2). We further adapted this failure prevention method for a social robot navigation scenario, where we used our causal model to predict perceived competence in a navigation task (RQ4). Our model matched or surpassed state-of-the-art performance while offering greater interpretability, and our method increased perceived robot competence by 72%. Additionally, we proposed three novel strategies for generating and transferring informed distribution priors based on the semantic similarity between two causal Bayesian networks (CBNs). We evaluated these methods in five transfer scenarios, demonstrating that the transferred priors enhanced the robot's ability to predict and prevent failures by up to 50%, particularly in zero-shot transfer situations.

Proactive failure prediction and retrospective failure explanations are essential for mitigating the negative effects of failures. However, given the complexity of human environments, robots may still encounter failures that they cannot resolve autonomously. In such cases, humans could assist robots by teaching the missing action. Therefore, the second goal of this thesis was to enable robots to learn how to perform these tasks by integrating and reusing previous experiences alongside newly acquired knowledge. This knowledge should be shareable among robots in different environments and adaptable to various scenarios. To make this accessible to non-experts, we proposed a virtual reality system that enables users to intuitively teach missing capabilities. Our system automatically segments and classifies demonstrations, generating symbolic, robot-agnostic actions that integrate into the robot's action library. This approach achieved a 92% success rate in learning task abstractions from

Research Questions	**Papers**
RQ1: How can we detect and learn cause-effect relationships in robot tasks involving timely shifted and erroneous action effects?	B, C
RQ2: How can a robot use the previously obtained causal models to predict and prevent future failures?	C, G
RQ3: How can a robot use previously obtained causal models to generate contrastive and selective explanations for task failures?	B
RQ4: How can we increase the data efficiency for learning causal models? How can we transfer causal models between tasks?	D, G
RQ5: How can a robot learn tasks in an efficient, agnostic, and interpretable way by observing human demonstrations?	A, F
RQ6: How can we extend current Learning from Demonstration methods from teaching single- to multi-agent tasks?	E

Table 5.1: Overview of Research questions and corresponding papers

a single demonstration. Additionally, guided learning from demonstration reduced demonstration time by 61% by leveraging combinatorial search to identify missing plan elements (RQ5). Finally, we proposed guidelines for extending our current learning-from-demonstration (LfD) systems to multi-agent scenarios, particularly for tasks that involve shared resources or require synchronized actions. Our approach achieved a 93% teaching success rate (RQ6). These contributions pave the way for future robots that can continuously learn and develop new capabilities, enabling them to adapt to dynamic human environments and evolving challenges.

Table 5.1 provides a summary of the addressed research questions and the corresponding relevant papers.

5.1 Future Work

Personalized Explanations

This thesis laid the foundation for enabling robots to provide failure explanations when needed by humans, which, as literature has shown, is key to building trust and acceptance of robots in human environments. In Paper G, we began exploring interactive human-robot tasks, marking the first step toward collaborative problem-solving scenarios. Moving forward, we aim to expand this research to focus on tasks where explanations are crucial for tackling complex challenges together. This approach emphasizes not only explaining failures but also using those explanations to enhance teamwork between humans and robots.

Another important step in this direction involves enhancing personalization in failure explanations. By integrating high-level task descriptions with low-level robot-specific details, explanations could be tailored to provide either an external or internal perspective, depending on the user's expertise and needs. Inspired by Occam's razor principle, our current approach delivers the simplest explanations and preventive actions, minimizing interval changes without requiring human domain knowledge. While effective, this uninformed BFS approach assigns equal importance to all causally relevant variables, potentially producing multiple equally optimal explanations. One concrete step forward could be incorporating heuristic prioritization into the search process. This would involve assigning higher weights to certain variables based on specific criteria, as outlined in our Workshop Paper J. For example:

- Temporal priority: Variables linked to recent events might be considered more relevant.
- Domain knowledge: Human preferences could guide the prioritization of certain variables.
- Action-based insights: Investigating common causes across tasks (e.g., stacking, pouring) could help identify universally significant variables.
- Ease of manipulation: Variables that are easier to adjust, such as cube positioning compared to cube size, could be given higher priority.

