SNAPE: A Sequential Non-Stationary Decision Process Model for Adaptive Explanation Generation

Amelie Sophie Robrecht[a] and Stefan Kopp[b]
Social Cognitive Systems Group, Faculty of Technology, Bielefeld University, Bielefeld, Germany

Keywords: Explainable AI, Explanation Planning, Non-Stationary Hierarchichal Decision Processes.

Abstract: The automatic generation of explanations is an increasingly important problem in the field of Explainable AI (XAI). However, while most work looks at how complete and correct information can be extracted or how it can be presented, the success of an explanation also depends on the person the explanation is targeted at. We present an adaptive explainer model that constructs and employs a partner model to tailor explanations during the course of the interaction. The model incorporates different linguistic levels of human-like explanations in a hierarchical, sequential decision process within a non-stationary environment. The model is based on online planning (using Monte Carlo Tree Search) to solve a continuously adapted MDP for explanation action and explanation move selection. We present the model as well as first results from explanation interactions with different kinds of simulated users.

1 INTRODUCTION

Current AI systems are increasingly required to be explainable, i.e. to be understandable by stakeholders when provided with some kind of additional explanation. Such explanations can range from visualizations of internal parameters, to dedicated analysis techniques to identify a classifier's sensitivities, to textual information on natural language requests. Which kind of explanation is suited best depends on different aspects including the to-be-explained object as well as the recipient of the explanation (Gunning et al., 2021). A recent trend in XAI is thus to consider explanations as an adaptive and interactive process (Miller, 2019). Ultimately, this requires a technical system and an end user to clarify the user's information needs and to tailor the explanation in an interactively co-constructed process (Rohlfing et al., 2021). Yet, it is not clear how such a process can be established and supported by an intelligent explanation system.

We address this problem for the case of natural language explanations that are constructed in a dialogical interaction between a human addressee (the explainee, henceforth) and an A.I. system. That is, we aim to support a natural language interaction that is co-constructed by an intelligent agent and a user, resulting in an effective and successfully personalized explanation. We focus specifically on the problem of generating explanations in a way that is adaptive to the explainee's understanding and information needs, as they become apparent through the course of a dialogical interaction. This requires an agent to autonomously decide upon the explanation's content (what is to be explained next), the general strategy (e.g. whether to introduce new information, to elaborate on previously given information, or to provide an example), and the specific realization (the concrete verbal utterance). Adaptivity must be enabled at each of these levels. We will focus on the dialog level at which two things are determined: the information that needs to be explained next and how it is done best. Both decisions depend on a model of the current situation as well as the partner. To that end, the explainee's understanding and other relevant characteristics are extracted from the interaction by analysing the feedback and are stored in a dynamic partner model. As this partner model becomes more and more specific over time, the decision process and the partner model influence each other. In result, the generation problem changes from time to time or, in other words, is non-stationary.

Adopting a broad view on explanations, we define an explanation goal not only as to give reasons (i.e. answering *why-questions*) but also to enable the

[a] https://orcid.org/0000-0001-5622-8248
[b] https://orcid.org/0000-0002-4047-9277

Robrecht, A. and Kopp, S.
SNAPE: A Sequential Non-Stationary Decision Process Model for Adaptive Explanation Generation.
DOI: 10.5220/0011671300003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 1*, pages 48-58
ISBN: 978-989-758-623-1; ISSN: 2184-433X

recipient to act or to help her understand (i.e. answering *how-questions* or *what-questions*). One argument for this is that providing a *why*-explanation requires a certain degree of common ground with regard to *what-/how*-understanding. More generally, we build on models of rational communication and treat the generation of explanations as a problem of rational decision-making. In this, an agent makes decisions by considering possible actions and choosing the one that maximizes a desired outcome weighed against its potential costs (Russell and Norvig, 2021). The standard approach is to model this as a (partially observable) markov decision process ((PO)MDP), whose solution is an action policy that can be determined via reinforcement learning (RL) (Sutton and Barto, 2018) or through some form of (usually approximate) planning such as value iteration (Lapan, 2018) or Monte Carlo tree search (Silver and Veness, 2010). A particular challenge arises when the environment dynamics, i.e. the way specific actions bring about specific effects, is changing over time. This is the case for the co-constructed explanation process as outlined above due to (1) the growing (non-)understanding of the explainee and (2) the correspondingly changing effects of explanation moves the agent may take. Indeed, different explanation strategies lead to various results depending on expertise, interest or communication skills of the explainee (Buhl, 2001; Grice, 1975): While presenting the information in small pieces and elaborating it with reformulations or additional information might suit a novice explainee, it will be more and more inappropriate as the explainee gets increasingly informed.

In this paper we propose SNAPE, a model that formalizes explanation generation as sequential, hierarchical decision-making in a non-stationary environment. In this model, a global explanation plan determines blocks of information to be explained. Then, local decision problems (MDPs) concerning the selection of explanation actions and moves are continuously revised and reformulated due to the changing partner model, and then solved efficiently through online planning. We demonstrate the approach in the domain of explaining the board game *Quarto!*, which is of a manageable size, comprises both rules as well as strategic elements, and allows for testing the explainee's understanding by letting her play the board game subsequent to the explanation. We start by discussing related work and underlying concepts in Section 2. Section 3 will describe the concept and structure of the model focusing on how it provides explanations sequentially and adaptively, based on continuous updates of a partner model. We will then present first results (Section 4) obtained with the model in simulations of explanation processes with different explainees. We will also show dialogue examples displaying different strategies SNAPE uses in result of varying partner models. Finally, a conclusion and discussion of future plans is given in Section 5.

