Explaining Teleo-reactive Strategic Behaviour

Nausheen Saba Shahid
Computer Science Department
Fatima Jinnah Women University
Rawalpindi, Punjab, Pakistan
nausheen_saba@fjwu.edu.pk

Agnieszka Mensfelt
Department of Computer Science
Royal Holloway, Univ. of London
Egham, Surrey, UK
agnieszka.mensfelt@rhul.ac.uk

Kostas Stathis
Department of Computer Science
Royal Holloway, Univ. of London
Egham, Surrey, UK
kostas.stathis@rhul.ac.uk

***Abstract*—Game-theoretic simulations are a powerful tool for exploring strategic interactions and guiding decision-making in fields ranging from business to economics and politics. However, simulations of such complex systems are often difficult to understand intuitively and debug. In this context, particularly challenging and hard to detect are logical (intentional) errors resulting from faulty logic in an agent's strategy. To address this challenge, we develop a framework for question-based explanations, enhancing the understanding of agents' behaviour. We focus on teleo-reactive agents in game-theoretical simulations, specifically in tournament and evolutionary contexts. Our approach centres on trace-based explanations, utilising behavioural logs to identify the steps leading to particular outcomes. We formally describe explanation templates for "why" and "why not" question types, linking them to agents' goals, beliefs, and condition-action rules. Furthermore, we provide formal definitions of the answers to these questions, linking them to output templates. The methodology is demonstrated through example dialogue scenarios, showing how these explanations can improve debugging efficiency by offering high-level insights into agents' behaviours.**

***Index Terms*—Explaining Agent Behaviour, Explanation of teleo-reactive program, Game-theoretic strategies**

I. Introduction

The ability to create complex simulations based on game theory [1] is becoming increasingly important as it enables investigating strategic interactions between diverse actors, anticipating outcomes of these interactions, and making informed decisions in various fields such as economics, politics, and business. These simulations are highly beneficial because they help us create and test policies, identify stable outcomes, and study how strategies change and adapt over time. However, an accurate simulation model of a complex system with multiple actors and diverse strategies comes at the cost of being difficult to understand and debug, which may hinder its trustworthiness.

In game-theoretic simulations, where software agents represent actors, one of the most challenging types of errors is a logical (intentional) error [2], [3] caused by faulty logic in an agent's strategy. These errors may not cause the simulation to terminate, but instead, they can result in incorrect or unexpected agent behaviours, leading to misleading conclusions. Therefore, there is a critical need for debugging support that can explain the reasons behind the agents' behaviour using concepts that are understandable to humans. It is also aligned with research on Explainable Artificial Intelligence (XAI) [4], which emphasizes the importance of providing high-level explanations [5], [6].

In this work, we have developed question-based explanations to facilitate interactive dialogue, which helps in understanding the behaviour of teleo-reactive agents in tournament [7] and evolutionary simulations [8] after the simulation has finished. Specifically, we have focused on trace-based explanations that use behavioural traces or logs to highlight the underlying steps leading to a specific outcome. The concept originates from early work on explanation [9] but has been recently adapted and expanded for symbolic agent models [10]. Similar approaches have been developed in related research for debugging support across various simulation contexts [11]–[13], showing significant improvements in debugging efficiency [14].

Our key contributions in this context are as follows: we formally describe explanation templates to generate an explanation for the concept of "why" and "why not" question types and link them to goals, beliefs, and condition action rules; we also offer a formal definition of the answers to these question types and linking them to output templates; and provide a demonstration of the overall approach using some example dialogue scenarios.

The rest of the paper is structured as follows. Section II reviews the related work, highlighting the key contributions and gaps in the existing literature. In Section III, we introduce teleo-reactive players, detailing their design and implementation. Section IV describes the intentional trace. Section V presents explanation templates, detailing the framework for generating human-understandable explanations of the players' actions. Section VI provides a comprehensive example to illustrate the practical application of the concepts discussed. Finally, Section VII concludes the paper, summarizing the main findings and suggesting directions for future research.

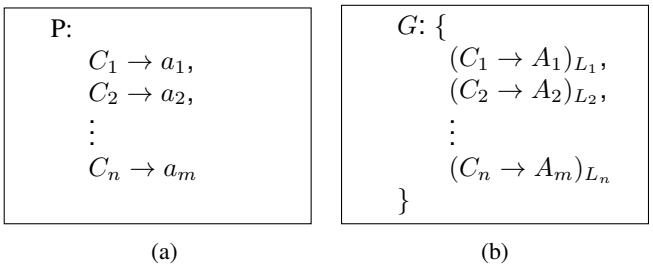
II. Related Work

Lately, there has been considerable interest in explainable AI [15], especially with regard to the inferences made by black-box machine learning models. This type of explainability helps in identifying model biases [16], building trust in a model's inferences [17], and refining both the architecture and future deployment of these models in practical applications [18]. In contrast to this recent effort, we are focusing on explaining the behaviour of BDI-like agent models [19]

using symbolic AI techniques such as logic programs. These AI systems are considered interpretable as we can attribute beliefs, desires, and intentions to their internal behaviour. Our specific focus is on using the internal behaviour specified as a teleo-reactive (TR) model [20] to explain agent strategies that have a natural BDI reading.

We have drawn inspiration from a series of works explaining BDI-like agents. This series begins with Hindriks' work on debugging and explanation for 2APL agents [10], followed by Winikoff's more formal work on program traces and explanation of AgentSpeak agents [12], and most recently, Dennis and Oren's use of dialogue to explain SimpleBDI agents [13]. Like these approaches, we also recognize the need for dialogue to unpack agent behaviour to understand why (or why not) an agent believes a proposition, selects a goal or carries out an action. Additionally, we also assume that this dialogue can be traced back to the agent's beliefs, goals, and rules that determine the agent's behaviour.

