



Finanziato dall'Unione Europea
NextGenerationEU

"Progetto Finanziato dall'Unione europea – NextGenerationEU"



**Università
di Brescia**

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

settore scientifico disciplinare
ING-INF-05

CICLO
XXXVIII

TITOLO TESI

Fast and Robust Goal Recognition through Deep Learning

Lorenzo Serina
Matricola:
715995

Primo SUPERVISORE
Alfonso Emilio Gerevini, Università degli Studi di Brescia
Secondo SUPERVISORE
Ivan Serina, Università degli Studi di Brescia

Abstract

Questa tesi si inserisce nel campo del Goal Recognition (GR) sostituendo l'utilizzo di pianificatori computazionalmente intensivi con metodi di deep learning efficienti, accurati e robusti nei confronti dei comportamenti avversari.

In primo luogo, stabiliamo una nuova connessione tra la pianificazione automatica e l'elaborazione del linguaggio naturale introducendo PlanBERT, un modello che tratta le sequenze di azioni come strutture linguistiche. La nostra architettura dimostra di essere capace di ricostruire sequenze di planning e di svolgere il task di Goal Recognition Offline, validando che le architetture Transformer possono apprendere efficacemente le dinamiche dei domini di pianificazione.

Per il Goal Recognition Online (OGR), dove le previsioni devono essere aggiornate incrementalmente all'arrivo delle osservazioni, adattiamo GRNet, approccio allo stato dell'arte basato su deep learning, come solida baseline neurale e introduciamo CLERNet (Causal Link Enhanced Recurrent Network), un'architettura neuro-simbolica che integra la conoscenza simbolica di pianificazione attraverso strategie di etichettatura basate sui causal link. Questa architettura dimostra capacità superiori rispetto allo stato dell'arte, sia in velocità di predizione che in accuratezza.

Infine, affrontiamo scenari avversari formalizzando attacchi di inganno e interferenza che degradano le prestazioni dei sistemi OGR. Proponiamo PAC-OGR, un framework di difesa che combina la pre-elaborazione simbolica con l'analisi euristica, migliorando significativamente la robustezza contro la manipolazione delle osservazioni e preservando le capacità di inferenza in tempo reale.

Abstract

This thesis contributes to the field of Goal Recognition (GR) by replacing computationally intensive planners with deep learning methods that are efficient, accurate, and robust to adversarial behaviours.

First, we establish a novel connection between automated planning and natural language processing by introducing PlanBERT, a model that treats action sequences as linguistic structures. Our architecture demonstrates the ability to reconstruct planning sequences and perform Offline Goal Recognition, confirming that Transformer-based models can effectively learn the dynamics of planning domains.

For Online Goal Recognition (OGR), where predictions must be updated incrementally as observations arrive, we adapt GRNet, state-of-the-art deep learning-based approach, as a strong neural baseline and introduce CLERNet (Causal Link Enhanced Recurrent Network), a neuro-symbolic architecture that integrates symbolic planning knowledge through Causal Link-based labelling strategies. This architecture achieves state-of-the-art performance in both prediction speed and accuracy.

Finally, we address adversarial scenarios by formalising deception and interference attacks that degrade the performance of OGR systems. We propose PAC-OGR, a defence framework that combines symbolic pre-processing with heuristic trend analysis, significantly improving robustness against observation manipulation while preserving real-time inference capabilities.

Contents

1	Introduction	1
1.1	Planning Language Modelling and Offline Goal Recognition through Transformers	3
1.2	Online Goal Recognition with Neuro-Symbolic Architectures . .	3
1.3	Adversarial Robustness in Online Goal Recognition	4
1.4	Thesis Organisation	4
2	Background	6
2.1	Automated Planning	6
2.1.1	Classical Planning	7
2.1.2	Goal and Plan Recognition	12
2.1.3	Deceptive Planning	18
2.2	Deep Learning	24
2.2.1	Learning via Gradient Descent	24
2.2.2	Word Representations	25
2.2.3	Recurrent Neural Networks	28
2.2.4	Attention Mechanisms	32
2.2.5	Transformer	37
2.2.6	Large Language Models for Planning	42

3	State of the Art	43
4	Transformer Based Architectures for Offline Goal Recognition	51
4.1	PlanBERT: BERT for Automated Planning	53
4.1.1	Planning Language Modelling	53
4.1.2	Goal Recognition Fine-tuning	58
4.1.3	Goal Recognition Inference	59
4.2	Discussion	60
5	Online Goal Recognition through Deep Learning Approaches	61
5.1	GRNET for Online Inference	62
5.2	CLERNet (Causal Link Enhanced Recurrent Network)	63
5.2.1	Architecture Overview	64
5.2.2	Training via Causal Link Labelling	67
5.3	Discussion	74
6	Deceptive Online Goal Recognition	75
6.1	Adversarial Scenarios: Attacks on OGR	76
6.1.1	Interference	77
6.1.2	Deception	78
6.2	Enhancing OGR Systems: PAC-OGR	80
6.2.1	Pre-Processing	80
6.2.2	Post-Processing	81
6.3	Discussion	83
7	Experimental Results	84
7.1	Experimental Evaluation of PlanBERT	84
7.1.1	Benchmark Suite and Datasets	84
7.1.2	Results and Analysis	88

7.1.3	Robustness to Noisy Observations	92
7.1.4	Discussion and Limitations	98
7.2	Experimental Evaluation of CLERNet	100
7.2.1	Experimental Setup	100
7.2.2	Results and Analysis	102
7.2.3	Discussion and Limitations	113
7.3	Experimental Evaluation of PAC-OGR	114
7.3.1	Experimental Setup	115
7.3.2	Discussion and Limitations	123
8	Conclusion	126
8.1	Summary of Contributions	126
8.2	Limitations and Challenges	128
8.3	Future Research Directions	130

Chapter 1

Introduction

Goal Recognition (GR), the task of inferring an observed agent’s intent from a sequence of actions [43, 75, 52], is a critical capability for intelligent systems across a wide range of domains. In cybersecurity, GR enables intrusion detection systems to identify malicious intent from sequences of observed network actions, distinguishing advanced persistent threats from benign operations, and to trigger proactive countermeasures before critical systems are compromised [78]. In human–robot collaboration, robots that accurately infer human goals can proactively hand over tools, adapt their navigation trajectories, and provide timely assistance in manufacturing, healthcare, and domestic environments [78, 19]. In ambient assisted living, smart-home systems recognise activities of daily living to monitor elderly residents and detect safety-critical anomalies [78]. These applications share a common, stringent requirement: *real-time recognition* within milliseconds to seconds, a constraint that fundamentally shapes the design of any practical GR system.

Traditionally, this problem has been framed as *inverse planning* [63, 64, 8], relying on symbolic planners to validate hypothesis sets by computing optimal

plan costs for each candidate goal. While formally robust, these approaches are often computationally prohibitive for real-time applications: symbolic planning is inherently NP-hard to PSPACE-complete [15], and goal recognition requires multiple planner invocations per observation, one for each candidate goal, leading to recognition latencies of seconds to minutes in domains of realistic complexity [69, 86, 77].

This thesis advances beyond existing approaches along three complementary dimensions. First, while prior deep learning methods for GR treat observed action sequences as plain token sequences [19, 2], we exploit the structural analogy between plan traces and natural language to apply Transformer-based architectures [79, 21], modelling the “grammar” of planning domains through *Planning Language Modelling*. Second, whereas existing neuro-symbolic approaches to GR combine learning and planning knowledge post-hoc [5], we introduce CLERNET, which integrates symbolic causal-link relationships from partial-order planning theory directly into the neural network training process, yielding an architecture that is both data-efficient and structurally informed. Third, we address a critical gap in the literature by formalising and analysing adversarial robustness in on-line GR, proposing the first defence framework against deception and interference attacks in security-sensitive applications. Taken together, these contributions demonstrate that the integration of symbolic planning knowledge into deep learning architectures produces recognition systems that are simultaneously fast, accurate, and robust—advancing the state of the art at the intersection of automated planning, deep learning, and agentic AI [68, 69].

1.1 Planning Language Modelling and Offline Goal Recognition through Transformers

Our investigation begins by exploring the structural parallels between natural language sentences and plan traces. Just as Large Language Models (LLMs) capture the syntax and semantics of text, we posit that Transformer-based architectures can learn the "grammar" of planning domains. We introduce the concept of *Planning Language Modelling* (PLM), inspired by Masked Language Modelling (MLM), to train our BERT-based architecture (Chapter 4). Then, using this architecture as a foundational model, we fine-tune it to solve Offline Goal Recognition tasks. Our experiments demonstrate that the bidirectional attention mechanism of BERT can effectively capture long-range dependencies and infer agent goals from completed traces with high accuracy, treating plans as a language to be understood.

1.2 Online Goal Recognition with Neuro-Symbolic Architectures

While offline recognition provides a strong theoretical foundation, real-world scenarios often require *Online* Goal Recognition (OGR), where predictions must be updated incrementally as observations arrive. To address this, we first investigate the adaptability of existing deep learning models by porting GRNET, originally designed for offline tasks in [19], to the online setting. Our extensive testing reveals that while the adapted GRNET establishes a strong learning-based baseline, it lacks specific mechanisms to handle the streaming nature of OGR efficiently.

Building on these insights, we introduce CLERNet (Causal Link Enhanced Recurrent Network), a novel architecture specifically designed for OGR (Chapter 5). CLERNet integrates symbolic planning knowledge directly into the training loop via *Causal Link* labelling strategies. This neuro-symbolic approach allows the system to map observed actions to the specific goal fluents they support, significantly outperforming both the symbolic state-of-the-art OGR architectures and the online adaptation of GRNet in terms of convergence speed and recognition accuracy.

1.3 Adversarial Robustness in Online Goal Recognition

Standard OGR systems operate under the assumption of a rational, truthful agent. However, security-sensitive applications require resilience against active manipulation. This thesis extends the OGR framework to adversarial settings, analysing robustness against *Deception* (where agents obfuscate their goals) and *Interference* (where observations are tampered with) (Chapter 6). We propose **PAC-OGR**, a defence framework that combines symbolic pre-processing with heuristic trend analysis to detect and mitigate these attacks, ensuring reliable recognition even in the presence of malicious actors.

1.4 Thesis Organisation

The remainder of this thesis is organised as follows:

Chapter 2 provides background on automated planning, goal recognition, and deep learning techniques that form the foundation of this work.

Chapter 3 provides an overview of the current state of the art in goal recognition.

Chapter 4 introduces PlanBERT, our Transformer-based approach to offline goal recognition using Planning Language Modelling. We experimentally demonstrate that BERT can learn planning domain dynamics and achieve competitive performance on goal recognition tasks.

Chapter 5 presents CLERNet, a neuro-symbolic architecture for Online Goal Recognition that integrates causal link knowledge from planning theory. We show that this approach significantly outperforms both purely symbolic and purely neural baselines.

Chapter 6 addresses adversarial scenarios in goal recognition, formalising attacks and proposing the PAC-OGR defence framework. We detected improved robustness against both deception and interference attacks.

Chapter 7 presents comprehensive experimental evaluations of all proposed approaches across multiple planning domains and metrics.

Chapter 8 summarises the contributions of this thesis, discusses limitations, and outlines promising directions for future research.

Chapter 2

Background

This thesis bridges symbolic and neural paradigms by combining model-based planning with learning-based approaches. Automated planning provides the formal framework for representing agent behaviour through domain theories, while deep learning enables efficient pattern recognition from data. This chapter provides the necessary background in both areas.

The first section covers **Automated Planning**, introducing classical planning formalisms, goal recognition theory, state-of-the-art approaches, and deceptive planning strategies. The second section covers **Deep Learning**, presenting the neural network architectures and attention mechanisms that enable our learning-based goal recognition systems.

2.1 Automated Planning

A core challenge in Artificial Intelligence (AI) is enabling intelligent behaviour, i.e., the ability of an autonomous agent to use knowledge of the environment to decide what to do in previously unseen situations. In this context, we focus on

the action selection problem, which consists of determining the next action an agent should perform. This task is known as *Automated Planning*. The literature identifies three general strategies to address this problem:

- **Programming-based strategy:** the behaviour is explicitly defined by the programmer through rules or procedural code.
- **Model-based strategy:** the behaviour is computed automatically using a model that describes actions, states and objectives.
- **Learning-based strategy:** the behaviour is inferred from data and experience, typically using techniques such as machine learning, deep learning, or reinforcement learning.

This thesis focusses on a specific subclass of automated planning known as Classical Planning.

2.1.1 Classical Planning

In this setting, the environment is deterministic and fully observable, and both the initial state and the goals are completely specified.

A classical planning problem is defined by the tuple

$$\mathcal{S} = \langle S, s_0, S_G, A, f, c \rangle \quad (2.1)$$

where:

- S is a finite set of discrete states encoding all possible configurations of the environment;
- $s_0 \in S$ is the initial state;

- $S_G \subseteq S$ is a non-empty set of goal states;
- $A(s) \subseteq A$ is the set of actions applicable in state s ;
- $f(a, s)$ is a deterministic transition function, yielding $s' = f(a, s)$, with $s \in S$ and $a \in A(s)$;
- $c(a, s)$ is a positive execution cost for action a in state s .

A plan π is a sequence of actions $\pi = a_0, \dots, a_n$ generating a state trajectory $s_0, \dots, s_{n+1}$ such that each action is applicable and $s_{n+1} \in S_G$. The cost of π is:

$$\text{cost}(\pi) = \sum_{i=0}^n c(a_i, s_i) \quad (2.2)$$

That is, the sum of the action costs $c(a_i, s_i)$ with $i \in [0, n]$. A solution plan is optimal if it has the minimum cost among all possible solution plans. A common cost structure consists of setting all action costs $c(a, s)$ to 1. In this case, the cost of the plan is given by its length, and the optimal plans are the shortest ones. In this thesis, we will rely on this setting.

Classical planning models can be succinctly expressed using a language that employs boolean state variables [32]. The most basic language for classical planning is STRIPS [24], which is structured around boolean variables. In STRIPS, the boolean variables that define a state are referred to as *facts*, *fluents*, or *atoms*. A planning problem in STRIPS is represented by a tuple $P = \langle F, I, A, G \rangle$ where:

- F represents the set of atoms or propositions of interest;
- $I \subseteq F$ represents the initial state;
- A represents the set of actions;
- $G \subseteq F$ represents the goal.

```

(define (domain blocksworld)
  (:requirements :strips :equality)
  (:predicates (clear ?x) (on-table ?x) (arm-empty) (holding ?x) (on ?x ?y))
  (:action pick-up
    :parameters (?ob)
    :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
    :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
                 (not (arm-empty))))
  (:action put-down
    :parameters (?ob)
    :precondition (and (holding ?ob))
    :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
                 (not (holding ?ob))))
  (:action stack
    :parameters (?ob ?underob)
    :precondition (and (clear ?underob) (holding ?ob))
    :effect (and (arm-empty) (clear ?ob) (on ?ob ?underob)
                 (not (clear ?underob)) (not (holding ?ob))))
  (:action unstack
    :parameters (?ob ?underob)
    :precondition (and (on ?ob ?underob) (clear ?ob) (arm-empty))
    :effect (and (holding ?ob) (clear ?underob)
                 (not (on ?ob ?underob)) (not (clear ?ob)) (not (arm-empty))))
  )
(define (problem pb2)
  (:domain blocksworld)
  (:objects block_a block_b)
  (:init (on-table block_a) (on-table block_b) (clear block_a)
         (clear block_b) (arm-empty))
  (:goal (and (on block_a block_b)))
  )

```

Figure 2.1: An instance of the BLOCKSWORLD domain expressed in PDDL

In STRIPS, each action $a \in A$ is formally represented as a triple

$$a = \langle Pre(a), Add(a), Del(a) \rangle, \quad (2.3)$$

where $Pre(a)$, $Add(a)$, and $Del(a)$ are subsets of the fluent set F :

- **Preconditions** $Pre(a)$: the set of fluents that must hold in the current state s for action a to be *applicable*:

$$a \text{ is applicable in } s \iff Pre(a) \subseteq s.$$

- **Add effects** $Add(a)$: the set of fluents that become true after the execution of a .
- **Delete effects** $Del(a)$: the set of fluents that become false after the execution of a .

The successor state s' resulting from applying a in state s is therefore:

$$s' = (s \setminus Del(a)) \cup Add(a). \quad (2.4)$$

The Planning Domain Definition Language (PDDL) [50] builds upon the STRIPS language by incorporating several new syntactic elements, such as the inclusion of negative literals in preconditions and goals. As represented in Figure 2.1, we outline the widely recognised BLOCKSWORLD domain, where the agent's objective is to construct one or more stacks of blocks, with the constraint that only one block can be moved at a time, all articulated in PDDL. We will rely on this domain throughout all the examples presented in this thesis. In Figure 2.1, PDDL problems are articulated in two segments within a “define” clause:

- the **domain** section: this represents the abstract domain. Here, actions are described using generic atoms, which are composed of predicate names, variables, and constants;
- the **problem** section: this details a specific instance within the domain. In this section, the names of objects that will substitute the variables are listed, alongside the atoms that describe the initial state and a formula that specifies the goal state.

As illustrated in Figure 2.1, in the BLOCKSWORLD domain the **predicates** are `clear`, `on-table`, `holding`, `arm-empty`, and `on`. In PDDL, predicates define the *types of relations and properties* over domain objects, and their grounded instances (obtained by substituting variables with concrete objects) constitute the fluents of the planning problem. The predicates `clear`, `on-table`, and `holding` represent unary properties of blocks (expressed through the variable `?x`); `arm-empty` is a zero-arity property of the environment, while `on` expresses a binary relation between two objects.

Following the definition of predicates, we define four **action schemas**: `Pick-Up`, `Put-Down`, `Stack`, and `Unstack`. Unlike predicates, action schemas define *state transitions*: each schema specifies parameters, preconditions, and effects, as formalised above. `Pick-Up` lifts block `?ob` from the table; `Put-Down` places it back. `Stack` places `?ob` on top of `?underob`; `Unstack` performs the reverse operation.

In practice, a planner does not operate directly on action schemas but on their **grounded** (instantiated) counterparts. *Grounding* is the process of replacing all schema variables with concrete objects from the problem instance. Formally, let $a(x)$ be an action schema with parameters $x = (x_1, \dots, x_k)$ and let $\mathcal{O}$ be the set of domain objects. Grounding generates all actions $a(o)$ where $o \in \mathcal{O}^k$ satisfies the relevant type constraints. The resulting set of grounded actions A_g constitutes the finite action space used by the planner during search.

Mutex Relations and Causal Links

Two additional concepts from planning theory are central to the architectures developed in this thesis: *mutex relations* and *causal links*.

Mutex Relations. A **mutex** (mutual exclusion) relation between two fluents p and q indicates that no reachable state of the planning problem can satisfy both simultaneously. Formally, p and q are *mutex* if there exists no state s reachable from the initial state I such that $\{p, q\} \subseteq s$. In other words, any valid execution of the domain guarantees that p and q are never both true at the same time.

In the context of this thesis, mutex relations are pre-computed offline from the domain model and stored as a static lookup table $\mathcal{R}$, where $\mathcal{R}(f)$ denotes the set of fluents mutually exclusive with fluent f . This information is then exploited by the Aggregation Component of CLERNET (Chapter 5) at inference time: when a fluent f is predicted with high confidence, all fluents in $\mathcal{R}(f)$ are immediately eliminated as goal candidates, since no valid plan can achieve both f and any fluent mutually exclusive with it.

Causal Links. A **causal link** is a triple $\langle a_i, p, a_j \rangle$ drawn from Partial-Order Causal Link (POCL) planning [9], indicating that action a_i establishes fluent p as a precondition of action a_j . Formally:

$$\langle a_i, p, a_j \rangle \iff p \in \text{Add}(a_i) \cap \text{Pre}(a_j).$$

A *threat* to a causal link $\langle a_i, p, a_j \rangle$ arises from any action a_k whose delete effects include p and which is not constrained to execute before a_i or after a_j ; POCL planners resolve threats by introducing ordering constraints.

2.1.2 Goal and Plan Recognition

Goal recognition, along with its broader counterpart, plan recognition, is a long-standing challenge in artificial intelligence [27]. These tasks are linked through their common emphasis on identifying patterns within sequences of actions and

are widely acknowledged as fundamental elements of human cognition. The ability to discern patterns in others' actions is essential for handling daily activities, such as driving, conversing, or engaging in sports. Given that these terms are sometimes used interchangeably in the literature, we will offer an informal definition based on [78] and will refer to these areas of study using the defined terms throughout this thesis.

Goal recognition (GR) is the problem of identifying the intention (goal) of an agent from observations about the agent behaviour in an environment. These observations can be represented in different ways, such as an ordered sequence of images, each representing a state, or an ordered sequence of actions identified as text or indices. The agent's goal can be expressed either as a set of propositions or as a probability distribution over alternative sets of propositions (each forming a distinct candidate goal).

Plan recognition is more general than GR and concerns both recognising the goal of an agent and identifying the full ordered set of actions (plan) that have been or will be performed by the agent in order to reach that goal; like GR, typically, plan recognition takes as input a set of observed actions performed by the agent [16].

Goal Recognition Setting

In this thesis, we will refer to the approach to GR known as “goal recognition over a domain theory” [64, 67, 73, 78]. In this setting, the available knowledge consists of an underlying model of the behaviour of the agent and its environment. Such a model represents the agent/environment states and the set of actions A that the agent can perform; typically, it is specified by a planning language such as PDDL, as described in subsection 2.1.1.

An **instance of the GR problem** $T = (\Pi, I, O, \mathcal{G})$ in a given domain is then

specified by:

- $\Pi = \langle F, A \rangle$ is a planning domain where F specifies the set of possible fluents and A represents the set of actions
- an initial state I of the agent ($I \subseteq F$), where F consists of all the possible state fluents of the considered domain;
- a sequence $O = \langle o_1, \dots, o_n \rangle$ of observations ($n \geq 1$), where each o_i is an action in A performed by the agent;
- and a set $\mathcal{G} = \{G_1, \dots, G_m\}$ ($m > 1$) of possible goals of the agent, where each G_i is a set of fluents over F that represents a partial state.

More precisely, a **partial state** is a set of fluents $X \subseteq F$ that specifies conditions on the world without fully describing it: fluents in $F \setminus X$ are left unspecified. A goal $G_i \in \mathcal{G}$ is satisfied in any complete state s such that $G_i \subseteq s$. Throughout this thesis, goals are thus represented as partial states over the fluent set F .

We define the **plan trace** of a plan $\pi = \langle a_1, \dots, a_k \rangle$ as the ordered sequence of actions generated by executing π from the initial state I , i.e., the sequence of labelled transitions

$$I \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_k} s_k,$$

where each $s_i = f(a_i, s_{i-1})$ and s_k satisfies the agent's true goal G^* . In the action-based view adopted in this thesis, the plan trace of π is identified with the ordered action sequence $\langle a_1, \dots, a_k \rangle$ itself. The **observation sequence** $O = \langle o_1, \dots, o_n \rangle$ is then a (possibly non-consecutive) subsequence of such a plan trace, preserving the ordering of actions as they appear in π . Solving a GR instance consists of identifying the $G^* \in \mathcal{G}$ that is the hidden goal of the agent.

Model-based Goal Recognition *Model-based goal recognition* (MBGR) is an approach that relies on an explicit model of the agent’s actions and the states of the agent and environment. Under this view, GR is characterized as a reasoning problem that can be tackled using automated planning methods [32, 52].

Model-free Goal Recognition An alternative approach is *model-free goal recognition* (MFGR) [14, 26]. In this approach, GR is formulated as a classification task addressed through machine learning.

MFGR requires minimal information about the domain actions and can operate without the specification of an initial state, which can be completely unknown. Moreover, since running a learnt classification model is usually fast, a MFGR system is expected to run faster than a MBGR system based on planning algorithms. On the other hand, MFGR needs a dataset of solved GR instances from which to learn a classification model.

Offline Goal Recognition Offline Goal Recognition characterises the task when the following two conditions hold:

1. The observation sequence O consists of possibly non-consecutive actions from the full plan π , ordered as they appear in π .
2. A single candidate solution is generated once, given the complete (though potentially partial) observation sequence O .

Formally, an offline GR system receives the full tuple $T = (\Pi, I, O, \mathcal{G})$ with O fixed, and outputs a ranking or posterior distribution over $\mathcal{G}$ computed in a single pass. In this setting, the trace is typically provided after the agent has fully executed its plan, so the recogniser has access to all available observations simultaneously.

Example 1. To better understand the task, we will use a very simple Offline GR instance in the well-known BLOCKSWORLD domain. A goal recognition instance in the presented domain is illustrated in Figure 2.2, and it is composed of an initial state I , an observation sequence O , and a goal set $\mathcal{G}$. The initial state is $I = \langle (\text{On-Table Block_F}), (\text{On-Table Block_C}), (\text{On-Table Block_H}), (\text{On-Table Block_G}), (\text{On Block_B, Block_H}), (\text{Clear Block_F}), (\text{Clear Block_C}), (\text{Clear Block_B}), (\text{Clear Block_G}) \rangle$, while the observation sequence represents $O = \langle (\text{Pick-Up Block_C}), (\text{Stack Block_C Block_B}), (\text{Pick-Up Block_F}) \rangle$. Finally, the goal set $\mathcal{G}$ of the instance example consists of the two goals $G_1 = \{ (\text{On Block_F Block_C}), (\text{On Block_C Block_B}) \}$ and $G_2 = \{ (\text{On Block_G Block_H}), (\text{On Block_H Block_F}) \}$;

By processing the input, the task of an Offline Goal Recognition system is to select the most probable agent's objective from the goal set $\mathcal{G}$.

Online Goal Recognition (OGR) Online Goal Recognition (OGR) is the task of understanding the intention of an agent as it executes a plan, updating the goal hypothesis each time a new action is observed. OGR is characterised by the following conditions, following the assumptions stated in [82]:

1. The observation trace is *complete and incremental*: at time step t , the recogniser receives the prefix $O_t = \langle o_1, \dots, o_t \rangle$, and the full sequence $O_k = \pi$ is only revealed at the end.
2. A candidate solution is generated after each new observation o_t , without prior knowledge of the total plan length k .

Formally, OGR constitutes a sequence of GR problems with and incremental observations $\{T_t\}_{t=1}^k = \{(\Pi, I, O_t, \mathcal{G})\}_{t=1}^k$, each solved as new observations arrive. This contrasts with offline GR, which solves a single problem for a fixed obser-

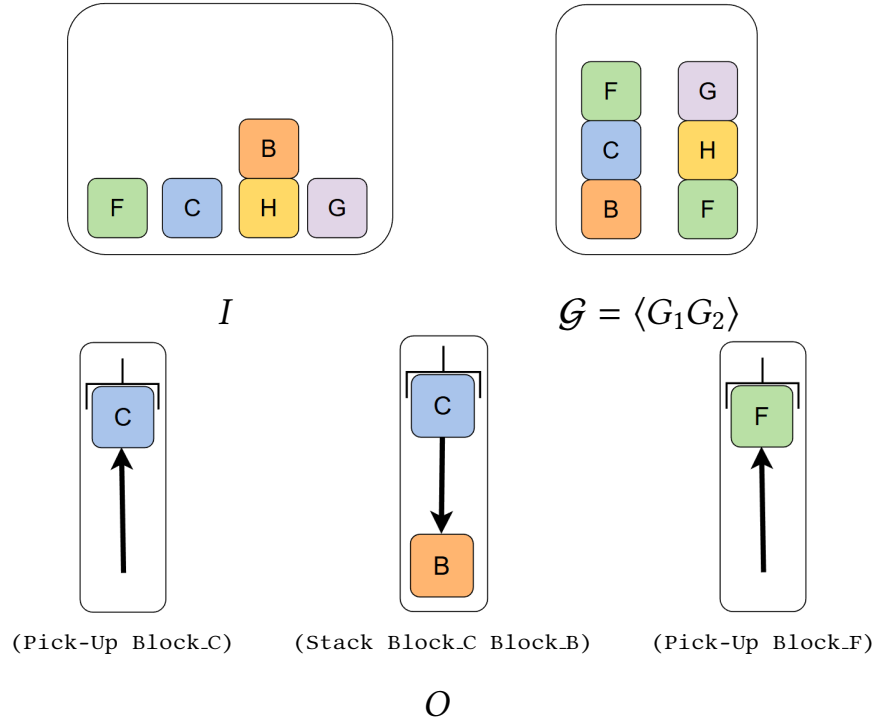


Figure 2.2: Representation of a toy goal recognition instance in the BLOCKSWORLD domain. (a) The initial state I . (b) The goal set $\mathcal{G}$ composed of G_1 and G_2 . (c) The observation sequence O showing subset of actions executed by the agent (partial observability).

vation sequence O . A crucial requirement in OGR is therefore that the recogniser must produce a hypothesis after each additional observation with no prior knowledge of whether o_t is the final observation.

Example 2. Following the previous example, we will now see an OGR instance in the BLOCKSWORLD domain.

