Mysterious Murder - MCTS-driven Murder Mystery Generation

Corinna Jaschek*, Tom Beckmann†
Hasso Plattner Institute
Potsdam, Germany
corinna.jaschek@student.hpi.de*, tom.beckmann@student.hpi.de†

Jaime A. Garcia‡, William L. Raffe§
University of Technology Sydney
Sydney, Australia
jaime.garcia@uts.edu.au‡, william.raffe@uts.edu.au§

***Abstract*—We present an approach to procedurally generate the narrative of a simple murder mystery. As a basis for the simulation, we use a rule evaluation system inspired by *Ceptre*, which employs linear logic to resolve valid actions during each step of the simulation. We extend Ceptre's system with a concept of believable agents to make consecutive actions appear to have a causal connection so that players can comprehend the flow of events. The parts of the generated narratives are then presented to a player whose task it is to figure out who the murderer in this story could have been. Rather than aiming to replace highly authored narratives, this project generates puzzles, which may contain emerging arcs of a story as perceived by the player. While we found that even a simple rule set can create stories that are interesting to reason about, we expect that this type of system is flexible enough to create considerably more engaging stories if enough time is invested in authoring more complex rule sets.**

***Index Terms*—MCTS, Believable Agents, Ceptre, Linear Logic, Murder Mystery**

I. Introduction

Murder mysteries as a genre first became popular in the 19th century in literature. These detective stories reached their height of popularity in the 1920s and 30s, which has since been referred to as the Golden Age of Detective Fiction [1]. Many authors from that time are still prevalent in today's pop culture, as can be seen for example in the 2017 movie adaptation of Agatha Christie's novel *Murder on the Orient Express*. Novels from that era often follow very strict rules as defined for example by Van Dine's *Twenty Rules for Writing Detective Stories* [2] or Knox' *Ten Commandments* [3].

Today, murder mysteries have found their way into many forms of media. Television series and movies feature them especially often, with crime shows coming in a wide variety of sub-genres, be it the classical detective story or a more outlandish science fiction settings. Meanwhile, in games, the board game *Cluedo* is one of the longest selling well-known games in history, letting the players compete on who can solve the murder of Dr. Black the fastest[1].

Similar to traditional literature, murder mysteries in video games often follow a linear, pre-written path, but may also include limited branching options. The extent to which games divert from this mode of storytelling will be explored in Section II. Rather than replacing authors by generating stories automatically, this papers aims to provide a tool for authors to craft rules that inform the mechanics of a world. These mechanics, combined with a linear logic evaluator and procedural simulation, then yield a generated narrative whose quality will largely depend on the complexity of the pre-authored rules.

Using this system instead of a hand-crafted branching narrative promises a higher degree of variability and the possibility of emergent, perceived creativity in the stories on each playthrough. The characters in the simulation, modeled as agents using a Monte-Carlo-Tree-Search-based system (MCTS), use the low-level rules to act in ways that the player recognizes as intelligent, planning behavior, further aiding in the generation of creative yet believable murder mysteries. Additionally, the stories generated in this way lend themselves better to be displayed in an amusing puzzle setting, rather than a highly dramatic story that is authored to provide the best pacing and suspense.

In Section II we will introduce related work in the fields of murder mysteries in video games, procedural story generation, and the story generation system Ceptre. Section III will explain our approach to our own linear logic evaluator, our rule set, believable agent system, and the game's user interface. The Section IV discusses, in particular, our opinion on linear logic in this system and the challenges coming with the authorship of the rules. Sections V and VI will talk about possible next steps that would drive this project further and provide a summary of our findings.

II. Related Work

In this section, we will give an overview on existing murder mystery video games and examine how they approach the various challenges of the genre, survey projects that have used procedural generation in the context of game narratives and present the rule-specification language Ceptre created by Martens [4], which our own system is based on.

A. Murder Mystery Video Games

While prevalent in many genres, as seen in Section I, murder mysteries have only been explored in video games rarely, when compared to other genres. This may be due to the complexity involved in detective work as portrayed by literature and film. Most murder mystery games rely on handwritten narratives that are similar in plot to a novel or movie. The player has the opportunity to follow the detectives along the story and

[1]https://en.wikipedia.org/wiki/Cluedo

gather evidence, interview suspects, or follow leads. In some cases, they may also expose lies and find holes in presented testimonies. However, most games do not allow the players to make connections between the evidence and testimonies and let them come to their own conclusions. With most steps represented as multiple choice questions, the players just have to pick the correct choice or guess until they get it right [5].

This method is employed in games like *LA Noire* (Team Bondi, 2011), where falsely accusing suspects will make them offer less information to the player, but will never have game-changing consequences. This ensures that players will always be able to complete the game; they will always be led to reach the next scripted story point in the branching or linear narrative, even if they took wrong decisions before.

While this is a valid choice, it has been noted that the resulting game may be less compelling. When the answers are all given within the game, the player cannot become fully immersed in playing the clever detective [6]. However, when players are left on their own to collect all evidence, it can lead to games like *Murder on the Mississippi* (Activision, 1986), which becomes unsolvable if players miss a piece of information early on in the game [7].