In contrast to previous work, we utilize TR programs to construct the decision-making element of the agent. These programs do not require explicit management of agent intentions (as in 2APL [10] and AgentSpeak [12]); only one intention is implicitly selected at a time, based on the ordering of condition-action rules (not randomly when more than one rule is applicable [13] while detecting inconsistencies between traces is beyond the scope of our work). To the best of our knowledge, we are the first to discuss how to explain the teleo-reactive model of execution in an agent setting. Additionally, we are the first to apply these concepts in simulation for game-theoretic experiments, demonstrating an extension of previous work in this field.



(a) (b)

Fig. 1: (a) The typical structure of a Teleo-Reactive (TR) program P; (b) A TR strategy for goal G with labelled condition-action rules.

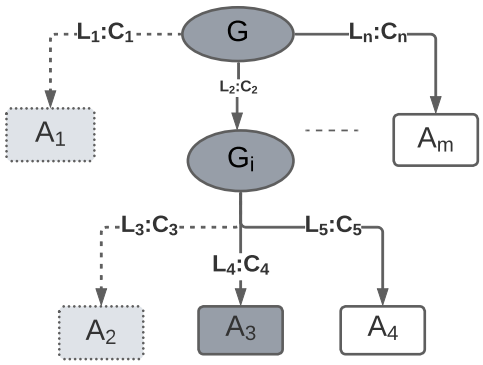


Fig. 2: Nodes and edges represented in dotted lines depict tried and failed conditions of rules. Dark grey nodes represent what is currently pursued, while white nodes represent rules that have not been attempted yet. More than one rule can point to the same action.

III. Teleo-reactive Players

We develop explanations for players who use strategies specified using the Teleo-Reactive (TR) paradigm. This high-level programming model, first introduced in [20] and further explored in [21], [22], offers a robust approach for directing an agent toward a goal (hence *teleo*) while accounting for state changes in the environment (hence *reactive*). In this section, we cover the general structure of TR programs and the agent control that interprets them, and we provide an example of how a simple strategy can be defined using a TR program.

A. Specification of Teleo-Reactive Behaviour

Fig.1a shows a simple teleo-reactive program consisting of an ordered set of *Condition-Action* rules specifying the agent's behaviour. Each of C_i contains conditions that if they hold in the state of the system then trigger a_k, which can be either an action to be carried out by the system in the environment or a call to another TR program. In our framework, we use these kinds of TR programs to represent player strategies, i.e. *teleo-reactive* behaviours directed at achieving a goal.

As shown in Fig.1b, G is the goal that the agent is pursuing and C_i ($i \in \{1...n\}$) and A_j ($j \in \{1...m\}$) are a set of conditions and the action of the condition-action rule, labelled with a unique $L_i \in \{1...n\}$ for easy reference. An action A_j is either an atomic action or a sub-goal G_i of G. Rules that specify how to achieve a sub-goal G_i are defined similarly.

Interpreting a TR program involves trying to achieve the goal G by evaluating the conditions C_1 of the first rule L_1. If they succeed, then the action A_1 is performed (or if $A_1 = G_1$ i.e. it is a sub-goal and evaluated recursively). Otherwise, the next rule L_2 is evaluated in the same manner, and so on. This depth-first evaluation of condition-action rules makes their ordering important; the conditions of the i^{th} rule implicitly imply the negation of some of the conditions in the previous $i-1$ rules [21]. Figure 2 illustrates the interpretation of a behaviour as a tree, displaying a particular path (in grey) that leads to selecting an action (A_3) to achieve goal G, from a set of possible ones ($A_1, ..., A_m$).

One interesting aspect of TR rules is that they can be used to create plan libraries for agents, similar to BDI agent languages like AgentSpeak [19]. In game-theoretic terms, these plan libraries represent a player's strategy. Moreover, the tree of possible paths the agent can select is similar to a BDI agent choosing from a set of intentions. However, in the simplest case, where we interpret the TR rules beginning with the top-level goal, there is no need to manage intentions separately, as it is automatically handled by the execution model.

B. The Player's Knowledge Base

We execute TR rules by assuming an observe-decide-act cycle for the agent, as in [7], that is called before a player has to make a decision according to the TR behaviour. As in [7] and [8], the TR behaviour represents a strategy executed by a TR interpreter denoted as $\vdash_{SIM_{TR}}$. This interpreter is implemented in Prolog and assumes an internal set of components representing the knowledge base KB of an agent player as follows:

$$KB ::= TR\_Beh \cup DKB \cup (TT \cup Percepts) \cup SKB \quad (1)$$

The TR_Beh component contains the teleo-reactive behaviour in terms of condition-action rules as discussed in Section III-A. A behaviour includes a temporal dimension in the sense that it is what makes an agent act at a specific time T. Consequently, the conditions of condition-action rules are parameterised with T and are either *derived beliefs* from rules the agent believes or derived directly from *atomic beliefs*. These atomic beliefs are essentially temporal facts that change over time due to the agent's observations, also known as fluents. In our approach, these fluents are evaluated using the Event Calculus (EC) [23], but an efficient version that caches fluents (CEC) as in [24].

Conditions that are derived from rules parameterised with the time T are described in the DKB, a knowledge base that defines a condition C of a teleo-reactive rule as a derived belief with a rule schema of the form:

$$C \ :- \ A_1, A_2, \ldots, A_k, \backslash+B_1, \ \backslash+B_2, \ldots, \backslash+B_l \quad (2)$$

Here $k, l \geq 0$. Such a rule is read as in Prolog, viz.: C can be derived in the agent's KB, if the conditions $A_1, A_2, \ldots, A_k$ hold in the KB, while conditions $B_1, B_2, \ldots, B_l$ do not (the symbol $\backslash+$ here is read as 'not' and is interpreted as negation by failure [25]). If an $A_i \in \{A_1, \ldots A_k\}$ or $B_j \in \{B_1, \ldots B_l\}$ is also defined in the DKB, then it is defined with the same rule schema. Otherwise, conditions can access fluents using the Event Calculus via queries of the form:

$$holds\_at(Ag : F = V, T).$$

Ag here is the agent's name, F is the name of the fluent, V is its value, and T is the current time. Reasoning about whether a fluent holds is based on the $Percepts$ that the agent has observed in the environment at time T. These percepts are interpreted by a temporal theory TT, where each percept is treated as an event E that happens at time T, which in turn initiates or terminates fluents, according to the Event Calculus ontology shown in Table I.

