Goal Recognition Using Off-The-Shelf Process Mining Techniques

Artem Polyvyanyy
The University of Melbourne
artem.polyvyanyy@unimelb.edu.au

Zihang Su
The University of Melbourne
zihangs@student.unimelb.edu.au

Nir Lipovetzky
The University of Melbourne
nir.lipovetzky@unimelb.edu.au

Sebastian Sardina
RMIT University
sebastian.sardina@rmit.edu.au

ABSTRACT

The problem of *probabilistic goal recognition* consists of automatically inferring a probability distribution over a range of possible goals of an autonomous agent based on the observations of its behavior. The state-of-the-art approaches for probabilistic goal recognition assume the full knowledge about the world the agent operates in and possible agent's operations in this world. In this paper, we propose a framework for solving the probabilistic goal recognition problem using process mining techniques for discovering models that describe the observed behavior and diagnosing deviations between the discovered models and observations. The framework imitates the principles of *observational learning*, one of the core mechanisms of social learning exhibited by humans, and relaxes the above assumptions. It has been implemented in a publicly available tool. The reported experimental results confirm the effectiveness and efficiency of the approach, both for rational and irrational agents' behaviors.

KEYWORDS

Goal recognition; process mining; observational learning

ACM Reference Format:
Artem Polyvyanyy, Zihang Su, Nir Lipovetzky, and Sebastian Sardina. 2020. Goal Recognition Using Off-The-Shelf Process Mining Techniques. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

We propose a technique for goal recognition (GR) that forgoes the need of predefined plan libraries and dynamic models, two onerous requirements in current GR approaches. Goal recognition[1] involves identifying an agent's intent by observing its behavior [8, 18]. With robots and agent software becoming ubiquitous, from smart houses to autonomous driving to video games, the ability to understand what *other* agents—be software, robots, or humans—are trying to accomplish is paramount to the realization of truly intelligent behavior and effective human-machine interaction. An intelligent aircraft support system should figure out what maneuver the pilot is executing, and a smart house ought to understand whether the household is trying to cook, watch a movie, or sleep by sensing her behavior (e.g., household entered the kitchen, turned on the light, and set the coffee machine timer). As the need for more autonomous systems increases, so does the attention on the GR problem [35], with numerous applications, from the support of adversarial reasoning for games and the military [20, 36] to smart homes and wheelchair movement [10, 34] to trajectory/manoeuvre prediction [12, 19, 21] and human-computer collaboration [22].

In traditional GR approaches, observations of agent's actions are "matched" to a plan (the one judged to being carried out by the agent) in a predefined library encoding the standard operational procedures of the domain [8, 11, 18]. While the first proposals did not accommodate uncertainty, numerous subsequent probabilistic solutions have later been developed [35]. Nonetheless, the challenge to obtain or hand-code the possible set of activities in a domain has triggered research in GR techniques that *dispense from plan libraries altogether* [16]. In particular, more recent work by Ramirez and Geffner [32, 33] has sparkled a plethora of approaches to perform GR by exploiting planning systems to automatically generate plans relative to a domain theory, hence not needing any a priori set of plans. Appealing to the principle of rational behavior, an agent is assumed to be taking the "optimal" path to the goal: the more rational a behavior appears towards a goal, the more probable such goal is the agent's goal. Notwithstanding the fact that specifying dynamic models could be less demanding than hand-coding plans in some domains, and that "new" plans can be found, the acquisition of such domain models at the outset is still far from trivial [15].

Relying on the fact that, in many settings, logs of recorded traces are readily available, we propose an approach to probabilistic GR that analyzes deviations of the observed behavior from process models that are automatically discovered using *process mining techniques* [38]. In this way, we are able to perform GR without the need of either predefined plan libraries or planning domain models, just traces. Importantly, traces are *not* complex structured plans (and logs are not plan libraries), but they do stand for execution instances of some underlying "hidden" standard operational procedures of the domain. Because of all this, we argue our proposal sits between traditional plan recognition (based on reasoning over plans) and more recent planning-based approaches (based on reasoning over cost difference). In addition, it is designed to imitate the principles of *observational learning*, one of the core mechanisms of social learning exhibited by humans.

[1] While subtleties do exist among them, we shall use the terms *goal*, *intention*, *plan*, or *activity* recognition interchangeably.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

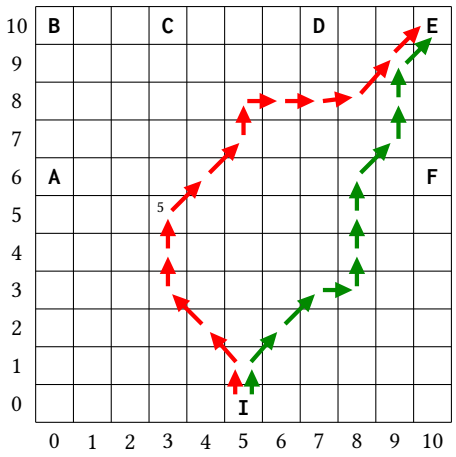


Figure 1: Two walks of an agent in a grid from initial cell I to goal cell E: rational (green) and irrational (red).

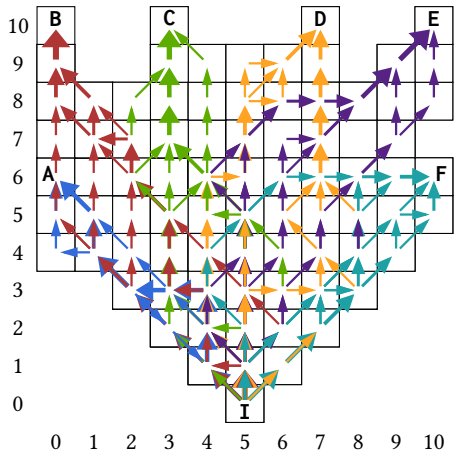


Figure 2: Observed agent behaviors from initial cell I to the six goal cells from Figure 1.

After explaining the approach, we report experimental results providing evidence of its effectiveness. In particular, we conducted goal recognition in three real-world domains from the Business Process Intelligence community *for which only event logs are available*, two regarding permits (building and environmental) and one on human daily activities. Such experiments show that, unlike previous approaches to GR, ours can be applied when no planning domains or predefined domain processes are available. For completeness, though, we also report experiments on domains where planning domains are readily available, by synthesizing traces compatible with such domains. Nonetheless, we stress that the aim of this work is *not* per se to improve on GR performance when predefined plans or dynamic models are at disposal (despite the fact that the conducted experiments evidence our approach is scalable), but to be able to achieve GR in their absence.

2 MOTIVATING EXAMPLE

Figure 1 shows two walks of an agent in an 11x11 grid that both start at the cell denoted by I and end at the cell denoted by E. Cells A to F represent six goals in the grid. Hence, the walks represent two observed agent's behaviors for achieving goal E starting from I. To achieve a goal, the agent can perform horizontal and vertical moves at the cost of 1, and diagonal moves at the cost of $\sqrt{2}$. The grid, the positions of the initial cell and the six goal cells, and the red walk from I to E are taken from [33]. The green walk in the figure has the cost of $7 + 4\sqrt{2}$. As this cost is close to the cost of an optimal walk from I to E, i.e., $5 + 5\sqrt{2}$, we say that it is *rational*. The red walk has the cost of $7 + 6\sqrt{2}$ and is, thus, *irrational*. Indeed, the red walk starts towards goal A and then diverges to visit cells close to goals C, D, and F prior to reaching target goal E.