Additionally, we plan to investigate whether failure explanations align with typical human reasoning or if they vary depending on the individual providing or receiving the explanation. If variability exists, we could personalize the explanations by incorporating individual variable rankings into the search process. Ultimately, these enhancements could make explanations more intuitive, human-like, and effective in collaborative human-robot interactions.

Learning Execution Policies
Regarding RQ5 and RQ6, our primary focus has been on learning descriptive models, which offer strengths such as explainability, transfer (because they are robot agnostic) and the ability to reason about missing action. However, an open question remains: how can these models be effectively executed? While this has not been the main focus of this thesis, we have begun exploring one approach in Paper K. Specifically, we investigated how high-level plans derived from descriptive models can guide reinforcement learning (RL) agents to learn the lower-level execution policies for individual operators. Our experiments demonstrated that using high-level plans as guidance enhances sample efficiency for RL agents. For example, in simulated robot manipulation tasks like stacking cubes, our approach reduced training costs by half compared to standard RL baselines and improved generalization to unseen stacking task tasks (e.g., stacking a pyramid or stacking three cubes instead of two). Furthermore, the kinesthetic teaching approach presented in Paper E complements this by enabling the robot to save waypoints from demonstrated trajectories. This allows the robot not only to learn from the demonstrations but also to execute the trajectories directly, bridging the gap between learning and execution. In future work, it would be interesting to explore bi-directional communication between high-level descriptive models and low-level execution policies, where lower-level insights could refine high-level symbolic descriptions and vice versa, ensuring a more cohesive and adaptable learning-execution framework.

References

[1] S. Schaal, "The New Robotics—towards human-centered machines," *HFSP Journal*, vol. 1, no. 2, pp. 115–126, 2007, ISSN: 1955-2068.

[2] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling, "Explainable agents and robots: Results from a systematic literature review," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19, Montreal QC, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1078–1088, ISBN: 9781450363099.

[3] J. T. Jun Kinugawa Hiroki Suzuki and K. Kosuge, "Underactuated robotic hand for a fully automatic dishwasher based on grasp stability analysis*," *Advanced Robotics*, vol. 36, no. 4, pp. 167–181,

[4] E. Broadbent, "Interactions with robots: The truths we reveal about ourselves," *Annual review of psychology*, vol. 68, no. 1, pp. 627–652, 2017.

[5] R. Bloss, "Mobile hospital robots cure numerous logistic needs," *Industrial Robot: An International Journal*, vol. 38, no. 6, pp. 567–571, 2011.

[6] J. Hu et al., "An advanced medical robotic system augmenting healthcare capabilities-robotic nursing assistant," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 6264–6269.

[7] A. Grau, M. Indri, L. L. Bello, and T. Sauter, “Industrial robotics in factory automation: From the early stage to the internet of things,” in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 6159–6164.

[8] J. Guiochet, M. Machin, and H. Waeselynck, “Safety-critical advanced robots: A survey,” *Robotics and Autonomous Systems*, vol. 94, pp. 43–52, 2017, ISSN: 0921-8890.

[9] S. Haddadin, A. De Luca, and A. Albu-Schäffer, “Robot collisions: A survey on detection, isolation, and identification,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[10] S. Honig and T. Oron-Gilad, “Understanding and resolving failures in human-robot interaction: Literature review and model development,” *Frontiers in psychology*, vol. 9, p. 861, 2018.

[11] Z. Liu, A. Bahety, and S. Song, “Reflect: Summarizing robot experiences for failure explanation and correction,” *arXiv preprint arXiv:2306.15724*, 2023.

[12] G. Steinbauer, “A survey about faults of robots used in robocup,” in *RoboCup 2012: Robot Soccer World Cup XVI*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 344–355.

[13] D. J. Brooks, M. Begum, and H. A. Yanco, “Analysis of reactions towards failures and recovery strategies for autonomous robots,” in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016, pp. 487–492.

[14] S. Honig and T. Oron-Gilad, “Expect the unexpected: Leveraging the human-robot ecosystem to handle unexpected robot failures,” *Frontiers in Robotics and AI*, vol. 8, p. 656 385, 2021.