The agent's internal state also contains a static knowledge base that we call SKB, which is assigned to the agent upon initialization; it is a logic program that contains immutable knowledge and can be directly executed in Prolog (whose inference is denoted as $\vdash_{SLDNF}$). The SKB is accessible from all other components and is provided as support functionality. The rules of SKB have the same structure as equation 2, but do not contain time parameters as the DKB, belong to the ontology of the Event Calculus as the $Percepts$ and temporal reasoning TT, or contain TR rules as TR_Beh.

TABLE I: Ontology of the Event Calculus used.

Predicate	Description
holds_at(F=V, T)	Fluent F has value V at time T.
holds_for(F=V, [Ts,Te])	Fluent F continuously has value V from time Ts to time Te.
broken(F=V, [Ts,Te])	Fluent F has changed value V from time Ts to time Te.
happens_at(E, T)	Event E happens at time T.
initiates_at(E, F=V, T)	Event E initiates value V for fluent F at time T.
terminates_at(E, F=V, T)	Event E terminates value V for fluent F at time T.

IV. Tracing an Intention

To explain actions in our framework, we need to inspect intentions that achieve goals (and their sub-goals). The explanation occurs in the following context: a simulation has been completed, and as a result of experimental analysis (e.g. from simulation data) or testing the system (e.g. via unit testing), we have found that we need to inspect the behaviour of a set of player strategies to understand why certain actions have been taken. For this purpose, we need to analyse the behaviours of specific agents on concrete simulation logs. This involves questioning actions that agents have taken, by examining their intentions at specific times.

As discussed earlier, an intention is a path that eventually enables the selection of an action from a set of possible ones. Given that each agent is assigned a top-level goal G to pursue, this goal will be a parameterised term in the general case, so when the goal is called, the call must contain a substitution θ of goal's variables with specific values, giving rise to a specific goal instance $(G)\theta$. Applying θ on G has the effect of propagating the instantiated variables of θ to conditions-action rules that share the same variables contained in θ, either in the conditions, actions (whether atomic or sub-goals), or both. In addition, after execution of a behaviour, additional variables that do not belong to θ may have been intermediately computed and used as input parameters to the action selected, giving rise to a new substitution σ for these intermediate variables. So to ask for an explanation of why a particular action A has been executed, in general, what we are asking is why a specific intention has led to a specific instance of the action $(A)\theta \circ \sigma$.

To explore intentions from the logs of an agent, we define a meta-interpreter [26] that reconstructs the intention of agent Ag by tracing the agent's top-level goal $(G)\theta$ at time $T \in \theta$ to bring about the action $(A)\theta \circ \sigma$ that we want an explanation for. We, therefore, define:

$$trace(Ag^T_G, Trace^T),$$

where Ag is a unique agent identifier, T holds a time stamp, and G is the agent's top-level goal that we want to trace. When Ag^T_G is provided as input to $trace/2$ we use $Trace^T$ to represent the trace of Ag with top-level goal G at a time T, resulting from the active intention that Ag has used to

$$Trace_i^T = \begin{cases} Trace_{i-1}^T & \text{if } Trace_{i-1}^T = [G : (Cs \rightarrow A)_L | Rest] \wedge \text{action}(A) \\ [G : (Cs \rightarrow A)_L | Trace_{i-1}^T] & \text{if } G : (Cs \rightarrow A)_L \in \text{TR\_Beh} \wedge \text{KB} \vdash_{SIM_{TR}} Cs \end{cases} \quad (3)$$

select the action we want to be explained. $Trace^T$ is built incrementally in n recursive calls ($n \geq 1$) using the meta-interpreter. This recursive calls produce a series of the form $Trace_1^T, Trace_2^T, \cdots Trace_n^T$, where each step i ($i \geq 1$) is defined by $Trace_i^T$ as shown in Equation 3.

Note that we use a list-based syntax as we believe this is easier to imagine the implementation in Prolog. Initially (i.e. step $i = 1$), $Trace_0^T$ is empty (represented by the empty list []). Then at each interpretation step i, a rule of the form $G : (Cs \rightarrow A)_L$ is checked and, if this rule belongs to the agent's teleo-reactive behaviour TR_Beh and the conditions Cs of the rule holds in the state of the agent $KB \vdash_{SIM_{TR}} Cs$, then the rule is added in the trace. Now the $Trace_i^T$ consists of the trace of the previous step $Trace_{i-1}^T$ and also the contents of the rule $G : (Cs \rightarrow A)_L$. Once this is done, the meta-interpreter interprets A for further processing and if A is a sub-behaviour then the same process applies recursively. This process of interpretation finishes when a rule leads to an action A so $Trace_i^T$ is returned as the $Trace_{i-1}^T$. At the end of the complete interpretation, the final trace $Trace_n^T$ ($= Trace^T$) is a list with a complex structure, consisting of a sequence of successfully interpreted rules. We give next an example of a trace to provide the reader with a concrete instance of its internal structure reversed:

$$\begin{aligned} Trace^T = [& \\ & G : (Cs_i \rightarrow G_i), \\ & G_i : (Cs_{i2} \rightarrow G_{i2}) \\ & G_{i2} : (Cs_{i2\ldots l} \rightarrow G_{i2\ldots l}) \\ & \ldots, \\ & G_{i2\ldots l} : (Cs_{i2\ldots lp} \rightarrow A) \\ ] & \ \theta \circ \sigma_1 \circ \sigma_2 \circ \ldots \sigma_n \end{aligned}$$

G is the top-level goal, G_i is the i^{th} sub-goal of G, G_{i2} is the 2^{nd} sub-goal of G_i and so on. As before, the top-level goal may be parameterised with its own instantiation of variables θ (possibly shared with the rest of the rules). As the recursive calls proceed, they generate substitutions σ_1 in the first iteration, σ_2 in the second, and so on, with σ_n in the n^{th}. These substitutions are composed to a single one $\sigma = \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_n$, which eventually applies to the returned action as $(A)\theta \circ \sigma$. This action has been decided by the agent at time T, selected via the application of a series of condition-action rules: the first rule labelled i selects sub-goal G_i, the next rule labelled $i2$ selects sub-goal G_{i2}, and this process continues until the rule labelled $i2...lp$. These numbers indicate that from the behaviour indexed by G the i^{th} rule succeeds, from the sub-behaviour G_i the 2^{nd} rule succeeds, etc.