Often, the environment the agent operates in and the moves it can take are not known. However, one can learn both by observing agent's behavior. For each of the six goals from Figure 1, Figure 2 shows "footprints" of six sample observed walks of the agent from cell I to that goal; the thickness of arrow denotes the frequency the corresponding move was taken (only explored cells are shown).

Using these observations, one can construct models that describe *skills* for accomplishing the goals. Then, new observations in the environment can be compared with the acquired models to infer a probability distribution over the goals. Intuitively, the more discrepancies between the observed behavior and a skill model, the less likely the agent is attempting to achieve the corresponding goal.

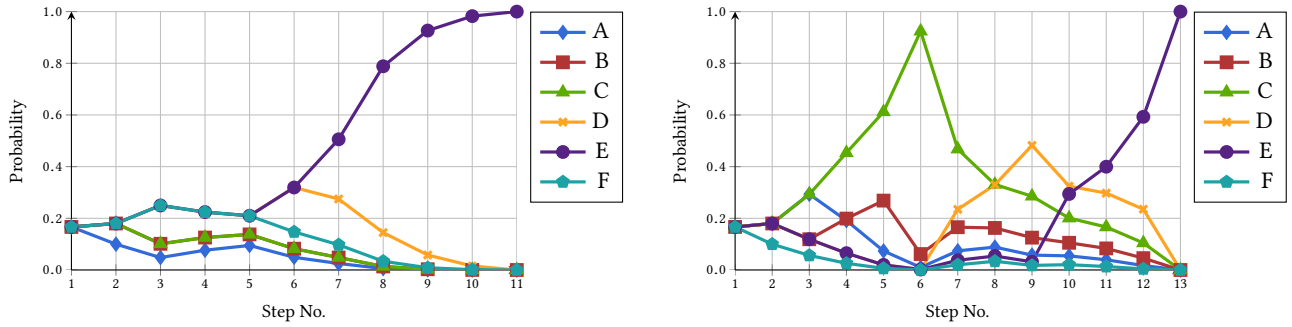
Figures 3(a) and 3(b) show probability distributions over the goals as functions of time (number of moves) for the walks from Figure 1. The distributions were inferred using the discrepancies captured in *alignments*, cf. Section 3.2, between the skill models discovered from the observations in Figure 2 and the walks. Goal E is consistently (one of) the most likely goal(s) along the rational walk. Note that goals A, B, C, and D prevail at the start of the irrational walk, while goal E is identified as the most likely goal towards the end.

3 BACKGROUND

3.1 Probabilistic Goal Recognition as Planning

A decade ago, Ramirez and Geffner [32, 33] proposed to move away from intention recognition over a *plan library* and considered the problem, instead, over a domain theory and a set $\mathcal{G}$ of possible candidate goals. To do so, they leveraged on the insight that a rational agent is expected to be taking the optimal, or close to optimal, plan to its (hidden) goal, also noted in [3, 27], so the probability of a plan can be linked to its cost. And candidate plans can be automatically synthesised using planning technology [14]! By doing this, the proposal eliminated the need to obtain (or hand-code) the typical processes in a domain, and allows the recognition problem to leverage advances in automated planning.

Technically, the input is a planning domain encoding the dynamics of the domain (i.e., a theory of action), an initial state, a set of *possible* goal states, and a sequence of observed actions. The solution is a posterior probability distribution which "prefers" those goals whose "best" plans satisfy observations. In their [2010] work, Ramirez and Geffner derive such probability distribution over the possible goals from Bayes' Rule based on the assumption that *the probability of a plan is inversely proportional to its cost*. Such assumption is encapsulated in the notion of *cost difference* between the (cost of the) optimal plan for a goal *matching the observed actions* and the optimal plan that could have been reached otherwise (not embedding the observed actions). Concretely, this yields a Boltzmann-like sigmoidal distribution with the important property that *the lower the cost difference, the higher the probability.*



(a) Rational walk in Figure 1.

(b) Irrational walk in Figure 1.

Figure 3: Inferred probability distributions over the six goals from Figure 1 based on the observed behaviors shown in Figure 2.

Several works have followed elaborating or extending Ramirez and Geffner's setup, or grounding it to specific interesting settings (e.g., navigation). We shall adopt a recent elaboration by Masters and Sardina [23–25], which refined the original set-up to achieve a simpler and computationally less demanding GR account, and one able to handle irrational agent behavior parsimoniously without counter-intuitive outcomes. Concretely, taking $optc(S, \tau, G)$ to denote the optimal cost of reaching goal G from state S by *embedding* the sequence of observations τ,[2] we first define the ***cost difference*** of reaching G from S via observations τ as follows:

$$costdiff(S, \tau, G) = optc(S_I, \tau, G) - optc(S, \epsilon, G).$$

When the agent is observed acting optimally for G, the cost difference is zero; as the agent becomes more suboptimal towards G, the cost difference increases. Using this, and assuming for simplicity that all goals are initially equally likely, the probability of a candidate goal $G \in \mathcal{G}$ given observations τ can be obtained as follows (note the denominator acts here as the normalization factor):

$$\Pr(G \mid \tau) = \frac{e^{-\beta \times costdiff(S_I, \tau, G)}}{\sum_{G' \in \mathcal{G}} e^{-\beta \times costdiff(S_I, \tau, G')}}, \quad (1)$$

where β is a parameter that allows the goal "recognition system developers to soften the implicit assumption of the agent being rational" [33]. In Masters and Sardina's self-modulating GR account [25], β is *dynamic* and adjusts the *confidence* of the GR system based on the degree of agent rationality observed so far; formally $\beta \in (0, 1]$ is defined as follows (here, η is a positive constant regulating how quickly confidence should drop when irrational behaviour is seen):

$$\beta = \left(\max_{G \in \mathcal{G}} \frac{optc(S_I, \epsilon, G)}{optc(S_I, \tau, G)} \right)^{\eta}. \quad (2)$$

Intuitively, β expresses the most optimistic rationality ratio among all goals, with $\beta = 1$ when the agent is seen fully rational towards *some* goal. By using this dynamic parameter, the more erratic the agent behavior (irrational to all goals), the more $\Pr(\cdot)$ approaches a uniform distribution.

Below, we will adapt and instantiate the above account to apply notions and techniques from process mining [38].

[2] Under no observations (i.e., $\tau = \epsilon$), $optc(S, \tau, G)$ reduces to optimal cost to G.

3.2 Process Mining

Process mining studies methods, techniques, and tools to discover, monitor, and improve processes carried out by organizations using the knowledge accumulated in event logs recorded by information systems that support execution of business processes [38]. An ***event log***, or *log*, is a collection of traces, where each ***trace*** consists of a sequence of timestamped actions observed and recorded during execution of a single case of a business process.

One can encode each collection of six sequences of actions towards each of the six goals from Figure 2 in an event log. Log $L_{\mathtt{A}} = \{\tau_1, \ldots, \tau_6\}$ that encodes the six walks towards goal A contains six traces each capturing the moves from the corresponding walk. For example, traces $\tau_3, \tau_5 \in L_{\mathtt{A}}$ are given as $\tau_3 = \langle \binom{5,0}{4,1}, \binom{4,1}{4,2}, \binom{4,2}{4,3}, \binom{4,3}{3,3}, \binom{3,3}{2,3}, \binom{2,3}{1,4}, \binom{1,4}{1,5}, \binom{1,5}{0,6} \rangle$ and $\tau_5 = \langle \binom{5,0}{4,1}, \binom{4,1}{4,2}, \binom{4,2}{4,3}, \binom{4,3}{3,3}, \binom{3,3}{2,3}, \binom{2,3}{3,2}, \binom{3,2}{4,1}, \binom{4,1}{4,2}, \binom{4,2}{4,3}, \binom{4,3}{3,3}, \binom{3,3}{2,3}, \binom{2,3}{1,4}, \binom{1,4}{1,5}, \binom{1,5}{0,6} \rangle$, where elements represent moves in the grid. For instance, the first element $\binom{5,0}{4,1}$ in τ_3 encodes the move from cell $(5, 0)$ to cell $(4, 1)$, a move away from cell I.