[15] M. Jung and P. Hinds, *Robots in the wild: A time for more robust theories of human-robot interaction*, 2018.

[16] K. Harrison et al., “The imperfectly relatable robot: An interdisciplinary workshop on the role of failure in hri,” in *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI ’23, Stockholm, Sweden: Association for Computing Machinery, 2023, pp. 917–919, ISBN: 9781450399708.

[17] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[18] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. USA: Basic Books, Inc., 2018, ISBN: 046509760X.

[19] G. LeMasurier, A. Gautam, Z. Han, J. W. Crandall, and H. A. Yanco, “Reactive or proactive? how robots should explain failures,” *2024 19th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 413–422, 2024.

[20] T. Hellström, “The relevance of causation in robotics: A review, categorization, and analysis,” *Paladyn, Journal of Behavioral Robotics*, vol. 12, no. 1, pp. 238–255, 2021.

[21] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner, and D. Miller, “Interactive hierarchical task learning from a single demonstration,” in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2015, pp. 205–212.

[22] K. Kawamura, P. Nilas, K. Muguruma, J. Adams, and C. Zhou, “An agent-based architecture for an adaptive human-robot interface,” in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, 2003.

[23] C. Kardos, Z. Kemény, A. Kovács, B. E. Pataki, and J. Váncza, “Context-dependent multimodal communication in human-robot collaboration,” *Procedia CIRP*, vol. 72, pp. 15–20, 2018, 51st CIRP Conference on Manufacturing Systems, ISSN: 2212-8271.

[24] K. Drnec et al., “The role of psychophysiological measures as implicit communication within mixed-initiative teams,” in *Virtual, Augmented and Mixed Reality: Interaction, Navigation, Visualization, Embodiment, and Simulation*, Springer International Publishing, 2018, pp. 299–313.

[25] D. Das, S. Banerjee, and S. Chernova, “Explainable ai for robot failures: Generating explanations that improve user assistance in fault recovery,” *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 351–360, 2021.

[26] S. Rosenthal and M. Veloso, “Mobile robot planning to seek help with spatially-situated tasks,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, pp. 2067–2073, 2021.

[27] R. A. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus, “Recovering from failure by asking for help,” *Autonomous Robots*, vol. 39, pp. 347–362, 2015.

[28] S. Van Waveren, E. J. Carter, O. Örnberg, and I. Leite, “Exploring non-expert robot programming through crowdsourcing,” *Frontiers in Robotics and AI*, vol. 8, p. 646 002, 2021.

[29] S. van Waveren, C. Pek, J. Tumova, and I. Leite, “Correct me if i’m wrong: Using non-experts to repair reinforcement learning policies,” in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2022, pp. 493–501.

[30] A. M. Zanchettin, “Symbolic representation of what robots are taught in one demonstration,” *Robotics and Autonomous Systems*, vol. 166, 2023.

[31] S. van Waveren, “Towards automatically correcting robot behavior using non-expert feedback,” Ph.D. dissertation, KTH Royal Institute of Technology, 2022.

[32] S. Tolmeijer et al., “Taxonomy of trust-relevant failures and mitigation strategies,” in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI ’20, Cambridge, United Kingdom: Association for Computing Machinery, 2020, pp. 3–12, ISBN: 9781450367462.

[33] J.-C. Laprie, “Dependable computing and fault-tolerance,” *Digest of Papers FTCS-15*, vol. 10, no. 2, p. 124, 1985.

[34] G. M. O’Hare, R. Collier, and R. Ross, “Demonstrating social error recovery with agentfactory,” in *AAMAS’04 Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, IEEE, 2004.

[35] M. Ghallab, D. Nau, and P. Traverso, *Automated planning and acting.* Cambridge University Press, 2016.

[36] D. J. Brooks, “A human-centric approach to autonomous robot failures,” Ph.D. dissertation, University of Massachusetts Lowell, 2017.

[37] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance.* Springer Science & Business Media, 2006.