2 RELATED WORK

2.1 Explanation Generation

In an explanation, an *explainer* and an *explainee* are interacting to establish sufficient understanding (Rohlfing et al., 2021). The object to be explained is referred to as the *explanandum*, while *explanans* refers to the particular way how the explanandum is explained. Generally, explanations can vary largely with regard to the explainer's stance or goal, or the causal patterns laid out by the explanans (Keil, 2006). Most current research in XAI looks at introspective, post-hoc explanations of a single decision of an AI system using, e.g., contrastive or counterfactual explanation patterns. This is fueled by the intrinsic interest of understanding deep neural networks (DNNs) and related "blackbox" algorithms, but also by external requirements such as the General Data Protection Regulation (GDPR). Consequently, current approaches in XAI aim at creating transparency, interpretability and explainability amongst others (Anjomshoae et al., 2019; Stepin et al., 2021). We focus on achieving explainability, which is mainly concerned with how to make a description understandable to the explainee (Rohlfing et al., 2021) through an appropriate, adaptive explanation process.

An explanation process can be defined as a sequence of phases that consist of explanation and verification blocks (El-Assady et al., 2019). The optimal pathway though an explanation and the best explanation strategy can differ depending on the targeted level of detail, the targeted explainee, or the desired level of interactivity. Current work in XAI has started to develop context-aware methods, making up 43% of explanation studies, while user-aware methods are still relatively underrepresented (only 10%) (Anjomshoae et al., 2019). One major challenge for the latter is that (1) there is not one global explanation that fits all (Sokol and Flach, 2020), and (2) prior assumptions about an explainee (user models) need to be adapted continuously as the explanation process unfolds. Therefore an explanation cannot be fully preplanned. Recent research also has been directed to explanations in human-robot interaction (Stange et al., 2022) or to eliciting explanations towards an artificial explainee. In Zare et al. (2019) the agent learns a game by playing and asking questions to a human explainer. Other approaches aim at developing an artificial explainer (Allameh and Zaman, 2021), e.g. in the form of a chatbot that is giving in-game support rather than an explanation for a given domain.

2.2 Markov Decision Processes

MDPs are a common approach to model sequential decision processes and agent-environment interactions (Lapan, 2018), with only few explanation generation approaches using MDPs so far (Anjomshoae et al., 2019). Classically, an MDP (Puterman and Feinberg, 1994) is defined by a tuple $< S, A, T, R >$, where S is a finite set of possible states and A is a finite set of actions. The transition model $T(s|a,s')$ describes the probability to reach a new state s' after performing action a in state s. The reward model $R(a,s)$ defines the immediate reward that is obtained after performing action a in state s.[1] Various extensions or sup-types of MDPs have been defined for specific domains, two of which are relevant for the present model are briefly discussed in the following.

2.2.1 Non-Stationary MDPs

MDPs typically assume the environment to be stationary, i.e the transition probability $T(s|a,s')$ to be constant throughout the whole decision process. Likewise, A and S are assumed to be fixed. However, in many real-life settings such as self-driving cars, stock market forecasting, or dialogue management, the environment is non-stationary. Different approaches to deal with non-stationary MDPs (NSMDPs) can be used (Lecarpentier and Rachelson, 2019). One option is to model several experts or policies and to compare their predictions in order to minimize *regret* (Cesa-Bianchi and Lugosi, 2006; Fruit et al., 2017). Usually regret minimization is used in competitive domains with at least one adversarial. A second approach – called Hidden-Mode MDP (HM-MDP) (Hadoux, 2015) – is to break down the NSMDP into different modes. Depending on the current mode the MDP is in, the transition and reward model differ. HM-MDPs have been used in several domains (Chades et al., 2012) and are employed in our model to adapt the transition probabilities according to the current partner model. Note that this is different from using so-called "non-stationary policies" (Scherrer and Lesner, 2012), which can change for certain parts of a decision sequence and were shown to be sometimes more efficient than classical policies.

[1]Alternate reward models may be $R(s|a,s')$ or $R(s)$ defined in regards to the domain.

No matter which kind of MDP is used, a crucial question is how to solve them for a policy. Classical offline planning aims to find the optimal policy beforehand. This is computationally costly in larger state-action spaces and infeasible in non-stationary environments, because each time the environment changes a new policy needs to be determined. An alternative is to use online planning with approximate methods such as Monte Carlo Tree Search (MCTS) (Pettet et al., 2022). This approach is described along with suitable application domains in Section 2.2.3. Here, we employ MCTS within a hierarchical model, in which local MDP models are formulated with smaller action-state spaces so that MCTS is applicable.