Two core problems tackled in process mining research are *process discovery* and *conformance checking*. Process discovery aims to automatically construct a "good" process model from an event log [2, 40, 41], while conformance checking measures and explains commonalities and discrepancies, i.e., the "goodness", of the model with respect to the log it is constructed from [9, 31, 39, 42]. In a nutshell, a *process discovery technique* takes a log as input and produces a ***process model***, e.g., a Petri net, BPMN model, UML activity diagram, or an EPC, as output [38].

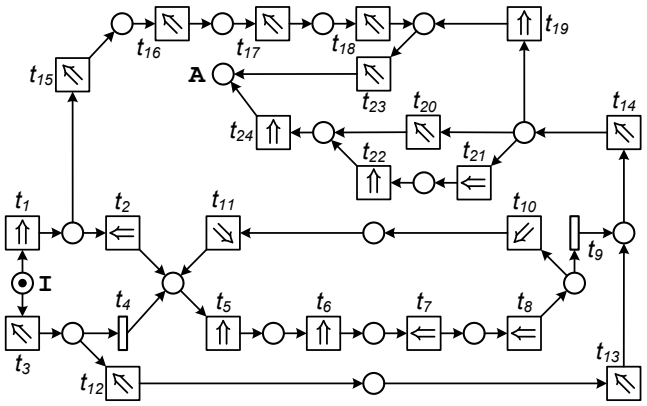


Figure 4: A net discovered from walks to goal A in Figure 2.

Let $\mathbb{L}$ and $\mathbb{M}$ be a universe of logs and a universe of process models, respectively. A ***process discovery technique*** is a function that maps event logs onto process models, i.e., $\pi : \mathbb{L} \to \mathbb{M}$. We say that the model $\pi(L)$, $L \in \mathbb{L}$, is discovered from L using technique π. For example, the model in Figure 4 is the Petri net discovered from log L_A mentioned above using the Split miner discovery technique [2]. The net aims to describe the traces from the log and generalize the "useful" behavior. In particular, it encodes all the six traces in log L_A and generalizes the repetitive fragment in trace τ_5, via transitions $t_5, t_6, t_7, t_8, t_{10}$, and t_{11}. Note that every transition in the net encodes a move in the grid. For example, transitions t_1 and t_5, despite both capturing a move towards north, describe two different moves in the grid, namely $\binom{5,0}{4,1}$ and $\binom{4,1}{4,2}$, respectively.

One of the central concepts in conformance checking is the concept of an ***alignment***. An alignment describes a relation between a trace and an execution of a process model as a sequence of *steps* [39]. Model and trace make a ***synchronous step*** in the alignment if they agree, i.e., they both take the same action. If trace and model disagree, either the model or trace takes an action that is not mimicked by the counterpart in the alignment. An ***optimal alignment*** between a model and trace is an alignment that yields the lowest cost among all the possible alignments between the trace and model according to some predefined cost scheme over the steps.

Synchronous steps are often assigned zero cost, while all other steps are given some positive cost. For such a cost scheme, an optimal alignment describes some minimal disagreements between the model and trace. Then, a trace and model ***agree perfectly***, i.e., every action in the trace can be mimicked by an action in the model, iff there exists an optimal alignment of zero cost between them.

It is convenient to represent alignments as tables. Below, γ_1 is an optimal alignment between the prefix of size five of the irrational walk from Figure 1 and the net in Figure 4, while γ_2 is an optimal alignment between the irrational walk and the net discovered from the traces towards goal E from Figure 2.[3]

$\gamma_1 =$

$\binom{5,0}{5,1}$	$\binom{5,1}{4,2}$	$\binom{4,2}{3,3}$					$\binom{3,3}{3,4}$	$\binom{3,4}{3,5}$
⇑	⇖	⇖	≫	≫	≫	⇑	⇑	
⇑	⇖	⇖	⇖	⇖	⇖	≫	≫	
$\binom{5,0}{5,1}$	$\binom{5,1}{4,2}$	$\binom{4,2}{3,3}$	$\binom{3,3}{2,4}$	$\binom{2,4}{1,5}$	$\binom{1,5}{A}$			
t_1	t_{15}	t_{16}	t_{17}	t_{18}	t_{23}			

$\gamma_2 =$

$\binom{5,0}{5,1}$	$\binom{5,1}{4,2}$					$\binom{4,2}{3,3}$	$\binom{3,3}{3,4}$	$\binom{3,4}{3,5}$	$\binom{3,5}{4,6}$	$\binom{4,6}{5,7}$		$\binom{5,7}{5,8}$	$\binom{5,8}{6,8}$	$\binom{6,8}{7,8}$	$\binom{7,8}{8,8}$	$\binom{8,8}{9,9}$	$\binom{9,9}{E}$	
⇑	⇖	≫	≫	≫	≫	⇖	⇑	⇑	⇗	⇗	≫	⇑	⇒	⇒	⇒	⇗	⇗	
⇑	⇖	⇑	⇗	⇑	⇖	≫	≫	≫	≫	⇗	⇗	≫	≫	⇒	⇒	⇗	⇗	
$\binom{5,0}{5,1}$	$\binom{5,1}{4,2}$	$\binom{4,2}{4,3}$	$\binom{4,3}{5,4}$	$\binom{5,4}{5,5}$	$\binom{5,5}{4,6}$					$\binom{4,6}{5,7}$	$\binom{5,7}{6,8}$			$\binom{6,8}{7,8}$	$\binom{7,8}{8,8}$	$\binom{8,8}{9,9}$	$\binom{9,9}{E}$	

In a table that describes an alignment, steps are encoded as columns. Two successive columns in a table refer to two successive steps in the alignment. In γ_1, each column has five rows. The top two rows of a column correspond to the *trace contribution* to the step. They encode either an action from the trace, e.g., a direction and coordinates of a move by the agent in the grid, or a ***skip*** denoted by '≫'. The bottom three rows of a column correspond to the *model contribution* to the step. They encode either an execution of an action in the model, i.e., an occurrence of a transition that describes a move of the agent in the grid, or again, a *skip* '≫'. For example, the first step in γ_1 represents the occurrence of transition t_1 in the net from Figure 4 that describes the move from cell $(5, 0)$ to cell $(5, 1)$. Steps at positions one, four, and seven in γ_1 are, respectively, examples of synchronous, asynchronous (on model), and asynchronous (on trace) steps. Note that in γ_2 we omit the fifth row.

Alignments γ_1 and γ_2 have costs of $2 + 3\sqrt{2}$ and $6 + 5\sqrt{2}$, respectively, assuming zero cost of all synchronous steps, cost of $\sqrt{2}$ of all asynchronous diagonal steps, and cost of 1 of other asynchronous steps, following the cost model of the motivating example.

4 APPROACH

In this section, we first present an approach for performing probabilistic GR based on alignments with mined processes (Section 4.1). After that, we present a GR agent architecture that mimics the principles of *observational learning* (from social cognitive learning theory) and can be operationalized with process mining techniques and our alignment-based inference approach (Section 4.2).