V. Explanation Templates

In the explanation process, first, the relevant logs, expressed as the history of events and knowledge of an agent, are loaded from relevant archival files (e.g. similar to [8]) into the memory, and then different questions can be asked.

In our framework, based on the core cognitive concepts such as goals, actions, and beliefs available in the agent's knowledge, the following types of questions can be asked for explanation purposes.

- Why did agent Ag perform an action A at time T?
- Why did agent Ag believe B at time T?
- Why did agent Ag select goal G at time T?
- Why did agent Ag not perform action A at time T?
- Why did agent Ag not believe B at time T?
- Why did agent Ag not select goal G at time T?

Table II presents *explanation templates* for explaining each question type in terms of the knowledge components to be included in the explanation generation where the values of the variables can be instantiated through the query. While our framework supports inquiries regarding static knowledge and all possible scenarios related to actions, goals, and beliefs, we have provided explanation templates for only a subset of questions due to space constraints. The generation of explanations for the remaining scenarios can be achieved using a similar methodology.

To explain the rationale behind an action, a backward process is adopted using the beliefs and goals, directly above the action in the tree hierarchy. Users can ask follow-up questions about the contents (i.e. beliefs, goals) of the provided explanation, leading to new explanations for each question asked. This allows for concise explanations at each step, and users can keep asking questions to gain a deeper understanding, moving towards the top-level node of the goal hierarchy tree. The diagram in Figure 3 illustrates the flow of this dialogue, typically beginning with 'why an act' or 'why not an act' questions and leading to further questions.

For example, it can be asked why an agent has performed an action at a simulation time point T (provided as the reference point). We assume that $act(A)$ includes the agent's ID and the fully instantiated action that the agent executed. To provide an explanation for this inquiry, we need to first check the event history to verify whether the agent indeed performed this action. If the action was not carried out, the explanation should state that the question is invalid.

If the action has been executed, the trace $Trace^T$ would be constructed using the internal state of the agent. This trace is built using a meta-interpreter similar to the one employed for decision-making during the simulation. The components of this explanation template consist of the immediate goal G (the lowest goal in the goal hierarchy tree) that the agent was pursuing at time T, and the (triggering) conditions $Conds$ under which the action $(A)\theta$ was selected. These components are presented in the template as shown in Table II.