[38] E. Khalastchi and M. Kalech, "On fault detection and diagnosis in robotic systems," *ACM Comput. Surv.*, vol. 51, no. 1, 2018, ISSN: 0360-0300.

[39] V. Verma, G. Gordon, R. Simmons, and S. Thrun, "Real-time fault diagnosis [robot fault diagnosis]," *IEEE Robotics & Automation Magazine*, vol. 11, no. 2, pp. 56–66, 2004.

[40] Z. Zhang and J. Chen, "Fault detection and diagnosis based on particle filters combined with interactive multiple-model estimation in dynamic process systems," *ISA transactions*, vol. 85, pp. 247–261, 2019.

[41] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, pp. 85–126, 2004.

[42] M. M. Quamar and A. Nasir, *Review on fault diagnosis and fault-tolerant control scheme for robotic manipulators: Recent advances in ai, machine learning, and digital twin*, 2024.

[43] I. Raouf, P. Kumar, H. Lee, and H. S. Kim, "Transfer learning-based intelligent fault detection approach for the industrial robotic system," *Mathematics*, vol. 11, no. 4, 2023, ISSN: 2227-7390.

[44] R. Ueda, Y. Kakiuchi, S. Nozawa, K. Okada, and M. Inaba, "Anytime error recovery by integrating local and global feedback with monitoring task states," in *2011 15th International Conference on Advanced Robotics (ICAR)*, IEEE, 2011, pp. 298–303.

[45] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 5678–5684.

[46] R. Ghzouli, T. Berger, E. B. Johnsen, A. Wasowski, and S. Dragule, "Behavior trees and state machines in robotics applications," *IEEE Transactions on Software Engineering*, vol. 49, no. 9, pp. 4243–4267, 2023.

[47] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and ai," *Robotics and Autonomous Systems*, vol. 154, p. 104 096, 2022.

[48] J. Styrud, M. Iovino, M. Norrlöf, M. Björkman, and C. Smith, "Combining planning and learning of behavior trees for robotic assembly," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 11 511–11 517.

[49] S. Elfwing and B. Seymour, "Parallel reward and punishment control in humans and robots: Safe reinforcement learning using the maxpain algorithm," in *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2017, pp. 140–147.

[50] A. Inceoglu, E. E. Aksoy, and S. Sariel, "Multimodal detection and classification of robot manipulation failures," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1396–1403, 2024.

[51] S. A. Sloman and D. Lagnado, "Causality in thought," *Annual review of psychology*, vol. 66, no. 1, pp. 223–247, 2015.

[52] B. M. Rottman, D. Gentner, and M. B. Goldwater, "Causal systems categories: Differences in novice and expert categorization of causal phenomena," *Cognitive science*, vol. 36, no. 5, pp. 919–932, 2012.

[53] D. Gunning and D. W. Aha, "Darpa's explainable artificial intelligence program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[54] L. Longo et al., "Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions," *Information Fusion*, vol. 106, p. 102 301, 2024, ISSN: 1566-2535.

[55] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019.

[56] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, 2017, pp. 8–13.

[57] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu, "Interpreting cnns via decision trees," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6261–6270.

[58] M. T. Ribeiro, S. Singh, and C. Guestrin, “" why should i trust you?" explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[59] M. Lázaro-Gredilla, D. Lin, J. S. Guntupalli, and D. George, “Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs,” *Science Robotics*, vol. 4, no. 26, eaav3150, 2019.

[60] A. Niculescu-Mizil and R. Caruana, “Inductive transfer for bayesian network structure learning,” in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, M. Meila and X. Shen, Eds., ser. Proceedings of Machine Learning Research, vol. 2, San Juan, Puerto Rico: PMLR, 2007.

[61] V. Rodríguez-López and L. E. Sucar, “Knowledge transfer for causal discovery,” *International Journal of Approximate Reasoning*, vol. 143, 2022, ISSN: 0888-613X.

[62] Y. Hou et al., “Bearing fault diagnosis under small data set condition: A bayesian network method with transfer learning for parameter estimation,” *IEEE Access*, vol. 10, 2022.