4.1 Goal Recognition over Alignments

In Section 3.1, we explained how probabilistic GR can be realized by relying on the so-called cost-difference for each candidate goal. To get such cost-differences, optimal plans are synthesized relative to a (PDDL) model of the domain.

To get away from the burden of obtaining or crafting domain models, we shall propose a GR probability distribution *relative to a set of Petri net models* that are assumed to have been automatically extracted from recorded past executions, one model per candidate goal (for example, by observing and recording the household's actions for a period of time). More concretely, given a set of goals $\mathcal{G}$ and a sequence of observations τ, we adapt Equation (1) by *re-stating cost-difference in terms of (level of) misalignment* between τ and each learned Petri net model α_G, with $G \in \mathcal{G}$. The less misalignment the observed behavior τ displays against the skill model α_G learned for goal G, the most probably G is the true goal.

So, considering $\omega(\tau, \alpha_G)$ to be an alignment weight (defined below) of τ against the model α_G learned for goal G, we follow Masters and Sardina [23] in using a true Boltzmann distribution instead of a sigmoidal, and re-write Equation (1) as follows:

$$\Pr(G \mid \tau) = \frac{e^{-\beta \times \omega(\tau, \alpha_G)}}{\sum_{G' \in \mathcal{G}} e^{-\beta \times \omega(\tau, \alpha_{G'})}}. \tag{3}$$

Here, $\omega(\tau, \alpha_G) \geq 0$ and the "temperature" β controls the GR level of confidence; can also be interpreted as the trust over the learned models. For the temperature parameter, we define:

$$\beta = \frac{1}{1 + \min_{G \in \mathcal{G}} \omega(\tau, \alpha_G)}. \tag{4}$$

While apparently different, this actually follows Equation (2), simplified given that the best case scenario is an alignment cost of zero.[4] As the minimum (for all goals) alignment weight ω increases,

[3] The net discovered from walks to goal E is not shown. However, all the logs shown in Figures 1 and 2 and the Petri nets discovered from the logs that describe walks towards the six goals in Figure 2 captured using the XES standard (https://xes-standard.org/) and PNML standard (http://www.pnml.org/), respectively, can be accessed here: https://github.com/zihangs/fp1189_aamas2020/tree/master/training_examples.

[4] This implies we inherit the confidence-based properties described in [25].

the agent is arguably more "irrational", β approaches zero and the GR probability distribution resembles more the uniform one.

Finally, the ***alignment weight*** between an observation trace $\tau = \langle a_1, \ldots, a_n \rangle$ and a skill model α_G is defined as:

$$\omega(\tau, \alpha_G) = \phi + \lambda^m \times \sum_{i=1}^{n} \Big(i^{\delta} \times c(\tau, \alpha_G, i) \Big), \text{where} \qquad (5)$$

- $c(\tau, \alpha_G, i)$ is the cost of the step in an optimal alignment between trace τ and model α_G that involves trace action a_i;
- i^{δ}, with $\delta \geq 0$, is a discount factor that emphasizes later—more recent—disagreements;
- $\phi \geq 0$ is a constant to "smooth" the likelihoods of various goals when the trace results in (close-to-)perfect alignments (indirectly setting the max. β allowed); and
- λ^m, $\lambda \geq 1$ is a constant and m is the number of consecutive asynchronous steps on trace at the end of the optimal alignment to penalize when the suffix of the trace cannot be aligned.

We observe that, as shown in the next section, all these constants are useful in the presence of irrational agents, but can be omitted in practice if all agents are (close to) rational. To avoid artificially penalizing short traces, we only account for the cost of asynchronous steps on trace, as the model would typically complete a short trace with asynchronous steps on model until the goal is reached.

Given $\phi = 0$, $\lambda = 2$, and $\delta = 1$, the weight (ω) of alignment γ_1 from Section 3.2 is $0 + 2^2 \times (1^1 \times 0 + 2^1 \times 0 + 3^1 \times 0 + 4^1 \times 1 + 5^1 \times 1) = 36$; the computation only uses steps that involve trace actions to measure the inability to match the trace with the skill model. Here, we used costs of asynchronous steps from the motivating example. Note that the weight (ω) of the alignment of the same prefix of the irrational walk from Figure 1 and the model for skill E, based on the first nine steps of alignment γ_2 from Section 3.2, is equal to $0 + 2^3 \times (1^1 \times 0 + 2^1 \times 0 + 3^1 \times \sqrt{2} + 4^1 \times 1 + 5^1 \times 1) \approx 105.94$; same step costs were used. Because the weight of disagreement for goal E is greater than that for goal A, after five steps along the irrational walk, our approach would suggest that it is more likely that the agent aims for goal A than for goal E, as it is done in Figure 3(b).[5] Finally, the weight (ω) of alignment γ_2 from Section 3.2, assuming the constant values and step costs from above, is equal to $26 + 9\sqrt{2}$.

Putting it all together, Equation (3) provides a novel middle ground between traditional plan-library-based activity recognition and the more recent cost difference based approach from the planning literature. Indeed, observations are matched to a sort of library of plans, implicitly represented and aggregated in each Petri net skill model, by re-interpreting plan cost as level of misalignment.

Next, we show how the proposed GR account can be framed within an long-lived intelligent system.

4.2 Towards Goal Recognition Framework

While significant effort has been devoted to develop models for GR, not much emphasis has been put in studying how to operationalize the GR process within a full intelligent agent system. Such systems are not meant to merely solve one shot problems, but to run continuously for extended periods. Next, we present our framework consisting of components that can be selectively replaced to embed in such a system. The framework is inspired by the principles of *observational learning*, which is concerned with the acquisition of attitudes, values, and styles of thinking and behaving via observation of the examples provided by others [4]. Recently, observational learning was simulated by reinforcement learning with memory and without explicit modeling of the observed "teacher" agent [7]. Observational learning occurs via four functions:

(1) Attentional function (*Attention*) governs which observed stimuli a student agent selects from the teacher's behavior to implement learning, as not all observations may be of interest to the student;
(2) Representational function (*Retention*) concerns with transforming observations of the teacher's behavior into models suitable for informing the student's behavior;
(3) Motivational function (*Motivation*) controls when and how the learned knowledge is used by the student, including cost and benefit analysis of applying the knowledge in the environment;
(4) Production function regulates transformation of fresh observations and conceptions of student into a course of her actions.

In our work, the production function is concerned with applying the knowledge retained by the student to recognize the goal the teacher agent attempts to achieve (*Recognition*). Figure 5 shows our GR framework schematically. It consists of four parts that implement the four functions of observational learning, each composed of a collection of workflows intended for simultaneous execution:

(1) The *Attention* part (top-left of Figure 5) monitors teacher agents' actions and triggers two signals, viz. "*Action captured*" and "*Goal completion recognized*." The former is triggered when the student identifies a fresh action of a teacher, e.g., the figure shows that the student captured action w performed by teacher B, while the latter is triggered when the student recognizes that a teacher has achieved a goal, e.g., teacher A achieved goal α.
(2) The *Retention* part (top-right) processes the signals triggered within the *Attention* part. Once an "*Action captured*" signal is picked up, the corresponding action is added to the current trace of the teacher agent (cf. the "*Retain action*" activity in Figure 5), and then the "*Action retained*" signal is raised. Once the "*Goal completion recognized*" signal is received, the trace of the teacher that has led to the completion of the goal is added to the corresponding *skill library* ("*Retain skill trace*"); in the figure, trace $\alpha_5 = \langle e, e, e, n, e, w, s \rangle$ performed by teacher A to achieve goal α is added to the α-skill library. Then, a process discovery technique is applied to construct a *skill model* from the traces in the skill library to replace the old model for the corresponding goal ("*Discover skill model*"). Consequently, a skill model aggregates and generalizes the observed behavior for achieving the corresponding goal. In this work, we implemented the "*Discover skill model*" activity using the Transition System miner [40].
(3) The *Motivation* part (bottom-left) processes captured "*Action retained*" signals by filtering those that correspond to the observed actions of interest and subsequently triggering the "*Goal recognition initiated*" signal; e.g., in the figure, the student agent decides to initiate GR based on the observed action w by teacher B. Otherwise, the processing of the observed action terminates.
(4) The Recognition part (bottom-right) governs the inference of possible goals. The processing starts with the consumption of a "*Goal recognition initiated*" signal. Next, the corresponding

[5] Note that the probability distributions in Figure 3 were obtained using constants $\phi = 50$, $\lambda = 1.1$, and $\delta = 1$.