Question	Preconditions	Explanation
why_act(G, A, T)θ	$\texttt{Trace}^T$ $\wedge$ $((\texttt{G} : (\texttt{Conds} \rightarrow \texttt{A})_L)\theta \circ \sigma) \in \texttt{Trace}^T$ $\wedge$ action(A)	**While pursuing goal** $(G)\theta$, **I believe that if (Conds)**θ **hold, then I do** $(A)\theta$, **and at time** T **the** (Conds)$\theta \circ \sigma$ **hold,** **so I did** $(A)\theta$.
whynot_act(G, A',T)θ	$\texttt{Trace}^T$ $\wedge$ $(\texttt{G} : (\texttt{Conds} \rightarrow \texttt{H})_{L_1})\theta \circ \sigma \in \texttt{Trace}^T$ $\wedge$ $\texttt{H} \neq \texttt{A}'$ $\wedge$ action(A') $\wedge$ $(\texttt{G} : (\texttt{Conds}' \rightarrow \texttt{A}')_{L_2})\theta \in$ TR_Beh $\wedge$ $\texttt{Conds}' = (\texttt{Succ} \cup \texttt{C}_\texttt{f} \cup \texttt{Unexp})$ $\wedge$ $\texttt{L}_2 < \texttt{L}_1$ $\wedge$ KB $\vdash_{SIM_{TR}}$ (Succ)$\theta \circ \sigma$ $\wedge$ KB $\nvdash_{SIM_{TR}}$ $(\texttt{C}_\texttt{f})\theta \circ \sigma$	**If** $(Succ \cup C_f \cup Unexp)\theta$ **hold I do** $(A')\ \theta$, **but**$(C_f)\theta \circ \sigma$ **failed,** **so I did not do** $(A')\theta$
”	$\texttt{Trace}^T$ $\wedge$ $(\texttt{G} : (\texttt{Conds} \rightarrow \texttt{H})_{L_1})\theta \circ \sigma \in \texttt{Trace}^T$ $\wedge$ $\texttt{H} \neq \texttt{A}'$ $\wedge$ action(A') $\wedge$ $(\texttt{G} : (\texttt{Conds}' \rightarrow \texttt{A}')_{L_2})\theta \in$ TR_Beh $\wedge$ $\texttt{L}_1 < \texttt{L}_2$	**While pursuing goal** $(G)\theta$, **I believe that if** $(Conds)\theta$ **hold,** **I do** $(H)\ \theta$**, and** $(Conds)\theta \circ \sigma$**succeeded,** **so I selected** $(H)\theta$ **instead of** $(A')\theta$
why_goal$(G_i, G_{ij}, T)\theta$	$\texttt{Trace}^T$ $\wedge$ $((\texttt{G}_\texttt{i} : (\texttt{Conds} \rightarrow \texttt{G}_\texttt{ij})_L)\theta \circ \sigma) \in \texttt{Trace}^T$	**While pursuing goal** $(G_i)\theta$ **, I believe thatif** $(Conds)\theta$ **hold then I select** $G_{ij}\theta$ **and at time** T **the** $(Conds)\theta \circ \sigma$ **hold,** **so I selected** $(G_{ij})\theta \circ \sigma$
whynot_goal$(G_i, goal(G'_{ij}), T)\theta$	$\texttt{Trace}^T$ $\wedge$ $(\texttt{G}_\texttt{i} : (\texttt{Conds} \rightarrow \texttt{H})_{L_1})\theta \circ \sigma \in \texttt{Trace}^T$ $\wedge$ $\texttt{H} \neq \texttt{G}_\texttt{ij}'$ $\wedge$ $(\texttt{G}_\texttt{i} : (\texttt{Conds}' \rightarrow \texttt{G}_\texttt{ij}')_{L_2})\theta \in$ TR_Beh $\wedge$ $\texttt{Conds}' = (\texttt{Succ} \cup \texttt{C}_\texttt{f} \cup \texttt{Unexp})$ $\wedge$ $\texttt{L}_2 < \texttt{L}_1$ $\wedge$ KB $\vdash_{SIM_{TR}}$ (Succ)$\theta \circ \sigma \wedge$ KB $\nvdash_{SIM_{TR}}$ $(\texttt{C}_\texttt{f})\theta \circ \sigma$	**If** $(Succ \cup C_f \cup Unexp)\theta$ **hold I select** $(G'_{ij})\ \theta$, **but** $(C_f)\theta \circ \sigma$ **failed, so I did not select** $(G'_{ij})\theta$
”	$\texttt{Trace}^T$ $\wedge$ $(\texttt{G}_\texttt{i} : (\texttt{Conds} \rightarrow \texttt{H})_{L_1})\theta \circ \sigma \in \texttt{Trace}^T$ $\wedge$ $\texttt{H} \neq \texttt{G}_\texttt{ij}'$ $\wedge$ $(\texttt{G}_\texttt{i} : (\texttt{Conds}' \rightarrow \texttt{G}_\texttt{ij}')_{L_2})\theta \in$ TR_Beh $\wedge$ $\texttt{L}_1 < \texttt{L}_2$	**While pursuing goal** $(G_i)\theta$, **I believe that if** $(Conds)\theta$ **hold I do** $(H)\ \theta$, **and** $(Conds)\theta \circ \sigma$ **succeeded,** **so I selected** $(H)\theta$ **instead of** $(G'_{ij})\theta$
why_believe$(Ag : F = V, T)$	holds_for(Ag : F=V, [$\texttt{T}_\texttt{min}$, $\texttt{T}_\texttt{max}$]) $\in$ DKB $\wedge$ $\texttt{T}_\texttt{min} < \texttt{T} \leq \texttt{T}_\texttt{max}$ $\wedge$ happens_at(Event, $\texttt{T}_\texttt{min}$) $\in$ Percepts $\wedge$ (initiates_at(Event, Ag:F=V, $\texttt{T}_\texttt{min}$) $\leftarrow$ Body) $\in$ TT $\wedge$ KB $\vdash_{SIM_{TR}}$ (Body)	**I observed** E **that** $initiated$ $F = V$ **at** T_{min} **and until** T_{max} **nothing happened to** $terminate$ **it**
why_believe$(TConc, T)\theta$	(TConc $\leftarrow$ TConds)θ $\in$ DKB $\wedge$KB $\vdash_{SIM_{TR}}$ (TConds)$\theta \circ \sigma$	**I believe that if** $(TConds)\theta$ **then** $(TConc)\theta$ **also holds, and** $(TConds)\theta \circ \sigma$ **hold, so I concluded** $(TConc)\theta \circ \sigma$
whynot_believe$(Ag : F = V, T)$	X = {I $\mid$ holds_for(Ag:F=V, I) $\in$ DKB} $\wedge$ last(X) = [$\texttt{T}_\texttt{s}$, $\texttt{T}_\texttt{e}$] $\wedge$ $\texttt{T}_\texttt{e} < \texttt{T}$ $\wedge$ happens_at(E, Te) $\in$ Percepts $\wedge$ (terminates_at(E, Ag:F=V, $\texttt{T}_\texttt{e}$) $\leftarrow$ Conds) $\in$ TT $\wedge$ KB $\vdash_{\text{SIM}_{TR}}$ Conds	**I stopped believing** $F = V$ **because the happening of** E **at** T_e **terminated it**.
whynot_believe$(TConc, T)\theta$	Expl = {**I believe** (TConc **if** TConds)θ **but the condition** $(\texttt{C}_\texttt{f})\theta \circ \sigma$ **doesn't hold** $\mid$ (TConc $\leftarrow$ TConds)θ $\in$ DKB $\wedge$ TConds = (Succ $\cup$ $\texttt{C}_\texttt{f}$ $\cup$ Unexp)θ $\wedge$ KB $\vdash_{SIM_{TR}}$ (Succ)$\theta \circ \sigma$ $\wedge$ KB $\nvdash_{SIM_{TR}}$ $(\texttt{C}_\texttt{f})\theta \circ \sigma$ }	**Because** $Expl$

TABLE II: A subset of possible explanations. Here, the $Question$ column represents the question that the experimenter can ask, the $Preconditions$ column represents the context of the agent's internal state related to the question and how the explanation will be generated, and $Explanation$ represents the natural language text that will be generated as an explanation.

