

University of Kentucky

UKnowledge

Theses and Dissertations--Computer Science

Computer Science


2022

Don't Give Me That Story! -- A Human-Centered Framework for Usable Narrative Planning

Rachelyn Farrell

University of Kentucky, rac7hel@gmail.com

Author ORCID Identifier:

 <https://orcid.org/0000-0002-2106-7170>

Digital Object Identifier: <https://doi.org/10.13023/etd.2022.440>

[Right click to open a feedback form in a new tab to let us know how this document benefits you.](#)

Recommended Citation

Farrell, Rachelyn, "Don't Give Me That Story! -- A Human-Centered Framework for Usable Narrative Planning" (2022). *Theses and Dissertations--Computer Science*. 124.
https://uknowledge.uky.edu/cs_etds/124

This Doctoral Dissertation is brought to you for free and open access by the Computer Science at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Computer Science by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@sv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Rachelyn Farrell, Student

Dr. Stephen G. Ware, Major Professor

Dr. Simone Silvestri, Director of Graduate Studies

Don't give me that story! – A human-centered framework for usable narrative
planning

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Rachelyn Farrell
Lexington, Kentucky

Director: Dr. Stephen G. Ware, Professor of Computer Science
Lexington, Kentucky 2022

Copyright© Rachelyn Farrell 2022

ABSTRACT OF DISSERTATION

Don't give me that story! – A human-centered framework for usable narrative planning

Interactive or branching stories are engaging and can be embedded into digital systems for a variety of purposes, but their size and complexity makes it difficult and time-consuming for humans to author them. Narrative planning algorithms can automatically generate large branching stories with guaranteed causal consistency, using a hand-authored library of story content pieces. The usability of such a system depends on both the quality of the narrative model upon which it is built and the ability of the user to create the story content library.

Current narrative planning algorithms use either a limited or no model of character belief, which typically leads to undesirable stories and difficult domain authoring challenges. Domain authoring is further complicated by a lack of intelligent tools for summarizing the content that a domain can produce so that its author can effectively evaluate it. In this work I extend a prior narrative planning framework to model deeply nested character beliefs, thus avoiding common character omniscience problems without overburdening the domain author. Human subjects evaluations demonstrate that the belief model tracks nested beliefs correctly, and that it improves overall character believability in solution spaces over previous models. This model makes domain authoring more intuitive, but also adds complexity to the story generation algorithm, making the planner's output even harder for the author to predict.

As a step toward more intelligent domain authoring tools, I present a novel method for measuring story similarity by encoding important story information into a fixed-length numeric vector. This enables automatic clustering of stories based on their semantic similarity, facilitating high-level communication of large story spaces. I compare the story similarity metric to assessments made by humans, and find the metric to be highly accurate in judging how similar two stories are to each other. I then demonstrate its use in clustering solution spaces, and evaluate two strategies for summarizing the content of the resulting clusters. I find both techniques to be more effective than a control in communicating large story spaces to humans. These contributions together advance the usability of narrative planning algorithms by

improving their underlying narrative model and providing a basis for more intelligent domain authoring tools.

KEYWORDS: narrative planning, artificial intelligence, interactive narrative, belief, story similarity, authoring tools

Author's signature: Rachelyn Farrell

Date: December 14, 2022

Don't give me that story! – A human-centered framework for usable narrative
planning

By
Rachelyn Farrell

Director of Dissertation: Dr. Stephen G. Ware

Director of Graduate Studies: Dr. Simone Silvestri

Date: December 14, 2022

For my kids, Kaius and Lyra –
so that we may afford bigger, more expensive toys.

ACKNOWLEDGMENTS

I would like to express my sincerest appreciation to the National Science Foundation for sponsoring this research, and to the members of my defense committee for their guidance and expertise throughout this process.

No aspect of this work would have been possible without Dr. Stephen G. Ware, my advisor and mentor, who taught me everything I know about artificial intelligence, research, and many other things. Leading by example, he taught me to appreciate science, to ask intelligent questions, and to take the time to do things the right way. As a novice young story writer, I lacked the tools to create the kinds of stories I wanted, but Dr. Ware made me believe that this was no obstacle I could not overcome. I am profoundly grateful for the impact he has had on my life, my work in general, and of course this work in particular.

Many thanks are also in order for the contributions made to this work by fellow members of the Narrative Intelligence Lab. Dr. Alireza Shirvani helped design and evaluate the belief model presented in Chapter 2. Cory Siler contributed to the Sabre implementation of the belief model. Mira Fisher helped evaluate the story distance metric presented in Chapter 3. Lastly, Edward T. Garcia collaborated with me on a very early version of the solution space summarization project, which greatly shaped my understanding of the problem. I thank all of them for their friendship as well as their contributions, direct and indirect.

I am also extremely grateful to Dr. Lewis Baker, whose expert statistical advice and analysis has benefitted my work at many stages. Dr. Ben Samuel also lent his unique and valuable perspective during our group meetings at the University of New Orleans. I thank him especially for his humor and good nature, which never failed to bring me joy. I also thank the many other great minds who contributed to

my knowledge and understanding of these topics through conversations at academic conferences.

I have had the pleasure of learning from many outstanding computer science teachers over the years. I thank them all for their wisdom, most memorably the words of Dr. Pamela Lawhead, who taught us that the way to solve any computer science problem is to “Go sit under a tree, and be a person who has this problem.” Her advice resonated and stuck with me because it succinctly expresses what I love most about computer science—that the answers are not written down anywhere because no one has figured them out yet. I thank all those who helped me to see challenges as exciting opportunities for creativity and growth.

I thank my family for encouraging me to chase after my goals, and believing I could achieve them. Finally, no one has had greater impact on me than my husband, Chris Farrell. I am eternally grateful to him for supporting me in every way imaginable, and for being an endless well of intelligent conversation, without whom I would be much, much stupider.

TABLE OF CONTENTS

Acknowledgments	iii
Table of Contents	v
List of Figures	vi
List of Tables	vii
Chapter 1 Introduction	1
Chapter 2 Belief Model	6
2.1 Related Work	6
2.2 Blackbeard Domain	10
2.3 Problem Definition	12
2.4 Search	21
2.5 Evaluations	22
2.6 Limitations	25
Chapter 3 Summarizing Solution Spaces	27
3.1 Related Work	27
3.2 Grammalot Domain	32
3.3 Saliency Vectors	32
3.4 Saliency Distance	38
3.5 Distance Metric Evaluation	39
3.6 Clustering and Visualization	45
3.7 Evaluation of Solution Space Summaries	51
Chapter 4 Conclusion	63
Appendices	66
Appendix A: Accuracy for 18 Saliency Distance Variations	66
Appendix B: Best Performing Weights	67
Appendix C: Stories used for Domain Authoring Study	69
Appendix D: Qualitative Survey	70
Bibliography	72
Vita	80

LIST OF FIGURES

2.1	Example <i>Blackbeard</i> solution in state space	11
2.2	Six types of character plans	23
3.1	A depiction of the initial state in <i>Grammalot</i>	32
3.2	Distance metric accuracy	42
3.3	Hierarchical clustering of the 58 <i>Grammalot</i> solutions	46
3.4	Example cluster summary for <i>Grammalot</i> problem #193	48
3.5	Example tree summary for <i>Grammalot</i> (problem 193)	51
3.6	Depiction of the <i>Grammalot</i> initial state without the crossroads	52
3.7	Storytelling robot	54
3.8	Story display box	55
3.9	Number of stories having each number of configuration dependencies	57
3.10	Engagement factors	58
3.11	Participant accuracy	59
3.12	Confidence ratings	60
3.13	Perceived understanding of the story space (1=low, 5=high)	61
4.1	Accuracy results for 18 variations of Saliency Distance	66

LIST OF TABLES

2.1	Results after filter	25
3.1	Example stories X and Y	37
3.2	Salience vectors after each step in story X	37
3.3	Salience vectors after each step in story Y	38
3.4	<i>Grammalot</i> domain configurations	53
3.5	Number of solutions per problem	53
3.6	Configuration dependencies for each story question	57
4.1	Weights scoring 37 on the first analysis	67
4.2	Weights scoring 411 on the second analysis (both scored 36 on the first) .	68

Chapter 1 Introduction

Recent technology has found many applications for embedding narrative elements into human-facing systems in entertainment, education, training, therapy, journalism, and a variety of other fields. Digital interactive narratives can represent many possible stories as a branching tree structure, and allow the interactions of the user to affect which narrative trajectory they ultimately experience. Incorporating branching narratives into interactive systems can improve their overall effectiveness by increasing user engagement. However there is a major authoring bottleneck that limits how complex interactive narratives can be: Branching stories are fundamentally difficult to author because the more interaction points they contain, the more complex the story becomes, and the more difficult it becomes for the author to keep track of all the different branches.

Authors commonly deal with this problem by manually limiting the branching factor of the story—railroading the diverging branches back towards a central, linear storyline, or a small few of them. This is not an ideal solution, however, because it can leave readers longing for more interactivity, for a larger story with more possible endings, and more ways for their actions to matter or have an impact on the story. Since interactivity leads to some benefits already, it stands to reason that more benefits could be obtained by allowing more interactivity. One of the goals of story generation research is to enable authors to create larger branching stories by automatically combining smaller hand-authored elements into many different coherent stories. This changes the authoring task to that of creating reusable pieces of story content, rather than having to manually construct every single possible story.

Planning-based narrative generation systems, or narrative planners, can automate plot generation at a low level by modeling individual properties of the world and how they are changed by different events. A planning system can generate multiple branching paths from the beginning of a story through to multiple ends, in which the underlying properties and logic of the world remain consistent while the content of the individual stories varies by branch. A formal plan-based representation of story permits a great deal of authorial control for system designers and narrative authors. An author creates a planning domain, which defines everything that can be true in the story world, all events that can occur, when they can occur and what happens when they do. They can also specify constraints to define the kinds of stories to be generated, such as how the story should end (author goals) and key events that must happen at some point in the story (landmark events). System designers can embed models of specific narrative features (like conflict, suspense, character models, etc.) to further constrain stories in a variety of ways, using fine-grained information about the story world.

The theoretical benefits of the planning approach to story generation are enticing, but current systems suffer from a lack of usability for at least three reasons. First, usability requires the underlying narrative model to reliably produce acceptable story structures, but current models are imperfect and often allow stories that do not meet

human expectations of storiness. Second, humans must be able to effectively construct planning domains that model the kinds of stories they are intending to create; but this requires manually evaluating the generated story spaces, which are typically too large to read in full. These goals are interrelated: Insufficient narrative models leave domain authors responsible for handling complex problems; yet as narrative models grow in complexity to account for more nuances, it becomes even harder for domain authors to predict and control their output. Third, the planner also needs to be fast enough for practical use. The general problem of planning speed and efficiency is outside the present scope, but this work makes contributions toward the first two goals and discusses their implications for the third.