To make sure no clues can be missed while still allowing for more interaction than simple multiple choice options, games often feature in-game notebooks or a collection of relevant items and notes. For example in the game *Discworld Noir* (Perfect Entertainment, 1999) or *Contradiction: Spot the Liar!* (Baggy Cat Ltd, 2015). These written down clues often spoil the puzzles they represent as they need to feature the important aspect of the clue in order to be useful at all.

Some games have attempted to remove this step by letting players take their own notes and offering other option to receive more information, such as the in-game search engine in *Wadjet Eye Games, 2006* or the telephone book in *Sherlock Holmes: Consulting Detective* (ICOM Simulations, 1991), which, only when used correctly, open up potential new options for interaction. While this may increase the challenge and subsequent feeling of reward, it may still convey a feeling of just following in someone else's footsteps. The game *Her Story* (Sam Barlow, 2015) is an example of a different way of approaching this problem. The murder mystery game does not involve any sort of confrontation of the suspects. In fact, there is no in-game way of determining if you have come to the right conclusion at all. Players simply gather information by searching a police database until they are satisfied with their understanding of the events. We predict that reaching the correct solutions in this manner tends to be more satisfying to players, as it is a testament to their skills.

Yet, no matter to which of these categories a game belongs, it has a relatively low replay value, as once the murderer is known, the main motivation for playing the game ceases to exist. *ClueGen* [8] attempts to rectify this by using procedural generation on the plots of murder mysteries. A number of possible motives and histories are used to create a multitude of narratives. The addition of possible red herrings or lying non-player characters enhances the complexity even further. Even with a relatively small set of predefined options, the resulting mysteries were sufficiently interesting to players. The players recognized complex, underlying intentions, and structures, even in cases where these were not intended by the author of the rules informing the procedural generation program.

B. Procedural Narrative Generation

For the game PromWeek, McCoy et al. [9], [10] created a "model for socially-oriented gameplay" named *Comme il Faut*. The model draws from a corpus of social norms that inform the decisions of non-player characters each turn of the game. It then generates animations to visualize to players how characters react to their decisions. To reach this point, PromWeek required extensive manual authoring to embed social biases and conventions in a formalized rule system.

A similar demonstration of an engaging, fully realized social game is Façade [11]. Here, as well, it took the authors considerable effort to create the necessary rules. Instead of leaning heavily on exclusively generated content, Facade chose to define story events that may happen at any time if its preconditions are fulfilled, dynamically influencing the story and shaping it as much as possible based on the player's interaction with the game.

Rowe et al. [12] demonstrate the use of reinforcement learning to adjust stories. A narrative planning system was trained by having a large number of students play an educative story-based game and fill out surveys on their experience. Different branching story options were selected for different players. This allowed an automated analysis of the story variants that were particularly well received and had the highest impact on the education of the players.

Ceptre[2], which is used as the basis for the work presented in this paper, is a "rule specification language" for interactive narratives or simulations [4]. Its intention is to allow rapid prototyping by providing a language specific to this domain. It uses linear logic as the foundation for the interpretation. Linear logic is a formal logic that extends the classical logic used in mathematics. The classical logic concerns itself only with "stable truths". These can, however, not always be applied directly to real-life situations, as they lack causality [13]. While in classical logic all conditions stay the same after an evaluation, in reality, the conditions may be modified. For example, trading money for an object removes the possibility of spending this money again and changes the truths "I have money" to "I have this object". Linear logic focuses on its formulas as resources, rather than truths [14]. New resources may be added to the simulation's state or removed in the course of rule evaluations.

Ceptre allows a game designer to specify terms, rules, and predicates as well as an initial configuration S. The initial configuration S consists of a set of predicates that represent the collection of truths at this point, such as "I have money". The execution of a Ceptre simulation consists of a series of

[2]https://github.com/chrisamaphone/interactive-lp

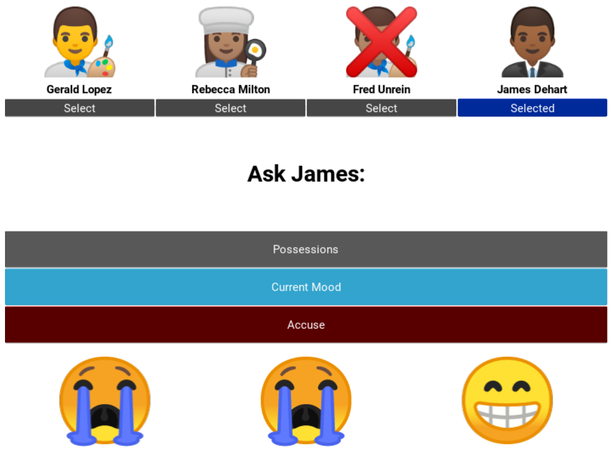

Fig. 1. Up top the four involved actors can be seen, with the murder victim crossed off in red. Players may now select characters and choose to ask them questions, either about their own feelings and possessions or about their relationship to other characters. Players may also choose to accuse a character, which will end the game and reveal whether their solution was the correct one. In this case, we asked James for his current mood and he answered twice with "sadness" and once with "joy", which may lead the player to make conclusions on this actor's motivations.