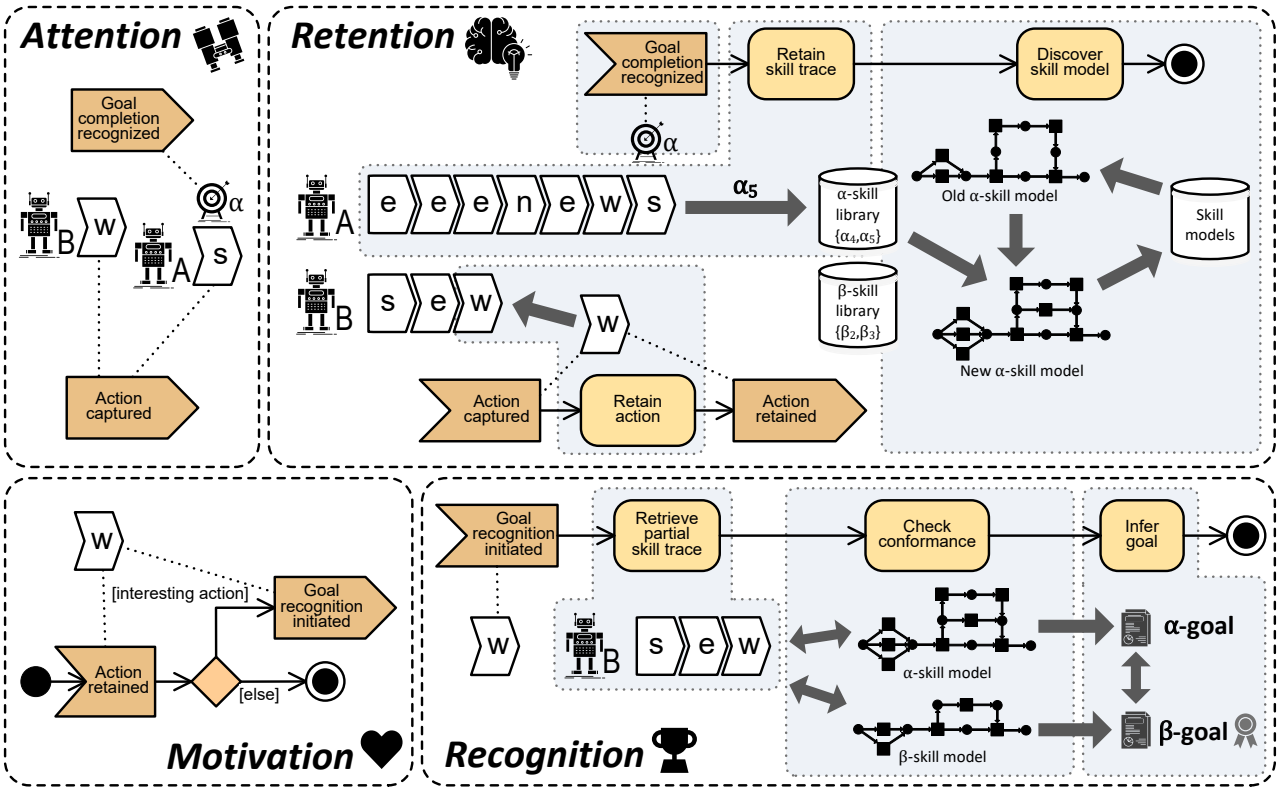


Figure 5: Schematic visualization of our goal recognition framework.

observed fragment of the agent's trace is retrieved ("*Retrieve partial skill trace*"); in the figure, $\langle s, e, w \rangle$ performed by agent B is retrieved. Then, the fragment is checked for conformance with all the retained skill models to produce *conformance diagnostics* ("*Check conformance*"). Finally, the diagnostics are used to infer the insights into the goal the observed agent attempts to achieve ("*Infer goal*"). In this work, the "*Check conformance*" activity is operationalized with alignments, refer to Section 3.2, between the retained models and the trace fragment. The operationalization of the "*Infer goal*" activity is detailed in Section 4.1; Figure 5 suggests that β was inferred as the true goal of agent B.

5 EXPERIMENTAL RESULTS

The presented approach for GR has been implemented and is publicly available.[6] Using this implementation, we evaluated our GR framework over five domains: three event logs made publicly available by the process mining community and two classical planning domains; collectively, these five domains constitute a heterogeneous collection of GR problems. The event logs are real-world records of performed actions towards different goals and, thus, are readily amenable for GR using our approach. To test the robustness of our probabilistic models for GR, we used classical domain models from planning to simulate collections of traces of different characteristics towards various goals. For each experiment, we computed two measures in terms of true positive (*TP*), false positive (*FP*), and false negative (*FN*) inferred goals: *precision* defined as $TP/(TP + FP)$ and *recall* defined as $TP/(TP + FN)$. In our experiments, $TP, FN \in \{0, 1\}$, as there is only one true hidden goal per problem, while $FP \in \{0, \ldots, |\mathcal{G}| - 1\}$, where $\mathcal{G}$ stands for the set of goal hypothesis. The traces and skill models used to obtain the results presented below are publicly available.[7]

All the experiments were conducted on a dual-processor Xeon(R) 'Gold' running a 3.00GHz, 24GB of RAM, using parameters $\phi = 50$, $\lambda = 1.1$, and $\delta = 1$. We also tested $\phi = 0$, $\lambda = 1$, $\delta = 0$ and noticed that results only change if traces are significantly sub-optimal.

Next, Section 5.1 presents the results of our experiments with real-world event logs, while Section 5.2 is devoted to the discussions of the results of the experiments with synthetic data.

5.1 Process Mining Domains

In this section, we evaluate our GR framework using three real-world event logs, namely activities of daily living of several individuals, and processes of handling building permit and environmental permit applications by five Dutch municipalities. Each trace of each event log can be seen as an instance of a business process aimed to achieve a goal. In the reported experiments, we use 80% and 60% of traces in each log for training, i.e., to discover the corresponding skill model, and the remaining 20% and 40% of traces for actual GR.