In this example, depicted in Figure 2.3, we assume that our OGR instance involves at most 4 blocks; subsequently, the fluent set F consists of 25 propositions. We then assume that the goal set $\mathcal{G}$ of the instance example consists of two

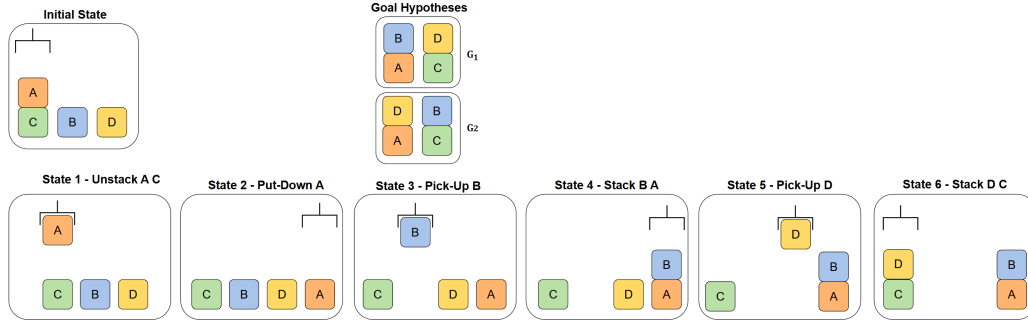


Figure 2.3: Online Goal Recognition running example in the BLOCKSWORLD domain. The figure illustrates an OGR instance with an initial state (left), two candidate goals G_1 and G_2 (top right), and a sequence of six actions executed by the agent (bottom). The system must predict which goal the agent is pursuing based on incrementally observed actions.

candidate goals $G_1 = \langle (\text{On Block_B Block_A}), (\text{On Block_D Block_C}) \rangle$ (which is the correct goal of the agent) and $G_2 = \langle (\text{On Block_D Block_A}), (\text{On Block_B Block_C}) \rangle$ (which is another hypothetical goal). The plan executed by the agent, which corresponds to the complete sequence of observations $O = \langle o_1, \dots, o_6 \rangle$, is made up of 6 actions: $\langle (\text{Unstack Block_A Block_C}), (\text{Put-Down Block_A}), (\text{Pick-Up Block_B}), (\text{Stack Block_B Block_A}), (\text{Pick-Up Block_D}), (\text{Stack Block_D Block_C}) \rangle$.

An Online Goal Recognition system wants to make a prediction for each new observation available, as depicted in Figure 2.4.

2.1.3 Deceptive Planning

The study of **deceptive planning** serves as a counterpart of the goal recognition problem. While goal recognition aims to infer an agent's hidden goal from observed behaviour, deceptive planning seeks to generate a sequence of actions that

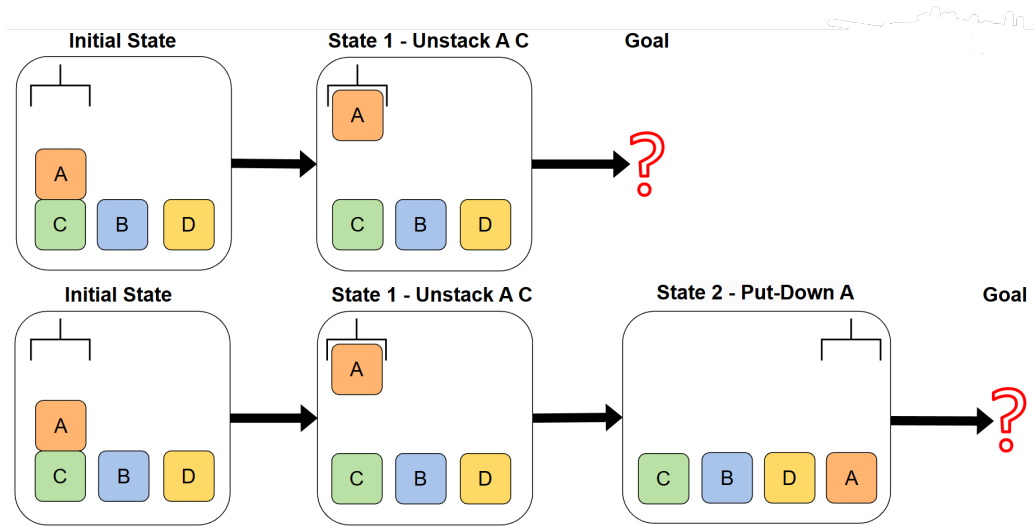


Figure 2.4: In the Online Goal Recognition task, predictions are made in sequence, with a new prediction generated each time additional information becomes available.

induces an observer to infer an incorrect goal or prevents them from determining the true goal until the last possible moment. Although the philosophical definition of deception, involving the manipulation of “false beliefs,” is complex when applied to machines, the field has coalesced around a behavioural definition: *an agent is deceptive if its behaviour has the potential to mislead*. Foundational theory in this domain identifies two primary mechanisms for deception:

- **Simulation** (“*showing the false*”): the agent deliberately produces behaviour that is strongly indicative of a false goal G_F . For instance, in the BLOCKSWORLD domain, if the agent’s true goal is to stack block A on block B, it may first build a tower consistent with stacking C on D—achieving landmarks specific to G_F —so that an observer predict G_F until a late stage of plan execution.

- **Dissimulation** (*“hiding the real”*): the agent behaves in a way that is compatible with many candidate goals, avoiding any action that uniquely reveals the true goal until the very last possible moment. For example, the agent may manoeuvre blocks into a “central” configuration from which multiple goals remain reachable at comparable cost, delaying any goal-specific commitment.

These two mechanisms correspond naturally to landmark-based strategies: simulation is realised by preferentially achieving landmarks exclusive to a specific false goal G_F (see DL-Similar below), whilst dissimulation is realised by restricting actions to the intersection of landmarks shared across many candidate goals (see DL-Shared strategy described below).

The earliest formalizations of this problem appeared in the Path Planning field, by Masters et al in [48].

Domain-Independent Deceptive Planning

While path planning focusses on spatial navigation, recent work by Price et al. [61] has extended these concepts to domain-independent classical planning (e.g., STRIPS domains like LOGISTICS or BLOCKSWORLD). In these domains, the complexity increases because states are not necessarily fully reachable, and distance is defined by the cost of actions rather than Euclidean geometry. Current research leverages two main techniques to generate deceptive plans in these complex environments: Landmarks and Relevant States.

Landmark-Based Strategies

Landmarks are facts (fluents) or actions that must be achieved or executed in all valid plans to reach a specific goal. In the context of deception, landmarks

are manipulated to control the observer’s belief update. By selecting landmarks associated with false goals, an agent can “simulate” a false intention. The authors proposed several distinct strategies for landmark selection:

DL-Similar (Most Similar Landmarks): Let $L(G)$ denote the set of landmarks for goal G . DL-Similar selects the false goal

$$G_F = \arg \max_{G \in \mathcal{G} \setminus \{G_r\}} |L(G_r) \cap L(G)|,$$

i.e., the goal whose landmark set most closely resembles that of the true goal G_r . The strategy then generates a plan that *first* achieves landmarks exclusive to G_F (including those also in $L(G_r)$), and only deviates towards G_r after those false-exclusive landmarks have been reached. This is a *simulation* strategy: its objective is to make G_F appear as plausible as possible for as long as possible. This attack will be studied in Chapter 6.

DL-Shared (Shared Landmarks): Rather than committing to any single false goal, DL-Shared focuses on the intersection of landmarks across the real goal and the set of false goals $\mathcal{G}' = \mathcal{G} \setminus \{G_r\}$:

$$\mathcal{L}_\cap = \bigcap_{G \in \mathcal{G}'} L(G) \cap L(G_r).$$

The planner generates a plan that achieves only the fluents in $\mathcal{L}_\cap$ before committing to G_r . By prioritising these common milestones, the agent remains *ambiguous* across multiple hypotheses simultaneously, a *dissimulation* strategy.

It is important to note that DL-Similar is **not** a special case of DL-Shared. Their optimisation objectives are fundamentally different: DL-Shared aims to preserve ambiguity across *multiple* goals by avoiding any landmark that would increase confidence in a particular hypothesis, including G_F . DL-Similar, by contrast, deliberately exploits landmarks *exclusive* to a single false goal G_F to drive

the observer towards a specific wrong conclusion. Whereas DL-Shared operates on a multi-goal intersection $\bigcap_{G \in \mathcal{G}'} L(G)$, DL-Similar operates on the pairwise overlap $L(G_r) \cap L(G_F)$ and then includes false-exclusive landmarks of G_F , precisely what DL-Shared avoids.

DL-Combined: This approach categorises landmarks into three sets to optimise the sequence of deception:

1. **Shared Landmarks ($\mathcal{L}_{\oplus}$):** Common to both the real and false goals.
2. **False Exclusive Landmarks ($\mathcal{L}_{N\otimes}$):** Necessary for the false goal but not for the real one.
3. **Real Exclusive Landmarks ($\mathcal{L}_{r\otimes}$):** Necessary only for the real goal.

The strategy orders the plan execution as $\mathcal{L}_{\oplus} \rightarrow \mathcal{L}_{N\otimes} \rightarrow \mathcal{L}_{r\otimes}$. This ensures that landmarks common to most goals are achieved first, followed by those that actively mislead the observer toward a false goal, delaying the revelation of the true goal until the very end.

DL-MostCommon: This strategy extends the concept to the entire set of possible goals in the domain. It identifies landmarks that are common to the highest number of candidate goals. By prioritising landmarks with the highest frequency of occurrence across all goals, the agent maintains maximum ambiguity, keeping the observer unsure among the largest possible set of candidates.

Complementary to deception-orientated strategies, work on *goal legibility* pursues the opposite objective: Bernardini et al. [11] demonstrate that, in cooperative multi-agent settings with partial observability, agent paths can be optimised to *minimize* ambiguity, enabling an observer to infer each actor's desti-

nation from the fewest possible observations by reformulating the problem as a classical minimum cost flow problem.

Centroids and Minimum Covering States

An alternative to landmark manipulation involves identifying specific states in the planning search space that inherently obfuscate the goal. These are referred to as Relevant States.

Centroid States (C): A centroid is a state that minimises the average cost to reach all goals in the set of potential goals $\mathcal{G}$. By navigating toward a centroid, an agent positions itself such that it is "close" to the majority of potential goals. This facilitates dissimulation, as the agent does not commit to a specific trajectory that rules out other goals. These kinds of states will be leveraged to perform an attack in Chapter 6.

Minimum Covering States (M): A minimum covering state minimises the *maximum* distance to any of the goals in $\mathcal{G}$. Geometrically, this corresponds to the centre of the smallest sphere containing all goal points. In planning, reaching an M state ensures the agent is not far from any potential goal, making it difficult for an observer—such as an adversary in a competitive domain—to predict the final destination or block the agent.

Computation: Computing these states in automated planning is challenging because the distance between states relies on the cost of optimal plans, which must be computed or estimated. Pozanco et al. [60] proposed the C&MCS algorithm, a Best-First Search that orders nodes based on an evaluation function e . To find optimal Centroids (C^*), the algorithm minimises the arithmetic mean μ

of distances to all goals; to find Minimum Covering States (M^*), it minimises the maximum distance. In deceptive planning, agents generate plans that traverse these relevant states before heading to the real goal, effectively maximising confusion or ambiguity.

Robust goal obfuscation is also studied from an optimisation-theoretic perspective by Bernardini et al. [10], who provide a complete foundational analysis of the problem and introduce efficient algorithms for computing exact solutions that protect the agent’s goal regardless of the observer’s strategy, including observers that may attempt to learn and exploit the agent’s behaviour.

2.2 Deep Learning

Having established the planning foundations, we now analyse the deep learning techniques that support our model-free goal recognition approaches. In this thesis, we frame goal recognition as a classification task addressed through neural networks. This section presents the fundamental concepts and architectures used in our work, including word representations, recurrent neural networks, attention mechanisms, and Transformer-based models.

2.2.1 Learning via Gradient Descent

All neural network models employed in this thesis are trained by minimising a **loss function** $\mathcal{L}(\theta)$ with respect to the model parameters θ via **gradient descent**. At each training step, the parameters are updated in the direction of the negative gradient of the loss:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} \mathcal{L}(\theta_t), \quad (2.5)$$

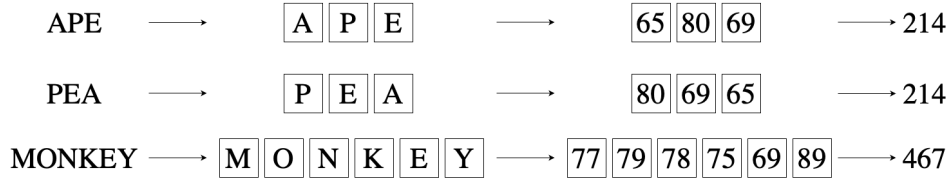


Figure 2.5: Example of naive ASCII summation encoding.

where $\eta > 0$ is the *learning rate*. In practice, stochastic variants are used: *mini-batch stochastic gradient descent* (SGD) approximates the true gradient using a randomly sampled subset of the training data at each step, reducing computational cost whilst preserving convergence properties. In this thesis, we employ the Adam optimiser [45], an adaptive learning-rate variant of SGD that maintains per-parameter momentum estimates and has become the standard choice for training deep learning models for sequential classification tasks.

2.2.2 Word Representations

To enable the machine learning system to address the Goal Recognition (GR) problem, observation components are modelled as discrete tokens, analogous to words in Natural Language Processing (NLP). In the computational domain, the representation of such tokens generally falls into two categories: *encoding*, where tokens are mapped to discrete numerical values or sparse vectors, and *embedding*, where tokens are projected into a continuous latent vector space of arbitrary dimensionality.

Encoding

Encoding is the process of transforming raw data into a rigorous format suitable for computational processing. Just as an image is represented as a matrix of pixel

	APE	PEA	MONKEY	
APE →	1	0	0	→ 100
PEA →	0	1	0	→ 010
MONKEY →	0	0	1	→ 001

Figure 2.6: One-Hot encoding representation.

intensities (e.g., values in $[0, 255]$), a word can be represented by mapping its constituent characters to numeric values using standards such as the American Standard Code for Information Interchange (ASCII). A naive approach to word encoding involves summing the ASCII values of a word's characters. Figure 2.5 illustrates this method applied to the words APE, PEA, and MONKEY. This example highlights a critical limitation of such naive aggregations: the words APE and PEA result in identical numerical values. For an algorithm lacking access to the raw string data, these distinct tokens become indistinguishable, introducing collision errors that render this specific method unsuitable for robust machine learning applications. To address the issue of uniqueness, *One-Hot Encoding* is commonly employed. Given a vocabulary of size K , each unique word is assigned an integer index $j \in [0, K - 1]$. The one-hot representation is a binary vector $v \in \{0, 1\}^K$, where the j -th component is 1 and all other components are 0. Figure 2.6 demonstrates this for a vocabulary of size $K = 3$ (APE, PEA, MONKEY), ensuring a distinct representation for each token. While One-Hot Encoding guarantees uniqueness, it introduces significant limitations. First, the dimensionality of the feature vector scales linearly with the vocabulary size K , leading to sparsity and the "curse of dimensionality," which can degrade the performance of many machine learning

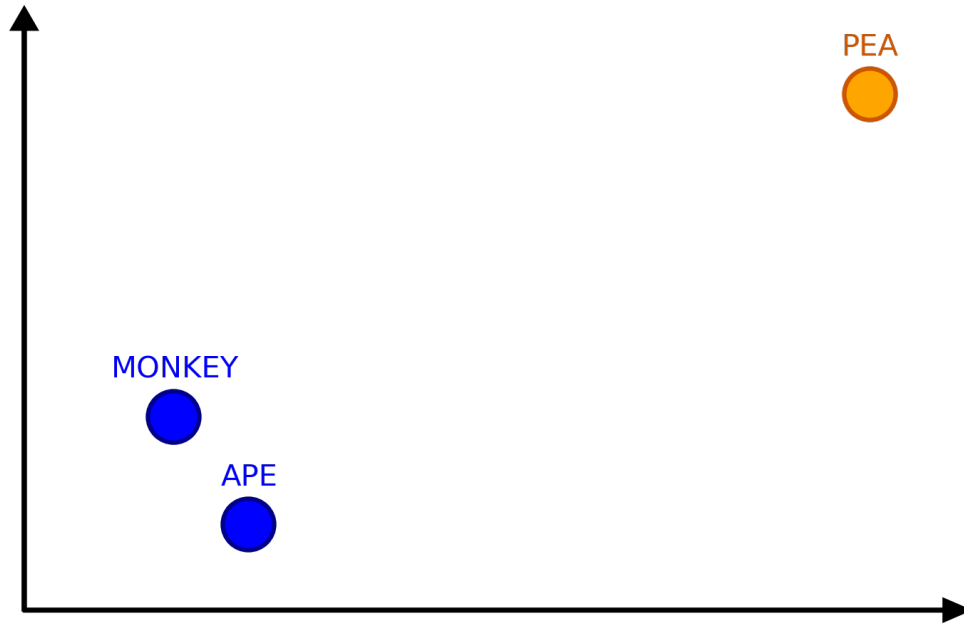


Figure 2.7: Bi-dimensional word embedding space.

models [80]. Second, this representation fails to capture semantic relationships. In a one-hot space, all word vectors are orthogonal (i.e., $v_{word_1} \cdot v_{word_2} = 0$ for any distinct pair), implying mathematical independence. Consequently, semantic similarities—such as the synonymy between APE and MONKEY—or grammatical relationships cannot be encoded or inferred.

Embedding

To overcome the limitations of sparse encodings, *embeddings* map high-dimensional categorical inputs into a continuous, relatively low-dimensional vector space. In NLP, this technique transforms sparse word representations into dense vectors that encapsulate semantic meaning and syntactic function [1]. Formally, each word is represented as a vector $v \in \mathbb{R}^n$, creating a distribution of points within

an n -dimensional vector space. An effective embedding space satisfies two primary geometric properties:

- **Semantic Proximity:** Semantically related tokens, such as synonyms, morphological variations, or conjugations, are mapped to adjacent regions in the vector space.
- **Semantic Distance:** Tokens with disparate meanings or distinct contexts are positioned at a significant distance from one another.

Figure 2.7 visualises a simplified two-dimensional embedding space. The synonyms APE and MONKEY cluster together in the lower-left quadrant, reflecting their semantic closeness, while the unrelated token PEA is situated distantly in the top-right quadrant. Several algorithms have been developed to generate such representations by analysing large corpora, including Word2Vec [53], GloVe [58], and FastText [42]. In the context of this research, we require token representations that are specifically optimised for the goal recognition task rather than for general linguistic patterns. Consequently, we compute embeddings end-to-end during the training of the supervised machine learning model. This approach generates *ad hoc* embeddings where the geometric arrangement of vectors is driven by the specific features necessary for classification.

2.2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [22] constitute a fundamental class of artificial neural networks specifically designed to process sequential data. Unlike standard feed-forward networks, where information flows unidirectionally from input to output, RNNs incorporate cycles within their computational graph. These cycles enable the network to maintain an internal state, or *memory*, allowing it to exhibit dynamic temporal behaviour. Formally, an RNN processes an

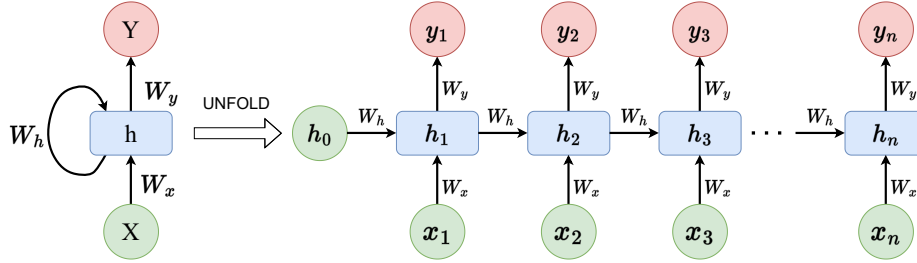


Figure 2.8: Compressed (left) and unfolded (right) representation of a Recurrent Neural Network processing an input sequence X of length n . W_x , W_h , and W_y denote the shared weight matrices.

input sequence one element at a time, updating its hidden state at step t based on both the current input and the hidden state from step $t - 1$. This mechanism enables the network to capture dependencies across time steps, making it particularly effective for tasks involving time-series data, natural language processing, and, in this context, sequential action observations.

Standard RNNs

Consider an input sequence $X = (x_1, x_2, \dots, x_n)$, where each element $x_t \in \mathbb{R}^k$ represents a k -dimensional feature vector (e.g., a word embedding) at time step t . Figure 2.8 illustrates the architecture of a standard RNN in both its compressed loop form (left) and its temporally unfolded form (right). In the unfolded view, the RNN can be interpreted as a deep network with n layers, where each layer corresponds to a time step and shares the same parameters. Let $h_t \in \mathbb{R}^d$ denote the hidden state at time t , where d is the dimensionality of the hidden layer. The transition dynamics of the network are governed by the following recurrence relations:

$$h_t = \sigma_h(W_x x_t + W_h h_{t-1} + b_h) \quad (2.6)$$

$$y_t = \sigma_y(W_y h_t + b_y) \quad (2.7)$$

Here, $W_x \in \mathbb{R}^{d \times k}$, $W_h \in \mathbb{R}^{d \times d}$, and $W_y \in \mathbb{R}^{m \times d}$ are the weight matrices connecting the input to the hidden state, the previous hidden state to the current hidden state, and the hidden state to the output, respectively. The vectors $b_h \in \mathbb{R}^d$ and $b_y \in \mathbb{R}^m$ represent the biases. The function σ_h is a non-linear activation function (typically tanh or ReLU), while σ_y is the output activation function tailored to the specific task (e.g., softmax for classification). A critical feature of RNNs is parameter sharing: the weights (W_x, W_h, W_y) remain constant across all time steps. This inductive bias allows the model to generalise patterns regardless of their position in the sequence and significantly reduces the number of parameters compared to fully connected networks treating the sequence as a fixed-size vector. Despite their theoretical power, standard RNNs suffer from two major limitations during training via Backpropagation Through Time (BPTT):

- **Exploding and Vanishing Gradients** [56]: As gradients are propagated backward through many time steps, they can grow exponentially (exploding) or decay exponentially (vanishing). The latter makes it effectively impossible for the network to learn dependencies between temporally distant events.
- **Long-Term Dependency Problem** [31]: Due to the vanishing gradient problem and the iterative transformation of the state, standard RNNs struggle to retain information over long sequences, often biasing the model toward recent inputs while "forgetting" early context.

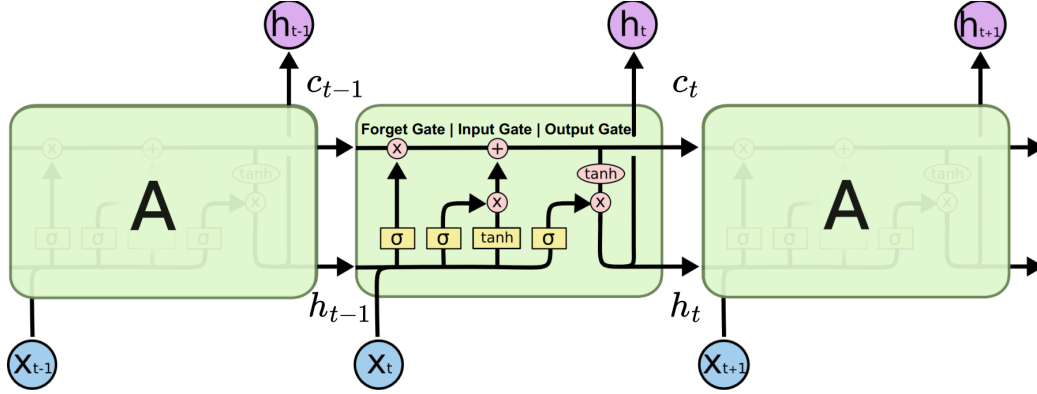


Figure 2.9: Schematic of an LSTM cell. Yellow blocks denote neural network layers, while pink circles represent pointwise operations. Arrows indicate the flow of vectors.

Long Short-Term Memory (LSTM)

To address the gradient stability issues of standard RNNs, Hochreiter and Schmidhuber [37] introduced the Long Short-Term Memory (LSTM) architecture. As depicted in Figure 2.9, the core innovation of the LSTM is the separation of the *cell state* c_t , which acts as a path for gradients to flow relatively unchanged, from the hidden state h_t . The flow of information into and out of the cell state is regulated by structures called *gates*. An LSTM cell at time t computes the following components:

Forget Gate (f_t) Determines what fraction of the previous cell state should be retained.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (2.8)$$

where σ is the sigmoid function, outputting values in $(0, 1)$.

Input Gate (i_t) and Candidate State ($\tilde{C}_t$) The input gate controls the extent to which new information is written to the cell state, while $\tilde{C}_t$ represents the new candidate memory content.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2.9)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (2.10)$$

Cell State Update The new cell state c_t is a linear combination of the previous state and the new candidate content, regulated by the gates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{C}_t \quad (2.11)$$

where $\odot$ denotes the Hadamard (element-wise) product.

Output Gate (o_t) and Hidden State (h_t) Finally, the output gate controls what part of the cell state is exposed as the hidden state h_t :

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (2.12)$$

$$h_t = o_t \odot \tanh(c_t) \quad (2.13)$$

By using additive updates for the cell state, LSTMs mitigate the vanishing gradient problem, enabling the learning of dependencies over much longer sequences than standard RNNs.

2.2.4 Attention Mechanisms

Despite the advancements introduced by recurrent architectures such as LSTMs, these models remain fundamentally constrained in their ability to capture long-range dependencies within sequential data [62]. Consider a sequence classification task over a document of length n , where the token at position t is represented

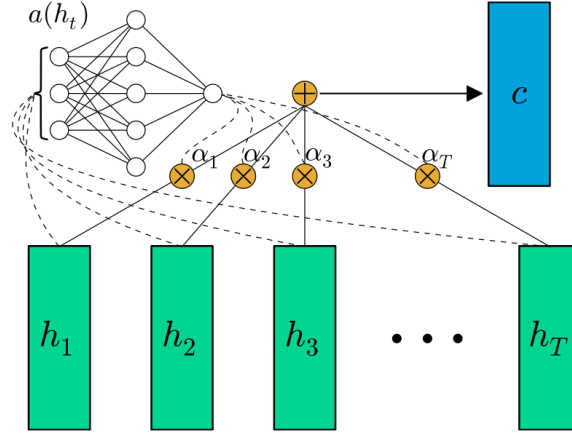


Figure 2.10: Schematic of the Feed-Forward Attention mechanism. A learnable function $a(h_t)$ computes a scalar energy score e_t for each hidden state h_t . These scores are normalized via a softmax function to produce attention weights α_t . The final context vector c is derived as a weighted sum of the hidden states [62].

by an embedding vector $w_t \in \mathbb{R}^k$. In standard recurrent processing, the hidden state h_n computed at the final time step is intended to encapsulate the entire semantic context of the sequence. However, as n increases, the influence of early tokens (e.g., w_1) on h_n diminishes due to the vanishing gradient problem and the iterative nature of state updates, leading to a representational bias toward recent inputs. To mitigate this limitation, *Attention Mechanisms* [7, 62] were introduced to allow models to dynamically focus on relevant parts of the input sequence, regardless of their positional distance.

Feed-Forward Attention

Feed-Forward Attention [62] adapts the attention paradigm originally proposed for sequence-to-sequence translation [7] to non-recurrent feed-forward architec-

tures. The mechanism is illustrated in Figure 2.10. Given a sequence of hidden states $H = (h_1, h_2, \dots, h_n)$ produced by a neural network, the goal is to compute a context vector c that aggregates information from the entire sequence based on the relevance of each state. Formally, the context vector c is defined as a weighted sum of the hidden states:

$$c = \sum_{t=1}^n \alpha_t h_t \quad (2.14)$$

where α_t represents the attention weight assigned to state h_t . These weights are derived by first computing an alignment score (or energy) e_t for each state using a learnable alignment function $a(\cdot)$, which is typically implemented as a feed-forward neural network:

$$e_t = a(h_t) \quad (2.15)$$

The scores are then normalised using the softmax function to form a valid probability distribution:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^n \exp(e_k)} \quad (2.16)$$

This mechanism allows the network to learn which elements of the input sequence are most significant for the downstream task, effectively creating a focused, fixed-size representation of variable-length inputs.