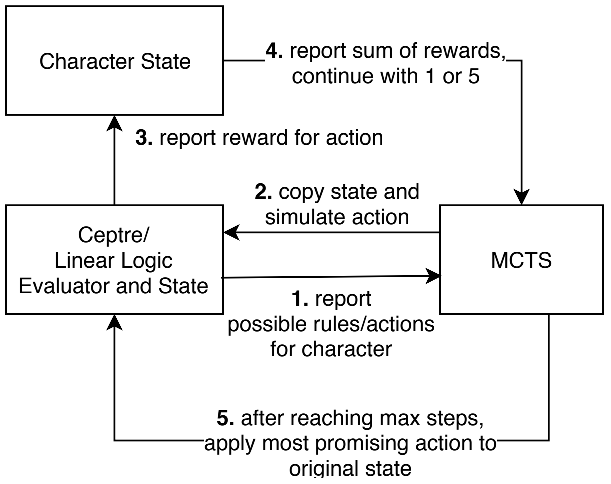
Character State
4. report sum of rewards, continue with 1 or 5
3. report reward for action
2. copy state and simulate action
Ceptre/ Linear Logic Evaluator and State
MCTS
1. report possible rules/actions for character
5. after reaching max steps, apply most promising action to original state

Fig. 2. The communication between the three core elements of our system. When tasked with deciding on an character's next move, the MCTS component continuously creates copies of the simulation state to explore which sequence of actions provides the given character with the highest reward, until a maximum number of steps has been taken.

repeated steps. During each step, it will first be determined which rules can be applied to the current state. A rule may have a set of preconditions, which is the resources that will be consumed from the state if that rule is applied, and a set of resulting resources. This corresponds to the possible actions an agent could take or in which a simulation may evolve.

A valid rule can either be chosen interactively by a player or randomly assigned. The chosen rule will then be applied to the state, creating a new instance of state S', on which the simulation step is then repeated until some terminal condition is reached, for example, when no more rules are available.

III. Approach

In this section, we describe our system for procedural generating compelling narratives in a bespoke murder mystery game. This includes an overview of the gameplay, our approach to a linear logic evaluator and how stories were generated to ensure that actors made choices that appear coherent, and finally the user interface we built to allow players to interact with the system.

The game revolves around murder mysteries that are generated in a simulation. In this simulation, multiple artificial actors get to take various actions, most of which involve an interaction with another actor. The actions are structured in a way that eventually a murder will happen, at which point the simulation stops. Players are then presented with the involved actors and who the victim was. Using the interface shown in 1, the player can then attempt to solve the murder by questioning the actors about their relationships and ask for testimonies of the circumstances leading up to the murder.

To generate the murder mystery simulations, our system[3] uses Ceptre [4] as a basis. Ceptre is a system to formulate and run interactive narratives or simulations. We extend Ceptre's core concept with non-deterministic outcomes and a reward system for actions to drive artificial agents. Our simulation is neither driven by a human player or random choices, which are the two options Ceptre offers. Instead, we implement MCTS agents that drive the story by evaluating the possible options to maximize their reward. Figure 2 illustrates how Ceptre, the MCTS system and the agents communicate.

A. Linear Logic Evaluator

Our approach on story generation is based on the approach used by the Ceptre story generation engine. This engine comes with a linear logic evaluator that takes rules that describe the transformation of resources to model the development of a story. The Ceptre engine is written in the Standard Meta Language and compiled via MLton[4], though to make predicting the outcome of the simulation multiple steps ahead feasible in terms of performance, we re-implemented Ceptre's engine in Python. Early in the project, we used too many constraints, modeled as resources, for Ceptre to handle. Execution speed would quickly deteriorate, especially as the simulation dragged on. As a consequence, we implemented our own reduced version of Ceptre, containing only the aspects that we were making use of. This meant that we dropped features like Ceptre's stages and its advanced type system with support for inheritance. The type system, which allowed for elegant rules like "has Character gun" for the predicate "has character object" and the subtype "gun" of "object", made the evaluation considerably more complex as each type instance would increase the possible permutations of each rule making use of it. For our simpler use case, we were able to avoid

[3]https://github.com/corinnaj/mysterious-murder Latest commit at time of writing: a2db58cefeae59b43818b35f44eb5f4c75974896

[4]http://mlton.org/

using types other than our actors by using rules of the form "has_gun Character".

Our linear logic evaluator takes a set of actors, a set of rules and a set of predicate instances that form the initial state of the simulation. Predicates have a name and an ordered list of placeholders that can be taken up by any actors. A predicate instance is a predicate of which all placeholders have been configured with concrete actors. Rules consist of a left-hand side and a right-hand side, both containing predicates. The predicate's placeholders are numbered consistently so that advanced rules that refer to the same actors across the predicates of a rule are possible. To this extent, our evaluator forms a strict subset of that of Ceptre.