2.2.2 Hierarchical MDPs

A key challenge with MDPs is that the state-action space tends to increase rapidly. One approach to deal with this, besides using Partially Observable Monte Carlo Planning (POMCP) (Silver and Veness, 2010) or combining RL and MCTS (Pettet et al., 2022), is to use hierarchical MDPs. This has been proposed, e.g., for a robot navigation task (Bakker et al., 2005): The key idea is to group states into clusters, which are states of a higher-level MDP but full state spaces of lower-level MDPs. At the lower level, the state space is decreased due to only solving the sub-problem of the associated cluster. The idea of using a hierarchy in an MDP is well established, for example in planning algorithms and decision-making. The challenge is to cluster the correct states together. To that end, Hierarchically Determinized MDPs (HDMDP) were introduced (Barry et al., 2010). Further, higher level clusters or macro-actions can be seen as local policies for solving sub-problems of the MDP (Hauskrecht et al., 2013). This allows for reusing the lower level policy several times. Nevertheless, in some cases using macro-actions can be costly too and lead to a less flexible and more abstract model (Hauskrecht et al., 2013). Notwithstanding these problems, many approaches on modelling hierarchical MDPs have been put forward (Kessler et al., 2017), all pointing out two main advantages: (i) the reduction of state and action spaces by breaking the problem down into sub-problems and (ii) the reusability of higher level actions.

2.2.3 Monte Carlo Tree Search (MCTS)

MCTS provides a popular way of solving MDPs online. A MCTS policy aims to find a balance between searching for new (exploration) and trying promising options (exploitation). Coulom (2007) first combined

Monte-Carlo evaluation with tree search, and Kocsis et al. (2006) introduced a first Upper Confidence Bound Algorithm (UCT1) that treats the choice of a child node as a multi-arm bandit problem. Generally, the MCTS algorithm consists of four steps that are repeated in each iteration: selection, expansion, simulation and backpropagation (Browne et al., 2012). The first step of the algorithm is the selection. Starting at the root node the algorithm moves down the tree until it reaches a node that is expandable, which means it is nonterminal and has unseen children. After selecting the node it is expanded, which means that at least one child node is added. How many children can be added to the tree depends on the available actions. Starting at the new node a simulation is run using the default policy and the result of the simulation is propagated back to the root node to update the probabilities.

3 THE SNAPE MODEL

In this section we introduce the SNAPE model which formalizes adaptive explanation generation as a sequential non-stationary decision process. It builds on the view that producing an explanation can be seen as a hierarchical process (Figure 1).

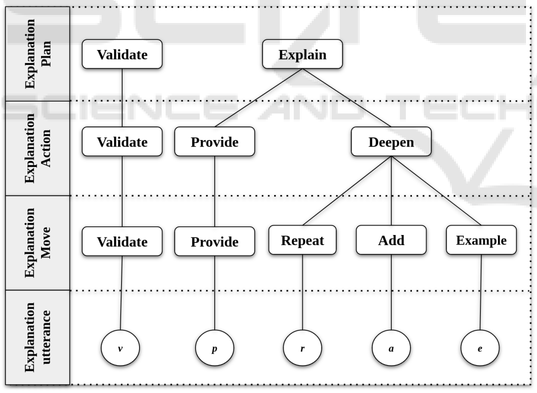

Figure 1: Assumed hierarchical structure of an explanation.

At the highest level of an explanation, the explainer follows an explanation plan that determines the next block. A block can either consist of explaining or verifying a given content (El-Assady et al., 2019). While providing an *explanation block* is possible as long as the block is not assumed to be understood by the explainee, providing a verification block is possible only if a content has been explained before. As soon as a block is selected, it is passed down to the next level which determines an appropriate *explanation action*. Here, the explainer either performs a validate action or, for an explanation block, chooses between providing new content or deepening the current one further. The action is transmitted to the *explanation move* level, at which the explainer decides for the best move to perform the given action. While, at the explanation action level, the focus is on which information to discuss, the move level evaluates how the information can be presented best. Finally, the move is passed down to the *explanation utterance* level, where a verbal utterance is generated. In sum, the explanation plan and action concern the content structure of the explanation, while the explanation move and utterance levels mainly determine the linguistic representation.

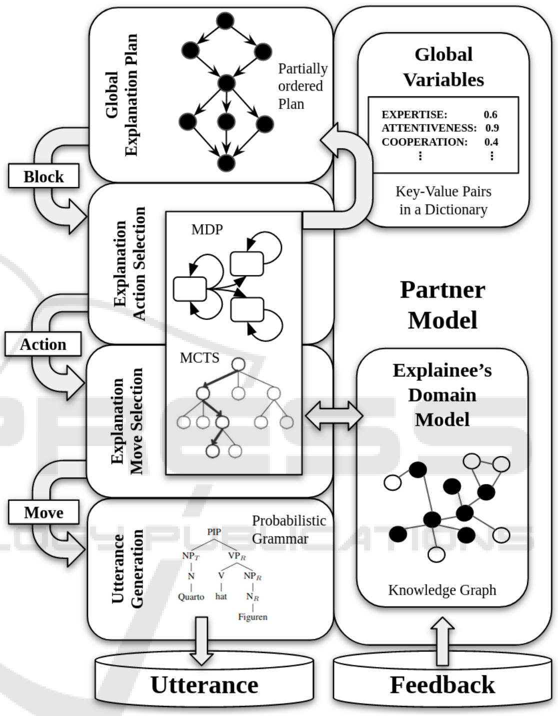

Figure 2: Structure of the SNAPE model.