Much of narrative planning research has aimed to define a model of narrative that reliably produces acceptable stories. As readers process the events of a story they take on the minds of the characters and attempt to understand their motivations and predict their future actions (Gerrig, 1993; Frith and Frith, 2001). When readers cannot make sense of a character’s behavior, they may disengage with the story or have an unfavorable reaction. This has led to models of character intentionality that constrain characters’ actions based on their individual goals, producing more acceptable stories by ensuring that characters only act in ways that might help them. Intentionality models prevent characters from acting against their interests or for no reason, but they also reveal a more subtle type of unwanted behavior related to character belief, which is the subject of Chapter 2.

Intentional narrative planners track individual character goals and ensure that, while the story achieves the author’s goal, characters can only take actions that are part of a plan to achieve their own goals. However, if such a system does not model individual character knowledge, then its characters may appear omniscient. This problem is often described in terms of the Belief Desire Intention model of agency (Georgeff *et al.*, 1998), which states that an entity’s behavior is considered believable (i.e. the audience will attribute agency to that entity) when it appears to be motivated by its own beliefs, desires, and intentions. Conversely, when an entity—even one that is motivated by its own goals—does not appear to possess its own set of beliefs, it is seen as less believable. If the underlying narrative model treats characters as omniscient, the system will likely allow many stories that humans deem unacceptable, unless the author manually encodes complex belief mechanics into the domain.

As an example, consider the following sequence, which might be generated by existing intentional narrative planners:

Ann is at her desk and wants coffee, so she goes to a nearby coffee shop. While she is there, her friend Bob delivers coffee to her desk. Ann leaves the coffee shop without buying anything, goes back to her desk, and drinks the coffee.

This is an unwanted sequence because it demonstrates Ann’s omniscience: how does she know that there is coffee at her desk if she was away when it arrived?¹

¹Of course, stories often contain gaps like these, and humans can readily infer unnarrated events to form coherent sequences (e.g. perhaps Ann received a phone call from Bob, informing her that

An intentional narrative planner can justify Ann leaving the coffee shop at that moment because it can help her achieve her goal to have coffee: the plan “walk back to my desk, drink coffee” is *possible* and it satisfies her goal. Yet this plan defies the audience’s expectation of limited belief: Ann should not have known there was coffee at her desk because she was away when it arrived. According to the reader, the plan “walk to my desk, drink coffee” does not make sense for Ann at this point because Ann should not *believe* it is possible, even though it is.

This shows why intentional planners should take into account characters’ beliefs when deciding what actions make sense for them. If they do not, it falls to the domain author to manually encode beliefs and observation rules into the domain, which can quickly become intractable. To demonstrate this process, here is one way we might fix the above example.

1. Add a property to the domain like *believes_coffee(?character, ?place)*, representing whether or not a character believes there is coffee at a place. Update the initial state to include any instances of this property that are true at the beginning, such as *believes_coffee(Ann, CoffeeShop)* — since Ann initially believes there is coffee at the coffee shop.
2. Using the *believes_coffee* property above, add to the effects of the action for delivering coffee: Any character currently located at the place where coffee is being delivered subsequently believes there is coffee at that place.
3. Add to the preconditions of the action for drinking coffee: The character drinking coffee must believe there is coffee at that place in order to drink it.

These adjustments prevent the problematic example: Characters can no longer make plans to drink coffee that was delivered while they were away. It is now impossible to drink coffee without believing there is coffee at that place, and the only way to acquire that belief is to be present when it is delivered there.

Of course, this only avoids one specific situation; if the author is concerned with the whole solution space, more work needs to be done to avoid similar problematic scenarios. For example, one would also want to add effects similar to those of step 2 to any other action that moves coffee, such as buying coffee from the coffee shop or walking from place to place while having coffee. Any other domain action that involves coffee (e.g. microwaving coffee) should be constrained as in step 3 so that characters cannot plan to do these things unless they believe the coffee is there. Also, no nested beliefs have yet been modeled, such as whether Bob believes that Ann believes there is coffee at a place, which may cause unexpected problems. Finally, only a single belief property has been modeled, that of the presence of coffee at a place. If the domain is to contain anything beyond this type of scenario, beliefs about many other properties will need to be modeled. The task becomes increasingly complex and difficult as the domain grows.

there is coffee at her desk); but this is undesirable because it introduces inconsistencies between the reader’s mental model of the story and the system’s model.

Another option for domain authors is to “hack” the domain to avoid revealing character omniscience, by which I mean using properties that do not actually reflect the intended mechanics of the world. For example, without modeling any beliefs, the unwanted example can be prevented by making *having coffee* a precondition of *leaving the coffee shop*. These are not the intended mechanics of the story world—that it should be physically impossible to leave the coffee shop without coffee—but it will prevent the unwanted story since the plan “walk to my desk, drink coffee” is impossible once Ann has reached the coffee shop, even after Bob has delivered coffee to her desk. However, the story is prevented for the wrong reason: Ann’s action should not be impossible, it should only be unreasonable for Ann. This type of domain modification is still tedious (a complete domain would require many such hacks), and is also likely to have unforeseen consequences in other story branches.

A better approach is for the underlying planning model to explicitly track individual character beliefs, automatically update them based on whether or not characters observe events, and use them when testing whether a plan makes sense for a given character. Chapter 2 presents a planning model that does this, tracking separate beliefs for each character including arbitrarily nested theory of mind (their beliefs about others’ beliefs, etc.). Using this model, characters can only act in ways they *believe* can contribute to their goal, which eliminates the omniscience problem and improves character believability over previous models. Furthermore the model allows characters to automatically interact with each other in more complex ways, such as anticipating each other’s actions, surprising each other, influencing, plotting against, and deceiving each other.

As mentioned previously, usability requires generated stories to be acceptable not just according to general narrative properties like causal coherence and character believability, but also according to the specific design goals of the author. Verifying this requires the author to observe the output of the narrative planning process, called the solution space, but this is fundamentally challenging because solution spaces can be so large. Reading all the stories is a slow process and is mentally taxing because they tend to be very similar to each other. Even for small domains it can be impractical to read all the stories, especially if doing so multiple times (e.g. after each domain modification). There is a need for tools that can intelligently help authors evaluate and debug solution spaces more efficiently.

The need for authoring tools is further underscored by the belief model presented in Chapter 2. While the model makes authoring easier in one sense, by removing the need to keep up with belief information manually or use hacks to avoid omniscient behavior, it also further complicates authoring because it makes observing and debugging solution spaces even more difficult. Adding belief makes solution spaces larger and harder to imagine, with more logic behind the scenes determining whether stories are acceptable or not. New features like this introduce new kinds of mistakes authors can make that propagate throughout the solution space and become difficult problems to debug. This is not limited to the specific belief model presented here, but rather is likely to be the case for any future improvements to models of character and other narrative elements. It is therefore important to address the authoring challenge, not only because it serves the ultimate goal of usability, but also to aid

ongoing research and development of these models.

Clustering algorithms are a logical starting point for the development of tools that help authors better explore and evaluate solution spaces. A tool that identifies groups of semantically similar stories could allow authors to get a high-level view of the full space just by reading a small number of stories (e.g. one from each cluster). It could also serve as a road map to explore specific groups of stories in more detail, and could help draw attention to outliers, which may be useful for identifying particularly interesting stories as well as for catching stories that are unwanted. Clustering solution spaces requires a meaningful measurement of the similarity between two stories, which is an open research topic. Existing plan comparison models can easily identify structural differences between two story plans, but these tend not to agree with how humans assess story similarity.

I address this need in Chapter 3 by presenting a novel method for measuring the similarity between two narrative plans that is based on human perception. The method frames story comparison as a comparison of human memories of stories; it encodes story information analogously to how the mind stores them in memory according to one psychological model. These encodings can then be compared using standard vector distance functions for a similarity measurement. I define the similarity model formally using the narrative planning framework defined in Chapter 2, as well as generically so that it can be more readily adapted to other story generation systems. I present an evaluation that demonstrates the model’s accuracy according to humans in an example domain, compared to several alternative approaches for measuring story similarity. I then apply the metric to clustering solution spaces, and propose two techniques for summarizing the resulting clusters. In a final evaluation, I find both techniques to be helpful for communicating story spaces to humans in a simulated domain authoring scenario.

To be usable at a large scale, narrative planning algorithms will need to run faster than they can presently; this is a high priority for future work. This work makes two significant contributions toward other current hindrances to usability. The first is a knowledge representation and planning framework that improves character believability in generated stories by ensuring that characters are limited to their own individual beliefs. Assuming a correctly authored domain, this causes solution spaces to contain more acceptable stories and fewer unacceptable stories, making the narrative model more effective in general. The second contribution is a model of story similarity that encodes important story information as vectors that can be compared using standard distance calculations. This enables meaningful clustering of solution spaces which can help domain authors better understand and debug the content they are creating. Together this work supports a narrative planning model that is more usable because its input can be authored more intuitively and its output can be more easily evaluated.

Chapter 2 Belief Model

To solve the omniscience problem described in the introduction, this chapter presents a narrative planning framework that extends a previous model to account for character beliefs. Modeling belief formally is challenging because people can possess not only their own beliefs, but also beliefs about each other’s beliefs. This leads to an infinite set of beliefs, even with only two agents (*I believe you believe I believe you believe...*). Limited belief models have been proposed for narrative planning that represent only first-level character beliefs, but many basic types of character interactions like deception, cooperation, and influence require at least two nested belief levels. For instance, to model Ann deceiving Bob, there must be a difference between what Ann believes to be true (1 level) and what Ann believes Bob believes to be true (2 levels). Furthermore, conveying deeply nested mental states (having 3 or more nested levels) either implicitly or explicitly is so common in fiction that it may be impossible to write fiction without doing so (Zunshine, 2022). Deeply nested belief states are therefore considered important for storytelling and thus the narrative model should not impose a limit on theory of mind depth, even though this incurs a computational cost.

This chapter describes a belief model for centralized narrative planning systems that features unlimited theory of mind for characters. The model is formally defined in Sections 2.3-2.4 and details are given for an implementation of this planner called Sabre. Sabre supports all the features defined by the Action Description Language (ADL; Pednault, 1989), including typed variables and equality, negated literals, quantified and disjunctive goals, and conditional effects. Sabre is the first narrative planner to support all of these features combined with intention, deep theory of mind, and numeric fluents.¹

Two human subjects evaluations demonstrate that the belief model accurately tracks nested beliefs, and that it improves character believability in solution spaces compared to previous intentionality models. Section 2.6 discusses the model’s limitations, including challenges for domain authoring that further motivate the work presented in Chapter 3.