We extended this system by introducing non-determinism. This means that the right-hand side of a rule may list multiple sets of predicates that are possible outcomes of activating this rule, together with a probability of each outcome occurring. For example, proposing marriage to another actor has a 90 percent chance of succeeding, while a rejection is less likely with a corresponding 10 percent chance. Prior to this, we had two rules for each possible outcome, for example, one for stealing from another actor successfully and another for getting caught in the act. This worked fine while randomly selecting rules, but as soon as we started using intelligent agents to select rules, the agents quickly learned that picking the rule where they get caught was not a good choice, rather than attempting to balance the potential risk-reward of a rule.

To further guide the agent in its decisions, we added a system of rewards to each rule. For each outcome of a rule, the agent could receive a reward or a penalty in different categories. As categories, we chose *social*, *sanity* and *fulfillment*, loosely based on the top three layers of Maslow's hierarchy of needs *belongingness and love*, *self-esteem and respect* and *self-actualization* [15]. As an example, the rule *steal_debt*, in which one actor steals from another in order to pay off their debts, results in a positive reward of 120 in fulfillment, while also entailing a negative reward of -10 on their sanity. These categories of rewards proved to be general enough so that each rule we developed could have a value that felt intuitive associated with it.

Lastly, to make the events more readable to players, we added a built-in templating system that used knowledge embedded in the system to formulate the rules as natural sentences. This allowed the formulation of templates such as:

```
Shocked that [0:his|her] [2:husband|wife]
was cheating, {0} murdered [2:his|her]
lover {1}.
```

where the each {i} will be replaced with the name of the i-th actor in the rule and each [i:a|b] will be replaced by a or b depending on if i-th actor is male or female.

B. A Murder Mystery Rule Set

As a basis for the rules, we chose to introduce resources that model the current relationships between characters and general tendencies in their character to act in certain ways. We adapted the *Big Five Personality Traits* [16], giving each character a balance of three resources for each opposing pair of traits. For example, a character might receive two *cautious* resources and one *curious* resource, or three *confident* resources and no *insecure* resources. This allowed authoring rules that require a certain tendency towards one of the extremes of each trait pair. To model the relationships between characters, we used *Plutchik's Wheel of Emotion* [17]. Again, we have opposing pairs that are not necessarily exclusive in our system. Spending time with another actor may yield *trusting* resources between the two, while fighting might yield *anger* resources. A *making up* rule would then allow removing *anger* resources. To model more extreme characters, we introduced a system similar to alignments as found in some role-playing games. We were inspired by the simple separation into *Good*, *Neutral* and *Evil* as found in *Dungeons and Dragons*[5]. Actors may follow one alignment which will enable them to pick certain options more easily than other characters.

In the next step, we generate relationships and attributes relevant to our domain of murder mystery. This includes whether a character is rich ("has_money Character"), their current relationship to other characters using the wheel of emotion, or whether the characters are either related or lovers or married.

Our rule authoring system is conceptually identical to that used in Ceptre. We, however, directly compose Python objects. This has the disadvantage of having to adhere to Python's syntax rather than being able to define a domain specific language, but the advantage that we can define functions that group often used sets of predicates, enable domain-specific consistency checks or toggle groups of rules with "if"-conditions. All of this would have also been possible in Ceptre, but would have required introducing new syntactical elements in the Ceptre language, while we effectively work on a meta-language level at all times.

Most of the rules pose a dramatic conflict between characters that generate resources like anger or suspicion. Typically, there is a countering rule that resolves some of this conflict by removing these resources again. The system is, however, biased towards escalating the conflict quickly, to ensure that stories do not get out of hand and become too complex to understand for a player. All rules require a motivation as the player would otherwise not have a means to follow the flow of events. For example, fighting with another character requires the preexistence of "anger" between the two. Making up would require characters to have developed two "anger" between each other beforehand, while there also exists a resource of "trust" between the two. Further, almost all rules have global preconditions, such as characters being alive or dead or not currently being married when proposing to another character. These preconditions are modeled in the same way that any other resources are. We did, however, observe that we started thinking about these conditions differently and created macros to help us keep track of this kind of state.