The SNAPE model follows this hierarchical structure and formalizes the corresponding decision problems (Figure 2). At the highest level, a partially-ordered plan (POP) specifies the logical order of the explanation blocks. In a partially-ordered plan some of the elements have a strict order, while others can be swapped to a certain extent. Our approach to form these explanation blocks differs from to the ones described in Section 2.2.2 as we receive the clusters from a data-set of human-human explanations of *Quarto!* that we translated into an ontology using *Web Ontology Language* (OWL 2) (Motik et al., 2009). The knowledge graph built from this ontology can be seen in Figure 3. While the ontology contains all the information connected to the domain, the knowledge graph includes only a subset of the obligatory triples (e.g., information like *the board is made of wood* or *you can play Quarto online* is included in the ontology but not in the knowledge graph for the game explanation). All the triples in the knowledge graph are assigned to the corresponding explanation block. In Figure 3 the blocks are indicated as gray areas. Correspondingly, the state space of the MDP is a subgraph of the knowledge graph and consists of a set of triples. The global explanation plan determines which block to perform next. The block includes a list of all triples that have to be understood by the explainee in order to complete the explanation successfully. The figure illustrates that a block corresponds to a state at the explanation plan level, while determining a state space at the action and move selection level.

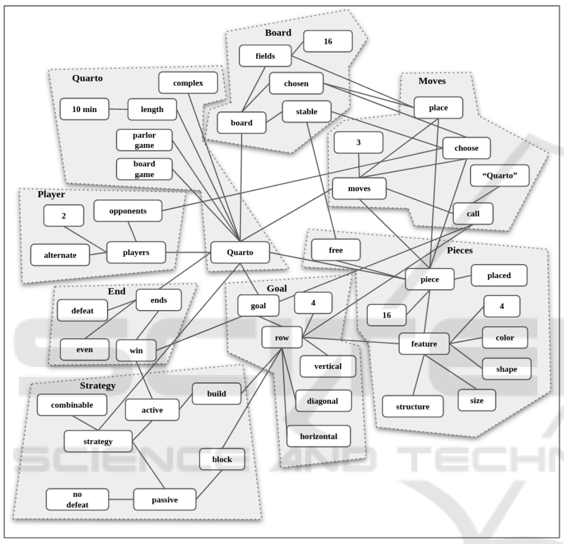

Figure 3: Knowledge graph of triples and associated explanation blocks for the game domain.

Each triple describes a relation (predicate) between two elements (a subject and an object). For illustration, the triple underlying the semantics of the sentence *"Quarto is a board game."* can be seen in Figure 4.

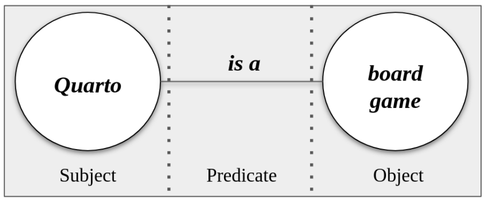

Figure 4: The semantics of the sentence *"Quarto is a board game."* in the form of a (entity-relation) triple.

Given the block, an MDP is built for solving explanation action and explanation move selection together. While the explanation action selection decides which triple to deal with next, the explanation move is more restricted to the linguistic representation of the triple. After choosing which explanation move to perform on which triple the information is passed down to the lowest level, the utterance generation. At the current state the utterance generation is a dictionary of move-template pairs, but will be extended to a probabilistic grammar (Collins, 2011) with probability distributions being influenced by the partner model.

One key innovation of the SNAPE model is the integration of an online partner model in the decision process in order to allow for creating an explanation adapted to the current state of the explainee's understanding and preferences. Different features of the partner model are directly connected to certain explanation levels of the model, while others serve as input for multiple levels (see below). In general, the partner model consists of two components that represent central assumptions an explainer holds about an explainee: (1) the partner's domain model covering the parts of the knowledge domain that are assumed to be understood, and (2) more global features that describes the partner's general ability and willingness to follow the explanation. In its current state, the partner model's features are the explainee's *expertise*, *attentiveness* and *cooperation*. The *expertise* refers to the general knowledge the explainee has of the domain. The *attentiveness* assesses how easily the explainee gets distracted, while *cooperation* estimates the extent to which the explainee is willing to give information about her current level of understanding.

Initially, the partner model starts as a very generic model (based on prior assumptions) but is then updated through the feedback the explainee provides. How powerful the integration of the partner model is, will become clear in the next two subsections that will provide a closer look at the MDP, the partner model and the adaptivity resulting from their combination.

3.1 Adaptive Explanation Action and Move Selection

As the explanation block is given by the global explanation plan, the set of triples and a list of templates for each action is passed down to the explanation action and move selection. Then, a local MDP is composed whose state consists of all triples of the respective explanation block, along with a representation of the level of understanding *(lou)* for each triple. In the initial state all *lou* values are set to *None*. A list of all the actions and their moves that can potentially be performed at the current state is composed. Possible actions always consist of a type (*action_type*) and the triple that is currently under discussion (*action_cud*).

There are two actions available: *provide* and *deepen*, which are mutually exclusive. *Provide* is available for all triples with the value *None*, while *deepen* is available only if the triple has been introduced already. Moves are more complex than actions. An explanation move describes the way of how to perform the action at the dialog level (see examples in Table 1) and can be seen as sub-actions. This means that not only the full explanation process but also the MDP has a hierarchical structure with actions at the higher and moves at the lower level. If the action *deepen* is chosen all the moves (*repeat, add, example*) are possible but with different transition probabilities and following states.