2.1 Related Work

Narrative Planning

Early work in story generation explored both a character-centric model that simulates stories using autonomous character agents (Meehan, 1977) and an author-centric model that uses a global “author” agent who controls characters like a puppet master (Dehn, 1981). The two highlight a fundamental trade-off between character believability and authorial control: Using autonomous characters, a system cannot

¹For benchmark results and more details about this implementation I refer the reader to (Ware and Siler, 2021).

fully control the trajectory and outcome of the story, though its characters will appear believable. Conversely, with a centralized storytelling agent, the system fully controls the story but its characters act in ways that do not appear believable to readers. Riedl and Bulitko (2013) survey story generation systems and classify them according to how they approach this trade-off. There are also data-driven approaches to story generation (McIntyre and Lapata, 2009; Li *et al.*, 2013), including recent applications of deep learning (Finlayson, 2017; Martin *et al.*, 2018; Yao *et al.*, 2019), which generate stories using large narrative datasets and do not explicitly model either character or author agents. This work, and narrative planning in general, takes the author-centric approach, prioritizing control over the story and having to account for character believability in other ways.

Kybartas and Bidarra (2016) note that story generation is always in some way mixed-initiative. They classify story generation systems by what components of narrative are being generated, and to what extent other components are manually authored. In the context of this survey, the present work is concerned with generating plots (sequences of events), as opposed to what Kybartas and Bidarra refer to as “space” (the entities and properties of the story world).² Narrative planning approaches use planning algorithms to automate the generation of plot sequences, relying on manually authored domain content including action templates, characters, entities, and setting information.

UNIVERSE (Lebowitz, 1985) is an early plan-based system that generates soap opera style plots from a library of manually authored plot fragments. The plot fragments themselves supply most of the melodramatic intrigue; each one represents an interesting series of events involving a number of characters. For example, in one fragment a character is threatened by their spouse’s domineering parent, causing the marriage to end and the spouse to begin a new relationship, among other effects. UNIVERSE is an author-driven system; its characters have no agency but instead have traits and other properties that the planning system uses to determine which plot fragments can apply and how. The system can continue storytelling after each fragment is executed by selecting new high-level author goals to pursue, such as to increase tension between two characters, have a character begin a new relationship, etc.

The character believability issue is largely avoided in the UNIVERSE model; the system relies on the manually authored content to adequately constrain plot fragments so that they can only apply in believable contexts. For example, the parent role in the fragment described above must be filled by a character who is not very nice. In this way the system produces stories that are generally consistent with characters’ personalities and current situations (if the author has modeled the fragments appropriately), but believability is far from guaranteed. UNIVERSE’s high-level event representation allows a great deal of authorial intent—in the sense

²Here the term “space” refers to the entities or “existents” of the story world, their properties, and their initial configuration, similar to Ryan (2012)’s definition of story space. Elsewhere in this document I use the term “solution space” in the planning sense, to refer to the set of solutions generated by a planner. Additionally, I use “space” in the psychological sense to refer to the spatial dimension of events.

that the human author can write content and get the kinds of stories they expect—but it precludes the system from doing any structural reasoning about the narrative. Furthermore it limits the system’s variety and capacity for the kinds of emergent behavior that are possible when plot is modeled at a lower level.

Young (1999) argued for the use of partial-order planning to generate plot, in part because its explicit representation of individual causal links between goals, preconditions, and effects allows it to model complex causal and hierarchical relationships between narrative events. Research has since evolved that incorporates planning and narrative in a variety of ways, e.g. integrating planning algorithms into story generation and interactive narrative systems (Cavazza *et al.*, 2002; Porteous *et al.*, 2010); and embedding models of narrative directly into the knowledge representation and search of the planner to create planning systems that automatically reason about universal narrative concepts like intentionality (Riedl and Young, 2010; Teutenberg and Porteous, 2013), conflict (Ware *et al.*, 2014), and suspense (Bae and Young, 2008; Cheong and Young, 2015). For a survey of planning-based approaches to narrative generation and interactivity, see Young *et al.* (2013).

The present work adopts methodology from previous planners that have embedded models of intentionality into the planner’s knowledge representation, adding to this representation a theory of mind for characters. It builds on the contributions of several planners beginning with the Intent-based Partial Order Causal Link (IPOCL) planner (Riedl and Young, 2010). IPOCL is a centralized planner that uses character goals to constrain the plan specification so that characters appear to be acting in pursuit of their own goals. The characters are not autonomous; they cannot independently pursue their goals. Rather, the author is the only agent and must search for a plan that both achieves its goal and does not violate certain constraints—namely, that each step in the plan is consistent with the motivations of the characters who take the action. IPOCL ensures this by tracking “intention frames”, plan-like structures that represent causally connected steps a character takes to achieve one or more of their goals. A character action is permitted only if that action exists within an intention frame for that character. Intention frames effectively “explain” why the characters take each of their actions.

Ware *et al.* (2014) extended IPOCL to allow failed character plans and conflict between characters in the Conflict Partial Order Causal Link (CPOCL) planner. CPOCL preserves intention frames that cannot be fully executed, e.g. due to conflicting actions from other characters. In other words, the existence of a plan to achieve the character’s goal is sufficient to justify the character carrying out actions from that plan, even if the rest of the plan does not actually happen in the story. Refining this notion of the search space as “possible worlds” that can be imagined by the characters and used to justify their actions, Ware and Young (2014) introduced Glaive, a state-space implementation of the CPOCL model. Glaive utilized advancements in state-space planning (Hoffmann and Nebel, 2001) to achieve significant improvements in speed and efficiency compared to its plan-space predecessors. The present work adopts Glaive’s state-space approach and its model of intentionality, where character actions can be explained by their existence within some suitable plan for that character, even if that plan is not part of the solution.

The addition of belief to this model constrains the planner to accept these character plans (henceforth called explanations) only if the character believes that they are possible and will achieve their goals.

Belief

Theory of mind has been modeled in character-centric narrative systems, where characters are agents with true partial observability. The Thespian framework (Si and Marsella, 2014) treats theory of mind as central to interactive narratives. Talk of the Town (Ryan *et al.*, 2015) models narrative characters that observe, misremember, and lie. These and others like them have robust character models, but leverage little or no centralized planning to coordinate the story. Centralized planning algorithms that are not designed for narrative generation have modeled agent intentions and beliefs for real world situations (Pollack, 1986), but they make assumptions that are unhelpful for narrative problems, e.g. that agents always cooperate (Grosz and Kraus, 1996), or always compete (De Rosis *et al.*, 2003), or that ignorance is always bad (Bolander and Andersen, 2011). In these systems, agents are designed to cope with an actual lack of knowledge—seen as a challenge to be overcome—rather than model it on purpose to tell interesting stories.

Several narrative planners have focused on character knowledge, partial observability, and wrong beliefs, but they tend to limit theory of mind depth. For example, Virtual Storyteller (Brinke *et al.*, 2014) and HeadSpace (Sanghrajka *et al.*, 2022) use a 1-layer theory of mind, meaning they represent what is true, and what each character believes is true, but stop short of representing what each character believes each other character believes is true. Some, like IMPRACTical (Teutenberg and Porteous, 2013) use a 1-layer model, but defer to a shared state for 2-or-more-layer beliefs. Instead of modeling what each character believes each other character believes, they use a single state representing “what everyone assumes everyone else believes”, which can be different from reality or any particular character’s beliefs. There are also narrative planners that include microtheories of character or audience beliefs that are used to achieve specific narrative qualities, such as suspense (Cheong and Young, 2015), ideal story structure (Robertson and Young, 2015), and deception (Christian and Young, 2004), but these are not intended to be general solutions to the belief problem.

Ostari (Mohr *et al.*, 2018) is an author-centric narrative planning system designed to allow agents to play games with asymmetric information. Like the present work, it models intentional agents with unlimited theory of mind, but Ostari also models uncertainty. Using dynamic epistemic logic, it represents all doxastically accessible possible worlds—all the worlds each agent believes could be the real world at a given moment. Ostari can be used for story generation, but the high cost of modeling uncertainty limits the scope of narrative problems it can solve. In the present work, character beliefs can be nested to any depth, but characters are committed to specific beliefs rather than considering multiple worlds to be possible at once. This trade-off allows the system to solve larger narrative problems while not significantly limiting the kinds of character interactions it can generate.

2.2 Blackbeard Domain

For a running example I introduce a toy narrative planning domain called *Blackbeard*. It includes three characters: James, Marley, and the pirate Blackbeard; and three locations: Tortuga, Skull Island, and Port Royal. Characters can sail from one location to another if they have a ship, or can go along with someone who has a ship. James and Marley each want to have the treasure for themselves. Blackbeard prefers for the treasure to be buried, but would accept having it himself over it being anywhere else. If the treasure is buried, it can only be dug up by Blackbeard or by someone who has a map. After someone digs up the treasure, anyone at that location can take it. Characters can also tell each other if they have a map.