[5]http://dnd.wizards.com/articles/features/systems-reference-document-srd

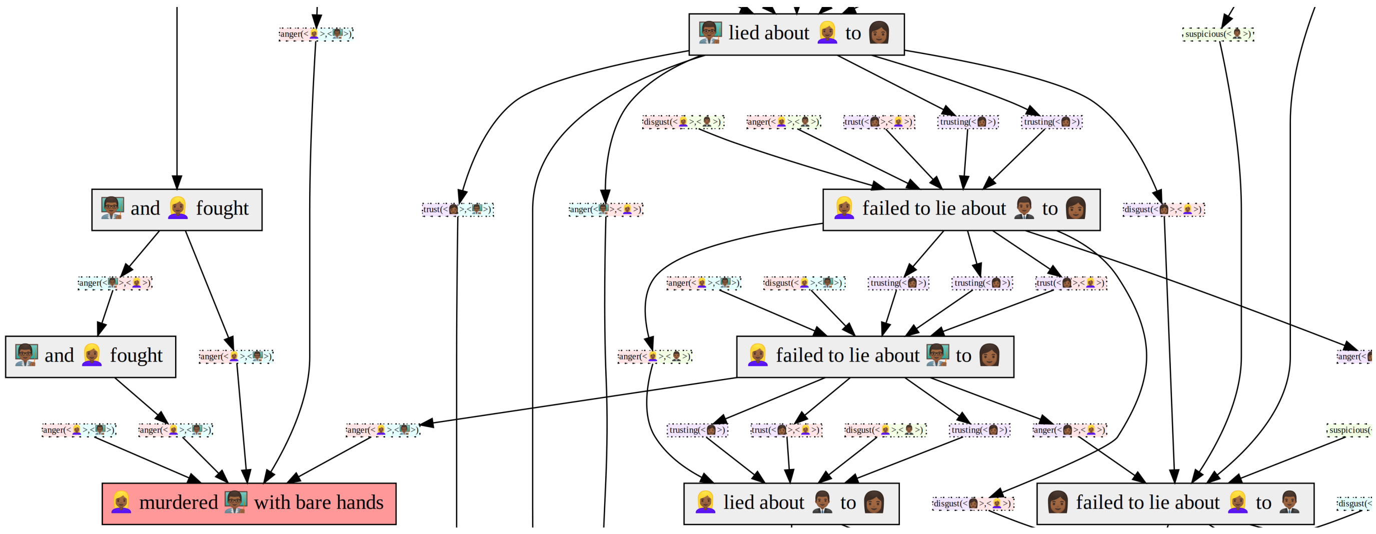

Fig. 3. An excerpt from a generated causality graph. The small nodes reflect resources that were generated or existed in the initial state. The larger boxes reflect the actions that agents chose to take. Marked in red is the rule that ended the simulation, the murder.

In our current rule set, there are three different motives for murder. Characters that have many "anger" resources to another character may choose to murder them over that. Spouses of characters that are cheating may choose to murder their spouse's lover. Greedy characters may choose to murder a rich character to acquire their money. All of these rules require the actor to first acquire a weapon, which delays the inevitable at least for one turn, typically making reason and consequences less obvious in the narrative. Further, we added one rule where characters may be able to kill another character without being in possession of a weapon if they have particularly many "anger" resources to another character, to make the "has_weapon" predicate slightly less of a definitive indicator of a murder.

Other rules to add substance to the narrative include characters attempting to gamble, which may yield debt that motivates them to acquire money. This can either be done by attempting to steal from other characters or even by murdering them. Characters can seduce each other, become lovers or married, or get divorced. Further, to make the origin of "anger" resources that play a central role in the murder motives less obvious, we added the concept of spreading the abstract concept of a lie about another character. In this way, a character X that does not like another character Y can attempt to spread a lie about Y to a character Z. If the successful outcome is chosen, Z now also becomes angry at Y. A more sophisticated system may allow characters to construct specific lies, which would add another layer of interest to the game, as characters would then be able to formulate testimonies which they believe to be true based on information they received from another character, but which the player can then identify as lies.

We added non determinism wherever possible, and in particular to those rules with high rewards for a positive outcome, to ensure that the artificial agents will choose options that may have a high pay off for it, but may also result in dramatic tension, as for example, one character catches another in an attempt to steal from them.

Having only a few motives for murder does, in fact, align with the most common structure of Agatha Christie's novels, which we took as a major influence. Typically, the reason for murder is one of those we listed above, with the depth of the story coming from the elaborate reasoning and the relationships between the characters [18]. As such, while the stories appear very varied in their turn of events, most of the time the final step that motivated the murder can be put into few categories. Putting a focus on delaying the murder for multiple steps is thus important to let agents interact and provide variability to the otherwise few motives.

C. Believable Agents

In the first iteration of the prototype, we used agents that took actions randomly. This resulted in stories that were varied, but were hard to follow, as no higher level intention of the agents became apparent in the story.

To improve the believability of the flow of actions, we added certain rewards to each action that an agent can take. Some actions may incur a penalty, like picking up a weapon which decreases the agent's sanity score, but ultimately may allow it to commit a murder for a reward in fulfillment if the agent has a fitting motive. The agent's actions became a lot more coherent in this way, as they follow the directions as given by the authored reward structures. With an ability to plan ahead for multiple steps, different agents may try to anger characters about others or find different ways to fulfill the preconditions of the rules with the highest rewards.

To enable this sort of planned behavior, we used Monte Carlo Tree Search (MCTS) with upper confidence bounds policy (UCB1) [19]. Since the simulation has no clear terminal state, we let the simulation play out a fixed number of steps every time instead. Considering a murder a terminal state would have been an option, but this might discourage murders from happening, as actors who do not commit a murder get to collect rewards from more steps during playout. An alternative to this model would have been to only determine win/lose per playout, ending a playout with "win" if the character commits a murder and "lose" if they fall victim to one. This, however, would discourage actors from considering other possibilities that would bring smaller rewards and always steer them directly towards murder, resulting in much smaller murder mysteries.