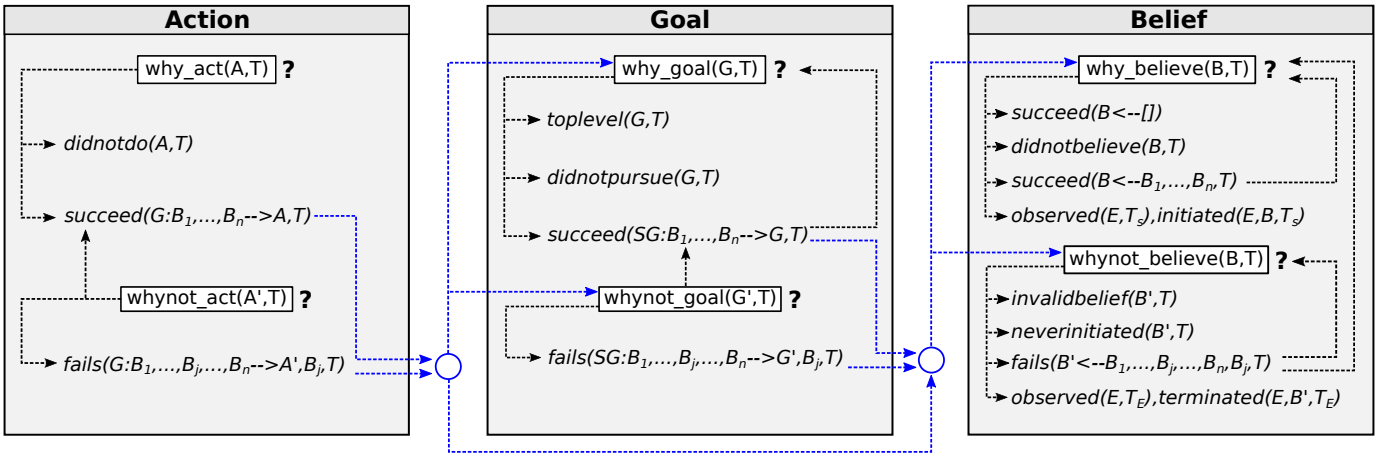


Fig. 3: Dialogue boxes supporting inquiries about an $Action$, a $Goal$, and a $Belief$ using our framework. Black arrows indicate dialogue flows within a box, while blue arrows represent flows between boxes. The process typically begins by asking why an action was carried out ($Action$ box), leading to explanations for the action's success or that the agent did not perform it. If the user expected a different action to succeed, they might ask why an action was not chosen, resulting in explanations of why that action failed or why another action succeeded. While inquiring about an action, users might also query a goal ($Goal$ box) or a belief ($Belief$ box). When asking why a goal was selected, explanations may state that it is the top-level goal, or that a rule succeeded that makes it a sub-goal of a higher-level goal, or that the goal was never pursued. If the user expected a different goal, they can ask why it was not selected, receiving explanations of failure or why another goal succeeded instead. Similarly, users can inquire about beliefs, with explanations detailing why a belief holds as a fact, a derived belief, or because of an observation that initiated it and still holds. Conversely, inquiries about why a belief does not hold might reveal it was never initiated, cannot be derived, or was terminated by an event.

We assume that the user will fully instantiate the action A to be explained by specifying all the free variables of the action in the substitution θ. The conditions $Conds$ of the condition-action rule L in the behaviour for G may have further (existentially quantified) variables instantiated at the time T, which are part of the substitution σ. We are now in the position to instantiate all components of the template.

Given a trace $Trace^T$, an experimenter can also ask why an action has not been selected at time T for a specific goal G. We assume that this question can be asked about an action of any goal G available in $Trace^T$. First, we check whether $G : (Conds \rightarrow H)_{L_1}$ is a member. If it is, we should check whether the action A' is in the head of a condition action rule L_2 within the behaviour indexed by the goal G that comes before L_1. If there is such a rule, then the conditions of that rule must have failed. For explanation purposes, we distinguish between the conditions of the rule that succeeded (denoted by $Succ$), the condition that failed (denoted by C_f), and the unexplored conditions (denoted by $Unexp$). Then, with some abuse of notation, we can write the rule as $(G : ((Succ \cup C_f \cup Unexp) \rightarrow A')_{L_2})\theta$. Another case is when the conditions of the rule $G : (Conds' \rightarrow A')_{L_2}$ were never checked because an earlier rule $G : (Conds \rightarrow A)_{L_1}$ was triggered. As in this case, an earlier rule was selected because its conditions $Conds$ hold. This is the case where the rule label L_1 of the triggering rule is less than the rule label L_2 of the un-evaluated rule.

An agent can be asked about why it has selected a goal. Similar to the *why action* template, if a rule of the form $G_i : (Conds \rightarrow G_{ij})_L$ is available in $Trace^T$ that led (contributed) to the agent selecting an action, then we provide the conditions of that rule in explanation.

Given a trace $Trace^T$, we should also be in a position to ask why an agent did not select an alternative sub-goal $(G'_{ij})\theta$ of G_i (instead of the current selection H) at time T. There are two cases: earlier sub-goal – when the rule of the sub-goal is positioned in the behaviour before the rule of the current selection; later sub-goal – when the rule of the sub-goal is positioned after the rule of the current selection in the behaviour. Explanations for both cases are generated accordingly.

An agent can also be asked why it believes something at a time point T. To answer that, it will be checked from the internal state of the agent whether the agent believes that knowledge at time T or not. If so, then it will be checked whether it is a fluent belief or a derived belief of an agent. In the case of an atomic belief, it is held in multi-valued fluents. In this case, first, holds_for/2 is used to retrieve the duration for which the fluent holds i.e. start and end time of that belief ($holds\_for(Ag : F = V, [T_{min}, T_{mx}])$) from DKB. If the start time (T_{min}) is not zero and the end time (T_{max}) is greater than the query's time point (T), then it is implied that this belief has been initiated by an event and nothing happened to change this belief until the (T). So the relevant event $happens\_at(Event, T_{min})$ is retrieved from the $Percepts$ and the temporal theory TT is searched for a relevant temporal rule of the form:

$$initiates\_at(Event, Ag : F = V, T_{min}) \ :- \ Body$$

to initiate this belief (in Table II the symbol ':-' is substituted by '$\leftarrow$'). $Body$ is derived from KB through SIM_{TR}. For Prolog rules denoted in the table as $Conc \leftarrow Conds$ in the agent's SKB, the SKB must be able to derive the $Conds$ through $\vdash_{SLDNF}$.