Self-Attention

The Self-Attention mechanism, depicted in Figure 2.11, was introduced by Vaswani et al. [79]. It generalises the attention concept to compute representations by relating different positions of a single sequence. Unlike RNNs, which process data sequentially, Self-Attention allows for parallel computation and direct interaction between any pair of tokens in a document, capturing dependencies regardless of their distance. Given an input sequence of n tokens, each represented by an embedding $x_t \in \mathbb{R}^d$, the mechanism first projects each input into three

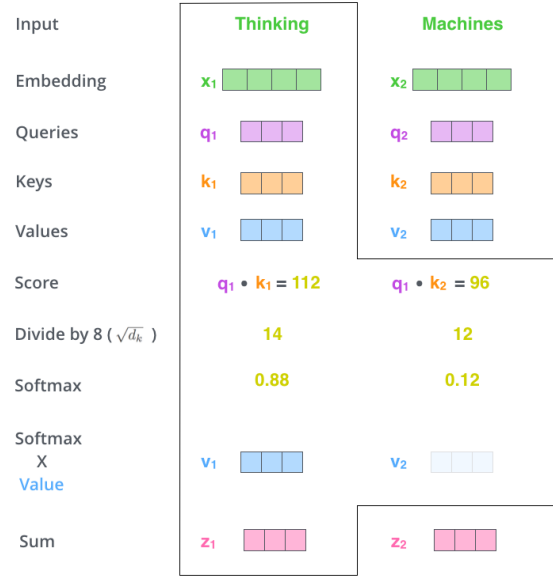


Figure 2.11: Operational steps of the Self-Attention mechanism.

distinct vector spaces: *Query* (q_t), *Key* (k_t), and *Value* (v_t). These projections are computed using learnable weight matrices $W_q, W_k, W_v \in \mathbb{R}^{d \times d_k}$:

$$q_t = W_q x_t, \quad k_t = W_k x_t, \quad v_t = W_v x_t \quad (2.17)$$

Here, d_k denotes the dimension of the key/query/value vectors (typically $d_k = 64$ in multi-head configurations). The output representation z_t for the token at position t is computed as a weighted sum of the value vectors of all tokens in the sequence:

$$z_t = \sum_{i=1}^n \alpha_{t,i} v_i \quad (2.18)$$

The attention weights $\alpha_{t,i}$ determine the relevance of token i to token t . These

are computed using the Scaled Dot-Product Attention formula:

$$\alpha_{t,i} = \text{softmax} \left(\frac{q_t \cdot k_i^T}{\sqrt{d_k}} \right) = \frac{\exp \left(\frac{q_t \cdot k_i^T}{\sqrt{d_k}} \right)}{\sum_{j=1}^n \exp \left(\frac{q_t \cdot k_j^T}{\sqrt{d_k}} \right)} \quad (2.19)$$

The dot product $q_t \cdot k_i^T$ measures the similarity between the query of token t and the key of token i . The scaling factor $\sqrt{d_k}$ prevents the dot products from growing too large in magnitude, which would otherwise push the softmax function into regions with extremely small gradients. This operation can be efficiently vectorised. Let $X \in \mathbb{R}^{n \times d}$ be the input matrix. The projections are computed as $Q = XW_q$, $K = XW_k$, and $V = XW_v$. The attention operation is then:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2.20)$$

The resulting matrix contains the context-aware representations z_t for all tokens.

Multi-Head Attention

To enhance the representational power of Self-Attention, Vaswani et al. [79] introduced *Multi-Head Attention*, depicted in Figure 2.12. This mechanism runs h parallel self-attention operations, or "heads," each with its own set of learnable projection matrices $\{W_i^Q, W_i^K, W_i^V\}_{i=1}^h$. This allows the model to jointly attend to information from different representation subspaces at different positions. For each head i , the output is computed as:

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \quad (2.21)$$

The outputs of all heads are concatenated and projected by a final output matrix $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.22)$$

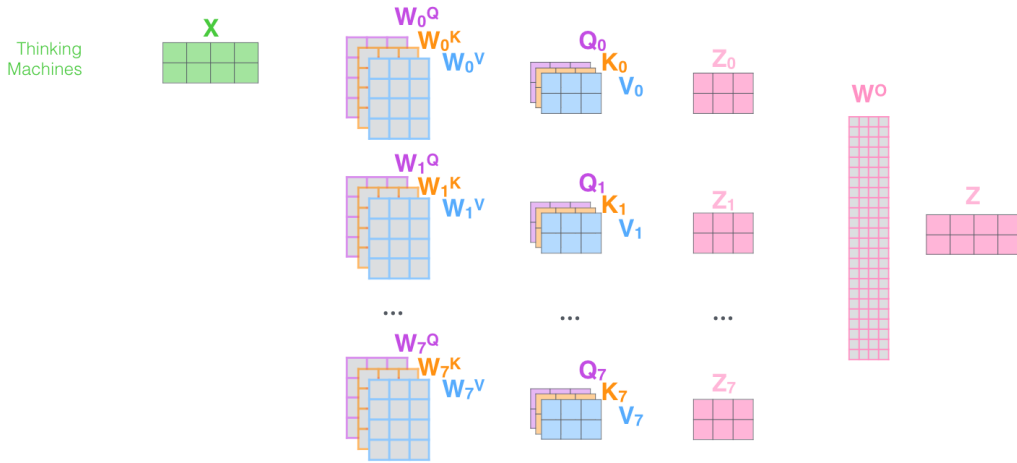


Figure 2.12: Multi-Head Attention architecture with $h = 8$ heads. The input matrix X is projected into h different subspaces, processed independently by scaled dot-product attention, concatenated, and finally projected linearly to form the output Z .

This multi-head structure enables the model to capture diverse types of dependencies (e.g., syntactic vs. semantic) simultaneously.

2.2.5 Transformer

The Transformer [79] represents a paradigm shift in deep learning architectures for sequence modeling. Unlike Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, which process data sequentially, the Transformer relies entirely on an attention mechanism to draw global dependencies between input and output. This architectural choice allows for significantly greater parallelization during training, addressing the computational bottlenecks inherent in sequential models. While originally designed for machine translation, the Transformer has become the foundational architecture for Large Language Mod-

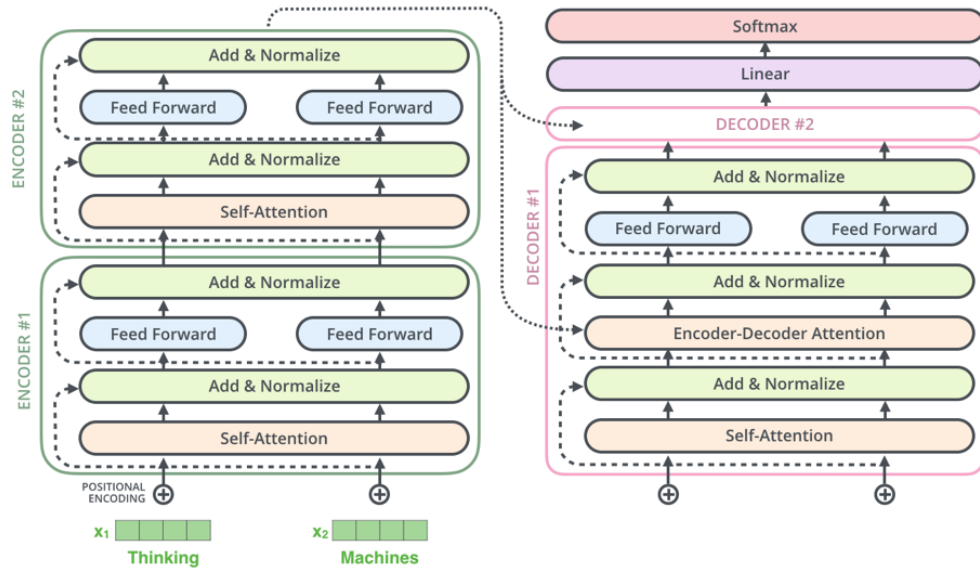


Figure 2.13: The Transformer architecture consisting of a stack of $N = 2$ encoders and decoders. The input sequence is represented by the matrix X , unfolded into token vectors x_1 and x_2 .

els (LLMs) and state-of-the-art performance across various Natural Language Processing (NLP) tasks.

Architecture

The Transformer follows the traditional encoder-decoder structure. The encoder maps an input sequence of symbol representations to a sequence of continuous representations, which is then fed into a decoder to generate an output sequence. The model is composed of a stack of N identical layers for both the encoder and the decoder components (in the seminal work by [79], $N = 6$). Figure 2.13 illustrates a simplified architecture with $N = 2$.

Input Embedding and Positional Encoding The input to the model is a matrix $X \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the embedding dimension. Because the Transformer contains no recurrence or convolution, it is invariant to the order of the sequence. To inject information regarding the relative or absolute position of the tokens, a *Positional Encoding* vector is added to the input embeddings. While positional encodings can be learned, [79] utilize fixed sinusoidal functions of different frequencies. For a position pos and dimension i , the encoding is defined as:

$$\begin{aligned} PE_{(pos, 2i)} &= \sin\left(\frac{pos}{10000^{2i/d}}\right) \\ PE_{(pos, 2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d}}\right) \end{aligned} \quad (2.23)$$

where $pos \in [0, n - 1]$ represents the position of the token in the sequence, and i indexes the dimension of the embedding. This formulation allows the model to learn to attend by relative positions, as for any fixed offset k , PE_{pos+k} can be represented as a linear function of PE_{pos} .

Encoder Stack The encoder is composed of a stack of N identical layers. Each layer has two sub-layers:

1. A Multi-Head Self-Attention mechanism (as defined in Equation 2.22).
2. A position-wise fully connected feed-forward network.

To facilitate gradient flow and training depth, a residual connection [33] is employed around each of the two sub-layers, followed by layer normalization. Formally, the output of each sub-layer is:

$$\text{LayerNorm}(x + \text{Sublayer}(x)) \quad (2.24)$$

where $\text{Sublayer}(x)$ is the function implemented by the sub-layer itself. To accommodate these residual connections, all sub-layers and embedding layers produce outputs of dimension $d_{\text{model}} = 512$.

Decoder Stack The decoder is also composed of a stack of N identical layers. However, in addition to the two sub-layers present in the encoder, the decoder inserts a third sub-layer, which performs Multi-Head Attention over the output of the encoder stack (often referred to as *cross-attention*). In this cross-attention layer, the queries come from the previous decoder layer, while the keys and values come from the output of the encoder. Let Z^E be the output of the final encoder layer and Z^D be the output of the masked self-attention sub-layer in the decoder. The projection matrices are computed as:

$$Q = Z^D W_{\text{enc/dec}}^Q, \quad K = Z^E W_{\text{enc/dec}}^K, \quad V = Z^E W_{\text{enc/dec}}^V \quad (2.25)$$

This allows every position in the decoder to attend over all positions in the input sequence. Furthermore, the self-attention sub-layer in the decoder is modified to prevent positions from attending to subsequent positions. This *masking* ensures that the prediction for position i depends only on the known outputs at positions less than i , preserving the auto-regressive property required for sequence generation.

BERT

Bidirectional Encoder Representations from Transformers (BERT), introduced by [21], represents a landmark advancement in the application of Transformer architectures. BERT is conceptually an encoder-only Transformer designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As illustrated in the left panel of Figure 2.14, BERT is pre-trained using two unsupervised tasks:

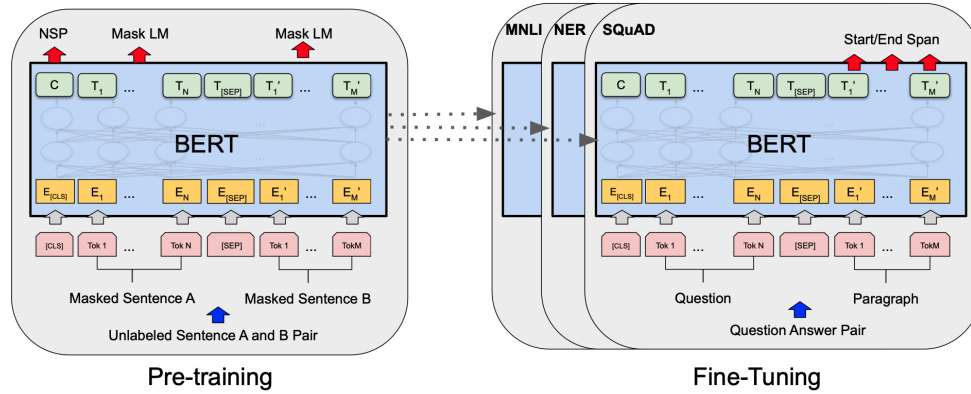


Figure 2.14: Overview of the BERT framework. The model undergoes semi-supervised pre-training (left) on large unlabeled corpora via Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Subsequently, it is fine-tuned (right) on specific downstream tasks with minimal architectural modification.

- **Masked Language Model (MLM):** Standard conditional language models can only be trained left-to-right or right-to-left. To enable deep bidirectional training, BERT randomly masks a percentage (typically 15%) of the input tokens. The model is then trained to predict the original vocabulary id of the masked word based only on its context. This objective forces the model to fuse context from both directions.
- **Next Sentence Prediction (NSP):** Many downstream tasks, such as Question Answering (QA) and Natural Language Inference (NLI), require an understanding of the relationship between two sentences. To capture this, BERT is trained on a binary classification task: given a pair of sentences (A, B) , the model predicts whether B is the actual next sentence that follows A in the corpus or a random sentence.

These two objectives are optimised simultaneously during pre-training. Once pre-trained, the model can be fine-tuned (right panel of Figure 2.14) by adding a single output layer to the core architecture. This transfer learning approach allows BERT to achieve state-of-the-art performance on a wide range of discriminative tasks with minimal task-specific parameter modifications.

2.2.6 Large Language Models for Planning

The success of Transformer-based large language models (LLMs) has recently prompted investigation into their applicability to automated planning tasks. Silver et al. [71] demonstrated that LLMs can be used to guide planning in text-described environments by leveraging world knowledge encoded during pre-training. Subsequent work by Silver et al. [72] explored the integration of LLMs into symbolic planning pipelines, using language models for domain grounding, heuristic guidance, and plan sketching in classical PDDL domains. Rossetti et al. [66] showed that Transformer architectures, when trained on planning traces, can learn to generate valid plans and capture planning domain dynamics effectively. These works collectively motivate the paradigm explored in this thesis: rather than using LLMs as end-to-end planners, we leverage the structural properties of Transformer architectures (in particular, BERT’s bidirectional attention) to model the “grammar” of planning domains for the purpose of goal recognition, bridging symbolic planning knowledge and neural sequence modelling.

Chapter 3

State of the Art

Goal and plan recognition have been extensively studied through two main paradigms: model-based approaches exploiting planning techniques [52, 59, 63, 67, 73] and matching techniques relying on plan libraries (e.g., [55]).

Among model-based approaches, two works are particularly relevant to this thesis. First, the seminal work of Ramírez and Geffner [63] was one of the first to apply classical planning techniques to solve plan and goal recognition problems. Second, the work of Pereira et al. [59], also proposed as an online goal recognition (OGR) system in [82], represents the current state-of-the-art for MBGR. Since this thesis proposes approaches to solve MFGR instances, we also review the most relevant MFGR approaches. Finally, we discuss state-of-the-art works in deceptiveness, introducing deceptive planning concepts and techniques.

This section is organised as follows. First, we review model-based approaches, focussing on classical planning techniques and landmark-based heuristics. Then, we examine model-free approaches that leverage neural networks. Finally, we discuss deceptive planning concepts and techniques.

Model-Based Goal Recognition Approaches

Plan Recognition as Planning The seminal work of Ramírez and Geffner [63] (henceforth PRP) reformulates the goal recognition problem as a planning problem. Rather than relying on plan libraries, this approach leverages domain theories and rational behaviour assumptions. The authors propose two methods: an exact approach based on optimal planning and an approximate approach based on suboptimal planning. The *exact approach* determines whether a goal $G \in \mathcal{G}$ is consistent with a sequence of observations O by comparing the cost of an optimal plan for G against the cost of an optimal plan for G that is constrained to satisfy O .

While theoretically sound, this method requires two calls to an optimal planner for every candidate goal, which is computationally expensive. The *approximate approach* relaxes the optimality requirement. It utilises a modified version of the Fast-Forward planner [38] to generate a “good” plan $\pi_{p'}$ for each goal. While significantly faster, this method does not offer the formal guarantees of the exact approach. A follow-up work [64] extends this framework to produce a posterior probability distribution over goals, identifying the optimal goal set as those minimising the cost difference $\Delta(G) = c_{p'}^*(G') - c_{p'}^*(G)$.

While PRP provides a solid theoretical foundation, its computational cost motivated the development of more efficient heuristic approaches.

Landmark-Based Heuristics for Goal Recognition To address the computational overhead of running full planners, Pereira et al. [59] (henceforth LGR) proposes a heuristic approach that bypasses the planning step entirely. LGR relies on *planning landmarks*—facts or actions that must be achieved or executed in every valid plan to reach a goal [39]. The authors introduce two specific heuristics: Goal Completion and Uniqueness. The *Goal Completion heuristic* estimates

progress toward a goal G by calculating the ratio of achieved landmarks to the total number of necessary landmarks.

The *Uniqueness heuristic* refines this estimate by weighting landmarks according to their information gain. Landmarks that appear in the landmark sets of fewer candidate goals are considered more “unique” and thus more discriminative.

The authors prove that the *Goal Completion heuristic* is monotonic and, under full observability conditions, strictly ranks the correct goal highest (soundness). However, like the approximate method of PRP, it does not guarantee correctness for partial observations in general cases where a false goal may share a high ratio of early landmarks with the true goal.

Building on the efficiency of landmark-based methods, researchers explored online recognition approaches that could operate during plan execution.

Goal Mirroring with Landmarks Vered and Kaminka [81] introduced Goal Mirroring, an online recognition approach inspired by the mirror neuron system in the human brain. Goal Mirroring utilises an off-the-shelf planner as a “black box” to dynamically generate recognition hypotheses during execution, similar to what is done in PRP. However, it incurs a high computational cost because it requires invoking a planner for every active goal hypothesis at each new observation step. To address this scalability issue, Vered et al. [82] proposed a hybrid approach combining Goal Mirroring with Landmarks. This method integrates the pruning power of planning landmarks with the generative capability of Goal Mirroring. The algorithm first checks if the observations are consistent with the landmarks associated with each goal. If an agent moves past a required landmark without achieving it, or achieves landmarks in an order inconsistent with a specific goal, that goal is pruned from the hypothesis set $\mathcal{G}$. The costly Goal

Mirroring process (calling the planner) is then executed only for the remaining valid hypotheses, significantly reducing runtime.

The system maintains the high accuracy of black-box planning methods while achieving computational efficiency comparable to heuristic approaches. The inherent tension between recognition speed and accuracy in time-sensitive scenarios is further formalised by Bernardini et al. [12], who model the interaction between a defender performing goal recognition and an attacker pursuing a sensitive target as a game-theoretic problem, providing an efficient method based on fictitious play to compute robust defender policies under strict time constraints.

In this work, to address discrete domains in an online scenario, the authors present an adaptation of LGR [59]. They refer to this method as *Online Recognition with Landmarks* (ORL).

Model-Free Goal Recognition Approaches

While the aforementioned approaches rely on explicit domain models, recent work has explored model-free alternatives using neural networks that do not require explicit domain models. These approaches can be categorised by their input representation and learning strategy.

Sequence-Based Approaches Several works use recurrent neural networks to process sequential observations, training a *single model* that generalises across multiple goal recognition instances within a domain. Some approaches focus on specific applications, such as game playing [54]. Amado et al. [2] combined a pre-trained convolutional encoder with an LSTM network to perform goal recognition from *image-based* domains, where observations are visual representations of states rather than symbolic action sequences; the encoder maps each image to a latent vector, which the LSTM then processes to infer the goal. Amado et al. [3]

trained an LSTM-based system to identify missing state observations, producing more complete sequences for downstream MBGR systems.

The key distinguishing feature of sequence-based approaches is that the trained model is *domain-level*: it is trained once over a large collection of plans and generalises to new goal recognition instances within that domain without retraining. This contrasts with instance-specific learning approaches (described below), in which a separate model is trained for each individual GR instance, i.e., for a fixed goal set $\mathcal{G}$.

Instance-Specific Learning Other approaches train models for specific goal recognition instances. Borrajo et al. [14] investigated XGBoost [17] and LSTM networks using only plan traces without domain knowledge. They train a separate model *for each goal recognition instance* (with a fixed goal set $\mathcal{G}$) using instance-specific datasets. Moreover, the experimental evaluation uses peculiar goal recognition benchmarks with custom-made instances.

Maynard et al. [49] compared model-based and deep learning techniques for goal recognition. As in [14], such a comparison is made using specific instances, and several types of neural networks are trained to predict the goal directly among a set of possible ones. This makes the trained networks specific to the goal recognition instances considered within a domain. Note that their work considers only observations from the beginning of the plan up to a given percentage, treating all subsequent observations as missing.

Reinforcement Learning Approaches Finally, the work by Amado et al. [4] introduces goal recognition as a reinforcement learning task by creating an instance-specific policy to select the most likely goal from the available options in the hypotheses set.

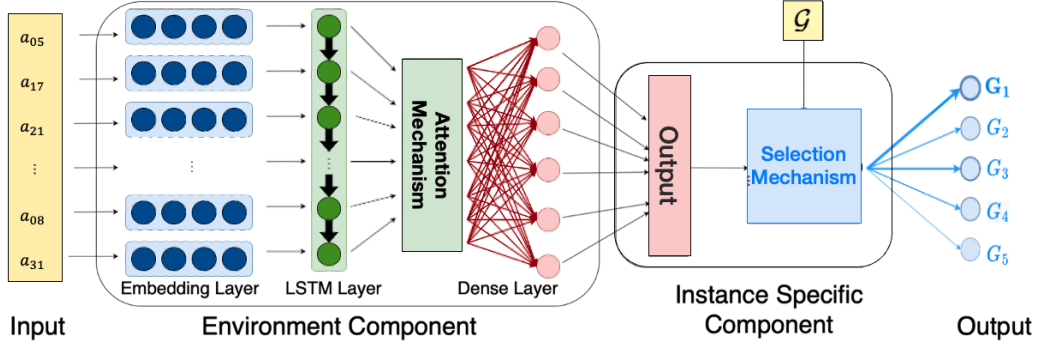


Figure 3.1: Architecture of GRNET. Observed actions are embedded and processed by an LSTM network. An attention mechanism computes a context vector, which feeds a dense layer with $|F|$ neurons (one per domain fluent). The instance component selects the goal with the highest score (G_1 in this example). The observed actions a_{05} , a_{17} , ..., a_{31} are ordered from top to bottom according to their execution order.

We now introduce GRNET [19], which represents the state-of-the-art for MFGR. This work serves as the foundation for this thesis, as it defines goal recognition as a deep learning task and establishes the core architectural principles upon which we build. It is proposed as an adaptation to approach the Online Goal Recognition task in Chapter 5.

GRNET Chiari et al. [19] proposed GRNET, an approach that formulates goal recognition as a machine learning classification task. It was originally designed to solve the Offline Goal Recognition task. Unlike model-based approaches that rely on planning algorithms at runtime, GRNET aims to improve both accuracy and classification speed by learning to solve recognition instances within a specific domain. The architecture is composed of two primary modules:

1. The **Environment Component**, which is general for a given domain (trained

once over the set of domain propositions F). It takes a trace of observations as input and outputs a likelihood score for every proposition in F .

2. The **Instance Component**, which is specific to a problem instance. It uses the proposition scores generated by the first component to evaluate and select a goal from the set of candidate goals $\mathcal{G}$.

The Environment Component: The input to the network is a sequence of observed actions O . As illustrated on the left side of Figure 3.1, each action label a_i is encoded into a vector of real numbers e_i via an embedding layer. These vectors are processed by a Recurrent Neural Network (RNN), specifically a Long Short-Term Memory (LSTM) network, described in Section 2.2.3, which is well-suited for processing sequential data. The output of the LSTM cells is processed by an Attention Mechanism, specifically the Feed-Forward Attention proposed by Yang et al. [84], addressed in Section 2.2.4, which computes weights representing the contribution of each element in the sequence to generate a unified *context vector* that represents the entire plan trace. This context vector is passed to a dense feed-forward layer with a *sigmoid* activation function.

This output layer contains N neurons, where N corresponds to the size of the domain fluent set ($N = |F|$). Consequently, the output of the i -th neuron, denoted as $\bar{o}_i$, corresponds to the i -th domain fluent f_i . The activation value of $\bar{o}_i$ represents the predicted likelihood (a rank between 0 and 1) that the fluent f_i is part of the agent’s goal. Effectively, the network is trained as a multi-label classification model, where each domain fluent is treated as a binary class.

The Instance Specific Component: This component solves the specific GR instance by evaluating the set of candidate goals $\mathcal{G}$ based on the predictions of the Environment Component. To select the most probable goal, GRNET employs a scoring function that aggregates the likelihoods of the individual fluents consti-

tuting a candidate goal. The score for a goal $G \in \mathcal{G}$ is defined as:

$$S_{\text{GRNET}}(G) = \sum_{f \in G} \bar{o}_f \quad (3.1)$$

where $\bar{o}_f$ is the network output for the specific fact f . The system sums the predicted values only for the neurons associated with the facts contained in G . The candidate goal with the highest aggregate score is selected as the solution. GRNET was subsequently expanded using a neuro-symbolic system, as detailed by Chiari et al. in [20]. This system draws inspiration from dual-process theories, combining a rapid, intuition-driven neural module (System 1), depicted by the GRNET environment component, with a more methodical, contemplative planner (System 2), depicted by PRP. This integration is coordinated by a meta-cognitive agent (MC) [25].

Chapter 4

Transformer Based Architectures for Offline Goal Recognition

The fields of automated planning and natural language processing (NLP) share fundamental similarities: both involve processing sequential structures (plans and sentences) composed of discrete elements (actions and words) that must follow domain-specific rules and constraints. Despite this parallelism, identified as early as 2007 [28], the two fields have largely evolved independently. While NLP has been revolutionised by deep learning techniques, particularly Transformer-based architectures like BERT [21], presented in Section 2.2.5, automated planning has seen limited adoption of these methods, primarily using deep learning for heuristic prediction [23] or sensor data processing [6].

This chapter explores the transfer of Transformer-based techniques from NLP to automated planning, specifically for offline goal recognition. We propose treating plans and actions analogously to sentences and words, enabling the application of language modelling techniques to planning domains. The central contribution is **PlanBERT**, a BERT-based model trained on action sequences

using a novel *Planning Language Modeling* (PLM) task, where the model learns to reconstruct incomplete action sequences.

PlanBERT serves as a *foundation model* for planning domains, a large-scale model trained on vast quantities of domain-specific data that can be adapted to various downstream tasks. We demonstrate this versatility through four evaluation tasks: Planning Language Modelling (reconstructing masked actions), Next Token Prediction (predicting subsequent actions), Previous Token Prediction (predicting preceding actions), and Offline Goal Recognition (identifying agent objectives from observations).

This chapter discusses and further develops the work originally presented in [68]. The base of this work is also reported in the doctoral thesis of Mattia Chiari [18]. The overlap between the two theses is the result of a collaborative research effort: the work was initiated by Mattia Chiari, who proposed the overall research direction and the application of Transformer-based architectures to automated planning as part of a broader research agenda on deep learning for goal recognition, and who supervised the author of this thesis during the project whilst working as a doctoral and post-doctoral researcher.

The contributions of this chapter are divided between the two researchers as follows. The entirety of the PlanBERT model, comprising the formalisation of the Planning Language Modelling (PLM) task, the design of the pre-training procedure, the fine-tuning strategy for Offline Goal Recognition, and the experimental evaluation across the PLM, Next Token Prediction (NTP), Previous Token Prediction (PTP), and Goal Recognition tasks — was designed and executed by the author of this thesis. The comparative evaluation against GRNET was conducted by Mattia Chiari, who combined the Goal Recognition results reported in [19] with the BERT results produced by the author of this thesis to form an unified experimental comparison, extended for this thesis by adding the compar-

ison to LGR. As original contribution of this thesis, a new experiment on noise is performed. These results can be found in Section 7.1.

4.1 PlanBERT: BERT for Automated Planning

Having established the parallelism between planning and NLP, we now present PlanBERT, our adaptation of the BERT architecture to automated planning domains. Adapting the Masked Language Modelling task for our use case, we conceptualised the *Planning Language Modeling* (PLM), a new task to train and test the BERT model on planning knowledge. This is the first step to develop a *foundational model* that can be fine-tuned to different down-stream planning tasks, as in this work, Offline Goal Recognition.

4.1.1 Planning Language Modelling

In order to perform *Planning Language Modelling*, we approach the planning domains as an NLP task. In Figure 4.1, we show a simplified representation of the model architecture performing the Planning Language Modelling task.

Task Definition In this technique, given a sequence of actions π composed of n actions a_i , with $i \in [1, n]$, we split each action into separate tokens, dividing the action name and the action’s predicates using the WordPiece tokenizer [83]. Next, we substitute a certain percentage of tokens with the special token [MASK] and provide this incomplete sequence as input to the model. The BERT model has the task of predicting the missing token from the overall context of the action sequence. If the model is capable of performing this operation, it is likely that it has the ability to understand how the planning domain works, the impact of the actions when they are performed by an agent, and the effects they have.

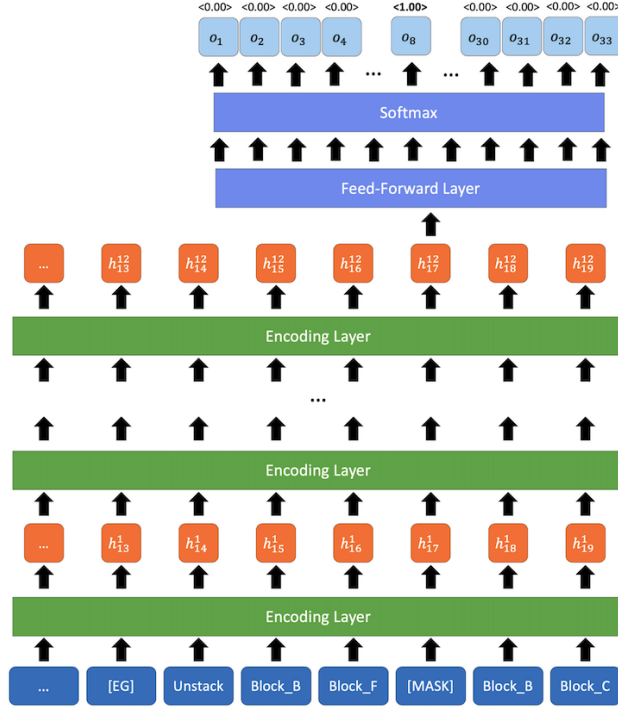


Figure 4.1: Schematic representation of a BERT model performing the Planning Language Modeling task. The input sequence (initial state, goal, and action plan with one masked token) is processed through the architecture. Analysing the masked token’s representation, it predicts the correct masked token from the domain vocabulary.