[63] Y. Zhou, T. M. Hospedales, and N. Fenton, “When and where to transfer for bayesian network parameter learning,” *Expert Systems with Applications*, vol. 55, pp. 361–373, 2016, ISSN: 0957-4174.

[64] A. S. Bauer, P. Schmaus, F. Stulp, and D. Leidner, “Probabilistic effect prediction through semantic augmentation and physical simulation,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9278–9284, 2020.

[65] A. Mitrevsk, P. G. Plöger, and G. Lakemeyer, “Ontology-assisted generalisation of robot action execution knowledge,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[66] K. Xu, E. Ratner, A. Dragan, S. Levine, and C. Finn, “Learning a prior over intent via meta-inverse reinforcement learning,” in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[67] K. Bousmalis et al., “Using simulation and domain adaptation to improve efficiency of deep robotic grasping,” in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018.

[68] M. R. Pedersen et al., “Robot skills for manufacturing: From concept to industrial deployment,” *Robotics and Computer-Integrated Manufacturing*, vol. 37, 2016.

[69] D. Weintrop, D. C. Shepherd, P. Francis, and D. Franklin, “Blockly goes to work: Block-based programming for industrial robots,” in *2017 IEEE Blocks and Beyond Workshop (B&B)*, IEEE, 2017.

[70] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Robot programming by demonstration,” in *Springer handbook of robotics*, Springer, 2008.

[71] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annual review of control, robotics, and autonomous systems*, vol. 3, no. 1, pp. 297–330, 2020.

[72] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, “From skills to symbols: Learning symbolic representations for abstract high-level planning,” *J. Artif. Int. Res.*, vol. 61, no. 1, pp. 215–289, 2018, ISSN: 1076-9757.

[73] M. Stenmark, M. Haage, and E. A. Topp, “Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation,” in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017.

[74] E. A. Topp, M. Stenmark, A. Ganslandt, A. Svensson, M. Haage, and J. Malec, “Ontology-based knowledge representation for increased skill reusability in industrial robots,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018.

[75] M. Mayr, F. Rovida, and V. Krueger, “Skiros2: A skill-based robot control platform for ros,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023.

[76] Z. Wang, Y. Gan, and X. Dai, “Assembly-oriented task sequence planning for a dual-arm robot,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, 2022.

[77] H. H. Zhuo, H. Muñoz-Avila, and Q. Yang, "Learning action models for multi-agent planning," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 2011.

[78] A. Mordoch, D. Portnoy, R. Stern, and B. Juba, "Collaborative multi-agent planning with black-box agents by learning action models," in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2022.

[79] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, e253, 2017.

[80] G. Puebla and S. E. Chaigneau, "Inference and coherence in causal-based artifact categorization," *Cognition*, vol. 130, no. 1, pp. 50–65, 2014.

[81] J. Kaddour, A. Lynch, Q. Liu, M. J. Kusner, and R. Silva, "Causal machine learning: A survey and open problems," *arXiv preprint arXiv:2206.15475*, 2022.

[82] B. Schölkopf et al., "Toward causal representation learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 612–634, 2021.

[83] K. C. Stocking, A. Gopnik, and C. Tomlin, "From robot learning to robot understanding: Leveraging causal graphical models for robotics," *Proceedings of the 5th Conference on Robot Learning*, vol. 164, pp. 1776–1781, 2022.

[84] T. E. Lee, J. A. Zhao, A. S. Sawhney, S. Girdhar, and O. Kroemer, "Causal reasoning in simulation for structure and transfer learning of robot manipulation policies," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4776–4782, 2021.

[85] T. E. Lee, S. Vats, S. Girdhar, and O. Kroemer, "Scale: Causal learning and discovery of robot manipulation skills using simulation," 2023.

[86] R. Cannizzaro and L. Kunze, "Car-despot: Causally-informed online pomdp planning for robots in confounded environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 2018–2025.

[87] R. Cannizzaro, M. Groom, J. Routley, R. O. Ness, and L. Kunze, "A causal bayesian network and probabilistic programming based reasoning framework for robot manipulation under uncertainty," *arXiv preprint arXiv:2403.14488*, 2024.