From experimentation, good values for the maximum number of expanded states is around 100, while the number of playout steps is limited to 10. Most of the time, simulations will be shorter than 30 steps, making 10 a good balance between performance and allowing actors a good amount of foresight. The accuracy of the MCTS agents is diminished by the non-deterministic choices. While expanding states, the actor will only get to know a single of those states and assume that this is the outcome. This is because we store the entire state after each expansion, since the computation of valid moves is costly, so subsequent visits in the selection of that node will not make a difference. In this way, agents will sometimes act more pessimistic or optimistic, based on their once observed outcome of a rule. As such, we found that not taking another step of optimization to enable the actors to explore all possible outcomes of actions results in a more believable behavior.

The MCTS agents encountered significant performance problems as they had to evaluate valid actions at the current state 300 times more often, with 30 expansions and 10 playout steps. Using pre-calculated hashes for all lookups into the resource state allowed the simulation to progress reasonably fast, at about a pace that allowed a human to follow the event flow as decisions are made. Evaluating decisions via MCTS is typically a fast option [20]. This is, however, based on the assumption that calculating rewards is the expensive action, while the random playout is negligible in terms of performance. For our simulation, however, each step of the simulation is costly, as the most expensive action is to find all valid rules, which has to happen on every step of the playout.

D. User Interface

The Python library Kivy[6] was used to develop our user interface (UI). This allowed us to directly use the Python objects that were created during the murder mystery generation, without the need for serialization when crossing language boundaries. It was important to represent the state, so that it could easily be understood by players. To achieve this, we relied on images to display our characters, objects, and rules. Specifically, we used emojis as our images. They presented us with a large collection of portraits to pick from for our actors and objects, as well as a simple way to display emotions without needing to write a lot of text. Using images helped

[6]https://kivy.org

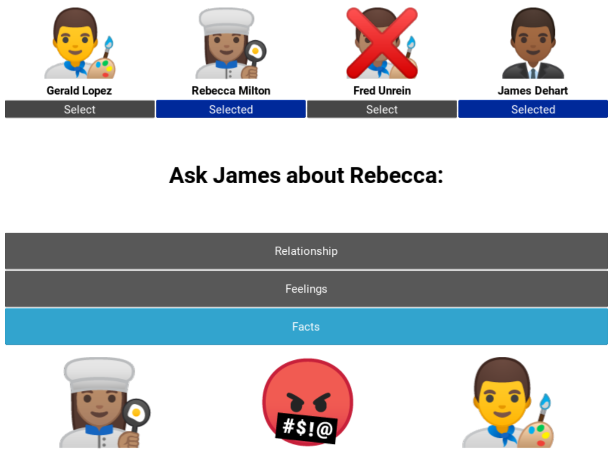

Fig. 4. The player is questioning one of the actors, James, on information she has about another actor, Rebecca. In this case, James knows that Rebecca was involved as an actor in the rule "fight" with another character, Gerald, as the victim.

players to quickly recognize the actors in each action, even if they did not remember the actors' names.

Rules were considerably harder to represent unambiguously using emojis. Some were quite simple, for example, murder, which we represent using a weapon and the two characters involved in the murder, as seen in Figure 4. Others were more complicated, such as the rule "lie to A about B" which is displayed as the "secret" emoji. Unfortunately, we were not able to find a recognizable emoji for all rules, for example paying a debt to someone or stealing from someone, which are both represented using different money symbols as there are no other, more obvious ones, to choose from.

IV. Discussion

This section presents some of the advantages and limitations of the system described earlier. While extensive user testing is yet to be conducted, a few insights have arisen through the development and initial player demonstrations.

A. Linear Logic

While linear logic is more suited to represent real-life situations than classical logic, it showed shortcomings when applied to our domain. In linear logic all rules are deterministic, meaning every rule has exactly one possible outcome. There is no possibility for chance. This initially forced us to have multiple rules to represent possible different resolutions to some rules. One example is the rule steal, which allows an actor to attempt to steal money from another. There were two possibilities when using this rule: the actor could be successful at stealing or he or she could be caught. These possibilities should obviously have a different effect on the state of the simulation. However, this kind of non-determinism was not supported by Ceptre. To remedy this, we extended our implementation of a linear logic evaluator to support multiple outcomes with authored probability as described in Section III-A.

Another drawback of linear logic was that it only knows the resources that exist in the current state. There is no possibility to test for the absence of a given resource. While this did not pose a significant problem, it made the rule's setup more complex, both in terms of authorship and in terms of computation. For many predicates, we now had to introduce an opposite and manually ensure that the resources always appeared mutually exclusive in the resource state.

For example, our system kept track of which characters in the game were related to each other via the predicate related Character Character'. When we introduced a new rule get_married, we wanted to check that the characters getting married were not related to each other. As linear logic does not allow this, we had to introduce a new predicate not_related and make sure to assign exactly one of these two predicated during the generation of the initial state.