Training Details Considering that the input sequence of actions π is a solution to a classical planning problem $P = \langle F, I, A, G \rangle$, we include the initial state I and the goal G in the input sequence, as they are crucial for correctly understanding the context. For this reason, we have added two new special tokens, [EIS] (End of Initial State) and [EG] (End of Goal), to help the model understand when the initial state and the goal sequences end. On the other hand, we do not want the model to learn to predict the initial state or the goal; for this

reason, during training, we use the [MASK] token only in the action sequence (i.e., after the [EG] token).

The general structure of the vocabulary for a planning domain consists of tokens for the ground objects that can appear in the domain, tokens for the possible actions, tokens for the possible predicates, and the special tokens [MASK], [EIS], and [EG].

The tokenised input sequences are processed by the embedding layer and by all the encoding layers, producing a final representation of each token as a vector. When a token is masked, its embedding vector is passed to a feed-forward layer with neurons corresponding to each possible token in the vocabulary, using the softmax activation function. The result produced by the feed-forward layer is compared with the desired output. This procedure is repeated for every training instance, evaluating the overall error made by the architecture and calculating a loss function, which will be used by the backpropagation algorithm to adjust the model weights.

Considerations Although BERT models typically treat a specific language, there are several multilingual models. Given that a planning domain has its own set of actions, predicates, and rules, we can consider it a sort of independent language. Therefore, the user should evaluate whether to train a BERT model for each domain, perhaps specialising and improving its predictive capabilities, or to build a multi-domain model by selecting which domains to consider and their quantity. To properly evaluate the capabilities in each domain separately, we opted to train a BERT model for each domain.

Next, an important constraint of the BERT model is that it must have a pre-defined vocabulary of tokens. Therefore, before the training process, the user must define a maximum number of ground objects that the model can handle in

each domain. Although this process makes the model sensitive to the chosen object names, in practice, this operation only affects the selection of the maximum number of objects. In fact, a mapping algorithm can be easily implemented to translate new names into predefined ones.

Example 3. To illustrate the planning language modelling process concretely, we present a complete example using the BLOCKSWORLD domain. We assume that our plans involve at most 22 blocks. In this case, our vocabulary consists of 22 tokens for the blocks (i.e., a different name for each block), 4 tokens for the possible actions (i.e., Stack, Unstack, Pick-Up, Put-Down), 4 tokens for the possible predicates (i.e., clear, on-table, holding, arm-empty), and 3 additional tokens (i.e., [MASK], [EIS], [EG]), for a total of 33 different tokens.

Consider the planning problem illustrated in Figure 4.2. The initial state, reported in Figure 4.2a, is $I = \langle (\text{on-table Block_F}), (\text{on-table Block_C}), (\text{on Block_B Block_F}), (\text{arm-empty}) \rangle$. The goal, reported in Figure 4.2b is $G = \langle (\text{on Block_B Block_C}) \rangle$. Finally, Figure 4.2c reports the solution plan $\pi = \langle (\text{Unstack Block_B Block_F}), (\text{Stack Block_B Block_C}) \rangle$.

Following the tokenization process described in Section 4.1.1, each action is split into separate tokens dividing the action name and the action’s predicates. For instance, the action $(\text{Stack Block_A Block_B})$ is divided into three separate tokens: Stack, Block_A and Block_B.

Therefore, the tokenized input sequence is “on-table” “Block_F” “on-table” “Block_C” “on” “Block_B” “Block_F” “arm-empty” “[EIS]” “on” “Block_B” “Block_C” “[EG]” “Unstack” “Block_B” “Block_F” “Stack” “Block_B” “Block_C”. Next, we mask 1 token after [EG] (17%), chosen randomly; in this example the Stack token is replaced by [MASK].

This sequence is then processed by the embedding layer and by all the encoding layers, producing a final representation of each token as a vector. Given

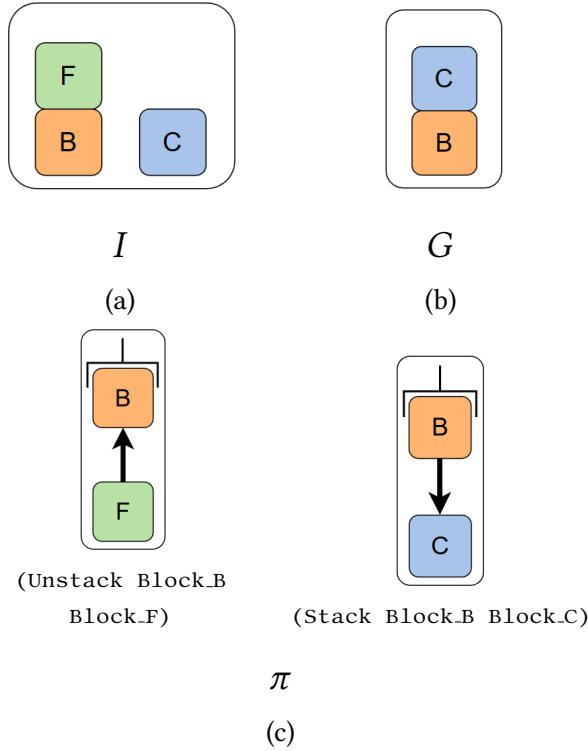


Figure 4.2: (a) The initial state I showing Block_B stacked on Block_F, with Block_C and Block_F on the table. (b) The goal G requiring Block_B to be stacked on Block_C. (c) A solution plan $\pi = \langle (\text{Unstack Block_B Block_F}), (\text{Stack Block_B Block_C}) \rangle$ that achieves the goal.

that the fourth element of the action sequence is masked, its embedding vector is passed to a feed-forward layer composed of 33 neurons (one for each possible token of the domain) with the softmax activation function. Supposing that the output neuron corresponding to the Stack token has index 8, we want in output a vector formed by all zeroes except for a 1 at index 8. Therefore, the result produced by the feed-forward layer is compared with the desired output, and the cross-entropy loss is calculated to adjust the model weights through back

propagation.

4.1.2 Goal Recognition Fine-tuning

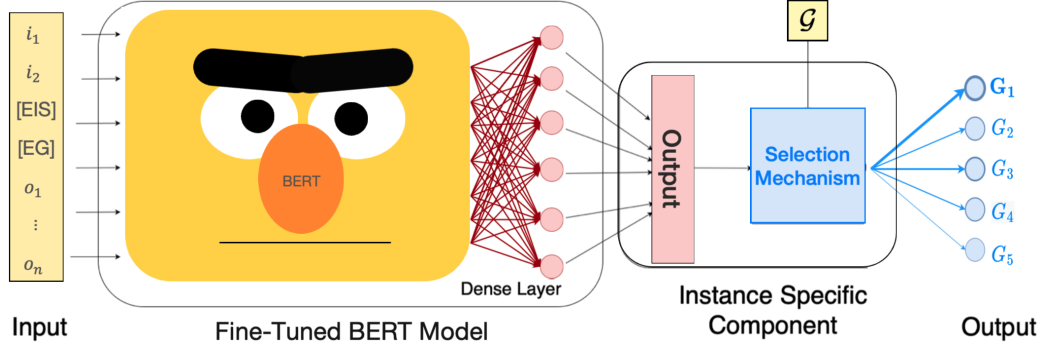


Figure 4.3: Architecture for goal recognition using a fine-tuned BERT model. The system receives as input the initial state I , an empty goal placeholder, and the observation sequence O . The fine-tuned BERT model outputs $|F|$ neurons (one per fluent in the domain), each representing the likelihood that the corresponding fluent is part of the true goal. Then, these fluent-level predictions are aggregated to score each candidate goal in $\mathcal{G}$, with the highest-scoring goal selected as the prediction.

As stated in Section 2.2.5, BERT models can be fine-tuned by replacing the PLM final feed-forward layer with another output layer. In order to perform Offline Goal Recognition, similarly to GRNET, we add an additional feed-forward layer that has $N = |F|$ output neurons with *sigmoid* activation. For the overall working of GRNET, please refer to Section 3.

As in the Pre-Training phase, we trained the model on this task using classical planning instances. Given that the input action sequence π is a valid plan for a classical planning problem P , we chose to include the initial state I in the input, while omitting the goal G , since predicting the goal is precisely the model's

objective. We kept the two special tokens, [EIS] and [EG], but left the goal segment empty. Next, we randomly deleted a specified fraction of actions from the plans, as required by the task definition. After the prediction, the model’s output is a probability for each one of the $|F|$ fluents of the domain, where the i -th output neuron $\bar{o}_i$ is aligned with the i -th fluent f_i , and the activation of $\bar{o}_i$ provides a ranking score for f_i being true in the goal state. These scores are then compared with the true goal G of the instance, and the back-propagation algorithm updates the model weights.

4.1.3 Goal Recognition Inference

The complete architecture is illustrated in Figure 4.3. Similarly to GRNET, it is composed of an Environment Component and an Instance Specific Component. For a goal recognition instance $T = \langle \Pi, I, O, \mathcal{G} \rangle$, the Environment Component takes as input the initial state I and the observation sequence O , where O is tokenised according to the procedure detailed in Section 4.1.1. This input is then processed by the model, and we obtain a ranking indicating how likely it is that f_i holds in the goal. These activation values are then forwarded to the Instance Specific Component, which uses these values to compute a score for each candidate goal from the candidate goal set $\mathcal{G}$:

$$S_{BERT}(G) = \sum_{f \in G} \bar{o}_f \quad (4.1)$$

The goal with the highest score is then selected as the predicted goal G^* .

4.2 Discussion

In this chapter, we introduced the overall architecture and the set of tasks designed to investigate the applicability of transformer-based models to automated planning domains. Specifically, we defined PlanBERT, a BERT-based architecture trained using the Planning Language Modelling (PLM) paradigm to enable the model to capture and reason over automated planning rules, and we proposed a fine-tuning strategy to adapt this model to the Offline Goal Recognition task.

To demonstrate that transformer-based architectures, and BERT in particular, can be effectively adapted to automated planning domains via PLM, we evaluated the model on three distinct tasks: Planning Language Modelling with varying masking ratios, Next Token Prediction (NTP), and Previous Token Prediction (PTP). Subsequently, we assessed the performance of its fine-tuned variant on the Offline Goal Recognition task. These tasks and the corresponding experimental results are reported in detail in Section 7.1.

Chapter 5

Online Goal Recognition through Deep Learning Approaches

The transition from offline to online goal recognition presents fundamental challenges that require rethinking how deep learning architectures process and predict from sequential observations. While traditional offline goal recognition systems can analyse complete observation traces before making a single prediction, online goal recognition (OGR) demands real-time inference capabilities with early convergence; the system must provide accurate predictions as soon as possible, updating them incrementally as new observations arrive. This demands a high-speed system capable of generating predictions within just a few milliseconds. Although BERT was a potential candidate, we chose a more compact and faster architecture built on Recurrent Neural Networks instead.

This chapter explores how deep learning approaches can be adapted and enhanced for OGR tasks. We begin by examining OnlineGRNet, a direct adaptation of the state-of-the-art offline system GRNet [19] to the online setting. While OnlineGRNet demonstrates the computational efficiency advantages of neural ap-

proaches, achieving inference speeds approximately 150 times faster than symbolic planning methods, it also reveals limitations in early convergence when relying solely on data-driven pattern recognition.

To address these limitations, we introduce **CLERNet (Causal Link Enhanced Recurrent Network)**, a novel neuro-symbolic architecture that integrates symbolic planning knowledge directly into both the training process and the inference mechanism. By leveraging causal link structures from planning theory, CLERNet learns not just to recognise action patterns but also to understand the functional role each action plays in achieving specific sub-goals. The work discussed in this chapter is presented in [69].

5.1 GRNET for Online Inference

While the state-of-the-art offline system, GRNet [19], established a strong baseline for using deep learning in goal recognition tasks, its architecture was originally designed to process a complete, although partially observable, sequence and output a single prediction once the agent has finished acting. To address the requirements of online goal recognition (OGR), specifically the need for early convergence and real-time prediction, an adapted version of this architecture, referred to here as OnlineGRNet, must be employed [69]. The core difference between the original implementation of GRNet and its online adaptation lies in the definition of the classification task. The original GRNet frames goal recognition as a “many-to-one” sequential task, where the entire input sequence maps to a single output vector. In the online context, the problem is reframed as a “many-to-many” task, as explained in Example 2.

To facilitate this during the testing and evaluation phases, the observation handling is restructured. Considering a plan π composed of m actions, the system

does not simply wait for the full sequence. Instead, it processes m distinct action traces. It begins with a trace containing only the first action executed by the agent, generating a prediction. It then incrementally adds subsequent actions one at a time, generating updated predictions at each step.

OnlineGRNet serves as a robust baseline using deep learning, and we can see its performance in Section 7.2. It relies primarily on the pattern recognition capabilities of the LSTM without explicitly leveraging domain-specific planning knowledge during the aggregation of its predictions. While OnlineGRNet shows that deep learning is feasible for OGR, a more specialised architecture that can reason on planning knowledge can combine the strengths of neural methods with symbolic reasoning, improving the results of the system while preserving its speed.

5.2 CLERNet (Causal Link Enhanced Recurrent Network)

The idea of integrating neural networks with symbolic systems motivated the development of the **Causal Link Enhanced Recurrent Network (CLERNet)**, introduced in [69]. CLERNet occupies a position between purely model-free and purely model-based goal recognition. Like model-free approaches, it relies on a trained neural network for inference and does not invoke a planner at recognition time, making it orders of magnitude faster than symbolic methods. However, unlike fully model-free approaches, it is *not* domain-agnostic: it incorporates structural knowledge derived from the planning domain model, specifically, causal links, the initial state, and mutex relations, both during training (to construct the labelled dataset) and during inference (in the Aggregation Com-

ponent). We therefore characterise CLERNet as a **neuro-symbolic** architecture: one that combines the computational efficiency and pattern-recognition capacity of deep learning with targeted symbolic knowledge from automated planning theory. Despite the symbolic information, we chose not to include the full PDDL model or explicit state sequences in the input data, because no additional knowledge is required to improve the system’s performance, and doing so could slow down the decision-making process.

5.2.1 Architecture Overview

The architecture of CLERNet is composed of three distinct modules, two of which have already been seen in GRNET and PlanBERT: the Neural Component, the Aggregation Component, and the Selection Component. This modular design allows the system to process raw observations via deep learning while refining the output using symbolic logic.

The Neural Component

The input to the system is an incremental sequence of observed actions. This component employs an embedding layer to transform each discrete action o_i into a vector of real numbers e_i . These vectors are processed by an LSTM layer, which captures sequential dependencies. Crucially, this component is designed for a many-to-many classification task: for every input action, a feed-forward layer outputs a prediction vector y_i . This vector contains a score in the range $[0, 1]$ for every possible fluent in the domain, representing the likelihood that a specific fluent is part of the agent’s goal. As we will discuss in the next section, this component is trained to predict only the goal fluents that are relevant to the last observed action, rather than the entire goal.

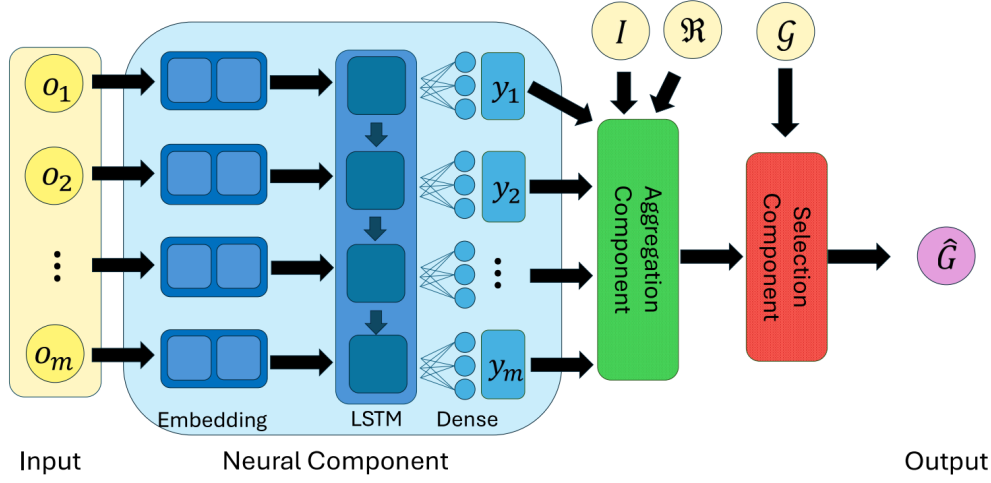


Figure 5.1: Architecture of CLERNet. Each observed action is processed by the Neural Component (in blue), which encodes the observations by embedding vectors, feeds them to an LSTM network and then to a feed-forward layer which, at each step, predicts the agent’s goal scores (y_i). These outputs are then combined together with the initial state (I) and the information related to mutually exclusive fluents ($\mathcal{R}$) by the Aggregation Component (in green). Finally, the Selection Component returns the predicted goal $\hat{G}$ by processing the aggregated score and the set of candidate goals $\mathcal{G}$ (in red).

The Aggregation Component

This component requires no training. It refines the raw predictions from the Neural Component by integrating two sources of symbolic domain knowledge: the Initial State (I) and Mutex relationships ($\mathcal{R}$). The behaviour of this component is formalised in Algorithm 1.

The choice of which domain knowledge to incorporate, and, equally importantly, which to exclude, is deliberate and merits justification.

Initial State. The initial state I provides the system with the baseline config-

uration of the world before the agent begins acting. Fluents that are already true in I are plausible goal candidates even before any action is observed: an agent may simply need to preserve or build upon an existing condition. Initialising the aggregation vector with a positive contribution α for fluents present in I encodes this prior knowledge directly, avoiding the cold-start problem in which all goals receive a score of zero at $t = 0$ regardless of the world state.

Mutex Relations. Mutex relations $\mathcal{R}$ encode pairs of fluents that cannot simultaneously be true in any reachable state of the domain. When the Neural Component predicts a fluent with high confidence, any fluent mutually exclusive with it can be immediately eliminated as a goal candidate, since no valid plan can achieve both. Incorporating mutex relations therefore allows the Aggregation Component to correct early prediction errors in a principled, logic-sound manner without requiring any additional learning.

Causal Links. As detailed in Section 5.2.2, causal links are used during training to label each action with the specific goal fluents it causally supports. This supervision signal teaches the Neural Component to predict not merely which goal is likely, but *which specific sub-goal each observed action contributes to*. Without this structured labelling, the network would only observe a uniform goal label for every action in the sequence, losing the fine-grained causal signal that enables early convergence.

Why not the full PDDL model? One might ask why CLERNet does not simply incorporate the complete PDDL domain model, including all action schemas, preconditions, and effects, as a symbolic oracle during inference. The reason is twofold. First, using a full planner at inference time would negate the computational advantage of the neural approach: the computational cost of planning is precisely what model-free methods are designed to avoid, and symbolic OGR systems such as [82] already represent that paradigm. Second, the three sources

of knowledge selected, initial state, mutex relations, and causal links, are *pre-computable* from the domain model offline, before any recognition takes place. They can therefore be stored as lightweight data structures and queried in constant time during inference, preserving the millisecond-level response speed that real-time OGR demands.

The Selection Component

The final module maps the aggregated fluent scores to the set of candidate goals $\mathcal{G}$. For each candidate goal $G \in \mathcal{G}$, the component sums the scores of the fluents that constitute that goal:

$$S(G) = \sum_{f \in G} a[f] \quad (5.1)$$

The goal with the highest cumulative score is selected as the predicted goal G^* .

5.2.2 Training via Causal Link Labelling

A critical innovation of CLERNet is how the training dataset is constructed. Simple labelling (assigning the final goal label to every action in the plan) is insufficient because different actions contribute to different fluents of a goal. To address this, CLERNet leverages **Causal Links**, structures that define which actions support the preconditions of subsequent actions. Given a plan π , it is possible to identify the set of *causal links* within π , representing the causal relationships between actions [57, 76]. These links indicate that the effect of one supporting action is necessary for the successful execution of another consumer action, without being threatened by any other action in the plan. Using this information, we can iterate backward from the end of a solution plan to rebuild the causal chain and associate each action only with the specific goal fluents it actually affects.

Algorithm 1: Pseudocode of the Aggregation Component. It integrates the initial state and mutex information to refine the prediction vector.

Input: The list of prediction vectors $Y = \langle y_i, \dots, y_n \rangle$, a set of mutually exclusive fluents $\mathcal{R}$, the initial state of the problem I and three numeric constants α, τ_1 and τ_2

Output: An aggregated prediction vector a

```

1  $a = [ ]$ 
2 for  $j \in |F_G|$  do
3   if  $F_G[j] \in I$  then  $a[j] = \alpha$ 
4   else  $a[j] = 0$ 
5 for  $y_i \in Y$  do
6    $t = \{j \mid y_i^j > \tau_1\}$ 
7    $mutex = \{ \}$ 
8   for  $j \in t$  do  $mutex.add(\mathcal{R}(j) \setminus t)$ 
9   for  $m \in mutex$  do  $a[m] = 0$ 
10  for  $j \in y_i$  do
11    if  $y_i^j < \tau_2$  then  $y_i^j = 0$ 
12   $a = a + y_i$ 
13 return  $a$ 

```

Causal links are widely used in Partial Order Planning [85], or in explaining the behaviour of executed plans [74]. In our context, plans are sequences of actions rather than partially ordered sets of actions, as in partial-order planning. With sequential plans, the computation of causal links can be performed in polynomial time [9].

Two labelling strategies are employed to generate effective training data:

- **Cumulative Labelling:** This strategy iterates backward from the end of a

Algorithm 2: Pseudocode of the CUMULATIVE labelling strategy. It assigns labels to actions based on their contribution to goal fluents via causal links.

Input: Planning Problem $\Pi = \{I, G, A\}$ and a totally ordered solution

plan $\pi = \langle a_1, \dots, a_n \rangle$ for Π

Output: A label associated to each action in π

```

1  Goals = G
2   $C\mathcal{L}_\pi = \{(a_i \xrightarrow{e} a_j) \mid i > j, e \in \text{Add}(a_i), e \in \text{Pre}(a_j), \nexists k \text{ s.t. } i < k < j, e \in \text{Del}(a_k)\}$ 
3  for  $i = n$  to 1 do
4      if  $\text{Goals} \cap \text{Add}(a_i) \neq \emptyset$  then
5           $a_i.\text{label} = \text{Goals} \cap \text{Add}(a_i)$ 
6           $\text{Goals} = \text{Goals} \setminus \text{Add}(a_i)$ 
7      else
8           $\text{acts}(a_i) = \{a_j \in A \mid (a_i \xrightarrow{e} a_j) \in C\mathcal{L}_\pi\}$ 
9           $a_i.\text{label} = \bigcup_{b \in \text{acts}(a_i)} b.\text{label}$ 
10     end
11 end
```

solution plan. If an action adds a goal fact, it is labelled with that fact. If an action supports another action (via a causal link), it inherits the labels of *all* the actions it supports. This creates a rich history of goal contributions for every action.

- **Proximity Labeling:** Similar to the cumulative approach, but with a tighter focus. An action only inherits the label of the *closest* action it supports in the causal chain.

Algorithm 2 describes the CUMULATIVE labelling strategy. It receives as input

a problem Π and a corresponding solution plan π . Without loss of generality, we assume that π is totally ordered; hence, each action of π is associated with an integer time step; in the case of partially ordered plans, the system can simply derive a total order from them. At line 2, all causal links are computed; then the **for** cycle (line 3) examines in reverse order all the actions of π to identify for each of them a set of label fluents that can be used during the training phase. In particular, if some additive effects of the current action are included in *Goals*, then the label of this action corresponds to these facts (line 5), and we remove them from *Goals*. Otherwise, we identify all the actions with a precondition supported by the current action (line 8), and we define the label of this action as the union of the labels of the previously identified actions; note that these labels correspond to goal facts already satisfied by actions causally ordered with the current action.

The algorithm for the PROXIMITY maintains the same structure as the previous one, but its selection procedure, at line 8, selects only the causally ordered action closest to the current one, i.e., $acts(a_i) = \operatorname{argmin}_j \{a_j \in A \mid (a_j \xrightarrow{e} a_i) \in C\mathcal{L}_\pi\}$, and in this case, the label of the current action corresponds to the label of the closest action in the causal links. The labels take the form of binary vectors, in which entries set to one correspond, via causal links, to the goal fluents that are relevant for a given action.

By training on datasets annotated with these strategies, the Neural Component learns not just to associate actions with goals but also to understand the functional role each action plays in achieving specific sub-goals. Experimental evaluations demonstrate that this planning-informed approach allows CLERNet to outperform both the standard GRNet and state-of-the-art methods like ORL in terms of convergence speed and recognition accuracy.

Example 4. Considering the problem described in Example 2, in Figure 5.2 we

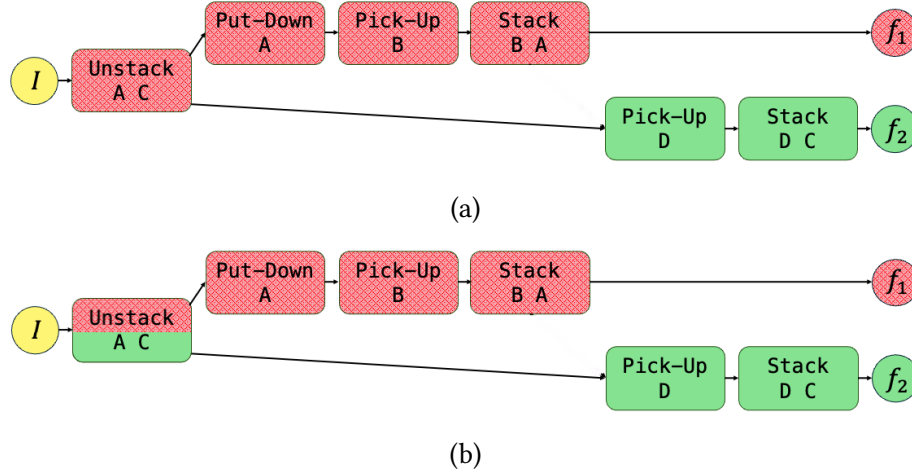


Figure 5.2: Visualisation of the PROXIMITY (a) and CUMULATIVE (b) labeling strategies for our example. Each rectangle represents an action and their colour represents which goal fluents are the label assigned to each action (red for f_1 , green for f_2). Each line represents a causal link.

show the label assigned by the PROXIMITY (5.2a) and CUMULATIVE (5.2b) strategies. In the figure, we show the initial state I , the actions progressively observed by the OGR system (represented as rectangles), and the fluents (represented as circles) belonging to the correct goal G_1 : $f_1 = (\text{On Block_B Block_A})$ (in red) and $f_2 = (\text{On Block_D Block_C})$ (in green). Each line between two actions or between an action and a fluent represents a causal link. Exploiting this information, we can label each action with one or more goal fluents that they achieve. If a causal link involves a goal fact, we assign that fluent as the label of the supporter action. For example, since the action (`Stack Block_B Block_A`) is the supporter action of f_1 , we assign f_1 as its label (in red). Recursively, the actions that have a causal link with (`Stack Block_B Block_A`) (such as (`Pick-Up Block_B`)) have f_1 as label too. The same happens with those that have a causal link with (`Pick-Up Block_B`) and so on. The main difference between the

CUMULATIVE and PROXIMITY strategies is on how labels are assigned to those actions that have a causal link leading to two different fluents. This is the case of (Unstack Block_A Block_C) which allows the agent to put Block_A on the table ((Put-Down Block_A)) and stack Block_B on top of it in order to achieve f_1 , as well to remove an object over Block_C, allowing the agent to pick-up Block_D to put on top of Block_C for achieving f_2 . With CUMULATIVE (Figure 5.2b) both fluents are assigned as a label. Instead, with PROXIMITY only f_1 is assigned because the action Put-Down A, which has label f_1 , is closer to (Unstack Block_A Block_C) than action (Pick-Up D), which has label f_2 .

The labels are structured as binary vectors. In our example, that involves at most 25 goal fluents, an index between 0 and 24 is assigned to each fluent and a vector of dimension 25 is created. Let's assume that f_1 has index 8. The action (Put-Down Block_A) and all others connected with f_1 will have as a label a vector with all zeroes except a 1 in position 8. With the CUMULATIVE strategy, if an action has two or more labels, the vector will contain a 1 for each label.