Similarly, if we wish to ask an agent why it did not believe something at time T. To explain an atomic belief, we would need to search for the latest holding interval of the fluent belief, and then from the Percepts, search for the event $happens\_at(Event, T_e)$ which terminated this belief $terminates\_at(Event, Ag : F = V, T_e)$ as specified in TT. From the agent's knowledge base KB we can derive the $Conds$. Then generate an explanation as shown in Table II. last$(X) = [t_s, t_e]$ is a function that takes a list of lists (of intervals) X and returns the last element as a term of the form $[t_s, t_e]$.

We can ask questions for inferred/derived beliefs in a similar manner. For explanations of such beliefs, we first search for a rule with conditions on which the derived belief is based. Then those conditions are checked whether they hold at time T. All the rules defined to infer derived belief should be checked and, for each rule, the failed condition C_f from the rule's conditions should be used as the explanation of failure. It is up to the implementation to present the explanation $Expl$ above. This can be done all at once, by providing all the rules defining $TConc$ and listing for each definition, the condition that fails. Alternatively, the rules can be presented incrementally, by producing one rule at a time and the reason it has failed, together with an additional option to ask for more solutions.

VI. Example

To demonstrate the working of our explanation framework, we consider an example scenario with two players; player1 and player2 who partake in a game-theoretic simulation. player1 plays the 'All_D' strategy and, as a result, defects with everyone. On the other hand, player2 plays 'Tit_For_Tat' and, consequently, cooperates with a stranger or anyone who has cooperated in the last move, or defects with anyone who has defected in the last move. Assume further that player1's strategy has been implemented correctly but player2's has not. In other words, the 'Tit_For_Tat' strategy contains an implementation-level error. A teleo-reactive behaviour for player2 is defined by the set of condition-action rules shown below.

player2 :: {
  tft(E, O, T):{
    (last_act(O)=defect:T $\rightarrow$ *defect)*$_1$*;*
    (last_act(O)=coop:T $\rightarrow$ *coop)*$_2$*;*
    (true $\rightarrow$ *coop)*$_3$
  }
}

We assume that the experimenter has implemented the scenario incorrectly and there is no rule defined to initialize last_act(O)=coop or last_act(O)=defect. We also assume that during the simulation, the experimenter observes that player1 and player2 have interacted with each other three times. Table III shows their precise action history, summarising the content of a log file.

TABLE III: Action History of Example

In encounter1 (time points 1-4), player1 defected with player2
at time step 2 and player 2 cooperated with player1 at time step 3.
In encounter2 (time points 5-8), player1 defected with player2
at time step 6 and player 2 cooperated with player1 at time step 7.
In encounter3 (time points 9-12), player1 defected with player2
at time step 10 and player 2 cooperated with player1 at time step 11.

According to 'Tit_for_Tat' we would expect player2 to defect in the second encounter. To investigate why player2 cooperated instead, we use the dialogue in Table IV. The dialogue assumes a simple natural language menu system that uses fixed phrases such as 'why did...' to initiate a dialogue. The system then uses grammar rules to guide the user in constructing what is valid to come next as a set of alternatives, step by step, i.e.: '...<player ><act> at time <Time>, while pursuing <goal>?'. The final statement, constructed in this way, has acquired all the specific parameters from the user, e.g.:

> 'why did player2 cooperate at time 7, while pursuing the top goal?'

and can be translated to a term that maps naturally to one of our framework's queries e.g.:

?- why_act(player2,tft(enc1, player1, 7), coop, Expl).

TABLE IV: Explanation dialogue where lines starting with '**U**:' show the question posed by the user via the menu interface, lines starting with '**S**:' represent the answer of the system using our trace/2 interpreter, and lines starting with the character '%' followed by a 'Q:' represent commented queries constructed by the menu interface.

U: Why did player2 cooperate at time 7 while pursuing topG?
% Q: why_act(player2,tft(enc1, player1, 7), coop, Expl).
S: Expl = While pursuing the tit-for-tat I do cooperate with player1 unconditionally.
U: Why did player2 not defect at time 7 while pursuing topG?
% Q: whynot_act(player2,tft(encounter1, player1, 7), defect,Expl).
S: Expl = While pursuing tit-for-tat(enc1, player1,7) strategy, if last_act(player1)=defect holds I defect, but last_act(player1)=defect failed, so I did not defect.
U: Why did player2 not believe that player1's last act is defect at 7?
% Q: whynot_believe(player2,last _act(player1,7)=defect,Expl).
S: Expl = I observed nothing that can make me believe that.

Using the dialogue of Table IV, the experimenter realizes there is a problem with the initiation of the belief last_act=defect in the implementation of 'Tit_For_Tat' strategy. This requires further interrogation with the system to

check the Event Calculus definitions written for belief initiation and amend them to support this scenario. This possible solution to discovering the error will allow the experimenter to revise the rules and create a more robust strategy. Due to space constraints, the detailed steps of this process are omitted.

VII. Conclusions

We have developed a framework for question-based explanations, with the aim of enhancing our understanding of an agent's strategic behaviour in game-theoretic simulations. We have been particularly concerned with agent strategies that are expressed using teleo-reactive programs that link goals and actions with condition-action rules on the belief base of the agent. Our approach centres on trace-based explanations, utilising behavioural logs to identify which series of condition action rules have succeeded in producing a particular outcome. In this context, we have formally described what we refer to as explanation templates for "why" and "why not" question types, corresponding to why a trace was successful or why alternative traces were not selected. These explanation templates aim at linking formal definitions of the answers to these questions, expressed in the template as natural language text that informs the user about the agent's behaviour at a specific simulation time-point. The methodology is demonstrated through example dialogue scenarios, showing how these explanations can improve debugging efficiency by offering high-level insights into agents' behaviours.