B. Authorship and Rule Set

The rules we used in our test runs generated interesting steps of events, but often lacked interest and the agents often fell back to following the same steps of actions to maximize their reward. We think that given more time, authors with more experience in storytelling would be able to use this kind of system to create a rule set that produces more compelling and varied simulations. It may be necessary to add more convenience to the pure linear logic system to lower the barrier of entry and allow keeping track of an increasingly complex system of rules. Further, the resulting outcome may be put into a shape that appears more appealing to players by adding additional flavoring to each rule and dynamically adapting the rendition of sequences of events to the actor, for example by making a more insecure actor's testimony show this character trait when they present the testimony.

When authoring our own rules, we became aware of a responsibility to balance the nature of the rules and the resulting societal norms between what we may think an interesting society should be like and matching what may be the players' expectations, so that they can understand and follow the agent's reasoning. This includes aspects like the preconditions for marriage, what the consequences for stealing or lying should be or the rules for when a murder may happen. This is further amplified by the requirement to attach rewards or penalties to each rule, such that the agents know which actions are desirable and which are better to avoid.

During limited usability tests, players showed excitement about the often surprising and funny rendition of certain actions by the emojis. At the same time, certain emotions, such as "relaxed" and "fear", were too ambiguous so that additional explanations were required. Early on, it became clear that the possession of a knife was too strong of an indicator that someone committed a murder. When multiple characters with knifes appeared, however, the players were most often able to solve the mystery by investigating their relationship to the victim and reasoning of the possible implications. As such, the rules we authored appeared to provide strong enough ties to the player's own experiences for them to comprehend the agent's motivation.

C. Flexibility of The System

To explore the versatility of this system, we ported the rule set to generate simple quest hooks for role-playing games, such as *Dungeons and Dragons*. The outcome is a tool that formulates the flow of events leading up to a moment where the adventurer party should get involved. Instead of stopping at the moment of a murder, we thus use a set of actions that involve hiring the adventurers as stop conditions. For example, in a dispute, a morally flexible adventuring party might be hired to settle the issue in favor of one of the opponents. A paranoid or threatened actor might ask the heroes for protection. Or, a rule we inherited from the murder mystery system, the party may simply be hired to investigate the murder of an actor. The system's output is kept intentionally abstract to let the human author using the tool improvise and fill in details about the story.

Even after investigating different reward models, involving more domain specific characteristics such as physical power or dexterity, we decided to keep using the existing measures, as they proved to be on a level that could easily be mapped to any actor's behavior. We found the generated stories to encourage thinking outside the box and fueling creativity when it came to developing short plot lines. However, the range of possible outcomes was of course limited to the ones we authored beforehand, again only providing points of variation in the circumstances that led to the hiring of the story's heroes.

V. Future Work

While the status of this project currently puts it in a playable state, the final outcome is unfinished. In this section, we will talk about the steps that are necessary before a version of this prototype can be tested by players to a larger extent, such as giving the simulated characters more distinct personalities and the ability to lie, as well as a concept of searching for additional evidence.

A. Enhancing Characters

The decision to use emojis to represent the characters has worked well, making recognition quick and easy. The characters themselves are, however, still rudimentary and lack personality. More options are need for the characters themselves, such as age and occupation, as well as using the already defined traits described earlier more effectively. For example, trusting characters could reveal information more easily, while others may try to keep secrets.

Relationships between characters might also be enhanced. Currently, we only generate whether two characters are related, we do not specify how. This would allow us to introduce more complex rules and allow for more interesting stories, such as rivaling siblings.

B. Enhancing the Investigation Stage

To make the game satisfying to players, they should have the option to uncover hidden motives. At the moment, none of the actors will lie to the player, or even hide information. As such, it is only a matter of dedication for the player to finally find the actor who admits to the murder.

To remedy this situation, we propose a gameplay mechanic, wherein local knowledge exists and certain actors can lie. More specifically, each actor is allowed to omit information that they consider sensitive, but only the murderer is allowed to also falsify statements. For any rule that is taken, uninvolved actors have a chance to witness this action happening. The probability that this happens would be defined per rule, to prevent sensitive information from being witnessed too frequently, such that more realistic testimonies come together. With these mechanics in place, the game's core objective for players will be to identify the one actor whose testimony contains false statements, thereby revealing themselves as the murderer. This is challenged by the fact that actors may omit certain interactions, or that the murderer primarily interacted with the victim and no other actors witnessed these interactions.

With this model, it is, however, questionable if the player will always be able to solve the mystery. It may happen that not enough information from actors is presented to find contradictions and identify the murderer. While this reflects real-life murder mysteries well, it may not lead to satisfying gameplay. As such, it would be desirable to detect how many statements contradict to skip potentially unsolvable mysteries.