Figure 2.1: Example *Blackbeard* solution in state space

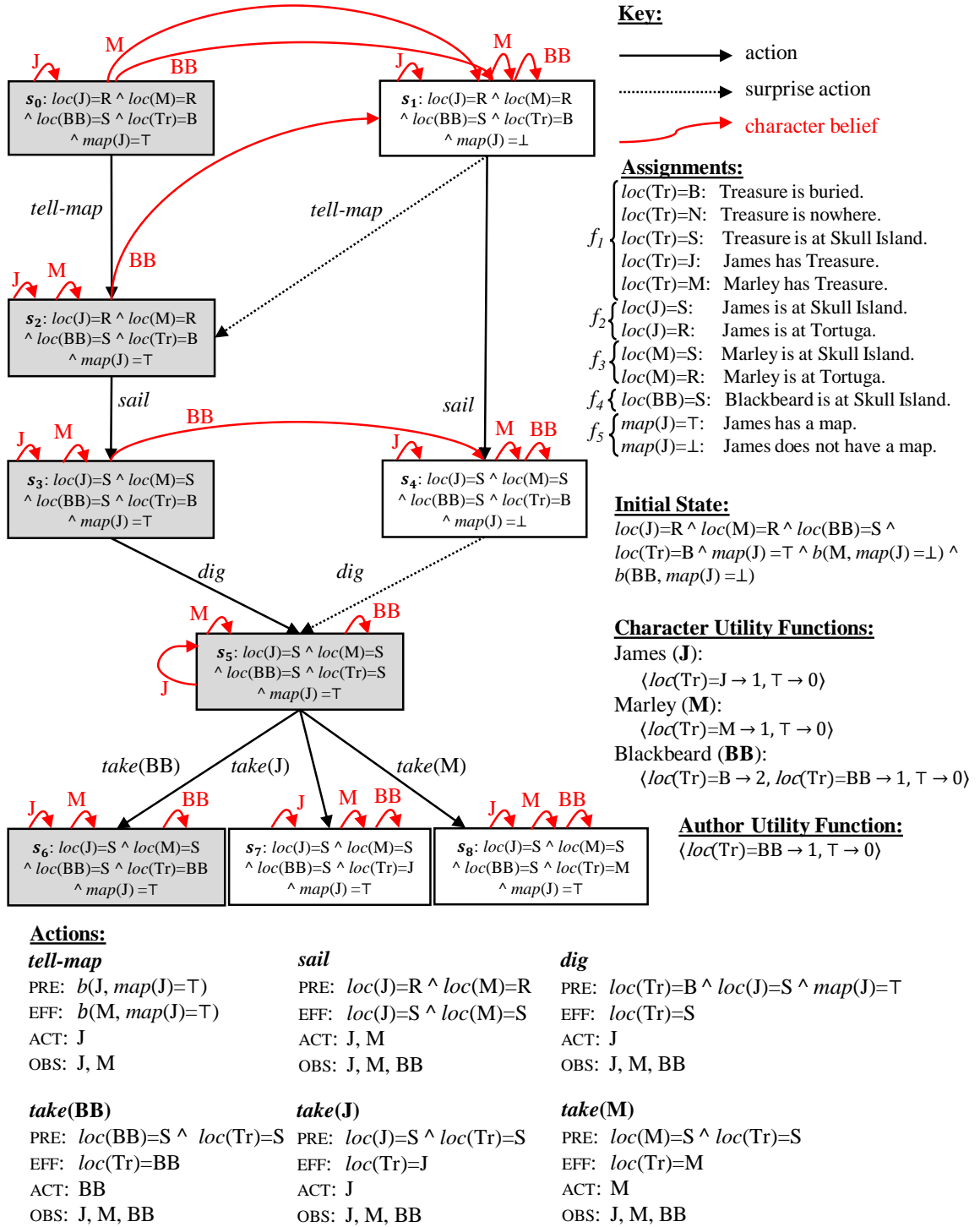


Figure 2.1 shows part of the state space for a *Blackbeard* problem in which James and Marley begin at Tortuga, and Blackbeard begins at Skull Island, where the treasure is buried. Marley and Blackbeard each have a ship, and James has a map but Marley and Blackbeard do not know this. The author's goal, i.e. the way the

story should end, is for Blackbeard to have the treasure. In this image, red curved edges represent the beliefs of a character in a given state, so in the initial state s_0 , James (J) has no wrong beliefs, while Marley (M) and Blackbeard (BB) both believe that s_1 is the real state. These can be interpreted as infinitely nested beliefs by following each belief edge in sequence. For example, in s_0 , Marley believes that James believes that Marley believes s_1 is the real state: $s_0 \xrightarrow{M} s_1 \xrightarrow{J} s_1 \xrightarrow{M} s_1$.

The example solution is four steps long and comprises the five states highlighted in gray in the left column of Figure 2.1. The actions *tell-map*, *sail*, *dig*, and *take(BB)* represent the story where James tells Marley that he has a map to the treasure so that she will agree to sail with him to Skull Island; They sail to the island, James digs up the treasure, and then Blackbeard takes it. The other states in the figure are those that the planner needs to consider in order to verify that this sequence is indeed a solution. They represent the beliefs of characters and the plans that explain characters' actions in this sequence.

2.3 Problem Definition

Characters and Fluents

A problem defines a finite number of *characters*, special entities which should appear to have beliefs and intentions. The term “character” is used rather than “agent” because the narrative planner is the only decision maker, though it creates the appearance that each character is an agent. James (J), Marley (M), and Blackbeard (BB) are the characters in the example domain.

A problem defines some number of state *fluents*, properties whose values can change over time. For some fluent f , let D_f denote the domain of f , or the set of possible values f can take on. Our implementation (Sabre) supports two kinds of fluents: nominal and numeric. For nominal fluents, D_f is a finite set of possible nominal values. For numeric fluents, $D_f = \mathbb{R}$. Numeric fluents are denoted as $f_{\mathbb{R}}$.

Seven types of logical literals l are supported. The first six are given by this grammar:

$$\begin{aligned} l &:= f = v \mid f \neq v \mid f_{\mathbb{R}} > n \mid f_{\mathbb{R}} \geq n \mid f_{\mathbb{R}} < n \mid f_{\mathbb{R}} \leq n \\ v &:= \text{any value in } D_f \\ n &:= v \mid f_{\mathbb{R}} \mid n + n \mid n - n \mid n \cdot n \mid n \div n \end{aligned}$$

These six kinds of literals have a fluent f on the left, a relation ($=$, \neq , $>$ etc.) in the middle, and a value on the right. For nominal literals, a value is one of D_f . For numeric literals, a value is a real number, a numeric fluent, or an arithmetic expression. Marley's location is an example of a nominal fluent whose domain is Tortuga, Skull Island, or Port Royal. The *Blackbeard* planning domain contains no numeric fluents, but for example a character's wealth could be represented numerically.

The seventh kind of literal takes the form *believes*(c, l), which we abbreviate $b(c, l)$, and which means that character c believes literal l is true. Beliefs can be nested, e.g. $b(c_1, b(c_2, f = v))$ means that character c_1 believes that character c_2 believes that

fluent f has value v . As mentioned previously, unlimited theory of mind is a key feature of this model and thus beliefs can be nested in this manner to any depth.

Logical Expressions

Three kinds of complex logical expressions are defined: *preconditions* that must be checked, *effects* that describe how states are modified, and *utility functions* that define preferences over states.

Preconditions are converted to disjunctive normal form during pre-processing. A process similar to Weld's (1994) is used to compile out first order quantifiers. Universal quantifications are replaced by conjunctions, and existential by disjunctions:

$$\begin{aligned}\forall v(f = v) &\leftrightarrow (f = v_1) \wedge (f = v_2) \wedge \dots \\ \exists v(f = v) &\leftrightarrow (f = v_1) \vee (f = v_2) \vee \dots\end{aligned}$$

In keeping with classical planners, preconditions and effects must be finite, so quantifiers over numeric fluents are not permitted. The constants \top and \perp are considered to be in disjunctive normal form. \top is a disjunction of one empty clause and always true, while \perp is a disjunction of zero clauses and always false. Negated literals are compiled out using these equivalencies:

$$\begin{aligned}\neg(f = v) &\leftrightarrow (f \neq v) \\ \neg(f > v) &\leftrightarrow (f \leq v) \text{ etc.}\end{aligned}$$

Complex belief expressions are compiled out using:

$$\begin{aligned}\neg b(c, x) &\leftrightarrow b(c, \neg x) \\ b(c, x \wedge y) &\leftrightarrow b(c, x) \wedge b(c, y) \\ b(c, x \vee y) &\leftrightarrow b(c, x) \vee b(c, y)\end{aligned}$$

Preconditions may not be contradictions. For example, the precondition $b(c, f = v) \wedge b(c, f \neq v)$ is not allowed.

Effects describe how the state after an event differs from the state before. Effects can be conditional, meaning they may not apply, depending on the state before the event. Like UCPOP (Penberthy and Weld, 1992) and Fast Downward (Helmert, 2006), Sabre represents all effects as having a condition, even if that condition is simply \top .

A single effect e can be described by this grammar:

$$\begin{aligned}e &:= p \rightarrow g \\ g &:= f = v \mid f_{\mathbb{R}} = n \mid b(c, g)\end{aligned}$$

All effects have a condition p in disjunctive normal form. The effect $p \rightarrow (f = v)$ means that, when p holds in the state before, fluent f has value v in the state after. Numeric fluents can be assigned numeric values following the grammar for n given above (i.e. n is a number, numeric fluent, or arithmetic expression). For example, $\top \rightarrow (f_{\mathbb{R}} = f_{\mathbb{R}} + 1)$ means that $f_{\mathbb{R}}$ has a value one higher in the state after.

Effects can modify character beliefs directly. $p \rightarrow b(c, f = v)$ means that, when p holds in the state before, character c believes fluent f has value v in the state after. Belief effects can also be arbitrarily nested, e.g. $p \rightarrow b(c_1, b(c_2, f = v))$ and so on. Having defined a single effect, we define an effect expression as a conjunction of effects:

$$(p_1 \rightarrow g_1) \wedge (p_2 \rightarrow g_2) \wedge \dots$$

In keeping with classical planning, effects must be deterministic, so disjunctions and expressions equivalent to disjunctions (like existential quantifications) are not permitted in effect expressions. Effects may not be contradictions.

Utility functions are compiled into a normal form similar to effects. They are conditional, but at least one condition must hold in any state. A utility function is an ordered sequence of m conditional numeric expressions $p \rightarrow n$:

$$\langle p_1 \rightarrow n_1, p_2 \rightarrow n_2, \dots, \top \rightarrow n_m \rangle$$

Here, $m \geq 1$, $p_{1\dots m-1}$ are conditions in disjunctive normal form, and $n_{1\dots m}$ are numeric values following the grammar for n above (a number, numeric fluent, or arithmetic expression). The last condition p_m must be \top to ensure that one case will always hold. Utility functions are similar to if/elseif/else statements in a programming language. To evaluate a utility function in a state, each conditional expression $p_i \rightarrow n_i$ is considered in order until one where p_i holds is found, then n_i is evaluated. In the *Blackbeard* example, Marley wants to have the treasure. Her utility function is:

$$\langle loc(Tr) = M \rightarrow 1, \top \rightarrow 0 \rangle$$

where $loc(Tr)$ is a fluent representing the location of the treasure, and M represents Marley. This means that if she has the treasure, her utility is 1, and otherwise her utility is 0.

Initial State and Goal

A problem defines an initial state s_0 as an assignment of a value to every fluent, i.e. $\forall f : f = v$ where $v \in D_f$, along with assignments for any wrong beliefs that characters initially hold. In the example problem, since James has a map ($map(J) = \top$) but Marley and Blackbeard believe that he does not, their wrong beliefs are included in the initial state: $b(M, map(J) = \perp) \wedge b(BB, map(J) = \perp)$.

Beliefs that are not specified in the initial state are inferred using the following variants of classical planning's closed world assumption (that anything not stated to be true is assumed false). First, if a character's belief about a fluent is not stated, it is assumed that they believe whatever is actually true. So if $s_0 \models f = v$, then assume $s_0 \models b(c, f = v)$ unless explicitly stated that $s_0 \models b(c, f = u)$ where $u \neq v$. Second, characters assume that other characters have the same beliefs they have, unless stated otherwise. If $s_0 \models b(c_1, f = v)$ and there is no explicit statement for $b(c_1, b(c_2, f = u))$ where $u \neq v$, then assume $b(c_1, b(c_2, f = v))$. These assumptions are only used for the initial state.

A problem defines an *author utility function*, which expresses preferences for the states the planner should attempt to reach. A problem also defines a utility function for each character. Recall that the planner is the only decision maker and it must ensure that characters appear realistic. The planner chooses when and how the characters act so that the author's utility is increased and the characters appear to be working towards increasing their own utility.