An intuition to how our system works is given in Figure 5.3, which shows the scores (calculated by the Neural Component and the Aggregation component) for each candidate goal. On the left, we show the score calculated for $G_1 = \langle (\text{On Block}_B \text{ Block}_A), (\text{On Block}_D \text{ Block}_C) \rangle$ (which is the correct goal of the agent), whereas on the right we show the score of $G_2 = \langle (\text{On Block}_D \text{ Block}_A), (\text{On Block}_B \text{ Block}_C) \rangle$. On the y-axis we have the initial state of our OGR problem and the actions progressively seen by our system. On the x-axis, we have the score given by the Aggregation Component for both fluents. As we explained in Section 5.2.1, this score is calculated incrementally, therefore we show the progressive contribution of each action to the prediction. We can see that the first three actions do not provide a valuable

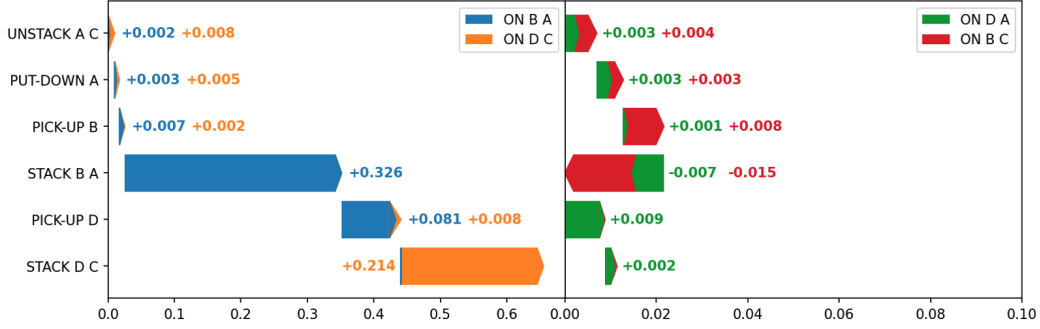


Figure 5.3: Visualisation of how the scores computed by the Neural and Aggregation components of CLERNet evolve incrementally over time as new actions are observed, for the running example of Example 2. Each row on the y -axis corresponds to a successive observation (initial state followed by actions in execution order); the x -axis shows the cumulative contribution of each observation to the fluent scores for each candidate goal (G_1 on the left, G_2 on the right). Arrows indicate the direction and magnitude of the score update produced by each individual action, illustrating how the system progressively converges towards the correct goal G_1 .

contribution to any of the goals. Instead, (Stack Block_B Block_A) makes the Neural Component and the Aggregation Component change their predictions and lean towards G_1 , recognising that such action is fundamental for (On Block_B Block_A). Instead, this action has a negative effect on the score of G_2 . This is due to the fact that (On Block_B Block_A) is predicted as *true* (i.e. with a score greater than τ_1). This causes the fluents (On Block_D Block_A) and (On Block_B Block_C) in G_2 , which are mutually exclusive with (On Block_B Block_A), to be set to 0. As expected, the remaining actions of the plan (in particular, (Stack Block_D Block_C)) increment the score of G_1 , which will be correctly chosen by the Selection Component.

5.3 Discussion

In this chapter, we transitioned the focus from an offline to an online setting by addressing the task of Online Goal Recognition. Specifically, we adapted the state-of-the-art MFGR architecture for Offline Goal Recognition, GRNET, to operate in an online scenario; its performance thus serves as a baseline for the evaluation of novel approaches. Subsequently, we introduced CLERNET, an architecture that integrates both neural and symbolic paradigms to perform Online Goal Recognition. We proposed a novel training methodology that exploits Causal Links to inform the neural network about the relationships between actions and goal fluents. Furthermore, we developed a new mechanism for incorporating relevant planning information to assist the model in selecting the correct goal. We evaluated OnlineGRNet and CLERNET against the state-of-the-art MBGR system ORL [82]. The corresponding experimental results are presented in Section 7.2.2.

Chapter 6

Deceptive Online Goal Recognition

Traditionally, OGR approaches operate under the *assumption of rationality*. They posit a cooperative or neutral setting where the observing agent has access to a reliable stream of observations, and the acting agent behaves rationally, typically optimally or near-optimally, to achieve its goal. Existing state-of-the-art systems, whether based on symbolic planning like ORL [82] or deep learning architectures like CLERNET [69], rely heavily on these assumptions to deliver high-accuracy predictions.

However, the transition from controlled experimental environments to real-world deployment exposes OGR systems to non-ideal conditions. In practical scenarios, the observation channel may be noisy, or the observed agent may be adversarial. We categorise these adversarial conditions into two distinct phenomena: *Interference* and *Deception*.

In an **Interference** scenario, the integrity of the observation channel is compromised. A third party, or environmental noise, tampers with the sequence of

actions perceived by the observer. This may involve the deletion of critical steps (sensor failure), the insertion of spurious noise (sensor hallucinations), or the replacement of actions, thereby disrupting the causal links that the OGR system relies upon.

Conversely, in a **Deception** scenario, the acting agent itself is adversarial. Rather than minimising cost, a deceptive agent actively modifies its plan to mislead the observer. Drawing from the literature on privacy-preserving planning, these agents may employ strategies such as *simulation* (feigning a false goal) or *dissimulation* (hiding the true goal until the last possible moment) [48, 61].

While the literature has addressed noisy observations and deceptive planning in isolation, there is a lack of comprehensive analysis regarding how modern OGR architectures withstand these adversarial conditions and how to effectively mitigate them in real-time. The performance of systems like ORL and CLERNet is known to degrade significantly when the premise of optimality or observational integrity is violated; yet, the extent of this vulnerability and the means to counteract it remain open research questions.

In this chapter, we address these challenges by formalising a taxonomy of attacks against OGR systems and proposing a robust defence framework. We introduce **PAC-OGR** (Pre-processing and Analysis for Clean OGR), a middleware framework designed to sanitise observation streams and mitigate the effects of adversarial planning before goal recognition takes place. The work presented in this chapter has been accepted for publication at ICAPS 2026 [70].

6.1 Adversarial Scenarios: Attacks on OGR

To evaluate the robustness of OGR systems, we define a comprehensive taxonomy of attacks divided into two main categories: Interference (third-party tam-

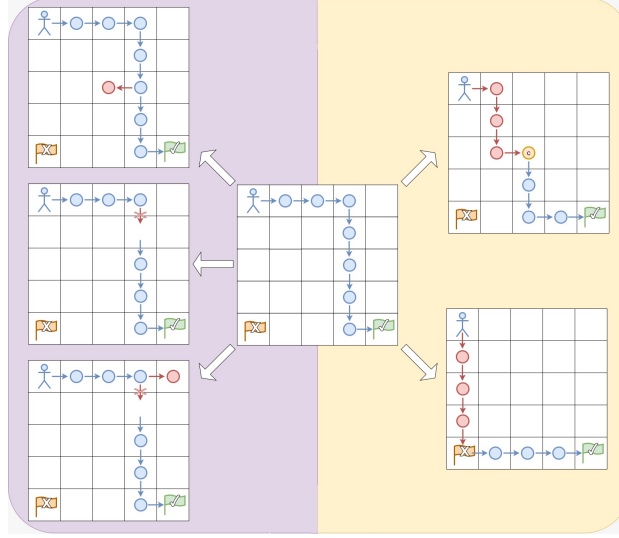


Figure 6.1: Visual illustration of attack vectors against OGR systems. Left: Interference attacks where a third-party intruder manipulates the observation stream through Insert (adding spurious actions), Remove (deleting valid actions), or Replace (substituting actions) operations. Right: Deception attacks where the agent itself employs Centroid (navigating toward states compatible with multiple goals) or Landmark-based strategies (initially pursuing a deceptive goal) to obfuscate its true objective.

pering of the observation channel) and Deception (adversarial behaviour by the observed agent itself).

6.1.1 Interference

A standard OGR system considers two actors: an Agent performing actions to reach a goal and an Observer that analyses the agent’s behaviour to infer its goal. In our Interference scenario, we consider an additional malevolent agent, called the *Intruder*. We assume that the Intruder has full knowledge of the Agent’s

action models and of the environment (plan states), just like the Observer. The Intruder can tamper with the observations received by the Observer without directly interacting with the Agent.

Therefore, while the Agent executes a plan π that achieves the correct goal G^* , the observation sequence O perceived by the Observer may differ from the true one. The observation sequence O is generated incrementally as the Agent performs actions. Consequently, any corruption introduced at observation o_i persists for all subsequent observations. After each action is performed, the Intruder may attempt an attack with a certain probability p . These attacks are depicted in the left column of Figure 6.1.

Insert attack. The Intruder injects a spurious observation (action) into O . The inserted action is drawn from $\mathcal{A}$, but it may be inapplicable in the state in which it is inserted.

Remove attack. The Intruder prevents some actions from being added to O , resulting in an incomplete observation sequence (the Observer is unaware of these missing observations).

Replace attack. The Intruder substitutes a valid observation in O with a fake one selected from $\mathcal{A}$.

6.1.2 Deception

In the Deception scenario, the OGR framework involves two actors: the Agent and the Observer. Unlike the Interference scenario, here the Agent itself is malicious and actively attempts to hide its true goal from the Observer for as long as possible. To this end, the Agent applies goal-obfuscating strategies that influence its action choices. Note that, despite these strategies, the Agent must act within the constraints of the environment and cannot interact with or tamper with the Observer. Thus, the observation sequence O fully reflects the actions executed

by the Agent.

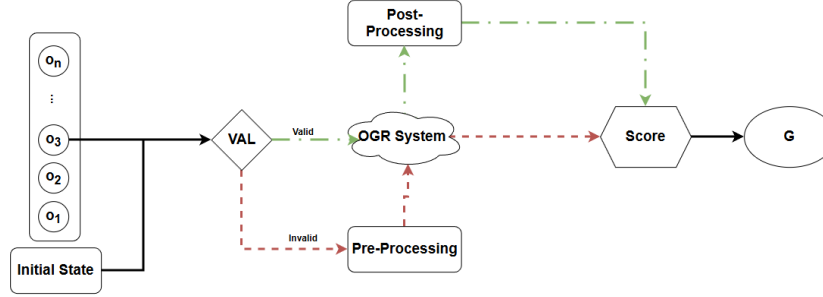


Figure 6.2: Architecture of the PAC-OGR framework. The system wraps existing OGR methods (such as ORL or CLERNET) with two defensive layers: pre-processing filters tampered observations using precondition-effect analysis, and post-processing re-scales goal predictions using heuristic trend analysis.

We consider two types of deceptive strategies: landmark and Centroid attacks.

Landmarks. The landmark-based attack was introduced in [61]. We adopt the *Most Similar Landmarks Goal* approach ($D\mathcal{L}_{Similar}$), as it has been shown to yield the strongest deception effects. The Agent initially executes a plan that achieves a wrong goal $G^- \in \mathcal{G} \setminus G^*$, chosen to maximise the landmark overlap of G^- and the true goal G^* . Specifically, for each candidate goal in $\mathcal{G} \setminus G^*$, we compute the set of its landmarks and select G^- as the goal whose landmarks have the largest overlap with those for G^* . The malevolent Agent executes a plan reaching G^- and then proceeds toward G^* , thereby misleading the Observer regarding the true target at least until it reaches G^- .

Centroid. A centroid C for an OGR task is defined as the state minimising the average estimated cost (here, the number of needed actions) of reaching each goal in $\mathcal{G}$ [60]. In this attack, the Agent first navigates toward C , and then continues to the true goal G^* . This delays explicit commitment to G^* , keeping the

observation sequence consistent with multiple goal hypotheses. Following [61], we adopt the *DC* strategy. To reduce computational cost and improve deception, we restrict the centroid computation to a subset $\mathcal{G}_C \subseteq \mathcal{G}$ of fixed cardinality (in our experiments, $|\mathcal{G}_C| = 3$). The subset is constructed by ranking goals according to their similarity with G^* and selecting the top $|\mathcal{G}_C|$ goals. We use the landmark overlap measure introduced for the Landmarks attack as the similarity measure. Note that G^* is always included in $\mathcal{G}_C$.

6.2 Enhancing OGR Systems: PAC-OGR

To mitigate the adversarial effects of Interference and Deception attacks, we introduce the *PAC-OGR* (Pre-processing and Analysis for Clean OGR) framework. PAC-OGR acts as a wrapper around existing Goal Recognition systems, pre-processing the observation stream to remove corrupted actions and re-scaling the output scores using symbolic domain knowledge. The complete system architecture is presented in Figure 6.2. As illustrated, the input sequence is initially evaluated with respect to its validity. If deemed valid, it is forwarded directly to the OGR system for processing and subsequently analysed by the post-processing component. Conversely, if classified as invalid, it is first parsed by the pre-processing component before being passed to the OGR system.

6.2.1 Pre-Processing

The pre-processing component addresses Interference attacks by filtering potentially tampered actions from the observation sequence.

This process is performed every time a new observation is added to $\mathcal{O}$. We first validate the observation sequence using VAL [40]. If the sequence is found to be invalid, indicating potential tampering, we invoke a custom action-validation

algorithm.

Given the initial state I and a sequence $O = \langle o_1, \dots, o_k \rangle$ of observed actions, we check the authenticity of each action o_i as follows. For every $o_i \in O$, we compute two scores: P_i and E_i . P_i indicates the percentage of preconditions of o_i that are satisfied in the state s_{i-1} generated by executing $O_i^- = \langle o_1, \dots, o_{i-1} \rangle$. Note that we construct s_{i-1} by applying all effects of the actions in O_i^- starting from I , without checking whether their preconditions are satisfied, and we calculate the percentage of preconditions of o_i that hold in that state. E_i indicates the percentage of effects of o_i that appear as preconditions for at least one subsequent action in $O_i^+ = \langle o_{i+1}, \dots, o_k \rangle$. These scores are motivated by the observation that tampered actions, which do not belong to the Agent's original plan, are unlikely to have their preconditions supported by prior actions or to produce effects that enable future actions. For instance, if action o_i has 4 preconditions and 3 are satisfied in s_{i-1} , then $P_i = 0.75$. We compare these scores against two thresholds ϕ_p and ϕ_e , optimised on a validation set, to classify actions as follows. If $P_i > \phi_p$, we consider o_i authentic, as most of its preconditions are satisfied. Otherwise, we check E_i : if $E_i > \phi_e$, we consider o_i authentic, since its effects contribute to the remaining execution. If neither condition holds, o_i is *removed* from O .

6.2.2 Post-Processing

The post-processing component addresses Deception attacks by analysing the trajectory of heuristic values to detect inconsistent goal hypotheses.

The post-processing component of PAC-OGR evaluates and re-scales the predictions made by the OGR system, which assigns a numeric score to each candidate goal entirely unaware of potential attacks. The idea is to examine the sequence of states generated by the observed actions to assess their coherence with each candidate goal and weight the OGR predictions accordingly. To do

this, we exploit the function $f(s) = g(s) + h(s)$ used in standard heuristic search to evaluate a search state s . Specifically, for every candidate goal G , we compute $f^G(s_i) = i + h^G(s_i)$, where i is the length of the observation sequence $\langle o_1, \dots, o_i \rangle$ (i.e., the number of observed actions executed by the Agent so far), s_i is the state generated at the end of the sequence, and $h^G(s_i)$ is a heuristic estimate of the distance from s_i to G . Thus, for a goal G , we obtain the sequence $F_i^G = \langle f^G(s_1), \dots, f^G(s_i) \rangle$.

Our method aims to penalise candidate goals for which the heuristic distances tend to increase over time, as this suggests that the Agent is moving away from them. However, the sequence of heuristic distances $\langle h^G(s_1), \dots, h^G(s_i) \rangle$ alone would not always be informative enough, as it may exhibit plateaus or small fluctuations even when the Agent is actually pursuing G . By considering the f^G values instead, we also incorporate the length of the observation sequence, and a lack of heuristic-distance increase can be compensated by the (increasing) first term of f .

The trend of the f_G values can be quantified by computing a linear regression over the sequence F^G in the form $y = ax + b$, and using the slope a as an indicator of goal deception: the larger a is, the less likely G the true goal is.

To ensure that a captures a reliable trend, the regression is performed only over the most recent portion of the sequence that exhibits a sufficiently strong linear correlation, measured using Pearson's coefficient. Starting from the first observation, earlier values are iteratively discarded until the absolute value of the coefficient exceeds a threshold ϵ , defined through a validation set. Finally, we integrate this slope information with the score s_G^{OGR} provided by the underlying OGR system. To combine the slope information with the OGR predictions, we define a penalty function based on the arctangent transformation. Specifically,

for each candidate goal G , we compute its final score as:

$$s_G = s_G^{OGR} \cdot \left(1 - \frac{\arctan(a)}{\pi/2}\right),$$

where negative slopes a are set to 0. The arctangent function maps the slope to the range $[0, 1]$, ensuring that larger positive slopes result in greater penalties. This penalises goals whose f^G trends suggest the Agent is moving away from them, making them increasingly implausible.

6.3 Discussion

This chapter investigates the robustness of Online Goal Recognition (OGR) under adversarial and non-ideal conditions, thereby challenging the standard assumptions of perfectly rational agents and reliable observations. We introduce a unified taxonomy of adversarial scenarios, distinguishing between Interference attacks and Deception strategies. To address these vulnerabilities, we propose PAC-OGR, a modular middleware framework that enhances existing OGR methods without requiring retraining or architectural changes. PAC-OGR combines a symbolic pre-processing filter that detects and removes corrupted observations with a post-processing module that mitigates deceptive behaviour through heuristic trend analysis. To validate our claims, we first assess the robustness of two state-of-the-art Online Goal Recognition systems, CLERNet and ORL, under the proposed adversarial scenarios. We then analyse the corresponding performance degradation induced by these attacks. Finally, we introduce PAC-OGR as an enhancement layer and evaluate its effectiveness in improving their robustness. The experimental setup and results are reported in [7.3](#).

Chapter 7

Experimental Results

7.1 Experimental Evaluation of PlanBERT

In this section, we evaluate PlanBERT’s performance across four tasks: Planning Language Modeling (PLM), Next Token Prediction (NTP), Previous Token Prediction (PTP), and Goal Recognition (GR). We demonstrate that PlanBERT learns meaningful representations of planning domain dynamics and achieves competitive or superior performance compared to domain-specific architectures. Furthermore, we assess its robustness when observation sequences are corrupted by spurious actions.

7.1.1 Benchmark Suite and Datasets

Domain Descriptions. We consider six well-known benchmark domains [47, 51]:

- **BLOCKSWORLD.** The domain consists of a hand robot that has to stack or unstack blocks, picking them up one at a time, in order to obtain a desired configuration of an available set of blocks.

- **DEPOTS.** The domain consists of actions for loading and unloading packages into trucks through hoists, and moving them between depots. The goals concern having the packages at certain depots.
- **DRIVELOG.** In this domain there are drivers (trucks) that can walk (drive) between locations. Walking and driving require traversing different paths. Packages can be loaded into or unloaded from trucks, that can be moved by drivers. The goal is to deliver (move) all packages (drivers) to their destinations.
- **LOGISTICS.** In this domain there are aircrafts that can fly between cities, trucks that can move between locations within a city, and packages that can be loaded into/unloaded from trucks and aircrafts. The goal is to deliver a set of packages to their delivery locations.
- **SATELLITE.** This is a scheduling domain in which one or more satellites can make certain observations, collect data, and download the collected data to a ground station. The goals concern having observation data at a ground station.
- **ZENOTRAVEL.** This is another variant of a transportation domain where passengers have to be embarked and disembarked onto aircraft that can fly between cities at two possible speeds. The goals concern transporting (move) all passengers (aircraft) to their required destinations.

Training Sets

Concerning the training set, for each domain, we randomly generated a large collection of (solvable) plan generation problems of different size. We considered the same ranges of the numbers of involved objects as in the experiments of

Domain	$ A $	$ F $	$ G_i $	$ \mathcal{G} $
BLOCKSWORLD	968	506	[4,16]	[19,21]
DEPOTS	13050	150	[2,8]	[7,10]
DRIVELOG	4860	156	[4,11]	[6,10]
LOGISTICS	15154	154	[2,4]	[10,12]
SATELLITE	33225	629	[4,9]	[6,8]
ZENOTRAVEL	23724	66	[5,9]	[6,11]

Table 7.1: Size of A , F , $G_i \in \mathcal{G}$ and $\mathcal{G}$ in the considered GR instances for each domain. Interval $[x, y]$ indicates a range of integer values from x to y .

Pereira et al. [59], shown in Table 7.1. For each of these problems, we computed up to four (sub-optimal) plans solving them using LPG [29, 30]. The generated training set consists of input sequences $\langle I, G, \pi \rangle$ where I is the initial state, G is the goal, and π is a plan leading from I to G . Table 7.2 reports the training set statistics for each domain. The overall size of our dataset is approximately 350 MB, substantially smaller than the dataset (~ 3 TB) used for training standard BERT architectures [21, 46].

Fine-tuning Dataset. For the goal recognition fine-tuning, for each domain we randomly selected 25k plans and derived observation sequences by randomly selecting actions from the plans (preserving their relative order), retaining between 30% and 70% of the plan actions. The generated dataset consists of pairs $(\langle I, O \rangle, G^*)$ where O is a sequence of observed actions obtained by sampling a plan π , and G^* is the corresponding hidden goal.

Domain	Training Set	Vocabulary	Avg.	Avg.
	Size	Size	Tokens	Actions
BLOCKSWORLD	200k	40	171.3	39.2
DEPOTS	192k	50	243.2	32.9
DRIVELOG	239k	84	204.5	30.9
LOGISTICS	225k	66	187.8	28.8
SATELLITE	154k	83	114.3	17.5
ZENOTRAVEL	252k	41	170.8	22.1

Table 7.2: Number of training plans, vocabulary size, average number of input tokens (Avg. Tokens) and average plan length (Avg. Actions) for each domain.

Architecture. Our model architecture follows Devlin et al. [21]: 12 encoding layers, each with 12 attention heads, and an embedding size of 768. We used *cross-entropy* as the loss function. For the PLM task, an additional output layer of size $N = |V|$ is added on top, where $|V|$ is the dimension of the vocabulary ; for Offline Goal Recognition, this is replaced by a layer of size $N = |F|$.

Evaluation Tasks and Test Sets

PLM, NTP, and PTP Test Set. The first test set comprises 2,000 plans withheld from training. The PLM task is evaluated using masked token percentages of 10%, 15%, 25%, and 50%. For NTP, given a token sequence of length l , we pass the first $l - 1$ tokens to the model and ask it to predict the l -th token. For PTP, the model predicts the first token given the remainder. Both tasks are evaluated at sequence lengths of 5, 10, 25, 50, and 100 tokens, and over the full plan (*Tot*). The initial state and goal are always included in the input; sequence length counts only tokens after the [EG] token.

Masked (%)	BLOCKS	DEPOTS	DRIVELOG	LOGISTICS	SATELLITE	ZENO
10	99.47	99.38	98.99	98.53	96.10	99.54
15	99.36	99.22	98.82	98.31	95.95	99.49
25	98.90	98.88	98.00	97.45	95.24	99.05
50	96.14	96.62	94.23	92.40	90.08	95.66

Table 7.3: PLM task accuracy. Rows report the percentage of masked tokens in the test sequences.

Offline Goal Recognition Test Set. For goal recognition evaluation, we use TS_{LGRGen} , introduced in [19], which extends TS_{LGR} [59] with a richer diversification of observation traces and a larger number of test instances. We consider action traces formed by 10%, 30%, 50%, 70%, and 100% of the plan actions, and use GR accuracy as the evaluation metric.

7.1.2 Results and Analysis

This analysis was conducted by me and included in the thesis of Chiari [18]. The results for PLM, NTP, and PTP are identical, whereas the Offline Goal Recognition comparison was originally carried out by Chiari and further extended by me in this thesis through the inclusion of LGR. In addition, a new noise-focused experiment was introduced as an original contribution of this work.

Planning Language Modeling Task. Table 7.3 shows accuracy on the PLM task across all domains and masking percentages. As expected, accuracy decreases as more tokens are masked. Performance is consistently above 90% in all domains and masking configurations, which is remarkable. In BLOCKSWORLD, DEPOTS, and ZENOTRAVEL, accuracy exceeds 95% even when half the tokens are

Seq. Length	BLOCKS	DEPOTS	DRIVELOG	LOGISTICS	SATELLITE	ZENO
5	99.9	84.85	36.70	21.40	45.60	58.80
10	80.85	67.55	68.10	32.00	56.45	51.85
25	82.30	74.10	71.80	70.50	74.55	67.10
50	81.65	74.30	80.60	76.80	78.65	75.55
100	94.28	90.95	85.00	90.95	79.40	89.10
<i>Tot</i>	100.0	100.0	100.0	100.0	100.0	100.0

Table 7.4: NTP task accuracy. Sequence Length counts tokens after [EG]. Row *Tot* uses the full plan minus the last token.

masked. The relatively lower performance in SATELLITE is due to the smaller training set and shorter average plan length in that domain (see Section 7.1.4).

Next Token Prediction Task. The NTP results are shown in Table 7.4. Accuracy increases with sequence length in all domains, with the exception of BLOCKSWORLD and DEPOTS at length 5, where the limited number of applicable first actions constrains the space of valid continuations and makes the task locally easier. The jump of approximately 15 accuracy points from 50 to 100 tokens (in all domains except SATELLITE) confirms that PlanBERT exploits the additional context to strongly constrain the prediction. When the full plan is given minus the last token, 100% accuracy is achieved in every domain.

It is worth noting that the NTP task is inherently ambiguous in domains where goals are composed of independent sub-goals. When multiple sub-goals can be pursued in any order, several actions may be equally valid continuations of a given prefix, meaning that the ground-truth next token is not unique. A prediction that does not match the specific next token in the test plan may nonethe-

Seq. Length	BLOCKS	DEPOTS	DRIVELOG	LOGISTICS	SATELLITE	ZENO
5	100.0	99.95	98.25	99.05	99.4	83.5
10	100.0	100.0	99.10	99.15	99.4	99.75
25	100.0	100.0	100.0	99.85	99.4	99.95
50	100.0	100.0	100.0	100.0	100.0	100.0
100	100.0	100.0	100.0	100.0	100.0	100.0
<i>Tot</i>	100.0	100.0	100.0	100.0	100.0	100.0

Table 7.5: PTP task accuracy. Sequence Length counts tokens after [EG]. Row *Tot* uses the full plan minus the first token.

less correspond to a perfectly valid and rational continuation. This structural ambiguity places a ceiling on achievable NTP accuracy that is unrelated to the model’s understanding of the domain, and partly explains the lower performance observed in LOGISTICS and DRIVELOG at short sequence lengths, where early actions are interchangeable across independent sub-goals.

Previous Token Prediction Task. The PTP results in Table 7.5 are remarkable: with only 5 input tokens, accuracy exceeds 95% in almost all domains. From 10 tokens onward all domains exceed 99%, and from 50 tokens onward 100% accuracy is achieved universally. PTP is structurally easier than NTP because PlanBERT can exploit all tokens that follow the masked one, which are typically more informative about the beginning of a plan than a short prefix is about its continuation.

<i>Plan %</i>	<i>Model</i>	BLOCKS	DEPOTS	DRIVELOG	LOGISTICS	SATELLITE	ZENO
10	LGR	20.39	25.42	25.43	27.20	41.26	28.37
	GRNET	22.85	32.95	39.80	39.15	45.50	48.00
	PlanBERT	23.00	36.07	38.87	32.29	44.13	41.76
30	LGR	38.92	47.52	41.27	55.50	73.95	49.70
	GRNET	52.35	59.80	65.40	68.80	75.10	76.90
	PlanBERT	56.31	74.81	75.40	66.84	71.93	81.14
50	LGR	53.02	65.88	59.07	73.93	84.35	69.90
	GRNET	71.10	74.70	78.40	80.40	88.30	89.20
	PlanBERT	78.06	89.27	91.11	82.17	84.73	96.64
70	LGR	72.25	78.78	76.32	85.06	92.34	88.18
	GRNET	84.90	84.90	86.20	89.20	96.10	96.80
	PlanBERT	89.70	95.19	96.94	91.17	91.63	99.67
100	LGR	88.62	93.17	89.94	90.14	96.44	98.71
	GRNET	92.02	91.95	90.78	93.94	98.73	98.73
	PlanBERT	87.01	96.39	98.21	93.30	93.63	99.99

Table 7.6: Goal recognition accuracy (% of instances correctly predicted) by LGR, GRNET, and PlanBERT on TS_{LGRGen} . Bold values indicate the best result for each domain and observation percentage.

Offline Goal Recognition Task. Table 7.6 reports goal recognition accuracy for LGR, GRNET, and PlanBERT on TS_{LGRGen} . PlanBERT consistently outperforms both baselines in four out of six domains (BLOCKSWORLD, DEPOTS, DRIVELOG, and ZENOTRAVEL) across most observation percentages, with particularly large improvements at intermediate observability: at 30% of the plan, PlanBERT exceeds GRNET by over 10 points in DEPOTS, DRIVELOG, and ZENOTRAVEL, demonstrating superior early-recognition capability. Both learning-based systems substantially outperform LGR in all domains and at all observation levels, with the exception of SATELLITE, where LGR remains competitive even at low observability due to the informativeness of its landmark-based reasoning in that domain.

In SATELLITE, GRNET consistently outperforms PlanBERT across all observation levels, for the structural reasons discussed in Section 7.1.4.