An open question in our approach is how to introduce more variety in template-based dialogue. The menu-based constructed questions and template answers, while allowing for different content instantiations, always maintain the same structure, which can appear repetitive to experimenters. A straightforward solution is to interface the input and output with templates to a language model, making the interaction more flexible. A more challenging approach would involve providing the language model with both our representation and the log of interest as input (as done in [27], [28]) and having it generate the explanations (similar to [29]).

Acknowledgment

We acknowledge support from the Higher Education Commission (HEC), Pakistan, and the Leverhulme Trust International Professorship Grant (LIP-2022-001).

References

[1] A. Rapoport, *Game theory as a theory of conflict resolution*. Springer Science & Business Media, 2012, vol. 2.

[2] R. Smith and S. Rixner, "The error landscape: Characterizing the mistakes of novice programmers," in *Procs. of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 538–544.

[3] G. Stergiopoulos, P. Katsaros, and D. Gritzalis, "Automated detection of logical errors in programs," in *Risks and Security of Internet and Systems*, J. Lopez, I. Ray, and B. Crispo, Eds. Springer International Publishing, 2015, pp. 35–51.

[4] M. Van Lent, W. Fisher, and M. Mancuso, "An explainable artificial intelligence system for small-unit tactical behavior," in *Proceedings of AAAI'04*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004, pp. 900–907.

[5] J. Broekens, M. Harbers, K. Hindriks, K. Van Den Bosch, C. Jonker, and J.-J. Meyer, "Do you get it? user-evaluated explainable BDI agents," in *MATES 2010, Germany, Sep. 27-29*. Springer, 2010, pp. 28–39.

[6] M. Harbers, K. van den Bosch, and J.-J. Meyer, "Design and evaluation of explainable BDI agents," in *Procs. of Web Intelligence and Intelligent Agent Technology*, vol. 2. IEEE, 2010, pp. 125–132.

[7] N. S. Shahid, D. O'Keeffe, and K. Stathis, "Game-theoretic simulations with cognitive agents," in *33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2021, pp. 1300–1305.

[8] N. S. Shahid, D. O'Keeffe, and K. Stathis, "A knowledge representation framework for evolutionary simulations with cognitive agents," in *35th International Conference on Tools with Artificial Intelligence, (ICTAI), Atlanta, GA, USA, November 6-8, 2023*. IEEE, 2023, pp. 361–368.

[9] "The epistemology of a rule-based expert system – a framework for explanation," *Artificial Intelligence*, vol. 20, no. 3, pp. 215–251, 1983.

[10] K. V. Hindriks, "Debugging is explaining," in *Principles and Practice of Multi-Agent Systems (PRIMA)*. Berlin: Springer, 2012, pp. 31–45.

[11] S. Chari, D. M. Gruen, O. Seneviratne, and D. L. McGuinness, "Directions for explainable knowledge-enabled systems," in *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*. IOS Press, 2020, vol. 47, pp. 245–261.

[12] M. Winikoff, "Debugging agent programs with" why?" questions," 2017.

[13] L. A. Dennis and N. Oren, "Explaining BDI agent behaviour through dialogue," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 2, p. 29, 2022.

[14] A. J. Ko and B. A. Myers, "Finding causes of program output with the Java Whyline," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1569–1578.

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

[16] B. Van Stein, D. Vermetten, F. Caraffini, and A. V. Kononova, "Deep bias: Detecting structural bias using explainable AI," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. NY, USA: ACM, 2023, p. 455–458.

[17] "The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable ai," *International Journal of Human-Computer Studies*, vol. 146, p. 102551, 2021.

[18] V. Belle and I. Papantonis, "Principles and practice of explainable machine learning," *Frontiers in Big Data*, vol. 4, 2021.

[19] A. S. Rao, M. P. Georgeff *et al.*, "BDI agents: from theory to practice." in *ICMAS*, vol. 95, 1995, pp. 312–319.

[20] N. Nilsson, "Teleo-reactive programs for agent control," *Journal of artificial intelligence research*, vol. 1, pp. 139–158, 1993.

[21] R. A. Kowalski and F. Sadri, "Teleo-reactive abductive logic programs," in *Logic Programs, Norms and Action*. Springer, 2012, pp. 12–32.

[22] K. Clark, "Rule control of teleo-reactive, multi-tasking, communicating robotic agents," in *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2018 - Volume 1, Porto, Portugal, July 29-31, 2018*, K. Madani and O. Gusikhin, Eds. SciTePress, 2018, pp. 5–15.

[23] M. Sergot and R. Kowalski, "A logic-based calculus of events," *New Generation Computing*, vol. 4, no. 1, pp. 67–95, 1986.

[24] O. Kafali, A. Romero, and K. Stathis, "Agent-oriented activity recognition in the event calculus: An application for diabetic patients," *Computational Intelligence*, vol. 33, no. 4, pp. 899–925, 2017.

[25] K. L. Clark, "Negation as failure," in *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. New York: Plemum Press, 1977, pp. 293–322.

[26] A. Brogi and F. Turini, "Metalogic for knowledge representation," in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991*, J. F. Allen, R. Fikes, and E. Sandewall, Eds. Morgan Kaufmann, 1991, pp. 61–69.

[27] A. Mensfelt, K. Stathis, and V. Tencsenyi, "Autoformalization of Game Descriptions using Large Language Models," in *1st International Workshop on Next-Generation Language Models for Knowledge Representation and Reasoning*, Hanoi, Vietnam, 2024.

[28] A. Mensfelt, K. Stathis, and V. Trencsenyi, "Logic-enhanced language model agents for trustworthy social simulations," 2024. [Online]. Available: https://arxiv.org/abs/2408.16081

[29] P. Bagga and K. Stathis, "Towards Explainable Strategy Templates using NLP Transformers," in *Workshop on Explainable AI in Finance (XAINF23)*. New York, USA: ACM, Nov 27, 2023.