[88] O. Ahmed et al., "Causalworld: A robotic manipulation benchmark for causal structure and transfer learning," *arXiv preprint arXiv:2010.04296*, 2020.

[89] A. A. Bhat, V. Mohan, G. Sandini, and P. G. Morasso, "Humanoid infers archimedes' principle: Understanding physical relations and object affordances through cumulative learning experiences," *Journal of the Royal Society Interface*, vol. 13, no. 120, 2016.

[90] J. Brawer, M. Qin, and B. Scassellati, "A causal approach to tool affordance learning," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8394–8399, 2020.

[91] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning task constraints for robot grasping using graphical models," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1579–1585, 2010.

[92] C. Uhde, N. Berberich, K. Ramirez-Amaro, and G. Cheng, "The robot as scientist: Using mental simulation to test causal hypotheses extracted from human activities in virtual reality," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8081–8086, 2020.

[93] J. Cheong, N. Churamani, L. Guerdan, T. E. Lee, Z. Han, and H. Gunes, "Causal-hri: Causal learning for human-robot interaction," in *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '24, Boulder, CO, USA: Association for Computing Machinery, 2024, pp. 1311–1313, ISBN: 9798400703232.

[94] L. Castri, S. Mghames, M. Hanheide, and N. Bellotto, "Causal discovery of dynamic models for predicting human spatial interactions," in *Social Robotics*, F. Cavallo et al., Eds., Cham: Springer Nature Switzerland, 2022, pp. 154–164, ISBN: 978-3-031-24667-8.

[95] F. Edström, T. Hellström, and X. De Luna, "Robot causal discovery aided by human interaction," in *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2023.

[96] M. Scutari, "Learning bayesian networks with the bnlearn R package," *Journal of Statistical Software*, vol. 35, no. 3, pp. 1–22, 2010.

[97] Q. Zhang, N. Tsoi, B. Choi, J. Tan, H.-T. L. Chiang, and M. Vázquez, *SEAN Together Dataset*, `https://sean-together.interactive-machines.com/`, [Online; accessed 2024], 2024.

[98] E. T. Hall, *The hidden dimension.* Anchor, 1966, vol. 609.

[99] Y.-C. Chen, T. A. Wheeler, and M. J. Kochenderfer, "Learning discrete bayesian networks from continuous data," *Journal of Artificial Intelligence Research*, vol. 59, no. 1, pp. 103–132, 2017, ISSN: 1076-9757.

[100] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[101] J. L. Lustgarten, S. Visweswaran, V. Gopalakrishnan, and G. F. Cooper, "Application of an efficient bayesian discretization method to biomedical data," *BMC bioinformatics*, vol. 12, pp. 1–15, 2011.

[102] D. Margaritis, "Learning bayesian network model structure from data," Ph.D. dissertation, Carnegie Mellon University, 2003.

[103] M. J. Vowels, N. C. Camgoz, and R. Bowden, "D'ya like dags? a survey on structure learning and causal discovery," *ACM Computing Surveys*, 2022.

[104] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[105] A. Sharma, V. Syrgkanis, C. Zhang, and E. Kiciman, "Dowhy: Addressing challenges in expressing and validating causal assumptions," *ICMAL Workshop: The Neglected Assumptions In Causal Inference*, 2021.

[106] J. Peters, D. Janzing, and B. Schlkopf, *Elements of Causal Inference: Foundations and Learning Algorithms.* The MIT Press, 2017, ISBN: 0262037319.

[107] Z. Ji, Q. Xia, and G. Meng, "A review of parameter learning methods in bayesian network," in *Advanced Intelligent Computing Theories and Applications*, Springer International Publishing, 2015.

[108] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009, ISBN: 0262013193.

[109] D. Das and S. Chernova, "Semantic-based explainable ai: Leveraging semantic scene graphs and pairwise ranking to explain robot failures," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3034–3041, 2021.

[110] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5831–5840, 2018.

[111] D. Altan and S. Sariel, "Probabilistic failure isolation for cognitive robots," in *FLAIRS Conference*, 2014.