Activities of Daily Living. This dataset consists of eight event logs capturing the actions of daily living of four individuals, e.g., sleeping, cooking, and cleaning; the dataset is publicly available.[8]

[6] https://github.com/zihangs/fp1189_aamas2020/tree/master/tools

[7] https://github.com/zihangs/fp1189_aamas2020/tree/master/datasets

[8] https://doi.org/10.4121/uuid:01eaba9f-d3ed-4e04-9945-b8b302764176

80 / 20	Daily Living			Build. PRMT			Env. PRMT		
%O	p	r	t	p	r	t	p	r	t
10	0.47	0.77	0.15	0.29	0.47	2.90	0.31	0.49	2.00
30	0.54	0.65	0.26	0.45	0.49	6.79	0.45	0.50	4.64
50	0.62	0.71	0.37	0.55	0.57	10.55	0.45	0.49	6.99
70	0.61	0.71	0.48	0.65	0.66	14.10	0.55	0.60	8.85
100	0.62	0.74	0.65	0.74	0.77	18.32	0.57	0.69	10.46
60 / 40	Daily Living			Build. PRMT			Env. PRMT		
%O	p	r	t	p	r	t	p	r	t
10	0.42	0.69	0.04	0.33	0.48	1.92	0.32	0.50	1.15
30	0.51	0.60	0.06	0.47	0.52	4.48	0.44	0.49	2.80
50	0.57	0.63	0.08	0.57	0.59	7.06	0.47	0.50	4.23
70	0.51	0.57	0.10	0.62	0.63	9.47	0.50	0.55	5.41
100	0.54	0.60	0.14	0.70	0.72	12.45	0.60	0.67	6.26

Table 1: GR results for the three real-world datasets showing for each % of sampled actions from observed trace (%O), avg. precision (p), avg. recall (r), and avg. time in seconds (t) to infer a goal based on a given trace fragment.

For each individual, two event logs were obtained: one for weekdays and one for weekends. Each trace was derived from sensor data and represents a daily routine of the corresponding individual at own home. The logs show both similar and different behavior patterns of the individuals. The event logs contain from 6 to 43 traces capturing from 368 to 4200 actions and were used in [37] for analysis and monitoring of personal processes.

The results of our GR experiment with this dataset are reported in Table 1 (column "Daily Living"). The table shows average precision, recall, and time to infer a goal, i.e., to determine the individual. Note that each skill model was discovered within one second, and often much faster; this performance is due to the small number of traces in the logs. It is evident that the increase of the number of training traces leads to qualitatively better predictions, as both precision and recall strictly increase when increasing the percentage of training traces from 60% to 80%. Interestingly, for this dataset, the precision and recall values are not sensitive to the percentage of sampled actions, i.e., they are similar across the experiments. A notable exception in this regard is precision for the 80% training experiments, which tends to increase with the increase in the number of sampled actions. Finally, we observe that, for this dataset, the approach achieves high precision and recall values for both sizes of training sets across different numbers of sampled actions. This indicates that once a sufficient amount of training is attained, the approach can be used as an early predictor of the agent's goal, i.e., can deliver accurate GR results based on a few observed actions.

Building Permit Applications. This dataset consists of five event logs recording processes for handling building permit applications by five Dutch municipalities, one municipality per log, over a period of approximately four years. The event logs contain from 832 to 1199 traces capturing from 44 354 to 59 083 actions. This dataset was made publicly available to the process mining community as part of the Business Process Intelligence Challenge 2015.[9] All the traces try to reach a similar goal using a similar set of actions, making this dataset particularly challenging for classifying an observed trace as originating from a particular municipality.

The results of our GR experiment with this dataset are reported in Table 1 (column "Build. PRMT"). Again, the average precision, recall, and time to infer a goal are reported; in this case for five goals pursued by the five municipalities. For this dataset, the goal inference took more time, namely from two to eighteen seconds on average, which is due to a high number of traces in the logs the skill models were learned from. The similar values of precision and recall for both percentages of training traces indicate that it is not always required to strive to learn skill models from high volumes of data. Instead, one should look for a point of "saturation" of skill models that permits high-quality GR. Finally, for this dataset, early predictions are of lower quality, with high precision and recall achieved only at and after observing 50% of sampled actions.

Environmental Permit Applications. This dataset consists of five event logs recording processes for handling environmental permit applications by five Dutch municipalities, one municipality per log. The event logs contain from 645 to 1087 traces capturing from 33 373 to 44 801 actions. This dataset was made publicly available to the process mining community as part of the CoSeLoG project executed under NWO project number 638.001.211.[10]

The GR results for this dataset are reported in Table 1, refer to the "Env. PRMT" column; again, the average values over five goals are shown. The general trends of the measurements for this experiment are consistent with those obtained for the building permit dataset. However, the measured precision and recall values for the environmental permit dataset are consistently lower, which may indicate that either more training is required, or the data is particularly challenging for distinguishing the goals.

5.2 Planning Domains

In this section, we evaluate our GR framework using two classical planning domains, namely *Navigation*, or grids, and *Blocksworld*. As planning domains are not directly applicable for our purpose, they describe rules one can follow to behave in the world and do not provide sample traces in the world, we generated training and testing traces by obeying the domain rules, refer to details below.

Navigation. The first domain we look at is a path-finding problem over *grids*, cf. Section 2, of different sizes, with initial location at bottom-middle cell and goals located at the left-, top-, and right-middle cells when only three goals are tested. Goals are evenly distributed across the left and right columns and top row when more goals are tested. For each problem, we learn skill models from 1000 traces, uniformly distributed among the goal candidates, and generated 100 new traces for testing. We test the implications of learning a model from traces with different levels of irrationality, namely traces whose costs are between $X\%$ and $(X + 10)\%$ larger than the optimal, while using observations for testing that match or differ from the level of rationality used for training.

The results in Table 2 show that the learned models tend to have perfect precision and recall for settings with three goals but for traces that contain only 10% or 30% of the observations (%O). As expected, with increasing number of goals, precision and recall drops as goals are less differentiable. Remarkably, when the model is learned from an agent acting with a level of irrationality that differs from the traces used to define the new observations (last multi-column in the table), still, the precision and recall remain almost perfect. The lowest scores occur in the smaller grids with more goals, as goals are closer to each other and traces share many