In the example problem in Figure 2.1, the author has specified that the planner should only accept stories where Blackbeard (*BB*) gets the treasure; The author's utility function is:

$$\langle loc(Tr) = BB \rightarrow 1, \top \rightarrow 0 \rangle$$

James and Marley have utility functions similar to the author's, except they each want to have the treasure themselves. Blackbeard's utility function is slightly different:

$$\langle loc(Tr) = B \rightarrow 2, loc(Tr) = BB \rightarrow 1, \top \rightarrow 0 \rangle$$

Blackbeard prefers for the treasure to be buried (*B*), but if it is not buried, he would rather have it himself (*BB*) than any other alternative.

Events: Actions and Triggers

Events change the world state. Domain authors can create two kinds of events: *actions*, which the planner can choose to take, and *triggers*, which must occur when they can.

An action *a* defines a precondition expression $PRE(a)$ and an effect expression $EFF(a)$. It also defines a set of 0 to many actors $ACT(a)$. Actors are the characters responsible for taking the action. For an action to make sense in a narrative plan, every actor needs a reason to take the action.

Actions also define when characters observe them occurring. Formally, an action *a* defines a function $OBS(a, c)$: For any character *c*, $OBS(a, c)$ returns a precondition expression *p* such that, when *p* holds, *c* observes action *a* occur. When a character observes an action, they update their beliefs based on the action's effects. When a character does not observe an action, their beliefs remain the same (unless explicitly modified by the effect).

For example, the *tell-map* action is defined below, where a character *c*₁ tells another character *c*₂ that they (*c*₁) have a map, at some location *l*:

$$\begin{aligned} \text{Action } a: \text{ tell-map}(c_1, c_2, l) \\ PRE(a) : & \quad b(c_1, \text{map}(c_1)) = \top \wedge loc(c_1) = l \wedge loc(c_2) = l \\ EFF(a) : & \quad \top \rightarrow b(c_2, \text{map}(c_1)) = \top \\ ACT(a) : & \quad \{c_1\} \\ OBS(a, c) : & \quad loc(c) = l \end{aligned}$$

Consider the instance of this action where *c*₁ is James, *c*₂ is Marley, and *l* is Tortuga. This action can only occur if James believes that he has a map and James and Marley are both at Tortuga ($PRE(a)$). After it occurs, Marley believes that James has a map

($\text{EFF}(a)$). Only James is an actor ($\text{ACT}(a)$), because he is the only one who needs a reason to take the action. The action can occur whether or not Marley wants it to happen. Finally, the action is observed by any character whose location is Tortuga ($\text{OBS}(a, c)$). They will know that Marley believes James has a map, whereas those not at Tortuga will not know.

Triggers are events that must happen when their preconditions are met. A trigger t defines $\text{PRE}(t)$ and $\text{EFF}(t)$ as above. Any time the world is in a state where $\text{PRE}(t)$ holds, $\text{EFF}(t)$ must immediately be applied to change the world state. Actions advance time, but triggers do not. In other words, after time is advanced by taking an action, any number of triggers may then apply to update the state, but they happen instantly. If multiple triggers can apply in a state, the system chooses arbitrarily. To ensure determinism, triggers should not be defined such that their outcome depends on order of execution.

Triggers are similar to the axioms and derived predicates of PDDL planners (Thiébaux *et al.*, 2005), but not identical. Notably, triggers modify state fluents directly rather than deriving new predicates from them. Triggers are convenient for modeling belief updates based on character observations. Consider this example, which means “When characters c_1 and c_2 are in the same place, c_1 sees that c_2 is there.”

$$\begin{aligned} \text{Trigger } t: & \text{ see}(c_1, c_2, l) \\ \text{PRE}(t) : & \text{ loc}(c_1) = l \wedge \text{loc}(c_2) = l \wedge b(c_1, \text{loc}(c_2) \neq l) \\ \text{EFF}(t) : & \top \rightarrow b(c_1, \text{loc}(c_2) = l) \end{aligned}$$

Suppose James is at Tortuga and believes Blackbeard is at Port Royal, but really Blackbeard is at Skull Island. If James then sails to Skull Island, he would share a location with Blackbeard while believing that Blackbeard is someplace else. This trigger would then apply, correcting James’s wrong belief about Blackbeard’s location.

Triggers represent rules of the world common to all, so they do not define actors or observation functions. Triggers are effectively observed by any character who believes their precondition holds. If a character believes a trigger can happen in a state, then it does, regardless of whether they want it to.

Action Results Action effects have a finite number of conjuncts as specified by the domain author, but because actions can be observed and because there is no limit on the depth of theory of mind, actions can cause infinitely many changes to the world state. In the *tell-map* example, when Marley’s belief is updated, James knows it, and knows that Marley knows it, and knows that any bystanders know it, and knows that they know that she knows it, and so on.

Let $\text{RES}(a)$ be the results of action a , a possibly infinite conjunction composed of the action’s explicitly authored effects and any effects implied by observations. Its conjuncts are in the same format as above, $p \rightarrow g$, where p is a condition and g is either an assignment of a value to a fluent or a belief. $\text{RES}(a)$ is defined by these two rules:

1. $\text{RES}(a) \models \text{EFF}(A)$
2. $\forall c : (\text{RES}(a) \models p \rightarrow g) \Rightarrow$
 $(\text{RES}(a) \models (\text{OBS}(a, c) \wedge b(c, p)) \rightarrow b(c, g))$

The first rule states the results of a include the effects of a . The second defines results implied by observations. If $\text{RES}(a)$ includes the effect $p \rightarrow g$, character c observes action a occur, and c believes the condition p holds, then c will subsequently believe g . If Blackbeard is at Tortuga when James tells Marley that he has a map, then Blackbeard will subsequently believe that Marley believes that James has a map, even though this was not explicitly authored as an effect of the *tell-map* action.

Fortunately, there are a finite number of preconditions that ever need to be checked—those that appear in event preconditions, action observation functions, effect conditions, and utility conditions. During pre-processing, Sabre calculates all *relevant* results (i.e. all results that would affect this finite list of preconditions), adding explicit effects to actions so that all relevant results are accounted for.

State Space

The planner starts at the initial state and searches forward through the space of possible future states for a solution. To save memory, Sabre does not store a state as an array of values for each fluent; instead, it stores the history of events that led to that state, and when it needs to know the value of a fluent, it looks back through the history for the last time the fluent was modified.³

Let s be a state, and let $\sigma(e, s)$ denote the state after event e (recall an event is an action or a trigger). For any literal l , we can determine if l holds in $\sigma(e, s)$ using this procedure:

```

if  $e$  has result  $p \rightarrow g$ , and  $p$  holds in  $s$ , and  $g \models l$  then
  return  $\top$                                      ▷ Case 1
else if  $e$  has result  $p \rightarrow g$ , and  $p$  holds in  $s$ , and  $g \wedge l \rightarrow \perp$  then
  return  $\perp$                                        ▷ Case 2
else if  $\text{PRE}(e) \models l$  then return  $\top$            ▷ Case 3
else if  $\text{PRE}(e) \wedge l \rightarrow \perp$  then return  $\perp$     ▷ Case 4
else if  $l$  holds in  $s$  then return  $\top$            ▷ Case 5
else return  $\perp$                                    ▷ Case 6

```

There are six cases. Case 1 states that l holds if e has a result (whose condition holds) which makes l true. Case 2 states that l does not hold if e has a result which contradicts l . Consider the precondition $f_{\mathbb{R}} > 1$. The effect $f_{\mathbb{R}} = 0$ is not an exact negation of it, but $f_{\mathbb{R}} > 1 \wedge f_{\mathbb{R}} = 0$ is a contradiction, so this effect would make that precondition \perp .

³The first proposed representation of this model used states that were complete assignments of all non-belief fluents, and beliefs were represented as epistemic edges between these states (Shirvani *et al.*, 2017), much like the graph shown in Figure 2.1. The graph method entails identifying when duplicate nodes have been created, which requires graph isomorphism checks. Since states are only identical if they contain all the same character beliefs, checking if two states are duplicates requires comparing not just the two states themselves, but also all the states the characters believe in those states, and what they believe in those states, etc. This makes expanding nodes costly; Sabre’s representation avoids this cost in exchange for the higher cost of evaluating a given state fluent.

In contrast with classical planning, $\sigma(e, s)$ is defined for any event from any state, regardless of whether the event’s precondition holds in that state. This is to account for characters observing events which they wrongly believed were impossible. When a character observes such an event, they first update their beliefs so that the action’s precondition holds, and then update their beliefs based on its effect. In Figure 2.1, s_4 represents the state Blackbeard believes to be the real state when the actual state is s_3 (i.e. after *tell-map* and *sail* have occurred). In s_4 , James does not have a map, so $s_4 \not\models \text{PRE}(\text{dig})$, yet *dig* nevertheless occurs and is observed by Blackbeard. When Blackbeard observes James digging up the treasure, he first learns that James has a map and then considers the effects of the action. Cases 3 and 4 express this—that l holds if it is implied by e ’s precondition, and does not hold if it contradicts e ’s precondition. Since cases 1 and 2 are checked first, 3 and 4 only apply when l was not changed by a result of the event. If none of these cases apply, event e has no bearing on l , so we check l in the previous state s (cases 5 and 6).

Recall that a trigger must be applied when its precondition holds, and triggers happen instantly. When the world state is s and there exists a trigger t such that $s \models \text{PRE}(t)$, the world must immediately transition to $\sigma(t, s)$. For an event e and state s , the notation $\alpha(e, s)$ refers to the state of the world after taking event e and then applying any relevant triggers. α is defined recursively as:

```
function  $\alpha(e, s)$ 
  if  $\exists$  trigger  $t$  such that  $\sigma(e, s) \models \text{PRE}(t)$  then
    return  $\alpha(t, \sigma(e, s))$ 
  else return  $\sigma(e, s)$ 
```

As shorthand, let $\alpha(\{a_1, a_2, \dots, a_n\}, s)$ represent the state after a sequence of n actions taken in that order from state s . Importantly, characters should not be able to anticipate an event whose precondition does not hold. We say that an event e is *foreseeable* in state s when $s \models \text{PRE}(e)$, and *unforeseeable* when $s \not\models \text{PRE}(e)$. Likewise, $\alpha(\{a_1, a_2, \dots, a_n\}, s)$ is foreseeable iff each action is foreseeable in the state before it occurs, i.e. $s \models \text{PRE}(a_1) \wedge \alpha(a_1, s) \models \text{PRE}(a_2) \wedge \alpha(\{a_1, a_2\}, s) \models \text{PRE}(a_3)$, and so on.