[112] Q. Zhang, N. Tsoi, B. Choi, J. Tan, H.-T. L. Chiang, and M. Vázquez, "Towards inferring users' impressions of robot performance in navigation scenarios," *arXiv preprint arXiv:2310.11590*, 2023.

[113] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017, Special Issue on AI and Robotics, ISSN: 0004-3702.

[114] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3, pp. 189–208, 1971.

[115] Jonsson, Morris, Muscettola, Rajan, and Smith, "Planning in interplanetary space: Theory and practice," in *International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, Breckenridge, CO, 2000.

[116] F. Ingrand, S. Lacroix, S. Lemai-Chenevier, and F. Py, "Decisional autonomy of planetary rovers," *Journal of Field Robotics*, vol. 24, 2007.

[117] P. Doherty, J. Kvarnström, and F. Heintz, "A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems," *Autonomous Agents and Multi-Agent Systems*, vol. 19, pp. 332–377, 2009.

[118] E. Erős, M. Dahl, A. Hanna, P.-L. Götvall, P. Falkman, and K. Bengtsson, "Development of an industry 4.0 demonstrator using sequence planner and ros2," in *Robot Operating System (ROS): The Complete Reference (Volume 5)*, A. Koubaa, Ed. Cham: Springer International Publishing, 2021, pp. 3–29, ISBN: 978-3-030-45956-7.

[119] M. Cashmore et al., "Rosplan: Planning in the robot operating system," in *ICAPS*, Jerusalem, Israel: AAAI Press, 2015, pp. 333–341, ISBN: 9781577357315.

[120] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 4th ed. Pearson, 2020.

[121] H. Geffner and B. Bonet, "A concise introduction to models and methods for automated planning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2013.

[122] D. McDermott et al., "PDDL – The Planning Domain Definition Language," New Haven, CT: Yale Center for Computational Vision and Control, Tech. Rep. CVC TR98003/DCS TR1165, 1998.

[123] J. Zhang, E. Dean, and K. Ramirez-Amaro, "Hierarchical reinforcement learning based on planning operators *," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 2006–2012.

[124] S. Alouneh, S. Abed, M. H. Al Shayeji, and R. Mesleh, "A comprehensive study and analysis on sat-solvers: Advances, usages and achievements," *Artificial Intelligence Review*, vol. 52, pp. 2575–2601, 2019.

[125] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, 2006.

[126] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A Survey of Robot Learning from Demonstration," *Robotics and Autonomous Systems*, 2009.

[127] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring Skills to Humanoid Robots by Extracting Semantic Representations from Observations of Human Activities," *Artificial Intelligence*, 2017.

[128] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming Up with Good Excuses: What To Do When No Plan Can be Found," in *ICAPS*, 2010.

[129] A. Herzig, M. V. de Menezes, L. N. De Barros, and R. Wassermann, “On the Revision of Planning Tasks,” in *ECAI*, 2014.

[130] S. Sreedharan, T. Chakraborti, C. Muise, Y. Khazaeni, and S. Kambhampati, “D3WA+ – A Case Study of XAIP in a Model Acquisition Task for Dialogue Planning,” in *ICAPS*, 2020.

[131] S. Grover, S. Sengupta, T. Chakraborti, A. P. Mishra, and S. Kambhampati, “RADAR: Automated Task Planning for Proactive Decision Support,” in *HCI Journal*, 2020.

[132] S. Grover, T. Chakraborti, and S. Kambhampati, “What Can Automated Planning do for Intelligent Tutoring Systems?” In *ICAPS Scheduling and Planning Applications Workshop*, 2018.

[133] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati, “Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy,” *Artificial Intelligence*, 2021.

[134] T. Wisspeintner, T. Van Der Zant, L. Iocchi, and S. Schiffer, “RoboCup@Home: Scientific Competition and Benchmarking for Domestic Service Robots,” *Interaction Studies*, 2009.

[135] D. L. Kovacs, “A multi-agent extension of pddl3.,” *WS-IPC 2012*, 2012.

[136] D. Furelos-Blanco and A. Jonsson, “Solving multiagent planning problems with concurrent conditional effects,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019.