[9] https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

[10] https://doi.org/10.4121/uuid:26aba40d-8b2d-435b-b5af-6d4bfbd7a270

Irrationality		0%–10% train&test			10%–20% train&test			20%–30% train&test			0%–10% train 20%–30% test		
Grid	%O	p	r	t	p	r	t	p	r	t	p	r	t
10x10	10	0.89	1.00	0.004	0.77	0.97	0.006	0.84	1.00	0.017	0.84	0.99	0.005
	30	1.00	1.00	0.004	0.99	1.00	0.007	0.98	1.00	0.022	0.96	1.00	0.004
3 goals	50	1.00	1.00	0.004	1.00	1.00	0.007	1.00	1.00	0.025	0.99	1.00	0.004
	70	1.00	1.00	0.004	1.00	1.00	0.007	1.00	1.00	0.025	1.00	1.00	0.004
	100	1.00	1.00	0.004	1.00	1.00	0.007	1.00	1.00	0.023	0.99	0.99	0.005
10x10	10	0.65	1.00	0.011	0.60	0.96	0.024	0.65	0.99	0.061	0.59	0.89	0.011
	30	0.84	1.00	0.010	0.81	0.98	0.028	0.80	0.97	0.074	0.73	0.88	0.010
6 goals	50	0.90	1.00	0.011	0.85	0.98	0.030	0.86	0.95	0.084	0.83	0.90	0.010
	70	0.90	0.99	0.010	0.87	1.00	0.031	0.88	0.95	0.088	0.82	0.89	0.010
	100	0.92	1.00	0.011	0.86	1.00	0.032	0.88	0.96	0.090	0.85	0.91	0.011
10x10	10	0.48	1.00	0.017	0.59	0.96	0.043	0.52	0.92	0.103	0.48	0.89	0.018
	30	0.71	0.97	0.020	0.67	0.94	0.051	0.69	0.93	0.121	0.64	0.88	0.020
9 goals	50	0.79	0.96	0.021	0.74	0.96	0.057	0.85	0.94	0.138	0.68	0.80	0.021
	70	0.82	0.97	0.021	0.74	0.87	0.059	0.87	0.97	0.147	0.71	0.82	0.022
	100	0.85	0.96	0.022	0.77	0.90	0.061	0.87	0.95	0.153	0.74	0.82	0.023
20x20	10	0.97	1.00	0.015	0.97	0.99	0.084	0.92	0.97	0.521	0.88	0.98	0.017
	30	1.00	1.00	0.019	1.00	1.00	0.111	1.00	1.00	0.739	0.99	1.00	0.020
3 goals	50	1.00	1.00	0.022	1.00	1.00	0.126	1.00	1.00	0.881	1.00	1.00	0.023
	70	1.00	1.00	0.022	1.00	1.00	0.131	1.00	1.00	0.857	1.00	1.00	0.023
	100	1.00	1.00	0.019	1.00	1.00	0.119	1.00	1.00	0.659	1.00	1.00	0.021
20x20	10	0.78	0.97	0.166	0.74	0.95	0.570	0.69	0.92	1.737	0.70	0.90	0.162
	30	0.87	0.98	0.201	0.88	0.98	0.740	0.87	0.96	2.283	0.81	0.91	0.200
9 goals	50	0.89	0.97	0.229	0.90	0.96	0.877	0.89	0.97	2.767	0.86	0.92	0.239
	70	0.91	0.96	0.242	0.91	0.97	0.941	0.92	0.97	3.012	0.88	0.92	0.261
	100	0.90	0.95	0.250	0.91	0.97	0.982	0.91	0.96	3.216	0.89	0.91	0.281
30x30	10	0.99	1.00	0.096	0.99	1.00	0.598	0.93	1.00	2.620	0.96	1.00	0.108
	30	1.00	1.00	0.147	1.00	1.00	0.887	1.00	1.00	4.080	0.99	1.00	0.150
3 goals	50	1.00	1.00	0.169	1.00	1.00	1.090	1.00	1.00	4.580	1.00	1.00	0.177
	70	1.00	1.00	0.146	1.00	1.00	0.999	1.00	1.00	4.220	1.00	1.00	0.177
	100	1.00	1.00	0.088	1.00	1.00	0.714	1.00	1.00	2.660	1.00	1.00	0.154
30x30	10	0.75	0.97	0.756	0.81	0.96	2.316	0.61	0.87	7.891	0.73	0.88	0.835
	30	0.89	0.97	1.014	0.86	0.93	3.295	0.71	0.88	11.367	0.80	0.85	1.156
9 goals	50	0.91	0.97	1.222	0.89	0.94	4.014	0.73	0.85	14.462	0.83	0.84	1.425
	70	0.92	0.97	1.285	0.89	0.92	4.297	0.74	0.84	16.101	0.87	0.88	1.589
	100	0.90	0.94	1.293	0.89	0.92	4.478	0.75	0.83	17.343	0.88	0.88	1.726

Table 2: GR results for grids showing for each % of sampled actions from observed trace (%O), avg. precision (p), avg. recall (r), and avg. time in seconds (t) to infer a goal based on a given trace fragment.

common observations. It took between 0.004 and 17 seconds to process observation traces, growing slower as a function of the length of the trace. Learning each model took 5 seconds in the smallest grid and up to 60 seconds in the biggest one. Table 3 shows the evolution of performance while learning models from new traces (6 goals in a 10x10 grid), exemplifying the continuous GR proposed in the previous section. Recall improves as the number of traces available for learning increases, but precision drops, due to sharing of sub-optimal traces among different goals. This indicates the existence of a trade-off between recall and precision when irrational agents are allowed and the need of "forgetting" old traces.

Blocksworld. The *Blocksworld* domain has been widely used by model-based GR systems [28], where a generative model of the problem is given in PDDL [15]. In our process mining (PM) framework, we learned models for each goal from 1000 traces generated by a planner [17], where most traces are optimal. Learning each model took on average one second for each of the 20 goals in the domain. We compared PM with existing model-based GR (R&G) [33] upgraded to use the best planner of the agile track from the International Planning Competition 2018 [13], and a version that

	10 traces			100 traces			1000 traces		
%O	p	r	t	p	r	t	p	r	t
10	0.61	0.89	0.009	0.63	0.94	0.052	0.59	0.94	0.290
30	0.80	0.91	0.008	0.83	0.96	0.062	0.79	0.98	0.360
50	0.88	0.91	0.008	0.88	0.97	0.069	0.86	1.00	0.410
70	0.94	0.96	0.008	0.91	0.99	0.071	0.88	1.00	0.420
100	0.95	0.96	0.009	0.90	0.99	0.077	0.86	1.00	0.430

Table 3: GR results for grids over different training sets.

	PM			Landmark		R&G	
%O	p	r	t	r	t	r	t
10	0.08	0.45	0.037	0.89	0.111	0.84	1.656
30	0.28	0.70	0.040	0.97	0.122	0.90	1.735
50	0.40	0.72	0.045	1.00	0.127	0.97	1.836
70	0.70	0.91	0.056	1.00	0.148	0.99	2.056
100	0.89	0.99	0.070	1.00	0.185	1.00	2.378

Table 4: Comparison of GR results using PM and model-based approaches over Blocksworld showing for each % of sampled actions from observed trace (%O), avg. precision (p), avg. recall (r), and avg. time in seconds (t) to infer a goal based on a given trace fragment.

trades off speed for accuracy relying on Landmark heuristics [29]. Performance metrics for the model-based approaches are taken from Pereira et al. [29], which only reports the number of times the true goal belongs to the candidate sets chosen by the algorithms (recall); hence, precision is not reported. The results reported in Table 4 show that precision and recall improve as observation traces become complete. While our approach is faster, the model-based GR has better recall as it has access to the full model of the problem.

6 CONCLUSION

We presented an approach to probabilistic GR based on aligning observations against process models that are automatically extracted using state-of-the-art process mining techniques for process discovery from recorded past behaviors, available in domain logs as sets of event traces. By doing so, GR can be performed in settings in which predefined plan libraries or planning domains may not be easily obtained. Indeed, we provided experimental results on three real-world datasets for which no such knowledge is readily available, but for which logs of traces do exist. We showed how our approach can be used to instantiate a GR framework inspired by the principles of observational learning from social cognitive learning theory. The framework constitutes a collection of components that can be selectively replaced to tune the performance of the system.

There exists work in (statistical) learning that proposes to build GR from previous behavior data. For example, [1] and [30] learn the underlying domain transition model that can then support planning-based GR, [5] learns the decision-making model of the observed agent when executing an HTN-style plan-library (which is known a priori), while [6] and [26] are cases of end-to-end learning from observed behavior to the intended goal. Like our work, their overarching objective is to ease the traditional requirement of having the observed agent model at hand. However, those approaches yield *black-box* type GR systems. In contrast, our approach produces judgments that can be directly interpreted by relying on the structured processes synthesized and identified process misalignments.

We acknowledge a range of limitations of our work so far. We have not tested with a wide range of process discovery and conformance checking techniques developed [9, 38], and we have only considered alignment steps that involve trace actions. We have used fixed parameters, which can be learned over time to yield the best performance. Finally, an interesting direction to pursue in future work is to look at non-stationary environments, i.e., environments that continuously change. In this context, the concepts of "forgetting" traces used to construct skill models and skill model "saturation" should be investigated to yield robust GR experience.

REFERENCES

[1] Leonardo Amado, Ramon Fraga Pereira, João Paulo Aires, Mauricio Cecilio Magnaguagno, Roger Granada, and Felipe Meneguzzi. 2018. Goal Recognition in Latent Space. In *Proc. of IJCNN*. 1–8.