Apart from the actions that are available, the MDP needs to know about the likely results of performing an action-move pair. To that end, all actions and moves available at the current state are simulated taking into account that an action can either be successful and lead to a new state, or can be unsuccessful so that the MDP remains in its current state. Crucially, the transition probabilities depend on the partner model's value of *attentiveness* p_a as shown in Table 1. The action *provide* always leads to a new state, where the level of understanding (*lou*) is the initial value (lou_{init}). The move *repeat* has a high probability to result in a state change, as we assume a repetition or reformulation does not need a high level of attentiveness or focus. The move *example* has a higher probability to fail than the *repeat* move, as it is more complex and engages the user to actively transfer knowledge.[2] The third move is to give *additional* information about the triple. This is not necessarily needed for the explanation but might help to understand or connect the information to prior knowledge. The transition probability of this move is in between those of the other two moves. It needs more attentiveness than a *repetition* but less than an *example*.

Upon conveying information to the explainee, the MDP changes to a new state that is also dynamically determined based on the complexity of the triple (c_t), the expertise of the user (p_e) and the kind of move (see Table 1). The complexity of the triple is stored in the dictionary and taken from human-human explanations, triples that caused confusion or frequently led to a misunderstanding are attached a high complexity level. The expertise capture's the explainee's domain-specific ability to understand. The kind of move is important as we hypothesise that different moves require different levels of communicative activity and therefore lead to different improvements of the level of understanding (*lou)*. If the current move is to *repeat* a triple the level of improvement is comparatively low, as a reformulation is not a very activating move for the explainee. Giving an *example* does need a high level of activity and therefore leads to a higher improvement. Again to add further information is the move that is in between. It is more active than a repetition, but less active than the example.

Further, the MDP needs to define what a terminal state looks like. In our case a state is terminal as soon as all the values have exceeded a certain level of understanding *lou* (in our case 0.75). The last element of the MDP that needs to be defined is the reward function, which assigns a reward to each possible state. For the MCTS only the final reward is needed, which in our case is the sum of all the rewards during the iteration. The reward is negative until the terminal state is reached. A non-terminal reward is set to the difference between all triples that exceed the needed level of understanding (0.75) and the full amount of triples. The reward at a final state is set to 100.

The decision process runs as follows: The best move (in terms of maximal expected reward) is calculated using MCTS and then performed by processing the template and offering the explainee the option to provide feedback. Currently the user can provide positive, negative or no feedback. If the feedback is positive, the information is assumed to be grounded (*lou* value of 0.9), while it is reset to its initial state if the feedback is negative. If no feedback is given, the MDP calculates a state parallel to the predicted state as described above. Then the current state is updated and the MCTS, again, finds the best move to perform. As the current explanation block is fully grounded, the MDP reaches its final state. Now the partner model is updated and the global explanation plan picks the next block to explain. How the partner model update is works in detail will be content of the next subsection.

3.2 Partner Model Update

As described above, different variables in the partner model influence the model's probability of changing into a new state as well as the new state itself. In result, changes to the partner model cause the generation process to be non-stationary by yielding new MDPs that steer the explanation actions/moves for a limited period of time. Initially, the partner model holds predefined values and an empty domain model. Then, the partner's domain model is updated constantly during the explanation process. Each triple

[2] Also, examples tend to be longer than repetitions of information. At the current state of the model this is implicitly included in the complexity level of the triple, later it will be encoded by the utterance complexity and influence the probability distribution of the probabilistic grammar.

Table 1: Overview of action-move combinations with examples, along with the (predefined) formulae for calculating new states and transition probabilities of the adapted MDP.

Action-Move	Example	New State	Transitions
Provide-Provide	*"To win you need to build a row."*	$lou'_i = \frac{p_e}{c_t}$	$t(s'\|s) = 1$ $t(s\|s) = 0$
Deepen-Repeat	*"You can win by creating a row."*	$lou'_i = lou_i + \frac{1-lou_i}{c_t} * \frac{0.4+p_e}{2}$	$t(s'\|s) = 0.9 * p_a$ $t(s\|s) = 1 - 0.9 * p_a$
Deepen-Add	*"A row has a length of 4 figures."*	$lou'_i = lou_i + \frac{1-lou_i}{c_t} * \frac{0.6+p_e}{2}$	$t(s'\|s) = 0.6 * p_a$ $t(s\|s) = 1 - 0.6 * p_a$
Deepen-Example	*"For example: If there are three small figures in a row and you have a small one, you can build a line."*	$lou'_i = lou_i + \frac{1-lou_i}{c_t} * \frac{0.9+p_e}{2}$	$t(s'\|s) = 0.4 * p_a$ $t(s\|s) = 1 - 0.4 * p_a$

that is introduced (*provide*) is added to the knowledge graph, each triple that is acknowledged with positive feedback or exceeds the needed *lou* is additionally labelled as *grounded*. Every time an explanation block is passed back to the global explanation plan the global variables in the partner model are updated as well. If the frequency of feedback given by the user is high, the level of *attentiveness* rises, which results in higher transition probabilities in the next MDP. In contrast, the attributed *expertise* is mainly influenced by the kind of feedback that is generated. Positive feedback on a complex triple hints to a high level of expertise, while the opposite, negative feedback on an easy information, is an indication for a lower level of expertise. Also it can be measured by asking open questions about the explainee's previous domain knowledge (in our game domain, e.g., *"You probably know Best of Four, right?"*). This was observed a lot in human-human game explanations.[3] Note also that the selection of the next explanation block depends on the current partner model and is hence subject to changes due to partner model updates. Specifically, whether to validate or to move on and explain the next block is significantly influenced by the partner's attributed *cooperation*. The *cooperation* value of the partner is derived from the quality of the produced feedback (see below).