7.1.3 Robustness to Noisy Observations

The experiments presented so far assume that observation sequences consist exclusively of valid actions drawn from the agent’s true plan. In practice, observation channels may be corrupted: spurious or erroneous actions can appear in the sequence, either due to sensor failures or adversarial interference. In this section, we evaluate the robustness of PlanBERT, GRNET, and LGR when the observation sequence is corrupted by randomly replaced actions.

The noise model adopted here is equivalent to the **Replace attack** introduced in Chapter 6 (Section 6.1.1): a set of spurious actions, drawn uniformly at random from the domain action set $\mathcal{A}$, are replaced with valid actions in the observation sequence at random positions. These replaced actions may be inapplicable in the state at which they appear and have no causal relation to the agent’s true plan.

Experimental Setup. We evaluate the three systems across the same six benchmark domains. Noise is injected at four concentration levels: 5%, 10%, 20%, and 30% of the total sequence length. Each noise level is combined with the five observability levels used throughout this chapter (10%, 30%, 50%, 70%, and 100% of plan actions). Missing predictions from any system are conservatively treated as incorrect. We use GR accuracy as the evaluation metric, consistent with Section 7.1.2.

Results. Table 7.7 reports the accuracy of the three systems averaged over all observability levels for each domain and noise concentration. Figure 7.1 complements this view by showing accuracy as a function of observability, averaged

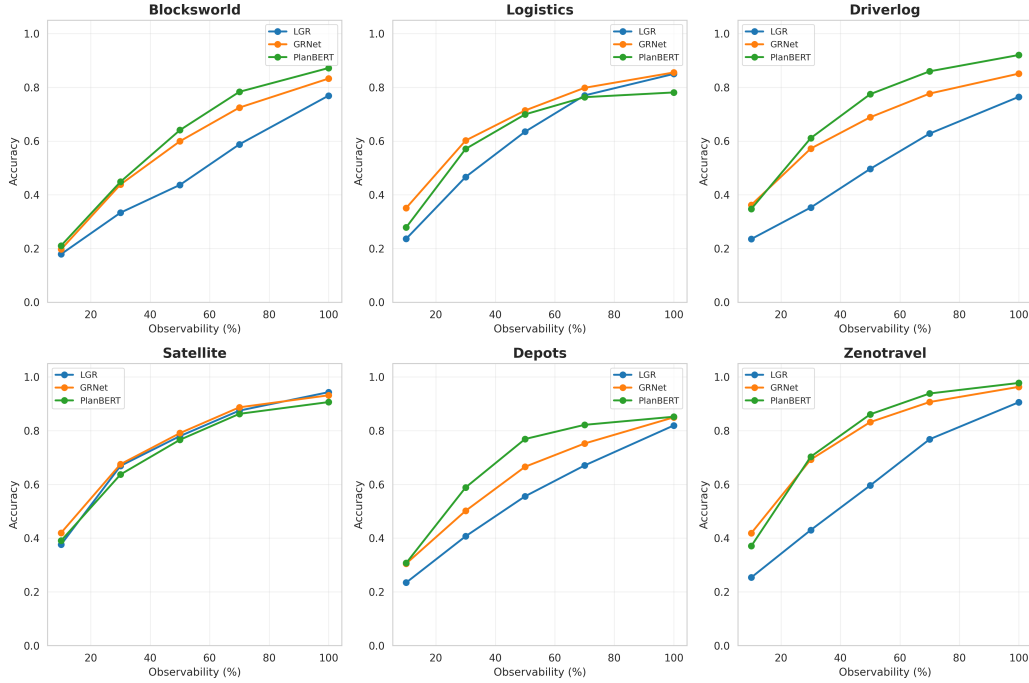


Figure 7.1: Accuracy of LGR, GRNET, and PlanBERT as a function of the percentage of observed actions (observability level), averaged over all noise levels, for each of the six benchmark domains.

over all noise levels, for each domain.

Across all experimental conditions, PlanBERT achieves the highest global average accuracy (67.41%), followed by GRNET (66.20%) and LGR (56.28%). Whilst the margin between PlanBERT and GRNET is modest (+1.21 points), the advantage over LGR is substantial (+11.13 points), confirming that learning-based approaches degrade considerably less under observation noise than the landmark-based symbolic baseline.

Domain-level Analysis. PlanBERT achieves the best performance in four out of six domains (BLOCKSWORLD, DEPOTS, DRIVERLOG, and ZENOTRAVEL) across all

noise levels, whilst GRNET is consistently superior in LOGISTICS and SATELLITE. In DRIVELOG and DEPOTS, PlanBERT’s advantage over GRNET is most pronounced: at 5% noise the gap reaches 5.58 and 7.02 points respectively, and it remains stable at 30% noise (+4.51 and +4.79 points). This sustained margin suggests that PlanBERT’s bidirectional attention mechanism is better able to contextualise spurious inserted actions and discount them when they are inconsistent with the surrounding sequence. In ZENOTRAVEL, PlanBERT leads at 5% noise (0.8133 vs. 0.8014) but the gap narrows progressively, and the two systems are statistically indistinguishable at higher noise levels (see below), consistent with the structural regularity of this domain.

<i>Domain</i>	<i>Noise (%)</i>	LGR	GRNET	PlanBERT
BLOCKSWORLD	0	0.5464	0.6464	0.6682
	5	0.5173	0.6139	0.6667
	10	0.4872	0.5869	0.6341
	20	0.4284	0.5314	0.5487
	30	0.3733	0.4670	0.4792
DEPOTS	0	0.6215	0.6886	0.7835
	5	0.5928	0.6711	0.7413
	10	0.5686	0.6519	0.7025
	20	0.5182	0.6005	0.6454
	30	0.4573	0.5255	0.5734
DRIVELOG	0	0.5841	0.7212	0.8011
	5	0.5530	0.7027	0.7585
	10	0.5267	0.6814	0.7391
	20	0.4705	0.6282	0.6788
	30	0.4197	0.5799	0.6250
LOGISTICS	0	0.6637	0.7430	0.7315
	5	0.6369	0.7231	0.7023
	10	0.6170	0.6954	0.6665
	20	0.5666	0.6400	0.5862
	30	0.5263	0.5839	0.5079
SATELLITE	0	0.7767	0.8075	0.7721
	5	0.7735	0.7913	0.7615
	10	0.7549	0.7662	0.7449
	20	0.7154	0.7195	0.6942
	30	0.6607	0.6780	0.6410
ZENOTRAVEL	0	0.6697	0.8193	0.8384
	5	0.6408	0.8014	0.8133
	10	0.6208	0.7886	0.7939
	20	0.5696	0.7449	0.7549
	30	0.5182	0.7067	0.7095
GLOBAL AVERAGE		0.5628	0.6620	0.6741

Table 7.7: Goal recognition accuracy of LGR, GRNET, and PlanBERT under varying Replace noise levels, averaged over all observability levels. Bold values indicate the best result per domain and noise level. The global average is computed across all domains, noise levels, and observability levels.

In LOGISTICS and SATELLITE, GRNET retains its advantage at all noise levels. In LOGISTICS, PlanBERT’s accuracy degrades more sharply with increasing noise (0.7315 at 0% vs. 0.5079 at 30%, a drop of 22.4 points) compared to GRNET (0.7430 vs. 0.5839, −15.9 points), suggesting that the tight causal coupling between actions in this domain makes spurious insertions particularly disruptive for PlanBERT’s attention-based mechanism. Causal coupling, as formalised by the causal graph of a planning task [34, 35], refers to the degree of interdependency among state variables induced by action preconditions and effects: an arc (u, v) exists in the causal graph whenever changing variable v requires a precondition on u , or whenever a single action affects both u and v simultaneously [34]. In a tightly coupled domain such as LOGISTICS, these dependencies form a dense, cyclic graph, loading or unloading a package simultaneously constrains both the vehicle-location and package-location variables, so that a single spurious action insertion propagates disruption across multiple interdependent state variables at once. In SATELLITE, all three systems converge at high noise levels, with the learning-based approaches increasingly penalised by the large action vocabulary that characterises this domain.

Degradation Rate. All three systems degrade monotonically as noise increases. LGR exhibits the steepest absolute degradation in most domains: in DRIVELOG its accuracy drops from 0.5530 at 5% to 0.4197 at 30% (−13.3 points). PlanBERT drops by a comparable absolute amount in DRIVELOG (−13.4 points) but from a much higher baseline. In BLOCKSWORLD, PlanBERT’s degradation rate is similar to GRNET’s (−18.75 vs. −14.69 points), causing the performance gap between the two to narrow under severe noise conditions.

<i>Domain</i>	<i>Noise (%)</i>	PlanBERT vs. LGR	PlanBERT vs. GRNET	GRNET vs. LGR
BLOCKSWORLD	5	< 0.001	< 0.001	< 0.001
	10	< 0.001	< 0.001	< 0.001
	20	< 0.001	< 0.05	< 0.001
	30	< 0.001	ns	< 0.001
DEPOTS	5	< 0.001	< 0.001	< 0.001
	10	< 0.001	< 0.001	< 0.001
	20	< 0.001	< 0.001	< 0.001
	30	< 0.001	< 0.001	< 0.001
DRIVELOG	5	< 0.001	< 0.001	< 0.001
	10	< 0.001	< 0.001	< 0.001
	20	< 0.001	< 0.001	< 0.001
	30	< 0.001	< 0.001	< 0.001
LOGISTICS	5	< 0.001	< 0.01	< 0.001
	10	< 0.001	< 0.001	< 0.001
	20	< 0.05	< 0.001	< 0.001
	30	< 0.05	< 0.001	< 0.001
SATELLITE	5	< 0.05	< 0.001	< 0.01
	10	ns	< 0.001	ns
	20	< 0.001	< 0.001	ns
	30	< 0.01	< 0.001	< 0.05
ZENOTRAVEL	5	< 0.001	< 0.05	< 0.001
	10	< 0.001	ns	< 0.001
	20	< 0.001	ns	< 0.001
	30	< 0.001	ns	< 0.001

Table 7.8: Pairwise Wilcoxon signed-rank test results. ns: not significant ($p \geq 0.05$).

Statistical Significance. To validate the reliability of the observed differences, we performed pairwise Wilcoxon signed-rank tests for each combination of domain and noise level. The results are summarised in Table 7.8.

PlanBERT significantly outperforms LGR in 23 out of 24 configurations ($p < 0.001$ in the large majority of cases), with the sole exception of SATELLITE at

10% noise ($p = 0.050$, ns), where the two systems are nearly equivalent. This robustness of PlanBERT’s superiority over the symbolic baseline is consistent across all domains and noise levels.

The comparison between PlanBERT and GRNET is more nuanced. In DEPOTS and DRIVELOG, PlanBERT is significantly superior across all noise levels ($p < 0.001$). In BLOCKSWORLD, significance holds at low-to-moderate noise ($p < 0.001$ at 5% and 10%; $p < 0.05$ at 20%) but disappears at 30% noise ($p = 0.095$), reflecting the convergence of the two systems under severe noise. In ZENOTRAVEL, PlanBERT is significantly better only at 5% noise ($p < 0.05$), with the two systems statistically indistinguishable at all higher noise levels. Conversely, in LOGISTICS and SATELLITE GRNET’s superiority over PlanBERT is confirmed to be statistically significant across most configurations ($p < 0.01$ or $p < 0.001$).

7.1.4 Discussion and Limitations

The results presented in Sections 7.1.2 and 7.1.3 show that PlanBERT can effectively accomplish three predictive tasks (PLM, NTP, PTP), can be fine-tuned for offline goal recognition, and maintains meaningful accuracy under observation noise equivalent to the Insert attack scenario of Chapter 6. This demonstrates that PlanBERT captures the structural rules of planning domains in a way that is resilient to moderate levels of spurious observation corruption. However, several limitations warrant discussion.

Training Data Requirements. As shown in Table 7.2, the number of plans necessary to train such a model is very high (approximately 200,000 per domain). This can be a significant obstacle in real-world applications where data collection is costly. Knowledge distillation techniques could potentially produce smaller, more efficient models [41].

Domain-Specific Training. Our approach requires training a separate model for each planning domain, limiting its applicability to new domains without re-training. Multi-domain training or transfer learning approaches could address this limitation.

Vocabulary Constraints. The vocabulary must be predefined with a fixed maximum number of objects, making the model sensitive to object names. A mapping algorithm can translate new names into predefined ones, but this adds a preprocessing step.

Performance Variability and the Satellite Domain. Performance varies across domains, with SATELLITE consistently showing lower results in all tasks. Several structural characteristics explain this: the SATELLITE training set contains approximately 50,000 fewer plans than other domains; plans are significantly shorter on average (17.5 actions vs. 22–39 in other domains); and the domain has the largest action space ($|A| = 33,225$), meaning each action is encountered less frequently during training. This combination makes SATELLITE structurally ill-suited for data-driven pattern recognition, and explains the consistently superior performance of the landmark-based system LGR in this domain even under noise.

Plan Representation. PlanBERT lacks a unified representation of the entire plan analogous to the [CLS] token in NLP. Developing a planning-specific task to learn an informative whole-sequence representation remains an open challenge, and could be valuable for downstream tasks such as heuristic prediction or plan quality assessment.

Despite these limitations, the results demonstrate that Transformer-based architectures can successfully learn planning domain dynamics, be adapted to mul-

multiple downstream tasks, and maintain robust performance under adverse observation conditions, validating the parallelism between automated planning and natural language processing.

7.2 Experimental Evaluation of CLERNet

In this section, we evaluate the performance of CLERNet and compare it against both symbolic planning-based methods (ORL) and the baseline deep learning adaptation (OnlineGRNet, referred to as GRNET). We demonstrate that CLERNet achieves both the computational efficiency of neural approaches and the recognition accuracy of symbolic methods.

7.2.1 Experimental Setup

For the experimental evaluation, we selected the same domains as before, with unit action costs [47, 51]: BLOCKSWORLD, DEPOTS, DRIVELOG, LOGISTICS, SATELLITE, and ZENOTRAVEL. For each considered domain, we created a training set with 55,000 instances using the labelling strategies described in Section 5.2.2 and the goal recognition problems from [19]. As a test set, we used the set of 6,000 goal recognition instances published and used in [19], which were generated with the planner LAMA [65]. As in the PlanBERT setting, in Table 7.1 we show the details of the problems used in our training and test sets. $|A|$ is the number of possible actions. $|F|$ is the number of fluents considered in the goal state, and this number defines the output dimension of our neural network. $|G_i|$ is the number of true fluents for a problem inside each goal hypothesis. $|\mathcal{G}|$ is the number of possible goal hypotheses inside a problem.

Evaluation Metrics. We evaluate our models using the following metrics.

Goal Recognition Accuracy (GR Accuracy). For offline evaluation (PlanBERT), we use GR accuracy, defined as the fraction of test instances in which the model correctly identifies the hidden goal G^* :

$$\text{Accuracy} = \frac{|\{T_i \in \mathcal{T} \mid \hat{G}_i = G_i^*\}|}{|\mathcal{T}|},$$

where $\mathcal{T}$ is the test set, $\hat{G}_i$ is the predicted goal for instance T_i , and G_i^* is the true goal.

Ranked First (RF). For a single OGR instance with k observations, RF is defined as the fraction of time steps at which the system correctly predicts the true goal:

$$\text{RF} = \frac{1}{k} \sum_{t=1}^k \mathbf{1}[\hat{G}_t = G^*],$$

where $\hat{G}_t$ is the predicted goal after observing the t -th action and $\mathbf{1}[\cdot]$ is the indicator function. Across the test set, RF is the average of per-instance values.

Convergence (CV). CV measures how early the system converges to the correct goal and maintains it. For a single OGR instance, let t^* be the earliest time step such that the system predicts G^* at every subsequent step:

$$t^* = \min\{t \in [1, k] \mid \forall t' \geq t, \hat{G}_{t'} = G^*\},$$

and define $t^* = k + 1$ if no such step exists. Then:

$$\text{CV} = \frac{k - t^* + 1}{k}.$$

By construction, $\text{CV} \leq \text{RF}$ for every instance, since any step contributing to CV also contributes to RF, but not vice versa. Across the test set, CV is the average of per-instance values.

More intuitively, consider the OGR instance in Examples 2 and 4 with $k = 6$ observations. A system predicting G_1 after the first action, G_2 after the second

and third actions, and G_1 from the fourth action onwards obtains $RF = (1 + 0 + 0 + 1 + 1 + 1)/6 = 4/6$ and $CV = (0 + 0 + 0 + 1 + 1 + 1)/6 = 3/6$. Considering the whole test set, RF and CV are defined as the average of the metrics across all test instances.

Execution Time (T). We also include the average prediction time per observation (T, in milliseconds), which does not include the training time (approximately 1 hour on CPU). Once trained, the network can be applied to multiple instances in the same domain without retraining.

Baseline Systems. We compare the deep learning-based approaches (GRNET and CLERNET) with the state-of-the-art symbolic planning method ORL [81]. For applying GRNET to OGR, considering a plan composed of m actions, we create m action traces, starting from one that contains only the first action executed by the agent and adding the subsequent actions one at a time. This poses a significant difference with the system application in [19] as, in that context, GRNET was applied to *offline goal recognition*, and thus it provided a single prediction for each incomplete trace.

7.2.2 Results and Analysis

The results of our comparison among CLERNET, GRNET and ORL, in terms of RF, CV and T, are available in Table 7.9.

OnlineGRNet vs. Symbolic Planning. First, we can observe that adapting GRNET to OGR provides better results compared to ORL in terms of RF in all domains except SATELLITE. The major improvements are reported in DRIVELOG (57.6 points versus 50.9), LOGISTICS (58.5 versus 49.5) and ZENOTRAVEL (69.2 versus 57.7). In terms of CV, the results of GRNET and ORL are comparable. In fact,

we can see an important improvement in ZENOTRAVEL (9.5 points) and in LOGISTICS (3.1 points), but a marked worsening in DEPOTS (4.3 points) and SATELLITE (9.5 points). On average, GRNET obtains a higher RF (57.1 versus 52.9) and a comparable CV (50.9 versus 51.1). However, the most important difference between these two approaches is in terms of time. In fact, while GRNET takes around 6 ms on average to compute the prediction after each observation, ORL takes about 150 times more (920 ms). Given that in the OGR context recognition time can be fundamental, as the agent keeps executing actions while the system is trying to understand its goal, GRNET overall performs better than ORL.

Domain	ORL			GRNET			CUMULATIVE			PROXIMITY		
	RF	CV	T	RF	CV	T	RF	CV	T	RF	CV	T
BLOCKSWORLD	38.9	36.3	580	41.5	36.4	11	<u>43.6</u>	<u>40.6</u>	11	46.6	44.2	11
DEPOTS	46.6	44.9	987	47.3	40.6	4	<u>49.1</u>	42.6	4	50.6	<u>44.7</u>	4
DRIVELOG	50.9	48.3	897	57.6	48.7	4	59.8	55.3	5	<u>59.5</u>	<u>53.8</u>	4
LOGISTICS	49.5	48.6	1118	58.5	51.7	4	<u>56.8</u>	<u>49.8</u>	4	55.0	51.7	4
SATELLITE	72.5	69.8	1260	66.8	60.3	7	<u>70.2</u>	68.0	7	70.1	<u>68.5</u>	7
ZENOTRAVEL	57.7	56.8	649	69.2	66.3	6	<u>75.2</u>	<u>71.0</u>	5	75.4	71.7	5
AVERAGE	52.9	51.1	920	57.1	50.9	6	<u>59.4</u>	<u>54.8</u>	6	59.8	56.0	6

Table 7.9: Results of ORL, GRNET and CLERNET for six planning domains. For each system, we consider the Ranked First (RF), the Convergence (CV) metrics and the execution time (T, in *ms*). Best results are marked in bold, second best underlined.

CLERNet vs. Symbolic Planning. An even more important difference can be seen comparing ORL to our new model, CLERNET, considering both the PROXIMITY and the CUMULATIVE labeling strategies. As can be seen in Table 7.9,

CLERNet obtains better results in all domains except SATELLITE in terms of RF. The most important improvements can be seen in ZENOTRAVEL (75.4 obtained by PROXIMITY) and DRIVELOG (59.8 obtained by CUMULATIVE, versus 50.9). In terms of CV, CLERNet performs better in all domains except SATELLITE and slightly in DEPOTS, in which ORL obtains a CV of 44.9 and CLERNet with the PROXIMITY strategy obtains a very similar 44.7. Analyzing SATELLITE, we can see a slightly less marked worsening of the performance (72.5 obtained by ORL versus 70.2 by CUMULATIVE and 70.1 by PROXIMITY, in terms of RF, and 69.8 versus 68.0 and 68.5 in terms of CV) with respect to the worsening that can be seen with GRNet. On average, our approach performs better than ORL by about 5 points both in terms of RF and CV. Similarly to what was previously observed for GRNet, we remark the important difference between CLERNet and ORL in terms of time (6 ms versus 920 ms).

CLERNet vs. OnlineGRNet. Finally, we compare the performance of GRNet and CLERNet. In terms of RF and CV, CLERNet obtains better performance for all domains except LOGISTICS (58.5 versus 56.8 obtained by CUMULATIVE and 55.0 obtained by PROXIMITY in terms of RF). The most important improvements can be seen in ZENOTRAVEL (about 5 point of RF and CV) and BLOCKSWORLD (44.2 obtained by PROXIMITY versus 36.4 of GRNet in terms of CV). On average, we improve by more than 2 points in terms of RF. In terms of CV, PROXIMITY obtains 5 points more w.r.t to GRNet. Since they are both LSTM based, CLERNet and GRNet have similar recognition time.

The strong performance across all systems in ZENOTRAVEL can be attributed to the structural regularity of the domain. ZENOTRAVEL has a relatively small fluent set ($|F| = 66$, as shown in Table 7.1) and a compact vocabulary of 41 tokens, which limits the hypothesis space that the model must navigate. Furthermore,

the goals in ZENOTRAVEL concern transporting passengers and aircraft to specific destinations: since each passenger must board, fly, and disembark in a fixed causal order, the causal link structure of plans in this domain is highly predictable and regular, making it easier for both the Neural Component and the landmark heuristics to identify goal-supporting actions early in the observation sequence.

Comparison of Labeling Strategies. Comparing our two labeling strategies, we can see that the results are quite similar, with an average difference of 0.4 for RF and about 1 for CV. There is however a notable exception: in BLOCKSWORLD, PROXIMITY performs better than CUMULATIVE, especially in terms of CV (44.2 versus 40.6). This is probably due to the high number of causal links that can be extracted from a plan in that domain, which complicates the learning process of the Neural Component with the CUMULATIVE strategy.

Accuracy Analysis. In terms of accuracy (see Figure 7.2), we observe the same trend we observed for RF and CV. In almost all cases, we can see that ORL (in blue) obtains lower accuracy for almost all the plan percentages. However, some peculiar exceptions are SATELLITE and DEPOTS with 10% of the plan. In such cases, the best performance is obtained by ORL. This is probably due to the landmark computation, which includes fundamental information that cannot be deduced from the small amount of observations in input. This slight difference of performance for DEPOTS is the main motivation behind the better CV, whereas in general GRNET and CLERNet obtain better performance. Comparing the deep-learning based approaches, we can see how CLERNet (green and red lines) obtains better accuracy compared to GRNET even with small percentages of plans. A notable exception is LOGISTICS, in which GRNET obtains higher results than our approaches, especially with lower plan percentages. With more information

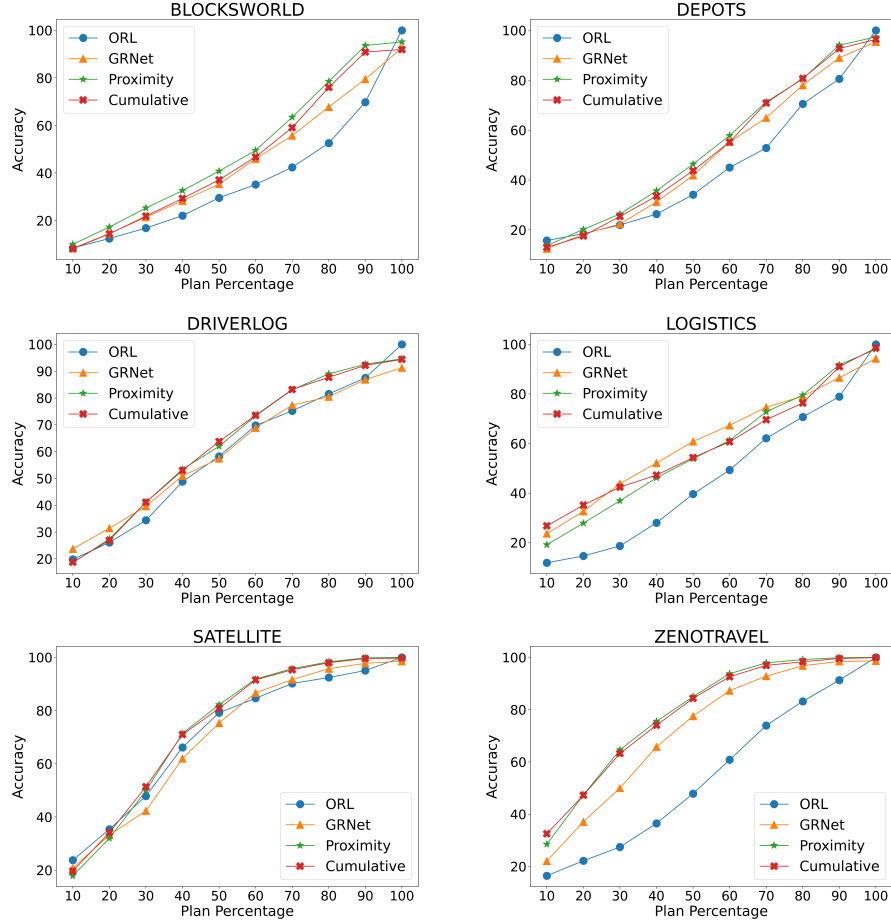


Figure 7.2: Performance of CLERNet with PROXIMITY (in green) and CUMULATIVE (in red) labeling strategies for each considered domain, compared with ORL (in blue) and GRNet (in orange) in terms of accuracy (on the y-axis). On the x-axis, we indicate the percentage of actions in the plan observed and given in input. The figure demonstrates that CLERNet achieves higher accuracy than both baselines across most plan percentages, with particularly strong performance in ZENOTRAVEL, DRIVERLOG, and BLOCKSWORLD domains.

(from 70%), both approaches perform very well.

Sensitivity Analysis. To evaluate the robustness of CLERNet’s Aggregation Component, we conducted a comprehensive sensitivity analysis examining how different threshold parameters affect the system’s performance. The Aggregation Component relies on two critical hyperparameters: τ_1 , which determines the confidence threshold for triggering the mutex mechanism (i.e., when the neural network is considered sufficiently certain about a predicted factual to apply symbolic constraints), and α , which represents the weight assigned to factials belonging to the initial state y_0 .

Table 7.10 presents the results of this analysis on the PROXIMITY dataset, systematically varying $\tau_1 \in \{0.2, 0.4, 0.6\}$ and $\alpha \in \{0.06, 0.1, 0.2\}$ across all six domains. The results demonstrate that CLERNet exhibits remarkable stability across different parameter configurations. For most domains, the performance variations are minimal (typically within 1-2 percentage points for both RF and CV metrics), indicating that the architecture is not overly sensitive to precise threshold tuning.

Our choice of $\tau_1 = 0.4$ and $\alpha = 0.1$ proves to be a robust configuration across all domains, achieving near-optimal or optimal performance in most cases. Notably, in domains like SATELLITE, the performance remains virtually unchanged across all parameter combinations (consistently around 70.1-70.2 for RF and 68.4-68.5 for CV), suggesting that the symbolic knowledge integration provides stable benefits regardless of the specific threshold values. In ZENOTRAVEL and LOGISTICS, we observe slightly more variation, but even in these cases, the performance differences remain modest. This robustness is particularly important for practical deployment, as it suggests that CLERNet does not require extensive hyperparameter tuning for each new domain, making it more generalizable and easier

Domain	$\tau_1 = 0.2$						$\tau_1 = 0.4$						$\tau_1 = 0.6$					
	$\alpha = 0.06$		$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0.06$		$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0.06$		$\alpha = 0.1$		$\alpha = 0.2$	
	RF	CV	RF	CV	RF	CV	RF	CV	RF	CV	RF	CV	RF	CV	RF	CV	RF	CV
BLOCKSWORLD	47.2	43.6	46.6	43.4	44.9	42.1	47.3	44.4	46.6	44.2	44.8	43.0	47.3	44.6	46.6	44.4	44.7	43.0
DEPOTS	50.2	42.2	50.3	42.4	50.4	42.3	50.6	44.6	50.6	44.7	51.0	45.1	50.4	45.2	50.5	45.3	50.8	45.6
DRIVELOG	60.4	53.6	60.0	53.9	59.0	54.2	60.1	53.5	59.5	53.8	57.5	53.5	59.8	53.7	59.2	53.8	57.0	53.4
LOGISTICS	55.3	50.4	53.5	49.3	50.8	47.3	56.7	52.7	55.0	51.7	52.2	49.2	56.7	53.0	55.0	51.9	52.3	49.5
SATELLITE	70.2	68.4	70.1	68.5	69.9	68.3	70.1	68.4	70.1	68.5	69.8	68.3	70.2	68.5	70.1	68.5	69.9	68.4
ZENOTRAVEL	74.4	70.0	74.8	70.7	76.0	72.8	75.2	71.3	75.4	71.7	75.1	72.5	75.2	71.1	75.5	71.9	75.2	72.8

Table 7.10: Results of the sensitivity study on the aggregation thresholds. τ_1 is the value we use to determine whether the net is sure about a predicted factual, in order to trigger the Mutex mechanism. α is the value that we assign to the factuals that belong to the initial state.

to apply in real-world scenarios.