Explanations and Solutions

Informally, a solution is a sequence of actions which can be executed from the initial state, leads to a state where the author’s utility is improved, and in which every action is explained, i.e. characters taking the action believe the action can lead to increasing their utility.

The notation $u(c_{\text{author}}, s)$ denotes the author’s utility in state s ; $u(c, s)$ denotes the utility of character c in state s ; and $\beta(c, s)$ refers to what a character c believes the state to be when the actual state is s . $\beta(c, s)$ is defined by this equivalence:

$$s \models b(c, l) \leftrightarrow \beta(c, s) \models l$$

An action a_1 is *explained* in state s when it is explained for all of $\text{ACT}(a_1)$, its actors. An action a_1 is *explained for* a character c in state s when:

1. There exists a sequence of $n \geq 1$ actions $\{a_1, a_2, \dots, a_n\}$ that starts with a_1 .

2. $\alpha(\{a_1, a_2, \dots, a_n\}, \beta(c, s))$ is foreseeable.
3. $u(c, \alpha(\{a_1, a_2, \dots, a_n\}, \beta(c, s))) > u(c, \beta(c, s))$.
4. All actions a_i where $i > 1$ are explained in the state before they occur; i.e. a_2 is explained in $\alpha(a_1, \beta(c, s))$, a_3 is explained in $\alpha(\{a_1, a_2\}, \beta(c, s))$, and so on.
5. No strict subsequence of $\{a_1, a_2, \dots, a_n\}$ also meets these 5 criteria and achieves the same or higher utility.

Requirement #1 states that there exists a plan which starts with action a_1 . #2 states that character c believes the plan can be executed. In other words, the whole plan must be executable from $\beta(c, s)$, even if it cannot be executed from s , perhaps because c has wrong beliefs. #3 states that c believes the plan will lead to a state where their utility is higher (even if it actually will not). #4 states that the rest of the actions (after a_1) must also be explained. This means that if the plan relies on actions by other characters, those actions must make sense for those characters according to c . #5 states that the sequence should not include redundant or unnecessary actions. For example, James cannot plan to sail to Skull Island, then sail back to Tortuga, then sail to Skull Island again, dig up the treasure and take it. This plan is redundant; the first two actions could be removed and it would still achieve the same utility.

As an example, consider the *sail* action in Figure 2.1, where James and Marley sail together from Tortuga to Skull Island. Would this action be explained in the initial state, s_0 ? Recall that in the initial state, Marley believes that James does not have a map.

To be *explained*, it must be *explained for* all of its actors (James and Marley in this case). It is explained for James: He can imagine a plan to sail to Skull Island, dig up the treasure, and take it. This plan begins with the action in question, sailing to Skull Island (#1). The plan is possible based on his current beliefs (#2), and he thinks it will lead to a state where his utility is higher (#3). James is the only actor for the *dig* and *take* actions, so these do not need to be explained for any other characters (#4), and there is no strict subsequence of this plan that would also work (#5). Therefore the sail action is explained for James in the initial state. However, Marley does not believe it is possible for James to dig up the treasure, since she wrongly believes he does not have a map. Although there is a plan that starts with *sail* and increases Marley's utility—sail with James, let James dig up the treasure, then take the treasure—it is not possible given her beliefs (#2). In fact there is no possible plan according to her beliefs that meets all the other criteria. The *sail* action is therefore not explained in the initial state, since it cannot be explained for one of its actors. It can, however, be explained after James tells Marley that he has a map, as in the solution shown in the figure.

Having defined explained actions, we can now formally define a solution. Let s_0 be the initial state. A solution to a problem is a sequence of n actions $\{a_1, a_2, \dots, a_n\}$ such that:

1. $\alpha(\{a_1, a_2, \dots, a_n\}, s_0)$ is foreseeable.

2. $u(c_{author}, \alpha(\{a_1, a_2, \dots, a_n\}, s_0)) > u(c_{author}, s_0)$.
3. All actions in $\{a_1, a_2, \dots, a_n\}$ are explained in the state immediately before they occur.

In short, the plan is possible, improves the author’s utility, and contains no unexplained actions.⁴

Pre-Processing and Simplification

A subtle challenge arises when dealing with triggers and beliefs. Triggers are not only checked in the current state s , but also in every character’s beliefs. Consider an example trigger, where x and y are shorthand for any two literals:

$$\begin{aligned} \text{Example trigger } t: \\ \text{PRE}(t) &= x \wedge \neg y \\ \text{EFF}(t) &= \top \rightarrow y \end{aligned}$$

And imagine c is a character and the state is:

$$x \wedge y \wedge b(c, x) \wedge b(c, \neg y)$$

The trigger does not apply in this state, but it does apply in $\beta(c, s)$. The state should immediately transition to:

$$x \wedge y \wedge b(c, x) \wedge b(c, y)$$

The above procedure defining α fails to capture this case. Even if it did, since Sabre does not limit how far theory of mind can be nested, it is difficult to know how many levels of beliefs need to be checked to make sure all triggers have been applied. A solution to this problem is proposed in (Shirvani *et al.*, 2017), but it requires expensive graph isomorphism checks. Sabre works differently.

During pre-processing, for every character c , if there exist a trigger t with the effect $p \rightarrow g$, and $b(c, g)$ is in the set of all possible preconditions, a new trigger t'

⁴Previous publications have included a fourth rule here similar to rule #5 for explanations: that no strict subsequence of $\{a_1, \dots, a_n\}$ also meets these criteria. In this document and all evaluations herein, that rule is not applied to solutions. The motivation to use the subsequence rule for solutions was to reject stories that contain unnecessary steps, or steps that do not contribute to the author’s goal (though they are explained for the characters). For instance, a character may walk back and forth between two locations if they have goals relevant to each one. They are effectively changing their mind about which goal to pursue, and since commitment to plans is not modeled, this behavior is allowed by the definition given here. Enforcing the subsequence rule for solutions is one way to exclude some unwanted stories like this, but it also removes other stories that are not unwanted.

For example, in the *Grammalot* domain (to be introduced in Section 3.2), enforcing this rule would exclude any story where the guard arrests the bandit, since the bandit being arrested does not contribute to either of the endings (Tom succeeding or failing). One could argue that if the author wants it to be possible for the bandit to be arrested, they can specify this in the author utility function; but I would contend that it is an unnecessary burden for the author to have to specify which character goals are allowed to be achieved. As long as the author’s utility is ultimately increased, characters should be allowed to take any action they have reason to, because this can help communicate their motivations, provide resolution, and contribute to the story in other ways.

Algorithm 1 The Sabre algorithm

```
1: Let  $\mathcal{A}$  be the set of all actions defined in the domain.
2: SABRE( $c_{author}, s_0, \emptyset, s_0$ )
3: function SABRE( $c, r, \pi, s$ )
4:   Input: character  $c$ , start state  $r$ , plan  $\pi$ , current state  $s$ 
5:   if  $u(c, s) > u(c, r)$  then
6:     if  $c = c_{author}$  or  $\pi$  is non-redundant then
7:       return  $\pi$ 
8:     Choose an action  $a \in \mathcal{A}$  such that  $s \models \text{PRE}(a)$ .
9:     for all  $c' \in \text{ACT}(a)$  such that  $c' \neq c$  do
10:      if  $a$  is unforeseeable in  $\beta(c', s)$  then return failure.
11:      Let state  $b = \alpha(a, \beta(c', s))$ .
12:      if SABRE( $c', \beta(c', s), \{a\}, b$ ) fails then return failure.
13:   return SABRE( $c, r, \pi \cup a, \alpha(a, s)$ )
```

is generated with $\text{PRE}(t') = b(c, \text{PRE}(t))$ and $\text{EFF}(t') = b(c, \text{EFF}(t))$. This ensures triggers also account for all relevant effects. To use the earlier example:

Original trigger t :		New trigger t' :
$\text{PRE}(t) = x \wedge \neg y$	\Rightarrow	$\text{PRE}(t') = b(c, x) \wedge b(c, \neg y)$
$\text{EFF}(t) = \top \rightarrow y$		$\text{EFF}(t') = \top \rightarrow b(c, y)$

The preconditions of these new triggers may add to the set of all possible precondition literals, so the process is repeated until no new triggers are needed. It is possible to construct triggers that would cause this process to run infinitely, in which case one would need to revert to the graph-based solution, but in practice such a domain has not been encountered.

Sabre also uses methods adapted from other planners (Hoffmann, 2003; Helmert, 2006) to simplify the problem by detecting propositions which must always be true or false. This sometimes allows the removal of fluents, actions, and triggers from the domain to reduce the time and memory required during search.

2.4 Search

Sabre's search procedure is given in Algorithm 1. It takes four inputs: the character c for whom to find a plan, the state r where the search began, a plan π , and the current state s . The plan π is the sequence of actions that can be executed from r to reach s . SABRE can find a plan for any character from any state. The initial call to SABRE (line 2) uses the special author character c_{author} , the initial state of the problem s_0 , and the empty plan \emptyset .

SABRE starts by checking if c 's utility is higher in s than it was in r (line 5). If so, then this is either a solution (if c is the author) or a possible explanation for a character. For an explanation to be valid, the plan must not contain any unnecessary actions. If the current plan is not a solution or a valid explanation, we nondeterministically choose an action a to add to the plan. Before exploring further,

Sabre checks whether that action can be explained for all its actors other than c (lines 9 to 12). When c is c_{author} Sabre needs to find an explanation for all the actors. When c is a character other than the author, that character must be able to anticipate the cooperation of other characters who are part of their plan. For example, James must expect that Marley will consent to the *sail* action in order to include it in his plan.

There are two cases where a cannot be explained for an actor c' . The first is when $\alpha(a, \beta(c', s))$ is unforeseeable (line 10), meaning c' does not think a is possible. The second is when c' cannot imagine a plan starting with a that improves their utility (line 12), so they have no reason to take it. For example, Marley cannot expect Blackbeard to dig up the treasure, because she knows that this wouldn't improve Blackbeard's utility. If the action can be explained for all actors besides c , we add a to the plan, advance the current state to $\alpha(a, s)$, and recursively call SABRE.

If the first call to SABRE on line 2 returns a plan, it is a solution to the problem; it is possible from the initial state (all actions' preconditions hold when they occur), it leads to a state where the author's utility is higher, and all actions can be explained for the characters who take them. SABRE may not terminate if no solution exists, so in practice we impose a maximum depth on the search.

2.5 Evaluations

A human subjects evaluation demonstrated that the model accurately tracks nested beliefs (Shirvani *et al.*, 2017). Three stories were generated, each in a different domain, that included narrative elements like deception, cooperation, anticipation, and surprise. Subjects were assigned one of the three stories, first reading a description of its initial state and then answering comprehension questions about the world and the characters' initial beliefs. Responses were removed from the analysis if the subject misunderstood the initial state (answered any of these comprehension questions incorrectly). Story actions, translated into simple English sentences, were then displayed one step at a time. At certain points throughout the story, subjects were asked questions about characters' current beliefs, including nested beliefs up to two levels. In total, data was collected for 54 of these belief questions across the three different domains.