Once these points have been realized, rigorous playtesting would be the next step. In particular, the number of characters will provide an interesting point of balance. Having more characters increases the load of information for the player, but also provides them with more points of reference for figuring out contradictions in a testimony and potentially a richer story with more complex character relationships. Since the interactions between actors are the only way the player gets to know them, giving actors the option to interact more will lead to players identifying actors in richer ways, as already shown in ClueGen [8]. An actor may become a lover of particularly many other actors or a family feud may arise.

VI. Conclusion

While the murder mystery generator presented in this paper showed promise with regards for variability in the stories it generated, it is clear that this system fundamentally relies on having a well-thought-out set of rules produced by an author. However, with the system in place, these rules were quick and easy to establish and prototype with. As such, the system would not take the place of authored, linear murder mystery games, but rather provide an option to quickly generate ideas for more elaborate plots or generate puzzles with a small scope that a player should be able to solve within 15 minutes. The explored approaches of combining the linear logic evaluator as implemented by Ceptre with Monte-Carlo-Tree-Search based artificial agents worked well to fulfill this purpose and has opened up an assortment of future research avenues.

References

[1] M. A. Ackershoek, "" the daughters of his manhood": Christie and the golden age of detective fiction," *CONTRIBUTIONS TO THE STUDY OF POPULAR CULTURE*, vol. 62, pp. 119–128, 1997.

[2] S. Van Dine, "The art of the mystery story," *American Magazine*, vol. 108, pp. 189–193, 1928.

[3] R. Knox, *Best Detective Stories of 1928-29*, 1929.

[4] C. Martens, *Ceptre: A Language for Modeling Generative Interactive Systems*. Palo Alto, California: AAAI Press, 2015, pp. 51–57. [Online]. Available: https://www.aaai.org/ocs/index.php/AIIDE/AIIDE15/paper/view/11536

[5] M. Brown, "What makes a good detective game?" 2017. [Online]. Available: https://www.youtube.com/watch?v=gwV_mA2cv_0

[6] H. Goldstein, "L.a. noire review - ign," 2018. [Online]. Available: https://m.au.ign.com/articles/2011/05/16/la-noire-review

[7] C. Petit, "Building a mystery: Gaming's crime investigation mechanics and the beauty of being unsatisfied," 2018. [Online]. Available: https://www.vice.com/en_au/article/5gj9bz/building-a-mystery-crime-investigation-mechanics-and-the-beauty-of-\being-unsatisfied-025

[8] *ClueGen: An Exploration of Procedural Storytelling in the Format of Murder Mystery Games*. Palo Alto, California: Association for the Advancement of Artificial Intelligence (AAAI), 2016. [Online]. Available: https://aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/view/14070

[9] J. McCoy, M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin, "Prom week: Designing past the game/story dilemma," in *Proceedings of the 8th International Conference on the Foundations of Digital Games, FDG 2013, Chania, Crete, Greece, May 14-17, 2013.*, G. N. Yannakakis, E. Aarseth, K. Jørgensen, and J. C. Lester, Eds. Society for the Advancement of the Science of Digital Games, 2013, pp. 94–101. [Online]. Available: http://www.fdg2013.org/program/papers/paper13_mccoy_etal.pdf

[10] J. McCoy, M. Treanor, B. Samuel, N. Wardrip-Fruin, and M. Mateas, "Comme il faut: A system for authoring playable social models," in *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'11. Menlo Park, California: AAAI Press, 2011, pp. 158–163. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014589.3014617

[11] M. Mateas and A. Stern, "Façade: An experiment in building a fully-realized interactive drama," 2003.

[12] J. Rowe, B. Mott, and J. Lester, "Optimizing player experience in interactive narrative planning: A modular reinforcement learning approach," in *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'14. Palo Alto, California: AAAI Press, 2014. [Online]. Available: https://www.aaai.org/ocs/index.php/AIIDE/AIIDE14/paper/view/9008

[13] J.-Y. Girard, *Linear Logic: its syntax and semantics*, ser. London Mathematical Society Lecture Note Series. Cambridge: Cambridge University Press, 1995, pp. 1–42.

[14] R. Di Cosmo and D. Miller, "Linear logic," in *The Stanford Encyclopedia of Philosophy*, winter 2016 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2016.

[15] A. H. Maslow, "A theory of human motivation." *Psychological Review*, vol. 50, no. 4, pp. 370–396, 1943.

[16] S. Rothmann and E. P. Coetzer, "The big five personality dimensions and job performance," *SA Journal of Industrial Psychology*, vol. 29, no. 1, 2003. [Online]. Available: https://sajip.co.za/index.php/sajip/article/view/88

[17] R. Plutchik, "The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice," *American Scientist*, vol. 89, no. 4, pp. 344–350, 2001. [Online]. Available: http://www.jstor.org/stable/27857503

[18] A. Verbogen and K. Macdonald, "Agatha christie and her murderers: a case study of the miss marple novels," Master's thesis, Universiteit Gent, 2008.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, May 2002. [Online]. Available: https://doi.org/10.1023/A:1013689704352

[20] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez Liebana, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," vol. 4:1, pp. 1–43, 03 2012.