Statistical Significance. To rigorously validate that the performance improvements of CLERNet over the baseline systems are not due to random chance, we conducted a comprehensive statistical significance analysis comparing ORL, GR-NET, and both variants of CLERNet (PROXIMITY and CUMULATIVE) across all domains.

We employed a two-stage statistical testing approach. First, we applied the Friedman test, a non-parametric statistical test suitable for comparing multiple related samples, to determine whether there exists a statistically significant difference among the four systems. The Friedman test is particularly appropriate for our evaluation because it does not assume normal distribution of the data and can handle the ordinal nature of our binary correctness assessments. For this analysis, we examined every prediction made by each system across all test instances. Specifically, for each observation in each of the 6,000 test instances per domain, we recorded whether the system correctly predicted the true goal

(True) or not (False), creating a comprehensive dataset of prediction outcomes for all four systems.

The Friedman test results confirmed that there is a statistically significant difference in performance among CLERNet, ORL, and GRNet across the evaluated domains, rejecting the null hypothesis that all systems perform equivalently. Following this global significance finding, we conducted the Nemenyi post-hoc test to perform pairwise comparisons and identify which specific system pairs exhibit significant differences while controlling for family-wise error rate. The Nemenyi test is the appropriate post-hoc procedure following a significant Friedman test, as it adjusts for multiple comparisons.

The results of the Nemenyi post-hoc test, visualized in the heatmap in Figure 7.3, reveal important insights. For five out of six domains (BLOCKSWORLD, DEPOTS, DRIVELOG, LOGISTICS, and ZENOTRAVEL), we observe statistically significant differences ($p\text{-value} < 0.05$) between PROXIMITY and both baseline systems (GRNet and ORL). This confirms that the performance improvements demonstrated by CLERNet in these domains are statistically robust and not attributable to random variation. The CUMULATIVE variant shows similar statistical significance patterns, further validating the effectiveness of our neuro-symbolic approach.

The notable exception is the SATELLITE domain, where no statistically significant difference exists between PROXIMITY and ORL ($p\text{-value} > 0.05$). This aligns with our earlier observations that ORL performs particularly well in SATELLITE due to its effective landmark-based reasoning, which captures critical domain structure that our current causal link-based approach does not fully exploit. This finding suggests that while CLERNet achieves competitive performance in SATELLITE, the integration of landmark information could potentially yield statistically significant improvements in this domain as well.

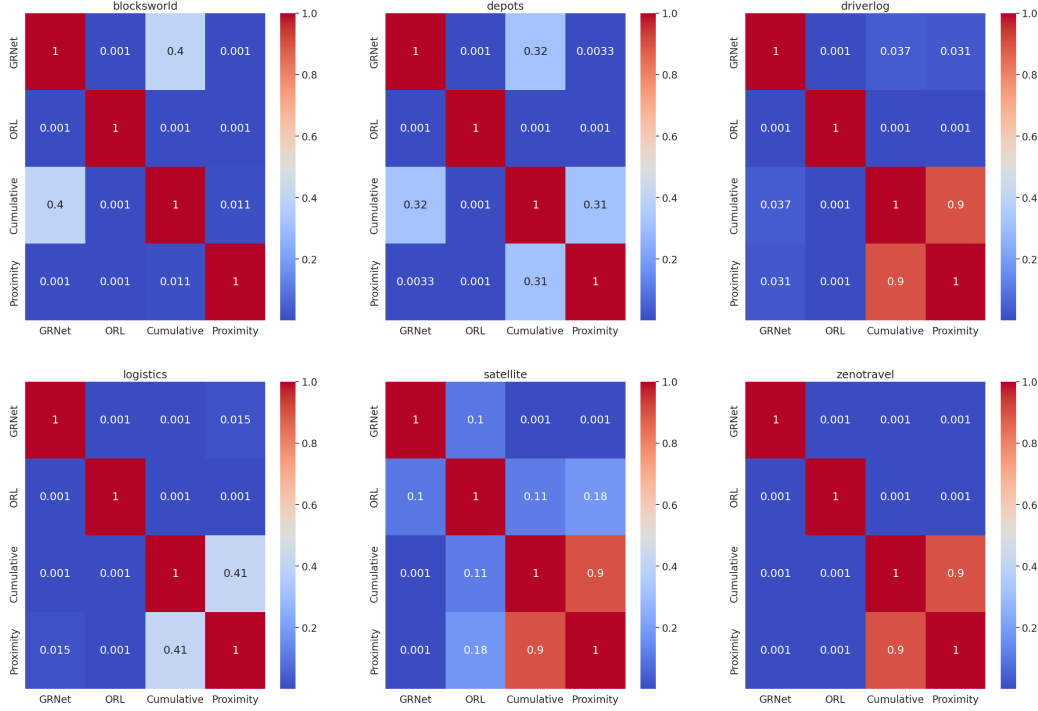


Figure 7.3: Nemenyi post-hoc heatmap for each prediction on all domains.

Ablation Study. To systematically assess the individual contributions of the symbolic planning knowledge components integrated into CLERNet’s Aggregation Component, we conducted a comprehensive ablation study on the model trained with the CUMULATIVE labelling strategy. This study isolates the effects of two key knowledge sources: (1) mutually exclusive facts (Mutex), which prevent the neural network from predicting logically incompatible facts simultaneously, and (2) the initial state information (y_0), which grounds predictions in the known starting configuration of the planning problem.

Table 7.11 presents four experimental configurations: *No* (baseline neural network without any symbolic knowledge), *Mutex* (including only mutex constraints), y_0 (including only initial state information), and *Both* (the full CLER-

Domain	No		Mutex		y_0		Both	
	RF	CV	RF	CV	RF	CV	RF	CV
BW	43.8	41.0	43.9	40.6	43.4	40.9	43.6	40.6
DEPOTS	43.4	39.2	47.6	41.6	44.3	39.7	49.1	42.6
DRIVELOG	58.5	52.3	58.6	52.3	59.5	55.0	59.8	55.3
LOGISTICS	55.1	49.6	55.1	47.5	56.7	51.7	56.8	49.8
SATELLITE	70.2	68.0	70.2	68.0	70.2	68.0	70.2	68.0
ZENO	71.9	67.6	72.3	67.9	74.4	70.5	75.2	71.0

Table 7.11: Results of the ablation study in terms of RF and CV on CUMULATIVE. *No* stands for no knowledge included, *Mutex* for including only mutually exclusive facts, y_0 including only the initial state and *Both* including both *Mutex* and y_0 .

Net model with both knowledge sources). The results clearly demonstrate that incorporating planning knowledge into the Aggregation Component yields substantial performance improvements across most domains.

The most dramatic improvements are observed in DEPOTS, where the full model (*Both*) achieves RF of 49.1 and CV of 42.6, representing gains of 5.7 and 3.4 points, respectively, compared to the knowledge-free baseline (*No*: RF 43.4, CV 39.2). Analysing the individual components reveals that mutex constraints contribute more significantly in this domain (RF 47.6, CV 41.6) than initial state information alone (RF 44.3, CV 39.7), suggesting that preventing contradictory predictions is particularly valuable in DEPOTS’ complex object manipulation scenarios.

Similarly, in ZENOTRAVEL, we observe substantial improvements of over 3 points in both metrics when comparing the full model (RF 75.2, CV 71.0) to the

baseline (RF 71.9, CV 67.6). Here, the initial state information (y_0) appears to be the more critical component (RF 74.4, CV 70.5), likely because knowing the initial locations of aircraft and passengers provides strong constraints on feasible goal configurations.

In DRIVELOG and LOGISTICS, both knowledge sources contribute positively, with the full model achieving the best performance. Interestingly, in SATELLITE, all configurations yield identical results (RF 70.2, CV 68.0), suggesting that the neural network alone already captures the relevant patterns in this domain, or that the specific symbolic knowledge we integrate (mutex and initial state) is less discriminative for satellite task structures.

The BLOCKSWORLD domain presents an intriguing case where the baseline actually achieves competitive performance, and adding knowledge components yields only marginal improvements or, in some cases, slight decreases. This may be attributed to the domain’s relatively simple causal structure, where the neural network can learn effective patterns directly from data, and the additional symbolic constraints may occasionally conflict with learnt heuristics.

Overall, this ablation study confirms that the integration of symbolic planning knowledge is not merely additive but synergistic: the combination of mutex constraints and initial state information (*Both*) generally outperforms either component in isolation, validating our neuro-symbolic architecture design. The domain-specific variations in component effectiveness also highlight opportunities for adaptive knowledge integration strategies that could further optimise performance.

Computation of Mutex Relations. The mutex relations $\mathcal{R}$ used in the Aggregation Component are pre-computed offline from the planning domain model before any recognition takes place. Starting from the initial state I , all states

reachable by applying sequences of valid actions are enumerated or approximated. Two fluents p and q are then identified as mutex if no reachable state contains both simultaneously. In practice, this approximation can be computed efficiently by propagating reachability constraints forward from I : a fluent f is reachable if there exists an applicable action whose add effects include f , and two fluents are considered mutex if no single applicable action can establish both, and no reachable state seen so far satisfies both. The resulting mutex set $\mathcal{R}$ is stored as a static lookup table and queried in constant time during inference, incurring no runtime overhead.

7.2.3 Discussion and Limitations

The investigation presented in this chapter demonstrates the efficacy of integrating symbolic planning knowledge into deep learning architectures to address the constraints of Online Goal Recognition. While the direct adaptation of the offline baseline, OnlineGRNet, confirmed that LSTM-based models can achieve real-time inference speeds drastically superior to symbolic approaches (6 ms versus 920 ms for ORL), its struggle with early convergence highlighted the limitations of purely data-driven pattern recognition in this domain.

The proposed CLERNet architecture successfully bridges this gap by embedding domain semantics directly into both the training process—via causal link labeling—and the inference mechanism via mutex-based aggregation. The experimental results confirm that this neuro-symbolic synergy allows CLERNet to retain the computational efficiency of neural networks while approximating, and often surpassing, the reasoning accuracy of state-of-the-art planning approaches.

However, several limitations warrant discussion. First, although standard planners like ORL remain competitive in specific domains heavily reliant on landmark detection (e.g., SATELLITE), this suggests that causal link structures alone

may not capture all relevant planning knowledge. Integrating landmark information into CLERNet could potentially address this limitation. Second, CLERNet requires domain-specific training data (55,000 instances per domain), which limits its applicability to new domains without substantial data collection and retraining efforts. Transfer learning approaches could help mitigate this requirement.

Third, the choice between CUMULATIVE and PROXIMITY labeling strategies appears domain-dependent, with PROXIMITY performing better in domains with complex causal structures (e.g., BLOCKSWORLD) and CUMULATIVE showing advantages in other domains. This suggests that an adaptive strategy selection mechanism could improve overall performance. Fourth, the mutex-based aggregation assumes complete and accurate domain knowledge, which may not always be available in real-world scenarios with partial observability or uncertain action effects.

Despite these limitations, CLERNet represents a significant advancement in bridging the gap between symbolic and neural approaches to goal recognition. The results suggest that the future of real-time goal recognition lies not in choosing between symbolic reasoning and deep learning, but in structurally aligning the latter with the logical constraints of the former. Future work should explore integrating additional planning knowledge (landmarks, heuristics), developing transfer learning mechanisms, and extending the architecture to handle partial observability and continuous action spaces.

7.3 Experimental Evaluation of PAC-OGR

In this section, we evaluate the robustness of state-of-the-art OGR systems to adversarial attacks and demonstrate how PAC-OGR enhances their defensive ca-

pabilities.

7.3.1 Experimental Setup

For our experimental evaluation, we use the two main state-of-the-art OGR systems: ORL and CLERNet. We evaluate their inherent robustness to Interference and Deception attacks, and demonstrate how our PAC-OGR defense mechanism enhances their performance. We evaluate their robustness to Interference and Deception attacks, and assess how our defense mechanism enhances it. The experiments are conducted over four classical planning domains: BLOCKSWORLD, DEPOTS, DRIVELOG, and LOGISTICS. For each domain, we have a test set with 350 problems extracted from the test set provided in [69], onto which we applied the described attacks. For Interference attacks, we set $p = \{0.10, 0.15, 0.2\}$ and use an equal number of instances for each value.

We evaluate the performance using the standard OGR metrics *Ranked First* (RF) and *Convergence* (CV), introduced in [82].

Parameter Configuration. The thresholds ϕ_p and ϕ_e used to detect tampered actions were determined through an iterative optimization process testing different values. This process was performed on a dedicated validation set composed of 50 instances, where 30% of the observed actions have been tampered. The optimal parameter pair was selected by comparing the actions predicted as tampered against the ground truth, to maximize the $F_{0.5}$ score. The $F_{0.5}$ score was specifically chosen to penalize false positives more heavily, as the system considers false positives to be particularly harmful. In fact, misclassifying a valid action as tampered could exclude valid information, potentially causing performance degradation. The threshold ϵ was set to 0.95.

Heuristic Configuration. We define the post-processing heuristic h^G as the average of four planning heuristics: two **admissible** heuristics, $h_{\max}$ [13] and LM_{cut} [36], and two **inadmissible** heuristics, h_{ff} [38] and h_{sa} [44]. An **admissible** heuristic never overestimates the true cost to reach the goal, i.e., $h(s) \leq h^*(s)$ for all states s , where $h^*(s)$ denotes the optimal cost from s ; thus, admissible heuristics provide conservative lower bounds on goal distance. In contrast, **inadmissible** heuristics may overestimate the true cost, but are often more informative in practice. The heuristic $h_{\max}$ computes the cost of achieving the most expensive precondition in the delete-relaxation of the problem, yielding an admissible but typically loose lower bound [13]. LM_{cut} , by contrast, is an admissible landmark-based heuristic that is at least as informative as $h_{\max}$ and often provides significantly tighter estimates in empirical evaluations [36]. The two admissible heuristics are therefore complementary in our framework: LM_{cut} is preferred when tighter estimates justify the additional computational effort, while $h_{\max}$ offers a simpler and faster lower bound that remains useful when landmark extraction is less discriminative. The two inadmissible heuristics, h_{ff} and h_{sa} , are included to mitigate the underestimation bias of admissible heuristics. h_{ff} is a relaxed-plan heuristic extracted from a relaxed planning graph and is known to be non-admissible but highly informative in practice [38]. h_{sa} is the set-additive heuristic for planning with action costs, which combines advantages of additive and relaxed-plan approaches and is likewise non-admissible [44]. By averaging over all four heuristics, h^G trades off the conservative lower bounds of admissible heuristics with the higher informativeness of inadmissible ones, yielding a single distance estimate that we found empirically to be more robust across domains with varying heuristic accuracy. The threshold ϵ was set to 0.95.

Domain	System	No Attacks		Interference						Deception			
				Insert		Delete		Replace		Centroids		Landmarks	
		RF	CV	RF	CV	RF	CV	RF	CV	RF	CV	RF	CV
BLOCKS	ORL	36.8	34.7	34.5	32.2	32.0	29.8	30.9	28.3	25.7	23.9	11.5	7.0
	ORL + PAC-OGR	36.8	34.7	34.7	32.5	31.2	29.1	30.6	28.2	31.1	27.3	19.5	11.5
	CLERNet	43.2	42.0	42.4	39.6	40.3	38.2	39.1	36.1	33.4	32.4	17.3	11.5
	CLERNet + PAC-OGR	43.2	42.0	42.8	40.3	39.5	37.3	38.8	35.5	36.0	31.1	20.3	13.2
DEPOTS	ORL	45.3	44.0	40.6	38.1	39.1	37.7	37.4	34.4	31.2	29.0	11.3	5.9
	ORL + PAC-OGR	45.3	44.0	42.0	40.1	38.0	36.3	37.5	35.4	40.0	33.3	15.4	8.2
	CLERNet	48.3	43.5	46.1	39.4	44.4	39.4	43.3	36.2	39.1	35.6	19.3	13.1
	CLERNet + PAC-OGR	48.3	43.5	47.2	42.0	42.7	37.7	42.8	37.0	41.4	34.6	19.2	13.0
DRIVELOG	ORL	48.6	46.7	42.8	39.8	41.8	39.9	39.2	35.9	32.6	28.9	13.3	7.2
	ORL + PAC-OGR	48.6	46.7	44.0	41.6	41.4	39.5	39.7	37.3	38.3	29.3	21.5	12.1
	CLERNet	58.3	52.3	56.2	48.8	55.1	49.1	51.4	43.6	39.5	32.3	24.8	16.8
	CLERNet + PAC-OGR	58.3	52.3	57.1	49.9	54.9	48.9	52.4	45.3	42.2	32.3	25.9	18.2
LOGISTICS	ORL	43.6	41.2	40.9	39.4	39.4	38.4	38.7	37.1	25.8	21.3	8.4	3.9
	ORL + PAC-OGR	43.6	41.2	41.2	40.0	38.9	38.0	38.8	37.3	25.9	20.0	7.9	2.6
	CLERNet	51.2	48.1	49.5	46.3	46.9	44.1	42.3	39.8	36.2	26.1	13.1	5.8
	CLERNet + PAC-OGR	51.2	48.1	50.7	47.4	46.5	43.6	46.7	43.6	35.1	23.4	12.8	5.5
All Domains	ORL	43.6	41.7	39.7	37.4	38.1	36.5	36.5	33.9	28.8	25.8	11.1	6.0
	ORL + PAC-OGR	43.6	41.7	40.5	38.6	37.4	35.7	36.7	34.5	33.8	27.5	16.1	8.6
	CLERNet	50.3	46.5	48.6	43.5	46.7	42.7	44.7	39.5	37.1	31.6	18.6	11.8
	CLERNet + PAC-OGR	50.3	46.5	49.4	44.9	45.9	41.9	45.2	40.4	38.7	30.4	19.6	12.5

Table 7.12: Experimental results comparing ORL and CLERNet without/with attacks, and considering the proposed defence methods. Best values between the standard and defence configurations are shown in **bold**.

Results for Interference Attacks

The results of our robustness analysis are shown in the left and central parts of Table 7.12.

Inherent Robustness. These experiments demonstrate that ORL and CLERNet possess high inherent robustness, particularly against the Insert attack. For this type of attack, the average performance drop is modest: approximately 4

points for ORL (e.g., RF drops from 43.6 to 39.7) and roughly 2 points for CLERNET in terms of both metrics. The Delete attack causes comparable but slightly higher degradation. However, the Replace attack poses more significant challenges, diminishing the RF and CV scores by several points for both ORL and CLERNET. Domains DEPOTS and DRIVELOG are the most vulnerable. The worst result can be seen for the Replace attack in DRIVELOG, where ORL dropped from 48.6 to 39.2 in RF. Conversely, LOGISTICS and BLOCKSWORLD are the most resilient domains, experiencing only minor performance decreases.

Impact of PAC-OGR on Interference Attacks. The application of PAC-OGR delivers the most significant benefits against the Insert attack, yielding consistent gains across all domains. We observed average increases of approximately 1 point in both metrics for CLERNET and ORL (e.g., ORL CV rose from 37.4 to 38.6). This is the attack type for which the pre-processing component is best suited: spurious inserted actions are identifiable precisely because they are inapplicable in context—their preconditions are unsatisfied and their effects do not contribute to subsequent actions—and the scores P_i and E_i are designed to detect exactly this signature.

The behaviour under the **Replace attack** is more nuanced. While the global average indicates a slight overall improvement (e.g., ORL RF increased marginally from 36.5 to 36.7), domain-level analysis reveals that the defence is counter-productive in BLOCKSWORLD, especially for CLERNET. In this domain, our pre-processing component misclassifies some valid actions as tampered, causing information loss and thus a slight performance decrease. Replacing a valid action with a fake one of the same type but different parameters can produce a sequence that still passes precondition checks partially, making the tampered action harder to distinguish from a genuine one. This structural ambiguity limits the discrimi-

native power of the P_i/E_i heuristic in such cases.

The **Delete attack** is the most challenging scenario for PAC-OGR, and it causes a slight performance drop in both RF and CV compared to the undefended baseline. The root cause is structural: the pre-processing component is designed to *identify and remove* tampered actions, not to *reconstruct* missing ones. When the Intruder deletes a valid action from O , the Observer receives an observation sequence with a gap. From the perspective of pre-processing, this gap is invisible: the remaining sequence may appear internally consistent, as the actions before and after the deletion can still satisfy their respective precondition and effect checks. Consequently, the pre-processing component has no signal to trigger reconstruction, and the OGR system must operate on an incomplete trace. This information loss directly penalises recognition accuracy, since the causal chain is broken and the neural or symbolic recogniser cannot compensate for the missing evidence. This represents an intrinsic limitation of the current architecture, discussed further in Section 7.3.2.

Results on Deception Attacks

The results of these experiments are shown in the left and right parts of Table 7.12, and they are further supported by the experimental results shown in Fig. 7.4 and Fig. 7.5.

Vulnerability to Deception. Both systems are vulnerable to deception attacks, as previously observed in [61] for ORL. For the Centroid attack, both ORL and CLERNET exhibit an average loss of 15 points in terms of RF (from 43.6 to 28.8 and from 50.3 to 37.1) and CV (from 41.7 to 25.8 and from 46.5 to 31.6), as the agent moves towards states compatible with multiple goals. The Landmarks attack causes a much more pronounced performance drop for ORL and CLER-

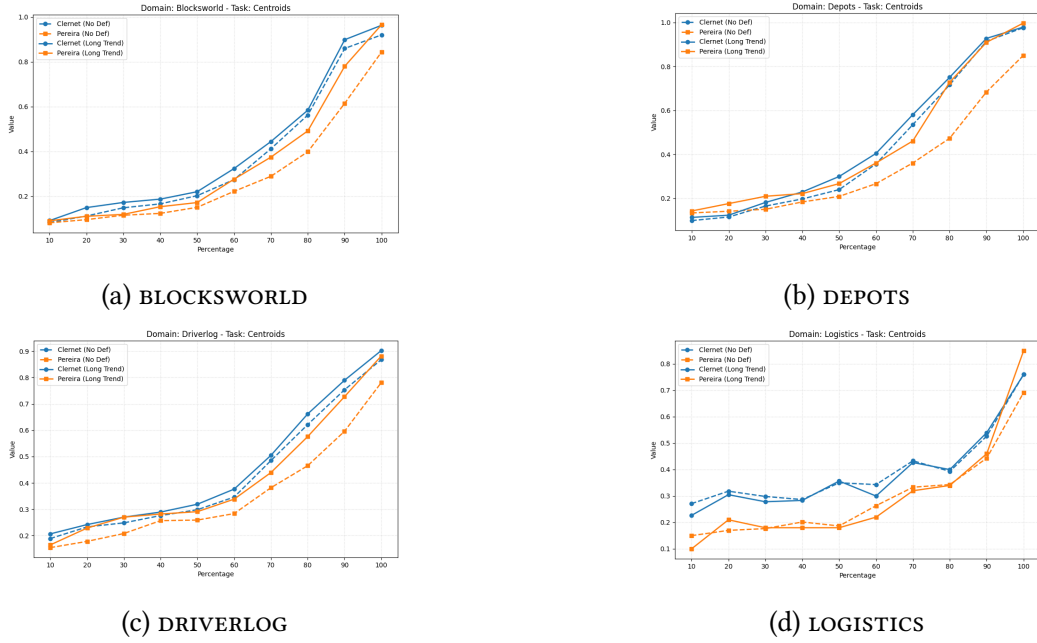


Figure 7.4: Accuracy of the PAC-OGR system against Centroids attacks across different domains.

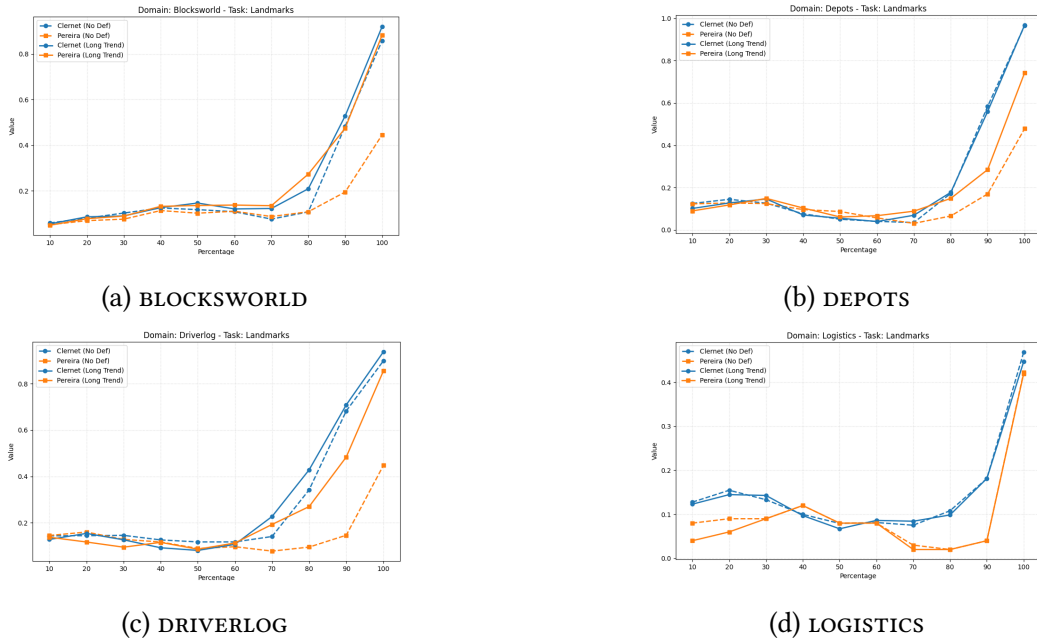


Figure 7.5: Accuracy of the PAC-OGR system against Landmarks attacks across different domains.

NET in terms of RF (from 43.6 to 11.1 and from 50.3 to 18.6) and CV (from 41.7 to 6.0 and from 46.5 to 11.8), since the agent moves towards a fake goal, thereby misleading the system.

Behavioural Analysis of PAC-OGR under Deception. The post-processing component of PAC-OGR is designed to detect deceptive behaviour by monitoring the trend of the f^G values over the observation sequence. Under the **Centroid attack**, the agent deliberately navigates towards a state equidistant from multiple goals before committing to its true objective. During this phase, the heuristic distances $h^G(s_i)$ to all candidate goals decrease simultaneously, producing f^G sequences with relatively flat or slowly increasing slopes. As a result, the post-processing component applies only mild penalties across all goals, preserving ambiguity without strongly penalising any single candidate. Once the agent departs from the centroid towards its true goal, the heuristic distances to the false goals begin to increase while that of the true goal decreases, generating a discriminative slope signal that the post-processing can exploit. The net effect is a modest but consistent improvement in RF, at the cost of a slight delay in commitment that reduces CV, producing the accuracy–convergence trade-off observed in the results.

Under the **Landmark attack**, the agent explicitly pursues a false goal G^- for the first portion of its plan before deviating towards its true goal G^* . During the deceptive phase, the f^{G^-} sequence decreases (or remains flat), generating a near-zero or negative slope for G^- , which the post-processing correctly refrains from penalising. Meanwhile, the f^{G^*} sequence increases as the agent moves away from G^* , generating a positive slope that triggers a penalty. Paradoxically, this means the post-processing component is actively penalising the true goal during the deceptive phase, which explains why the improvement over the undefended

baseline is modest for this attack type. The defence only begins to recover once the agent abandons G^- and starts approaching G^* , at which point the slope of f^{G^*} reverses. However, by this point the recogniser may already have accumulated significant score in favour of G^- . This inherent asymmetry between the deception and recovery phases constitutes the primary limitation of the heuristic trend analysis under strong Landmark attacks.

Mitigation with PAC-OGR. However, PAC-OGR mitigates the impact of these attacks, enhancing the robustness of OGR systems. The defense technique applied to ORL leads to a remarkable improvement: for Centroid, from 28.8 to 33.8 RF, and from 25.8 to 27.5 CV; for Landmarks, from 11.1 to 16.1 RF, and from 6.0 to 8.6 CV. Both metrics increase consistently in BLOCKS, DEPOTS, and DRIVELOG, with only a slight decrease observed in LOGISTICS, as illustrated in Figure 7.5. A smaller gain (1 point) can be seen for CLERNet in Landmarks, whereas for Centroid RF slightly increases (from 37.1 to 38.7) but CV decreases (from 31.6 to 30.4). Comparing the different domains, the best results are in DRIVELOG, where both metrics improve, whereas the worst results are obtained in LOGISTICS.

Analyzing the impact on the neuro-symbolic system CLERNet, we observe an interesting trade-off between accuracy and convergence, particularly in the Centroid attack (see Figure 7.4). While RF increases, CV decreases (from 31.6 to 30.4). This suggests that the defense mechanism induces a "cautiousness" in the system: by re-evaluating heuristic scores, it delays the commitment to a specific goal, protecting against early deceptive signals but slightly delaying the final correct prediction. Furthermore, domain-specific analysis reveals that LOGISTICS is particularly resistant to the defense mechanism, with CLERNet + PAC-OGR performing slightly worse than the baseline. This is likely due to the highly interconnected topology of the LOGISTICS state space, where centroid-based deception

is naturally harder to distinguish from optimal behavior using standard planning heuristics ($h_{\max}$, h_{ff}), which may be less discriminative here than the learned embeddings of the neuro-symbolic model.