Sabre's belief model, along with a 1-layer limited belief model and a model lacking belief, were tested for accuracy compared to the human data, on all questions for which subjects' answers were in agreement (94% of questions). Sabre achieved 100% accuracy, while the 1-layer and zero belief models achieved only 49% and 44%, respectively. This means that for these stories, readers tracked characters' beliefs in a way that previous models fail to capture, but that the Sabre model captures with complete accuracy.

Sabre's explanations represent the plans that explain or justify characters' actions and are responsible for ensuring character believability. To evaluate whether this belief model improves overall believability compared to previous models, we asked human subjects to assess the believability of different plans from the perspective of a character in a story (Shirvani *et al.*, 2018).

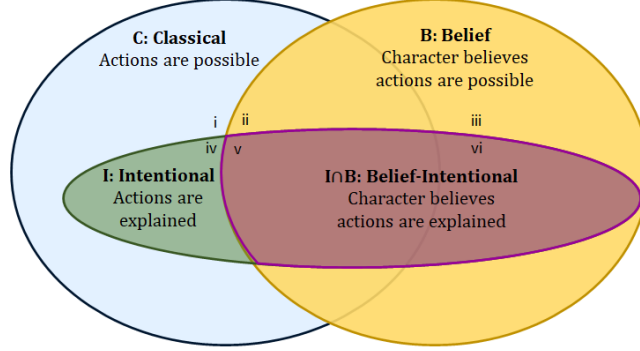


Figure 2.2: Six types of character plans

Figure 2.2 depicts the overlapping spaces of different definitions for character plans. The oval labeled C represents the classical plan constraint, where each action is possible when it occurs. B represents the belief constraint, where each action will be possible when it occurs, according to the beliefs of the character at the beginning of the plan. I is the constraint that plans must achieve the character’s goal (or increase their utility), either in reality or in the mind of the character. Previous intentional planners such as Glaive constrain character plans to be possible in reality (C) and to achieve the character’s goal (I), the combined regions labeled $iv + v$. Sabre plans are represented by regions $v + vi$: plans that the character believes are possible (B) and will achieve their goals (I), regardless of whether they are or will in reality.

Using a *Blackbeard* problem, we generated every possible pairing of a partial story (a sequence that is foreseeable from the initial state), and a plan for Marley that falls into any of the regions in Figure 2.2, where the story and plan lengths combined did not exceed 7 steps. From the state at the end of the partial story, all plans were either possible in reality or according to Marley’s current beliefs, or both. The plans were tagged based on whether they satisfied each constraint in the diagram for Marley in that state.

The evaluation aimed to compare the believability of Sabre plans and plans in the other regions, given the same partial story. For some partial story, two plans paired with that story were randomly sampled from two different spaces and shown to participants. The partial story was displayed first, with each step translated into a simple sentence in third person and present tense (e.g. “James and Marley sail to Skull Island”). After these steps, the two plans were displayed side by side, translated into future tense and first person from Marley’s perspective (e.g. “James and I will sail to Skull Island”). We asked subjects which of the two plans made more sense for Marley to be considering at that time in the story. Each data point therefore reflects a preference for one class of plans over another.

Regions of interest were grouped, and comparisons were made between groups by counting the number of times a subject preferred a plan from any region in the first group over a plan from any region in the second group, and vice versa. The binomial test then determines whether plans from one group were favored significantly more often than the other. Sabre plans ($v + vi$) were perceived as more believable than non-Sabre plans ($i + ii + iii + iv$) with $p = .014$. In the original publication, we were

unable to draw more specific conclusions, such as whether Sabre plans were preferred over Glaive plans. A later investigation into why this was the case revealed a factor that caused some plans to be less believable for a separate reason, which is relevant to this document because it pertains to domain authoring and suggests a direction for future work in character believability modeling. When we account for this factor via filtering, we do find a significant preference for Sabre plans over Glaive plans. The section below describes this additional analysis, which has not yet been published elsewhere.

Inefficient Plans

After subjects in the above experiment answered which plan made more sense between the two options, they were also asked to identify (via checking boxes) which steps in the rejected plan, if any, they deemed to be flaws. Examination of this data revealed that steps where Blackbeard sails to Tortuga were reported as flaws far more frequently than any other step.

In the *Blackbeard* problem used for this study, Blackbeard begins at Skull Island, and his goal is to have his treasure and be at Port Royal (note that this is different from the example in Section 2.2, in which he wants his treasure to remain buried). In this problem, Blackbeard almost never has a reason to be in Tortuga. However, since the domain allows characters to sail between any two islands, it is considered reasonable for him to sail to Tortuga just so that he can then sail from Tortuga to Port Royal, as if Tortuga is a step along the way. Even though sailing directly from Skull Island to Port Royal would have taken fewer steps, the existence of this shorter plan does not invalidate the longer plan (it is not a subsequence of it; see Section 2.3). Humans realize that the shorter plan makes *more* sense than the longer plan, and for some people this weakened the believability of the longer plan.

The model does not and should not limit characters to only form optimal plans. This would force characters to be perfectly rational, which is not typical of stories. Furthermore it can be useful towards the author’s goals for characters to be allowed to pursue longer plans than necessary. However, plan efficiency is one element by which people commonly evaluate character believability, and sometimes this behavior does not appear believable. It may be possible in future work to further improve character believability by constraining explanations to better account for plan efficiency. For now, the model errs on the side of inclusion and allows characters to make inefficient plans because this includes more stories in the solution set (which can be filtered out later).

To examine the effect of Sabre’s belief model on believability in the absence of the efficiency question, I carefully reviewed all responses and removed those where the participant reported a flaw that demonstrated an inefficient plan (a character sails to some location other than their actual destination at that moment). The filter disproportionately removed data where subjects had rejected Sabre plans over others. Inefficient plan steps represented the majority of flaws that were reported for Sabre plans. The results after this filter are shown in Table 2.1. Hypothesis 1 (Sabre plans are preferred over the rest) is now supported with $p < .001$, and additionally Sabre

Hypothesis	Regions	Successes / Total	<i>P</i> -Value
1	$v + vi \succ i + ii + iii + iv$	63 / 93	$p < .001$
2	$v + vi \succ iv + v$	30 / 46	$p = .028$
3	$v + vi \succ i + ii + iv + v$	62 / 93	$p < .001$

Table 2.1: Results after filter

plans are preferred over plans using previous intentionality models ($p = .028$) as well as classical plans ($p < .001$).

This study demonstrates that Sabre’s model of belief improves character believability over previous models. Notably, however, it can still allow characters to appear to do things unnecessarily: When a shorter (or for other reasons perceived better) plan is available, a pursued plan may seem unreasonable. For now, it falls to the domain author to recognize and eliminate this kind of behavior when it is not wanted.

2.6 Limitations

Modeling theory of mind improves character believability over previous intentionality models and allows more complex character interactions, but it is computationally expensive. An obvious direction for future work is to explore improvements to the search process via pruning and better heuristics that account for beliefs and intentions. This approach dramatically improved performance for other narrative planners like Glaive and IMPRACTical. It is also worth noting that although the model does not limit theory of mind depth, in practice a limit can be imposed in order to speed up search. For example, there is some research suggesting that 3-4 levels of nesting may be sufficient to cover most scenarios found in literature (Zunshine, 2022).

Uncertainty is not modeled, meaning characters cannot be unsure about the state of the world. They commit to a set of beliefs about everything, and do not consider any other world to be possible. Because of this, it may be difficult to construct certain types of domains for which information and deduction are a central focus, e.g. murder mystery stories (Eger, 2020). Modeling true uncertainty comes with a high computational cost, but a limited model could still improve the system’s capabilities and is worth exploring in future work.

Character believability is improved but by no means perfect. One example is described above, where believability suffered when characters pursued unnecessarily long plans. Similar problems might be caused by characters leaving plans incomplete or changing plans without reason. Future work may explore ways of eliminating this unwanted behavior, e.g. by modeling plan costs, preferences over plans, commitment to plans, etc. Believability may also be improved by modeling personality and emotion (Shirvani and Ware, 2019, 2020), which could give the author more control over how characters act on an individual basis.

The belief model improves authorability in the sense that it provides a simple

syntax for authoring belief mechanics that would otherwise be extremely difficult to program by hand. Yet the general problem of author burden remains. Domain authors must still define fluents, actions, triggers, and utility functions, and evaluating these definitions is difficult because solution spaces are so large and complex. Human authors will make mistakes and wrongly expect certain stories or sequences to be present or absent in the solution space, and it can be difficult both to notice these problems and to identify their cause. The belief model further complicates this issue because it adds more automatic reasoning that the planner does behind the scenes, and therefore more ways for unexpected situations to arise.

In fact, the same would likely be true for other models and features that improve believability, such as those suggested above. As narrative models continue to increase in complexity so that they can further hone in on the ideal set of acceptable stories, their solution spaces will become increasingly difficult to evaluate. It may therefore be essential to first address the problem of solution space summarization so that adequate domains can be created to facilitate and test these increasingly complex narrative models. The methodology presented in Chapter 3 aims to benefit future research in this area by enabling meaningful summarization of solution spaces that can help domain authors understand and debug domains more effectively.

Chapter 3 Summarizing Solution Spaces

For interactive narrative domain authors, it is critical to understand and be able to evaluate all the branches of the story world being created. Narrative planners can produce solution spaces that are far too large, detailed, and repetitious for humans to simply read and comprehend. This is by design—and the very reason for using narrative planning—but without a means of summarizing these spaces, domain authors cannot know if they are creating the kinds of stories they intend to create. Authoring tools can help people understand solution spaces better, e.g. by allowing them to test whether certain conditions, actions, or sequences are possible or impossible, or providing an interface for exploring solution spaces manually. These are useful tools, but they require the user to know what they are looking for. What is missing is a high-level view of the solution space, and a means of organizing the stories so that they can be explored more efficiently. Clustering stories based on their similarity can solve this problem; it can visualize the story space as a whole and guide authors to evaluate stories in meaningful groups, providing significant aid in designing, editing, and debugging story domains.

Clustering stories for humans requires a measurement of similarity, or a distance metric, that captures what humans perceive to be similar and different about two stories. This problem is challenging because humans often have very different ideas about what makes stories similar than the algorithms that created them do. This chapter presents a story distance metric that captures the human perspective of story similarity using a cognitive model of narrative comprehension and memory. It then demonstrates how the proposed distance metric can be used to cluster solution spaces into meaningfully different branches that help communicate their content effectively.