The detailed update rules for *expertise* E_b and *attentiveness* A_b with respect to a block b are as follows ($|FB_p|$: number of positive feedback signals, $|FB_n|$: number of negative feedback signals, $|FB|$ number of all feedback signals including *None*):

Expertise:

$$E_{b+1} = E_b * \frac{1}{|FB_{p+n}|} + \frac{act}{|FB_{p+n}|} * (1 - \frac{1}{|FB_{p+n}|})$$
$$with$$
$$act = 0.5 + \frac{tanh(2 * pos - neg)}{2}$$
$$pos = \sum_{i=1}^{|FB_p|} c_i, \;\; neg = \sum_{i=1}^{|FB_n|} 4 - c_i \quad (1)$$

Attentiveness:

$$A_{b+1} = (A_b + \frac{|FB_{p+n}|}{|FB|}) * 0.5 \quad (2)$$

4 RESULTS

In order to evaluate the SNAPE model we have implemented the described approach and tested it in explanation interactions with simulated explainees. Generally, our hypothesis is that the adaptive model will be able to generate user-specific explanations that can yield a better understanding of the domain, higher user satisfaction, and trust. While this is left to an evaluation study with human users, in this paper we present first results to demonstrate the model's ability to adapt to different explainees and to produce according explanations.

We created three partner models *Harry*, *Ron* and *Hermione* (Table 2). *Hermione* is highly interested, always stays focused and is an expert in nearly everything. *Harry* often gets distracted and is not much of an expert in the game domain. Finally, *Ron* does not have a high level of attentiveness as he easily looses track or gets bored when listening to an explanation, but he has a high level of game expertise. Note that, as we focus on the explanation blocks only, the *cooperation* value is not considered here. Also, due to

[3]This kind of questions is also a building block for explaining through analogy, which we leave to future work.

the simplified feedback, it is currently not possible to analyse the quality of feedback.

Table 2: The three partner models used for simulation.

Partner	Attentiveness	Expertise
Hermione	0.9	0.9
Ron	0.3	0.8
Harry	0.3	0.3

These models were used as different initial partner model of the explainer. Further, during the explanation process, feedback was generated according to three different feedback distribution types (Table 3).

Table 3: The three kinds of feedback distribution.

FB	Negative	Positive	None
A	0.2	0.6	0.2
B	0.01	0.39	0.6
C	0.39	0.01	0.6

In feedback type A there is a high frequency of feedback overall, which would be correlated with a high level of attentiveness. The feedback is mainly positive, which should indicate a high level of expertise. Feedback type B rarely produces feedback and therefore displays a low level of attentiveness. If feedback is produced, it is mainly positive, this correlates with a high degree of expertise. Finally, feedback type C should correspond to a low level of both attentiveness and expertise.

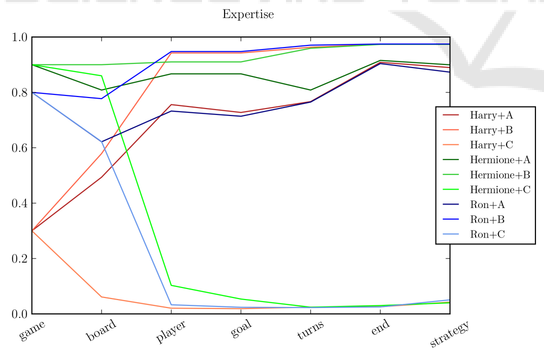

Figure 5: Expertise development during explanation.

In our evaluation, we have SNAPE explaining *Quarto* starting out with one of the three partner models and then receiving feedback according to one of the three types. If the feedback matches the initially assumed partner model (i.e. Hermione - feedback A; Ron - feedback B; Harry - feedback C), SNAPE should explain suitably to whom it thinks it is talking to. Adaptive explanation generation, however, would be observed if the partner does not behave as expected and the system changes its explanation behavior in result, i.e. produces feedback that does not match the initial attributions.

We combined each initial partner model with each of the possible feedback variants and let SNAPE explain for ten times each. As long as the partner model matches the feedback type, we expect the development of the partner model during the explanation to be stable. In the other cases we expect the partner model to update to the feedback and cause changes in the explanation behavior, which will be shown in this section as well.

Figure 5 shows how SNAPE adjusts the *expertise* value to the unexpected feedback distribution. The main changes appear in the updates after the first three explanation blocks. It can also be observed that – no matter with which initial partner model SNAPE started – the same feedback leads to a more or less equal *expertise* value in the end of the explanation.

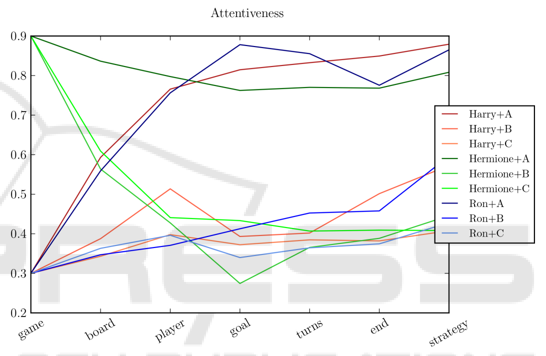

Figure 6: Attentiveness development during explanation.