7.3.2 Discussion and Limitations

Our experimental evaluation demonstrates that PAC-OGR significantly enhances the robustness of state-of-the-art OGR systems against adversarial attacks. The pre-processing component effectively filters tampered observations in Interference scenarios, while the post-processing component successfully re-calibrates predictions to mitigate Deception attacks.

A comparative analysis of the two systems reveals that PAC-OGR provides a substantially larger relative improvement for the symbolic system (ORL) compared to the neuro-symbolic one (CLERNet). For example, in Landmarks attacks, ORL sees a relative increase of over 40% in RF, whereas CLERNet improves by approximately 5%. This suggests that the ensemble of heuristics in PAC-OGR effectively compensates for the rigidity of standard symbolic approaches, while the neuro-symbolic system likely already captures some of these robust features within its learnt embeddings.

Furthermore, we observe a significant "Gap Closing" effect. In the Centroids attack scenario, the gap between the baseline neuro-symbolic system (CLERNet, RF 37.1) and the defended symbolic system (ORL + PAC-OGR, RF 33.8) shrinks from roughly 8 points to just 3.3 points. This implies that PAC-OGR can upgrade a purely symbolic, domain-independent system to achieve performance levels comparable to state-of-the-art neuro-symbolic models trained specifically on the domain, without requiring any training data itself.

Crucially, however, the defence preserves the performance hierarchy: even with the significant boost to ORL, the neuro-symbolic CLERNet consistently

yields higher performance. This confirms that PAC-OGR acts as an orthogonal and additive layer of robustness, empowering the underlying system without replacing the fundamental advantages of the neuro-symbolic architecture.

However, several limitations warrant discussion. First, the pre-processing component is designed to detect and remove tampered actions from the observation stream, but it has no mechanism to infer or reconstruct missing ones. This architectural choice has a direct consequence under the **Delete attack**: when the Intruder suppresses a valid action o_i from O , the Observer receives a sequence in which the deleted action leaves no trace. The scores P_i and E_i are defined over the actions that *are* present in O ; a deleted action, by definition, never enters the sequence and therefore cannot be evaluated or flagged. The surrounding actions may still appear mutually consistent—if the preceding state partially satisfies the preconditions of the action following the gap, or if the gap happens not to violate any precondition check—leaving the pre-processing component with no signal to act upon.

The downstream consequence is that the OGR system receives a trace with broken causal links. For CLERNET, which relies on causal link labels to assign fluent scores, a missing action means that the causal chain supporting a specific goal fluent may never be observed, permanently suppressing that fluent’s cumulative score. For ORL, which uses landmark counting, a deleted action may prevent a landmark from being marked as achieved, causing an underestimate of goal progress. In both cases, the information loss is irrecoverable within the current framework, which explains why PAC-OGR produces a net performance degradation under Delete attacks rather than an improvement: the pre-processing layer cannot compensate for what it cannot observe, and applying it introduces only a small additional risk of false positives without any corresponding benefit.

Second, the threshold parameters (ϕ_p , ϕ_e , ϵ) require domain-specific tuning

through validation sets, which may limit the generalizability of PAC-OGR across diverse planning domains without additional calibration. A more fundamental limitation is the **absence of an observation reconstruction mechanism**. The current PAC-OGR design operates under a *closed-world assumption* over the received observation stream: it can validate and filter what it sees, but it cannot hypothesise what it does not see. In the presence of Delete attacks, this means that the framework accepts the incomplete trace as the best available evidence and forwards it to the OGR system without attempting to fill the gaps.

A principled approach to addressing this limitation would require an *observation completion* or *plan repair* module capable of inferring, given the initial state I , the observed prefix O , and the domain model Π , which actions are likely to have been deleted from the sequence. Such a module could, for instance, apply backward chaining from the state following the gap to identify actions whose effects would bridge the inconsistency, or use a symbolic planner to generate candidate completions of the trace. However, incorporating such a module introduces computational overhead that the model-free and neuro-symbolic approaches to OGR are designed to avoid. Designing a reconstruction mechanism that is both computationally efficient and accurate under partial observability represents a significant open research challenge and a natural direction for future work.

Finally, in certain domains (notably BLOCKSWORLD), the pre-processing component occasionally misclassifies valid actions as tampered, resulting in false positives that degrade performance. This suggests that more sophisticated action-validation mechanisms could improve precision.

Despite these limitations, PAC-OGR represents a step toward OGR systems capable of operating in adversarial real-world scenarios. Future work should address these limitations through adaptive parameter learning, observation reconstruction mechanisms, and extensions to broader classes of planning problems.

Chapter 8

Conclusion

This thesis has established a comprehensive framework for learning-based goal recognition. By bridging deep learning and automated planning, we have showed that neural architectures can learn planning domain dynamics while maintaining the computational efficiency required for practical applications.

8.1 Summary of Contributions

This thesis makes several key contributions to the field of goal recognition:

- **Planning Language Modelling and Offline Goal Recognition through Transformers (Chapter 4):** Our foundational work with BERT confirmed that deep neural networks can internalize the preconditions and effects of planning domains solely through observation. The success of the *Planning Language Modelling* task showed that treating actions as tokens in a language sequence is a viable strategy for extracting semantic meaning from plans, yielding significant accuracy improvements in offline settings. PlanBERT achieves over 90% accuracy in reconstructing masked actions

and matches or exceeds domain-specific architectures in goal recognition tasks.

- **Neuro-Symbolic Online Goal Recognition (Chapter 5):** In the on-line domain, we provided a critical comparative analysis between general-purpose and specialized architectures. By adapting GRNET to the online setting, we established a robust baseline that validated the feasibility of LSTM-based OGR. However, our primary contribution, CLERNET, demonstrated that integrating domain-specific knowledge, specifically *Causal Links*, is essential for superior performance. CLERNET consistently outperformed the adapted GRNet, providing faster convergence and higher ranked-first accuracy across standard benchmarks. The Proximity labeling strategy proves particularly effective in domains with complex causal dependencies, achieving approximately 5 points improvement in convergence metrics over purely data-driven approaches.
- **Adversarial Robustness for Goal Recognition (Chapter 6):** We addressed the vulnerability of learning-based systems to adversarial inputs. Our evaluation of Deception and Interference scenarios highlighted the fragility of standard models. The introduction of the **PAC-OGR** framework proved that a hybrid approach, coupling the speed of neural networks with the rigor of symbolic validation, effectively recovers performance against observation tampering and goal obfuscation strategies. PAC-OGR significantly improves robustness metrics, particularly for Insert and Replace interference attacks, while maintaining real-time inference capabilities.

8.2 Limitations and Challenges

While this thesis demonstrates significant advances in learning-based goal recognition, each of the three proposed systems exhibits specific limitations that constrain its applicability and point towards concrete open problems.

Training Data Requirements. Both PlanBERT and CLERNet require substantial domain-specific training data: approximately 200,000 plans per domain for PlanBERT and 55,000 for CLERNet. In real-world scenarios where plan corpora must be generated by a planner or collected from deployed systems, this requirement entails significant computational investment even before any recognition takes place. Furthermore, this cost scales linearly with the number of domains to be covered, making large-scale deployment prohibitive without data-generation infrastructure. Knowledge distillation techniques [41] could potentially yield smaller, more efficient models with reduced data requirements, but their applicability to the planning domain has not yet been investigated.

Domain-Specific Training and Limited Generalisation. All three approaches—PlanBERT, CLERNet, and PAC-OGR—require either a trained model or pre-computed symbolic structures (landmarks, causal links, mutex relations) specific to a given planning domain. None of them generalises zero-shot to unseen domains. In particular, the vocabulary of PlanBERT and CLERNet is fixed at training time to a maximum number of objects, making the models insensitive to domain rescaling without retraining. While a mapping algorithm can translate object names, it cannot compensate for structural differences between domains. The SATELLITE results across all experiments illustrate the concrete consequence of this limitation: a domain with a large action space and short plans consistently underperforms relative to domains with more regular causal structure, regardless

of the architecture used.

Incompleteness of the Symbolic Knowledge in CLERNet. CLERNet integrates three sources of symbolic knowledge—causal links, the initial state, and mutex relations—but deliberately excludes the full PDDL model to preserve inference speed. This selective integration has measurable consequences: as demonstrated by the ablation study in Section 7.2.2, the SATELLITE domain shows no improvement from adding symbolic knowledge, suggesting that the specific structures chosen (mutex and initial state) are not discriminative for this domain. More generally, causal links alone do not capture all relevant planning knowledge; in particular, the landmark information that drives the strong performance of ORL in SATELLITE is absent from CLERNet’s training supervision. This represents a structural gap between the neuro-symbolic and purely symbolic approaches that the current architecture cannot close.

Limitations of PAC-OGR under Delete Attacks. The pre-processing component of PAC-OGR is designed to identify and remove tampered actions, but it has no mechanism to reconstruct missing ones. As a result, Delete interference attacks—where an intruder silently suppresses valid actions from the observation stream—cause a net degradation in recognition performance even when PAC-OGR is applied, because the pre-processing layer cannot detect what is absent. The broken causal chain that results from a deleted observation permanently penalises both CLERNet (which loses a causal link label) and ORL (which loses a landmark achievement signal). This asymmetry between the system’s ability to handle noisy additions versus silent deletions is a fundamental architectural limitation of the current design.

Sensitivity to Deception Phase Timing in PAC-OGR. The post-processing component of PAC-OGR relies on the trend of the f^G heuristic values to detect deceptive behaviour. Under Landmark attacks, however, the deceptive agent genuinely approaches a false goal G^- during the first phase of its plan, causing the f^{G^*} values to increase and triggering a penalty on the *true* goal. The post-processing component therefore actively penalises the correct hypothesis during the very period when the attack is most effective, recovering only once the agent abandons G^- . This phase-timing asymmetry limits the improvement achievable by PAC-OGR against strong Landmark attacks and represents a challenge that cannot be addressed purely through heuristic trend analysis.

Discrete Action Spaces. All proposed architectures are designed for symbolic planning domains with discrete, grounded action spaces. Extending these approaches to continuous or hybrid domains—where actions are parameterised by real-valued quantities, as in robotics or autonomous navigation—remains entirely open. The tokenisation scheme of PlanBERT and the binary causal link labels of CLERNet have no direct analogue in continuous spaces, and new representation and training paradigms would be needed.

8.3 Future Research Directions

The limitations identified above, together with the experimental findings of this thesis, define a set of concrete and well-motivated research directions.

Observation Reconstruction for Delete Interference. The most immediate gap left open by PAC-OGR is the absence of a mechanism to infer missing observations under Delete attacks. A principled solution would require an *observation*

completion module that, given the initial state I , an incomplete observation prefix O , and the domain model Π , hypothesises which actions are likely to have been suppressed. Backward chaining from inconsistent state transitions, or constrained plan repair guided by landmark and causal link analysis, are natural candidates. The central challenge is doing so within the millisecond-level inference budget of online recognition, which precludes full replanning at each step. Lightweight, approximate reconstruction heuristics that exploit the same causal link structure already present in CLERNet could provide a feasible path forward.

Integrating Landmark Knowledge into CLERNet. The experimental results consistently show that ORL retains an advantage over CLERNet in the SATELLITE domain, where landmark-based reasoning captures domain structure that causal links alone do not encode. A natural extension of CLERNet would incorporate landmark annotations alongside causal link labels in the training supervision signal, enabling the Neural Component to learn not only which actions causally support specific goal fluents, but also which actions achieve necessary milestones along every valid plan to the goal. Hybrid labelling strategies that combine causal link labels and landmark labels could be investigated, along with architectures that maintain separate memory streams for causal and landmark evidence.

Multi-Domain and Transfer Learning. All approaches in this thesis are trained per domain from scratch, with no mechanism for transferring knowledge across domains. Developing domain-agnostic representations—for instance, by encoding actions as structured objects (name, precondition set, effect set) rather than opaque indices—could enable pre-training on a diverse corpus of planning domains and few-shot adaptation to new ones. This direction is directly motivated

by the data requirements identified as a primary limitation: if the model can leverage structural similarities across domains, the number of domain-specific training instances required could be reduced substantially. Recent work on LLMs for planning [72, 66] suggests that Transformer architectures can generalise across planning domains when augmented with structural awareness, providing a natural starting point.

Adaptive Deception Detection in PAC-OGR. The heuristic trend analysis in PAC-OGR applies a fixed penalisation strategy that is blind to the phase structure of Landmark attacks. A more principled approach would model the deception process explicitly, for instance by maintaining a probability distribution over attack hypotheses (no attack, Centroid, Landmark) and updating it as observations arrive. Bayesian or game-theoretic formulations of this detection problem could yield a post-processing component that adjusts its penalisation strategy depending on the inferred attack type, rather than applying a uniform slope-based penalty across all scenarios.

Continuous and Hybrid Planning Domains. Extending the proposed architectures to continuous and hybrid domains would significantly broaden their applicability to robotics, autonomous vehicles, and other real-world control problems. For PlanBERT, this would require replacing the tokenisation scheme with a continuous action encoder capable of representing real-valued parameters. For CLERNet, new notions of causal links and mutex relations applicable to continuous state spaces would need to be developed, possibly drawing from work on causal inference in continuous dynamical systems. The PAC-OGR pre-processing component, which relies on discrete precondition and effect checking, would need to be replaced by a constraint satisfaction or probabilistic validity check

appropriate for continuous dynamics.

Real-World Evaluation and Deployment. All experimental evaluations in this thesis are conducted on benchmark planning domains from the International Planning Competition. Transitioning to real-world applications—such as human-robot collaboration in manufacturing, network intrusion detection in cybersecurity, or activity monitoring in ambient assisted living—introduces challenges that benchmark domains do not capture: imperfect action models, partial and noisy observations, non-stationary agent behaviour, and real-time constraints measured in wall-clock time rather than plan percentages. Field studies in these domains would provide the empirical grounding necessary to identify which limitations are critical in practice and to guide the algorithmic development of the next generation of goal recognition systems.

Goal Recognition in Multi-Agent Settings. The frameworks proposed in this thesis assume a single observed agent pursuing a single hidden goal. Many real-world scenarios involve multiple agents acting simultaneously, potentially cooperating or competing, with partially overlapping goal sets. Extending OGR to multi-agent settings would require new formalisms for representing joint observations, new training strategies for handling inter-agent dependencies, and new adversarial models for deception in which multiple agents coordinate to mislead the observer. The PAC-OGR taxonomy of attacks provides a natural starting point for formalising multi-agent adversarial scenarios.

Bibliography

- [1] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *CoRR*, abs/1901.09069, 2019.
- [2] Leonardo Amado, João Paulo Aires, Ramon Fraga Pereira, Mauricio Cecilio Magnaguagno, Roger Granada, and Felipe Meneguzzi. LSTM-Based Goal Recognition in Latent Space. *CoRR*, abs/1808.05249, 2018.
- [3] Leonardo Amado, Gabriel Paludo Licks, Matheus Marcon, Ramon Fraga Pereira, and Felipe Meneguzzi. Using Self-Attention LSTMs to Enhance Observations in Goal Recognition. In *Proceedings of IJCNN 2020*. IEEE, 2020.
- [4] Leonardo Amado, Reuth Mirsky, and Felipe Meneguzzi. Goal recognition as reinforcement learning. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 9644–9651. AAAI Press, 2022. doi: 10.1609/AAAI.V36I9.21198. URL <https://doi.org/10.1609/aaai.v36i9.21198>.
- [5] Leonardo Amado, Ramon Fraga Pereira, and Felipe Meneguzzi. Robust neuro-symbolic goal and plan recognition. In Brian Williams, Yiling Chen,

- and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 11937–11944. AAAI Press, 2023. doi: 10.1609/AAAI.V37I10.26408. URL <https://doi.org/10.1609/aaai.v37i10.26408>.
- [6] Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6094–6101. AAAI Press, 2018.
- [7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [8] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009. ISSN 0010-0277. doi: <https://doi.org/10.1016/j.cognition.2009.07.005>. URL <https://www.sciencedirect.com/science/article/pii/S0010027709001607>. Reinforcement learning and higher cognition.
- [9] Pascal Bercher. A closer look at causal links: Complexity results for delete-relaxation in partial order causal link (POCL) planning. In Susanne Biundo, Minh Do, Robert Goldman, Michael Katz, Qiang Yang, and Hankz Hankui Zhuo, editors, *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2021, Guangzhou, China (virtual)*,

- August 2-13, 2021, pages 36–45. AAAI Press, 2021. URL <https://ojs.aaai.org/index.php/ICAPS/article/view/15944>.
- [10] Sara Bernardini, Fabio Fagnani, and Santiago Franco. An optimization approach to robust goal obfuscation. In Diego Calvanese, Esra Erdem, and Michael Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 119–129, 2020. doi: 10.24963/KR.2020/13. URL <https://doi.org/10.24963/kr.2020/13>.
- [11] Sara Bernardini, Fabio Fagnani, Alexandra Neacsu, and Santiago Franco. Optimizing pathfinding for goal legibility and recognition in cooperative partially observable environments. *Artif. Intell.*, 333:104148, 2024. doi: 10.1016/J.ARTINT.2024.104148. URL <https://doi.org/10.1016/j.artint.2024.104148>.
- [12] Sara Bernardini, Fabio Fagnani, and Santiago Franco. Speed vs accuracy in goal recognition for time-sensitive applications: A game-theoretic approach. In Sanmay Das, Ann Nowé, and Yevgeniy Vorobeychik, editors, *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, pages 280–288. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025. doi: 10.5555/3709347.3743541. URL <https://dl.acm.org/doi/10.5555/3709347.3743541>.
- [13] Blai Bonet and Hector Geffner. Planning as heuristic search. *Artif. Intell.*, 129(1-2):5–33, 2001.
- [14] Daniel Borrajo, Sriram Gopalakrishnan, and Vamsi K. Potluru. Goal recog-

- nitition via model-based and model-free techniques. *Proceedings of FinPlan 2020*, 2020.
- [15] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artif. Intell.*, 69(1-2):165–204, 1994. doi: 10.1016/0004-3702(94)90081-7. URL [https://doi.org/10.1016/0004-3702\(94\)90081-7](https://doi.org/10.1016/0004-3702(94)90081-7).
- [16] Sandra Carberry. Techniques for plan recognition. *User Model. User Adapt. Interact.*, 11(1-2):31–48, 2001.
- [17] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.
- [18] Mattia Chiari. *Deep Learning Approaches to Goal Recognition*. Doctoral thesis, Università degli Studi di Brescia, Brescia, Italy, April 2024. Disponibile su Iris Unibs. Tutor: Alfonso Emilio Gerevini, Ivan Serina.
- [19] Mattia Chiari, Alfonso Emilio Gerevini, Francesco Percassi, Luca Putelli, Ivan Serina, and Matteo Olivato. Goal recognition as a deep learning task: the grnet approach. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 560–568, 2023.
- [20] Mattia Chiari, Alfonso Emilio Gerevini, Andrea Loreggia, Luca Putelli, and Ivan Serina. Fast and Slow Goal Recognition. In *AAMAS 2024 (Accepted)*, 2024.
- [21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

- [22] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- [23] Patrick Ferber, Malte Helmert, and Jörg Hoffmann. Neural network heuristics for classical planning: A study of hyperparameter space. In *ECAI 2020*, pages 2346–2353. IOS Press, 2020.
- [24] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 1971.
- [25] M Bergamaschi Ganapini, Murray Campbell, Francesco Fabiano, Lior Horesh, Jon Lenchner, Andrea Loreggia, Nicholas Mattei, Francesca Rossi, Biplav Srivastava, and Kristen Brent Venable. Thinking fast and slow in ai: The role of metacognition. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 502–509. Springer, 2022.
- [26] Hector Geffner. Model-free, Model-based, and General Intelligence. In *Proceedings of IJCAI 2018*, 2018.
- [27] Christopher Geib and David Pynadath. Plan, activity, and intent recognition. *AI Magazine*, 28(4):124, 2007.
- [28] Christopher W. Geib and Mark Steedman. On natural language processing and plan recognition. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1612–1617, 2007.
- [29] Alfonso Gerevini and Ivan Serina. Planning as propositional CSP: from walksat to local search techniques for action graphs. *Constraints An Int. J.*, 8(4):389–413, 2003.

- [30] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. *J. Artif. Intell. Res.*, 20:239–290, 2003.
- [31] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.
- [32] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. ISBN 978-1-107-03727-4.
- [33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [34] Malte Helmert. A planning heuristic based on causal graph analysis. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 161–170. Morgan Kaufmann, 2004.
- [35] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
- [36] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What’s the difference anyway? In *ICAPS. AAAI*, 2009.
- [37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [38] Jörg Hoffmann. Ff: The fast-forward planning system. *AI Mag.*, 22:57–62, 2001. URL <https://api.semanticscholar.org/CorpusID:9968823>.

- [39] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *J. Artif. Intell. Res.*, 22:215–278, 2004.
- [40] Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301. IEEE, 2004.
- [41] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics, 2020.
- [42] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
- [43] Henry A. Kautz and James F. Allen. Generalized plan recognition. In *Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI 1986, Philadelphia, PA, USA*, pages 32–37. AAAI Press, 1986.
- [44] Emil Keyder and Hector Geffner. Heuristics for planning with action costs. In *CAEPIA*, volume 4788 of *Lecture Notes in Computer Science*, pages 140–149. Springer, 2007.
- [45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. URL <http://arxiv.org/abs/1412.6980>.

- [46] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.
- [47] Derek Long and Maria Fox. The 3rd International Planning Competition: Results and Analysis. *J. Artif. Intell. Res.*, 20, 2003.
- [48] Peta Masters and Sebastian Sardiña. Deceptive path-planning. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4368–4375. ijcai.org, 2017. doi: 10.24963/IJCAI.2017/610. URL <https://doi.org/10.24963/ijcai.2017/610>.
- [49] Mariane Maynard, Thibault Duhamel, and Froduald Kabanza. Cost-Based Goal Recognition Meets Deep Learning. *Proceedings of PAIR 2019*, 2019.
- [50] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. Pddl—the planning domain definition language. 1998. URL <https://api.semanticscholar.org/CorpusID:59656859>.
- [51] Drew V. McDermott. The 1998 AI Planning Systems Competition. *AI Mag.*, 21(2):35–55, 2000.
- [52] Felipe Meneguzzi and Ramon Fraga Pereira. A Survey on Goal Recognition as Planning. In *Proceedings of IJCAI 2021*, 2021.
- [53] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

- [54] Wookhee Min, Bradford W. Mott, Jonathan P. Rowe, Barry Liu, and James C. Lester. Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks. In *Proceedings of IJCAI 2016*. IJCAI/AAAI Press, 2016.
- [55] Reuth Mirsky, Roni Stern, Ya’akov (Kobi) Gal, and Meir Kalech. Sequential Plan Recognition. In *Proceedings of IJCAI 2016*. IJCAI/AAAI Press, 2016.
- [56] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, 2013.
- [57] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *KR*, pages 103–114. Morgan Kaufmann, 1992.
- [58] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [59] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. Landmark-based approaches for goal recognition as planning. *Artif. Intell.*, 279, 2020.
- [60] Alberto Pozanco, Yolanda E-Martín, Susana Fernández, and Daniel Borrajo. Finding centroids and minimum covering states in planning. In J. Benton, Nir Lipovetzky, Eva Onaindia, David E. Smith, and Siddharth Srivastava, editors, *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2019, Berkeley, CA, USA, July 11-15, 2019*, pages 348–352. AAAI Press, 2019. URL <https://ojs.aaai.org/index.php/ICAPS/article/view/3497>.

- [61] Adrian Price, Ramon Fraga Pereira, Peta Masters, and Mor Vered. Domain-independent deceptive planning. In Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh, editors, *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pages 95–103. ACM, 2023. doi: 10.5555/3545946.3598624. URL <https://dl.acm.org/doi/10.5555/3545946.3598624>.
- [62] Colin Raffel and Daniel PW Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*, 2015.
- [63] Miquel Ramírez and Hector Geffner. Plan Recognition as Planning. In *Proceedings of IJCAI 2009*, 2009.
- [64] Miquel Ramírez and Hector Geffner. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Proceedings of AAAI 2010*. AAAI Press, 2010.
- [65] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, 39:127–177, 2010.
- [66] Nicholas Rossetti, Massimiliano Tummolo, Alfonso Emilio Gerevini, Luca Putelli, Ivan Serina, Mattia Chiari, and Matteo Olivato. Learning general policies for planning through GPT models. In Sara Bernardini and Christian Muise, editors, *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, pages 500–508. AAAI Press, 2024. doi: 10.1609/ICAPS.

- V34I1.31510. URL <https://doi.org/10.1609/icaps.v34i1.31510>.
- [67] Luísa R. A. Santos, Felipe Meneguzzi, Ramon Fraga Pereira, and André Grahl Pereira. An LP-Based Approach for Goal Recognition as Planning. In *Proceedings of AAAI 2021*. AAAI Press, 2021.
- [68] Lorenzo Serina, Mattia Chiari, Alfonso Emilio Gerevini, Luca Putelli, and Ivan Serina. A preliminary study on BERT applied to automated planning. In *Proceedings of IPS 2022*, volume 3345 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022. URL <https://ceur-ws.org/Vol-3345/paper73460.pdf>.
- [69] Lorenzo Serina, Mattia Chiari, Alfonso Emilio Gerevini, Luca Putelli, and Ivan Serina. Towards efficient online goal recognition through deep learning. In Sanmay Das, Ann Nowé, and Yevgeniy Vorobeychik, editors, *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, pages 1895–1903. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025. doi: 10.5555/3709347.3743826. URL <https://dl.acm.org/doi/10.5555/3709347.3743826>.
- [70] Lorenzo Serina, Mattia Chiari, Matteo Olivato, Luca Putelli, Nicholas Rosseti, Ivan Serina, and Alfonso Emilio Gerevini. Mitigating deception and interference in online goal recognition systems. In *Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2026, Dublin, Ireland, June–July 27-2, 2026*. AAAI Press, 2026.
- [71] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDL planning with pre-

- trained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL <https://openreview.net/forum?id=1QMMUB4zfl>.
- [72] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 20256–20264. AAAI Press, 2024. doi: 10.1609/AAAI.V38I18.30006. URL <https://doi.org/10.1609/aaai.v38i18.30006>.
- [73] Shirin Sohrabi, Anton V. Riabov, and Octavian Udrea. Plan Recognition as Planning Revisited. In Subbarao Kambhampati, editor, *Proceedings of IJCAI 2016*. IJCAI/AAAI Press, 2016.
- [74] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Foundations of explanations as model reconciliation. *Artificial Intelligence*, 301:103558, 2021. ISSN 0004-3702. doi: <https://doi.org/10.1016/j.artint.2021.103558>. URL <https://www.sciencedirect.com/science/article/pii/S0004370221001090>.
- [75] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P. Goldman, editors. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann, 2014.
- [76] Austin Tate. Generating project networks. In *International Joint*

- Conference on Artificial Intelligence*, 1977. URL <https://api.semanticscholar.org/CorpusID:538251>.
- [77] Douglas Antunes Tesch, Leonardo Amado, and Felipe Meneguzzi. Real-time goal recognition using approximations in euclidean space. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 3589–3596. IOS Press, 2024. doi: 10.3233/FAIA240914. URL <https://doi.org/10.3233/FAIA240914>.
- [78] Franz A. Van-Horenbeke and Angelika Peer. Activity, plan, and goal recognition: A review. *Frontiers Robotics AI*, 8, 2021.
- [79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. URL <https://arxiv.org/pdf/1706.03762.pdf>.
- [80] Naveen Venkat. The curse of dimensionality: Inside out, 2018.
- [81] Mor Vered and Gal A. Kaminka. Online recognition of navigation goals through goal mirroring. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1748–1750. ACM, 2017. URL <http://dl.acm.org/citation.cfm?id=3091426>.
- [82] Mor Vered, Ramon Fraga Pereira, Maurício Cecílio Magnaguagno, Gal A.

- Kaminka, and Felipe Meneguzzi. Towards online goal recognition combining goal mirroring and landmarks. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 2112–2114. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL <http://dl.acm.org/citation.cfm?id=3238089>.
- [83] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
- [84] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical Attention Networks for Document Classification. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of NAACL HLT 2016*, 2016.
- [85] Håkan L. S. Younes and Reid G. Simmons. VHPOP: versatile heuristic partial order planner. *J. Artif. Intell. Res.*, 20:405–430, 2003. doi: 10.1613/JAIR.1136. URL <https://doi.org/10.1613/jair.1136>.
- [86] Chenyuan Zhang, Charles Kemp, and Nir Lipovetzky. Goal recognition with timing information. In Sven Koenig, Roni Stern, and Mauro Vallati, editors, *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling, Prague, Czech Republic, July 8-13, 2023*,

pages 443–451. AAAI Press, 2023. doi: 10.1609/ICAPS.V33I1.27224. URL
<https://doi.org/10.1609/icaps.v33i1.27224>.