[2] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. 2018. Split miner: automated discovery of accurate and simple business process models from event logs. *KAIS* 59, 2 (may 2018), 251–284. https://doi.org/10.1007/s10115-018-1214-x

[3] Chris L Baker, Rebecca R Saxe, and Joshua B Tenenbaum. 2009. Action understanding as inverse planning. *Cognition* 113, 3 (2009), 329–349.

[4] Albert Bandura. 2008. *Observational Learning*. American Cancer Society. https://doi.org/10.1002/9781405186407.wbieco004

[5] Francis Bisson, Hugo Larochelle, and Froduald Kabanza. 2015. Using a Recursive Neural Network to Learn an Agent's Decision Model for Plan Recognition. In *Proc. of IJCAI*, Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 918–924.

[6] Nate Blaylock and James F. Allen. 2003. Corpus-based, Statistical Goal Recognition. In *Proc. of IJCAI*, Georg Gottlob and Toby Walsh (Eds.). Morgan Kaufmann, 1303–1308.

[7] Diana Borsa, Nicolas Heess, Bilal Piot, Siqi Liu, Leonard Hasenclever, Rémi Munos, and Olivier Pietquin. 2019. Observational Learning by Reinforcement Learning. In *AAMAS*. 1117–1124.

[8] Sandra Carberry. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11, 1-2 (2001), 31–48.

[9] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. 2018. *Conformance Checking—Relating Processes and Models*. Springer International Publishing. https://doi.org/10.1007/978-3-319-99414-7

[10] Eric Demeester, Marnix Nuttin, Dirk Vanhooydonck, and Hendrik Van Brussel. 2003. A model-based, probabilistic framework for plan recognition in shared wheelchair control: Experiments and evaluation. In *ICIRS*, Vol. 2. 1456–1461.

[11] Robert Demolombe and Erwan Hamon. 2002. What does it mean that an agent is performing a typical procedure? A formal definition in the situation calculus. In *IJCAI*. 905–911.

[12] Jonas Firl and Quan Tran. 2011. Probabilistic Maneuver Prediction in Traffic Scenarios. In *ECMR*. 89–94.

[13] Guillem Frances, Hector Geffner, Nir Lipovetzky, and Miquel Ramırez. 2018. Best-first width search in the IPC 2018: Complete, simulated, and polynomial variants. In *IPC–2018–Classical Tracks*. 22–26.

[14] Malik Ghallab, Dana S. Nau, and Paolo Traverso. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers Inc.

[15] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. 2019. *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool.

[16] Jun Hong. 2001. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research (JAIR)* 15 (2001), 1–30.

[17] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. 2018. A Novel Iterative Approach to Top-k Planning. In *Proc. of ICAPS*.

[18] Henry A. Kautz and James F. Allen. 1986. Generalized plan recognition, In Proc. of AAAI. *Proc. of AAAI*, 32–37.

[19] Julian Kooij, Nicolas Schneider, Fabian Flohr, and Dariu Gavrila. 2014. Context-based pedestrian path prediction. In *Proc. of ECCV*. 618–633.

[20] Alexander Kott and William McEneaney. 2006. *Adversarial reasoning: Computational approaches to reading the opponent's mind*. CRC Press.

[21] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal* 1, 1 (2014), 1–14.

[22] Neal Lesh, Charles Rich, and Candace L Sidner. 1999. Using plan recognition in human-computer collaboration. In *Proc. of UM*. 23–32.

[23] Peta Masters and Sebastian Sardina. 2017. Cost-based goal recognition for path-planning. In *Proc. of AAMAS*. 750–758.

[24] Peta Masters and Sebastian Sardina. 2019. Cost-Based Goal Recognition in Navigational Domains. *Journal of Artificial Intelligence Research (JAIR)* 64 (2019), 197–242. https://doi.org/10.1613/jair.1.11343

[25] Peta Masters and Sebastian Sardina. 2019. Goal Recognition for Rational and Irrational Agents. In *Proc. of AAMAS*. 440–448.

[26] Wookhee Min, Eunyoung Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. 2014. Deep Learning-Based Goal Recognition in Open-Ended Digital Games. In *Proc. of the AAAI Conference on AI and Interactive Digital Entertainment (AIIDE)*, Ian Horswill and Arnav Jhala (Eds.). AAAI Press.

[27] David Pattison and Derek Long. 2013. Accurately determining intermediate and terminal plan states using Bayesian goal recognition, In Proc. of AAAI. *Proc. of AAAI*, 32–37.

[28] Ramon Fraga Pereira and Felipe Meneguzzi. 2017. Goal and plan recognition datasets using classical planning domains. (July 2017). https://doi.org/10.5281/zenodo.825878

[29] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. 2017. Landmark-based heuristics for goal recognition. In *AAAI*.

[30] Ramon Fraga Pereira, Mor Vered, Felipe Meneguzzi, and Miquel Ramírez. 2019. Online Probabilistic Goal Recognition over Nominal Models. In *Proc. of IJCAI*. 5547–5553.

[31] Artem Polyvyanyy and Anna Kalenkova. 2019. Monotone Conformance Checking for Partially Matching Designed and Observed Processes. In *Proc. of ICPM*. 81–88.

[32] Miquel Ramirez and Hector Geffner. 2009. Plan recognition as planning, In Proc. of IJCAI. *Proc. of IJCAI*, 1778–1783.

[33] Miquel Ramirez and Hector Geffner. 2010. Probabilistic plan recognition using off-the-shelf classical planners, In Proc. of AAAI. *Proc. of AAAI*, 1121–1126.

[34] Patrice C Roy, Abdenour Bouzouane, Sylvain Giroux, and Bruno Bouchard. 2011. Possibilistic activity recognition in smart homes for cognitively impaired people. *AAI* 25, 10 (2011), 883–926.

[35] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. 2014. *Plan, activity, and intent recognition: Theory and practice*. Newnes.

[36] Gita Sukthankar and Katia Sycara. 2005. A cost minimization approach to human behavior recognition, In Proc. of IJCAI. *Proc. of IJCAI*, 1067–1074.

[37] Timo Sztyler, Josep Carmona, Johanna Völker, and Heiner Stuckenschmidt. 2016. Self-tracking Reloaded: Applying Process Mining to Personalized Health Care from Labeled Sensor Data. *T. Petri Nets and Other Models of Concurrency* 11 (2016), 160–180. https://doi.org/10.1007/978-3-662-53401-4_8

[38] Wil M. P. van der Aalst. 2016. *Process Mining—Data Science in Action, Second Edition*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49851-4

[39] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. 2012. Replaying history on process models for conformance checking and performance analysis. *Data Min. Knowl. Discov.* 2, 2 (jan 2012), 182–192. https://doi.org/10.1002/widm.1045

[40] Wil M. P. van der Aalst, Vladimir A. Rubin, H. M. W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, and Christian W. Günther. 2010. Process mining: A two-step approach to balance between underfitting and overfitting. *SoSyM* 9, 1 (2010), 87–111. https://doi.org/10.1007/s10270-008-0106-z

[41] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. 2004. Workflow Mining: Discovering Process Models from Event Logs. *IEEE TKDE* 16, 9 (sep 2004), 1128–1142. https://doi.org/10.1109/TKDE.2004.47

[42] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling, and Mathias Weske. 2011. Process compliance analysis based on behavioural profiles. *Information Systems* 36, 7 (nov 2011), 1009–1025. https://doi.org/10.1016/j.is.2011.04.002