The second partner model value we were interested in was *attentiveness*, which differed in the feedback distributions as well. While B and C generated comparably little feedback, we had a lot of feedback in A. Figure 6 confirms that, no matter what the initial partner model was, SNAPE ends up at a higher *attentiveness* value after getting feedback A. If the initial partner model does not match the actual feedback, the partner model is adjusted after the first four sequences.

To illustrate how SNAPE adapts to different partners, we now compare two example explanations of the explanation block *board*. SNAPE explains the block to Hermione and Harry, respectively, who give feedback in accordance to what SNAPE's partner model expects. First lets consider the conversation between SNAPE and Hermione. We can see two interesting behaviors in the dialog. The first is a proof to what we previously expected: The explanation is comparatively short and mainly consists of the move *provide* (P), as Hermione tends to give positive feedback, which generates a high level of understanding (*lou*) and does not leave any need to deepen the content currently under discussion (*cud*).

After SNAPE *provided* the triple t1 (P-t1), Hermione does not give any feedback, so he repeats the triple. Then, even though there is still no feedback given by Hermione, SNAPE moves on to the next triple. As he does not come back to the triple during the explanation, it is taken as grounded.

```
Snape: "Quarto has a game board." (P-t0)
Hermione: +
Snape: "The unused figures are put next
    to the board, the stable." (P-t1)
Hermione: None
Snape: "Quarto has a stable." (R-t1)
Hermione: None
Snape: "The board consists of individual
    fields." (P-t2)
Hermione: +
```

This differs from the explanation behavior toward Harry. As SNAPE provides the second triple (P-t2) Harry does not give any feedback, so SNAPE gives some *additional* information (A-t2). Harry again does not provide any feedback and, in contrast to the previous example, SNAPE sticks to the triple currently under discussion (*cud*) and generates an *example* (E-t2) for the same triple.

```
Snape: "Quarto has a game board." (P-t0)
Harry: -
Snape: "The board consists of individual
    fields." (P-t2)
Harry: None
Snape: "It consists of 16 fields." (A-t2)
Harry: None
Snape: "The board is structured like a
    net for example." (E-t2)
Harry: +
Snape: "The board is made of wood."
    (A-t0)
Harry: None
Snape: "The board is placed on the
    table." (A-t0)
Harry: +
Snape: "The unused figures are put next
    to the board, the stable." (P-t1)
Harry: None
Snape: "The stable is not part of the
    actual board." (A-t1)
Harry: +
```

Now Harry gives positive feedback and SNAPE moves to the next triple (A-t0). Also the explanation is longer than the one for Hermione, which is due to the lower expertise and attentiveness of Harry as the explainee. The example also shows that SNAPE is capable of changing to a new triple, even though the item currently under discussion (*cud*) is not grounded, and to come back to it at a later state.

5 CONCLUSIONS

In this paper we introduced SNAPE, an explanation generation model using hierarchical decision processes that integrate flexible, online partner models. This online update results in a non-stationary decision process, which is realized sequentially by composing and solving local, updated models of rational decision-making in explanation generation and communication. By embedding those in a hierarchical decision model we illustrate everyday life explanations, which are roughly preplanned, but provide a detailed plan from sequence to sequence. Such hierarchical decision models minimize state and action space significantly so that local models can be solved online using MCTS. We have shown how the system updates its partner model and the corresponding decision-making during the conversation, based on initial assumptions as well as the kind and frequency of the feedback provided by the explainee throughout the running interaction. We also reported two dialog examples produced for the same explanation block with different underlying partner models. In direct comparison it becomes clear how SNAPE already does adapt its explanation strategy to the partner model of the user.

In future work, we will improve the feedback interpretation to enable evaluation studies with human explainees. By now we only allowed simplified feedback, such as *"+"*, *"-"* and *"None"*. We want the agent to be able to process backchannels as well as more complex feedback, such as open questions, sentence completion or examples generated by the explainee, which may indicate and influence the level of cooperation. A high level of cooperation leads to a smaller need for verification blocks and hence is a partner model feature that influences the highest level of global explanation planning. Further, the utterance generation, which is currently based on predefined templates, will be improved towards a probabilistic grammar. This will allow for adapting to the partner also at the level of lexico-grammatical choices in accordance with the various levels of the partner model already incorporated in SNAPE. Ongoing work focuses on defining and including preconditions for each triple into the model. The level of understanding (*lou*) of the preconditions directly influences the reward that is given for a state. This results in lower costs for generating triples for which more preconditions have been introduced.

ACKNOWLEDGEMENTS

We are grateful to Fabian Heinrich and Markus Rothgänger for help with developing the model, and to Katharina Rohlfing, Heike Buhl, Josephine Fisher and Erick Ronoh for valuable conceptual discussions. This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/1 2021 – 438445824.

REFERENCES

Allameh, M. and Zaman, L. (2021). Jessy: A Conversational Assistant for Tutoring Digital Board Games. In *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*, pages 168–173, Virtual Event Austria. Association fpr Computing Machinery.

Anjomshoae, S., Najjar, A., Calvaresi, D., and Främling, K. (2019). Explainable Agents and Robots: Results from a Systematic Literature Review. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems.

Bakker, B., Zivkovic, Z., and Krose, B. (2005). Hierarchical dynamic programming for robot path planning. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2756–2761.

Barry, J., Kaelbling, L. P., and Lozano-Perez, T. (2010). Hierarchical Solution of Large Markov Decision Processes. *ICAPS-10 Workshop on Planning and Scheduling Under Uncertainty, 2010*, page 8.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Buhl, H. M. (2001). Partner Orientation and Speaker's Knowledge as Conflicting Parameters in Language Production. *Journal of Psycholinguistic Research*, pages 549–567.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press, Cambridge. OCLC: 70056026.

Chades, I., Carwardine, J., Martin, T., Nicol, S., Sabbadin, R., and Buffet, O. (2012). MOMDPs: A Solution for Modelling Adaptive Management Problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):267–273.

Collins, M. (2011). Probabilistic Context-Free Grammars (PCFGs). *Columbia University*.

Coulom, R. (2007). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In van den Herik, H. J., Ciancarini, P., and Donkers, H. H. L. M. J., editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg. Springer Berlin Heidelberg.

El-Assady, M., Jentner, W., Kehlbeck, R., Schlegel, U., Sevastjanova, R., Sperrle, F., Spinner, T., and Keim, D. (2019). Towards XAI: Structuring the Processes of Explanations. *Proceedings of the ACM Workshop on Human-Centered Machine Learning, Glasgow, UK*, 4:13.

Fruit, R., Pirotta, M., Lazaric, A., and Brundskill, E. (2017). Regret Minimization in MDPs with Options without Prior Knowledge. *Advances in Neural Information Processing Systems*, 30.

Grice, H. P. (1975). Logic and Conversation. *Speech Acts*, 3:41 – 58.

Gunning, D., Vorm, E., Wang, J. Y., and Turek, M. (2021). DARPA's explainable AI (XAI) program: A retrospective. *Applied AI Letters*, 2(4):e61.

Hadoux, E. (2015). *Markovian sequential decision-making in non-stationary environments: application to argumentative debates*. PhD thesis, Université Pierre et Marie Curie - Paris.

Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T., and Boutilier, C. (2013). Hierarchical Solution of Markov Decision Processes using Macro-actions. *arXiv preprint arXiv:1301.7381*, page 10.

Keil, F. C. (2006). Explanation and Understanding. *Annual Review of Psychology*, 57(1):227–254.

Kessler, C., Capocchi, L., Santucci, J.-F., and Zeigler, B. (2017). Hierarchical Markov decision process based on DEVS formalism. In *2017 Winter Simulation Conference (WSC)*, pages 1001–1012, Las Vegas, NV. IEEE.

Kocsis, L., Szepesvari, C., and Willemson, J. (2006). Improved Monte-Carlo Search. *Univ. Tartu, Tech. Rep*, 1:22.

Lapan, M. (2018). *Deep reinforcement learning hands-on: apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd.

Lecarpentier, E. and Rachelson, E. (2019). Non-Stationary Markov Decision Processes, a Worst-Case Approach using Model-Based Reinforcement Learning. *Advances in neural information processing systems*, 32:10.

Miller, T. (2019). Explanation in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence*, 267:1 – 38. arXiv:1706.07269 [cs].

Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al. (2009). Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, 27(65):159.

Pettet, G., Mukhopadhyay, A., and Dubey, A. (2022). Decision Making in Non-Stationary Environments with Policy-Augmented Monte Carlo Tree Search. arXiv:2202.13003 [cs].

Puterman, M. L. and Feinberg, E. A. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. *SIAM Review*, 38(4):689.

Rohlfing, K. J., Cimiano, P., Scharlau, I., Matzner, T., Buhl, H. M., Buschmeier, H., Esposito, E., Grimminger, A., Hammer, B., Hab-Umbach, R., Horwath, I., Hullermeier, E., Kern, F., Kopp, S., Thommes, K., Ngonga Ngomo, A.-C., Schulte, C., Wachsmuth, H., Wagner, P., and Wrede, B. (2021). Explanation as a Social Practice: Toward a Conceptual Framework for the Social Design of AI Systems. *IEEE Transactions on Cognitive and Developmental Systems*, 13(3):717–728.

Russell, S. and Norvig, P. (2021). *Artificial Intelligence, Global Edition : A Modern Approach*. Pearson Deutschland.

Scherrer, B. and Lesner, B. (2012). On the Use of Non-Stationary Policies for Stationary Infinite-Horizon Markov Decision Processes. *Advances in Neural Information Processing Systems*, 25.

Silver, D. and Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. *Advances in Neural Information Processing Systems*, 23:9.

Sokol, K. and Flach, P. (2020). One Explanation Does Not Fit All: The Promise of Interactive Explanations for Machine Learning Transparency. *KI - Künstliche Intelligenz*, 34(2):235–250.

Stange, S., Hassan, T., Schröder, F., Konkol, J., and Kopp, S. (2022). Self-Explaining Social Robots: An Explainable Behavior Generation Architecture for Human-Robot Interaction. *Frontiers in Artificial Intelligence*, page 87.

Stepin, I., Alonso, J. M., Catala, A., and Pereira-Farina, M. (2021). A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence. *IEEE Access*, 9:11974–12001.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning*. The MIT Press, 2 edition.

Zare, M., Ayub, A., Wagner, A. R., and Passonneau, R. J. (2019). Show me how to win: a robot that uses dialog management to learn from demonstrations. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, pages 1–7.