My approach to similarity is inspired by research into meaningful choices in interactive narratives. Although many other factors contribute to choice meaningfulness (Iten *et al.*, 2018), one study (Cardona-Rivera *et al.*, 2014) suggests that people find choices more meaningful when their expected outcomes are sufficiently *different from each other* in terms of their situational properties—whether they involve the same characters, locations, and other entities. This is based on the event-indexing situation model (EIM) (Zwaan *et al.*, 1995; Zwaan and Radvansky, 1998), a cognitive model that describes how narrative events are processed, stored, and retrieved in memory. Building on this capability of the EIM to capture meaningful differences between story events, this work extends the model to capture meaningful differences between entire stories.

3.1 Related Work

Situation Models in Narrative Planning

Psychological research in narrative comprehension has established the theory of cognitive *situation models* (Van Dijk *et al.*, 1983; Johnson-Laird, 1983), in which

a reader maintains a mental model of the current situation, and actively updates it as new information is perceived. The event-indexing model (EIM) (Zwaan *et al.*, 1995; Zwaan and Radvansky, 1998) posits that events are the focal points of narrative situations, and that they are indexed along at least five dimensions, labeled *protagonist*, *time*, *space*, *causality*, and *intentionality* (the who, when, where, how, and why of each perceived event); and that they are stored in short- and long-term memory as a network of nodes connected through links in these dimensions. During narrative comprehension, new events can be processed (integrated into the mental situation model) more quickly if they share any index with any previous events, and previous events are easier to recall if they share any index with the current situation model. Humans also appear to segment continuous perceived behavior in general into discrete events along these dimensions (Newtson, 1973; Radvansky and Zacks, 2011).

The *protagonist* index was originally described as linking together events that involve the story’s protagonist. This definition requires that a specific story protagonist be established, which is not always the case, so it is often generalized to include other important characters and even objects the reader is actively tracking. The *time* index links events that occur in the same time frame, where the time frame is updated whenever the narrative discourse indicates a time shift, such as night falling, weeks passing, etc. The *space* index links events that occur in the same location. The *causality* index links events that are causally related, i.e. there is a link between events if the earlier event enables the later event in some way. The *intentionality* index links events that are motivated by the same goals, i.e. they occur for the same reason.

These indices were first mapped onto narrative planning events in Indexter (Cardona-Rivera *et al.*, 2012), which explicitly tracks the salience of previous story events based on the indices they share with the most recently experienced event (representing the current situation model). The authors suggest this technique could be used to manipulate narrative qualities that are based on memory and expectation, like surprise or suspense. Indexter, along with the present work, assumes that characters, locations, and time frames are represented in the planning domain as unique symbols, making it easy to determine whether two events share any of these indices. Causality is modeled explicitly by the planner, and Indexter accounts for intentionality using IPOCL’s frames of commitment (Riedl and Young, 2010), which reflect what specific goal a character is pursuing when they take each of their actions.

Under the Indexter model, the salience of a past event is calculated based on the presence or absence of shared indices with the event currently being perceived. A value is given for each index: 1 if the past event being recalled, e_i , shares that index with the current event, e_n ; otherwise 0. Indexter also allows each index type to be assigned a weight coefficient corresponding to the strength of its individual contribution to salience, such that the total salience will be between 0 and 1. The salience of the past event e_i is calculated as:

$$salience(e_i, e_n) = w_t t_{e_n} + w_s s_{e_n} + w_p p_{e_n} + w_c c_{e_n} + w_i i_{e_n}$$

where t , s , p , c , and i are respectively the time, space, protagonist, causality, and

intentionality indices of an event, and w_* are the weights of each index. The indices are assumed to be equally weighted ($w_* = 0.2$), in lieu of a better set of weights supported by evidence. The present work does not provide a general answer to this question, but sheds some light on it by demonstrating the effects of different index weightings in an example domain.

A later study validated this model by measuring readers’ response time when asked questions about previous events (Kives *et al.*, 2015). Readers tended to respond faster when the past event shared one or more indices with the current event than when it did not. Cardona-Rivera and Young (2014) also proposed a more nuanced salience calculation that scales the salience of a past event according to the total index overlap between the situation model and all the events in the story. Thereby, for example, the strength of the recall effect from a story location that only appears a few times would be stronger than that of a frequently used location.

One study measured player agency through choice meaningfulness, finding that two story situations, or events, are *meaningfully different* when they differ along situational dimensions (Cardona-Rivera *et al.*, 2014).¹ This was captured with a *situation vector* that represents a person’s mental model of a situation. The vector is a four-tuple $\langle S, T, G, C \rangle$ encoding the perceived space, time, goals (of the player), and characters involved in the situation. Two situation vectors $v1$ and $v2$ are considered different when $\Delta(v1, v2) > 0$, where Δ of two vectors returns a number between 0 and 4 indicating the number of indices that are equal between them. Thus, two events are considered different when they differ by any dimension. Notice that the causality index is missing from this vector. Comparing the situational similarity of two events that are both one step in the future—rather than two events that occur at different points in the same linear story—renders the causality index useless. Two possible outcomes of the same choice cannot share the causality index, since one cannot be the causal ancestor of the other if they are both already enabled.

In my own previous work, we explored whether we could predict and influence players’ choices for story endings using Indexter’s model of salience (Farrell *et al.*, 2020). We found that people tend to prefer endings that share indices with previous choices they had made that were situationally different (i.e. more meaningful choices, according to the study described above). In other words, players preferred endings that reminded them of their meaningful choices. Once they have made decisions that cause certain entities—locations, times, goals, and characters—to be more salient than others, they are more likely to prefer endings that reuse those same entities. The causality index was not used in this study, either, for similar reasons.

These studies (Cardona-Rivera *et al.*, 2014; Farrell *et al.*, 2020) highlight the important role that the situational dimensions play in how humans perceive the branching of interactive stories. Situational differences between the outcomes of branching points (player choices) indicate to the player that the story is branching in a more meaningful way. Making a choice that steers the story in some direction

¹This is based on Murray’s (1997) definition of agency as the satisfying ability to make meaningful choices and observe their consequences—though other definitions also exist (Wardrip-Fruin *et al.*, 2009).

along these dimensions establishes an important reference point for the player that can be used to model their preferences for the ending. We assume this is because the most satisfying ending is the one that makes the player’s choices feel most relevant or impactful. Notably, the causality index was not used, so this is not the same as reincorporation (Tomaszewski, 2011), whereby a player’s action is made *necessary* by the ending. In this case the ending simply reminds the player of their previous choice because it involves some of the same story entities. The present work extends these notions to model overall difference or similarity between stories by comparing the salience of all story entities at the end. The idea is that stories, like events and branching directions, are perceived as more similar when the same entities are used, and less similar when different entities are used.

Story Similarity

It is useful to distinguish between two broadly defined types of similarity that can exist between stories: those that describe the discourse, or presentation of the stories (non-structural similarities); and those that describe the stories’ fictional events and the relationships between them (structural similarities). Studies have shown that without reflection, humans often base their assessments of story similarity on discourse elements like presentation style, setting, types of events, linguistic features, and more (Löwe, 2011; Fisseni and Löwe, 2012). However, there is also evidence that humans are capable of identifying structural similarities and comparing stories independently of the presentation layer, especially when prompted to do so (Fisseni and Löwe, 2012). The present work deals with generating and analyzing narratives at the plot level, and so is specifically concerned with structural similarities.

Several formal frameworks have been proposed for modeling structural similarities between narrative texts (Löwe, 2011; Bod *et al.*, 2012; Reiter, 2014). These seek to capture the extent to which humans would judge two stories to be “the same story”, based exclusively on structural features like causality, the role of the protagonist, temporal ordering, expectations, and more. Narrative plans already define many of the same structural features; the present work aims to formalize a model of overall similarity between two narrative plans.

One way to compare narrative plans is to treat them simply as classical plans and use classical plan distance metrics. For example, it is common to measure the difference between two plans as the set difference between the sets of unique actions in each plan (Srivastava *et al.*, 2007). A more nuanced method is the edit distance, or number of editing operations required to convert one plan into the other (Levenshtein, 1966). Edit distance can be applied at different levels of granularity, e.g. an editing operation may be to replace an action with another action, or to replace a single parameter within an action with a different symbol. These are useful metrics of structural similarity, but they are limited. They only measure syntactic differences between the plans, and do not utilize any story-specific information modeled by the narrative planner, such as the relative importance of characters (as opposed to other symbols) and their intentions, conflicts between them, and so on.

Story plans can also be translated into natural language, at which point linguistic methods could be used to assess their overall similarity. Methods like BLEU and ROUGE (Papineni *et al.*, 2002; Lin, 2004) evaluate the quality of system-generated text based on the degree of n-gram overlap with a target text written by humans. The same techniques can be applied to measure the similarity of two translated stories based on their lexical overlap; however these metrics have many well known weaknesses. For example, they do not consider the meaning of words (synonyms are not recognized as similar), nor their morphology (variations of the same word are not recognized as similar). They also do not model the relative importance of different words, so for example articles and proper nouns are given equal importance. Another linguistic approach could be to use a language transformer like BERT (Devlin *et al.*, 2019) to project both story texts into a latent space where their distance can be measured. BERT is a large model, trained on millions of words from books and Wikipedia articles, and so accounts for many of the drawbacks of the above methods, but incurs a significant computational cost. These methods would primarily capture non-structural similarities, and may be unreliable for this task since they are sensitive to different text realizations of the story plans. Moreover, they would not be ideal for summarizing clusters of stories, since they do not provide a concise justification for their similarity assessments that could meaningfully label a cluster. My approach is to group stories based on which story entities are most important, so that these entities can be used to label and summarize the clusters they establish.

Kypridemou and Michael (2013) validate the general approach of measuring story similarity by summarizing each story and comparing the two summaries. Subjects in their study first rated the appropriateness of a single summary as a summary of each of two stories, and then rated the similarity between those two stories. The more appropriate the summary was for both, the higher the similarity between them. One summary-based story plan comparison metric has been proposed, called ISIF distance (Amos-Binks *et al.*, 2016); it summarizes a CPOCL (Ware and Young, 2011) narrative plan based on its important steps (IS) and summaries of its intention frames (IF). The important steps are those events with highest causal degree (total number of incoming and outgoing causal links), in line with earlier work in narrative summarization suggesting that more important events have more causal connections (Lehnert, 1981; Trabasso and Sperry, 1985). ISIF’s intention frame summaries capture how character goals are motivated and achieved by encoding the motivating and satisfying steps for each character goal. The authors propose a Jaccard-based distance metric that compares the overlap of the stories’ important steps and intention frame summaries. This method captures important structural differences that other domain-independent metrics do not, but its scalability is limited. For example, it exclusively compares sets of grounded actions, but similar actions do not always have identical signatures. By comparison, the present work models more narrative features (i.e. characters, time, and space, in addition to causality and intentionality) and uses a numeric vector representation that captures more nuanced information.

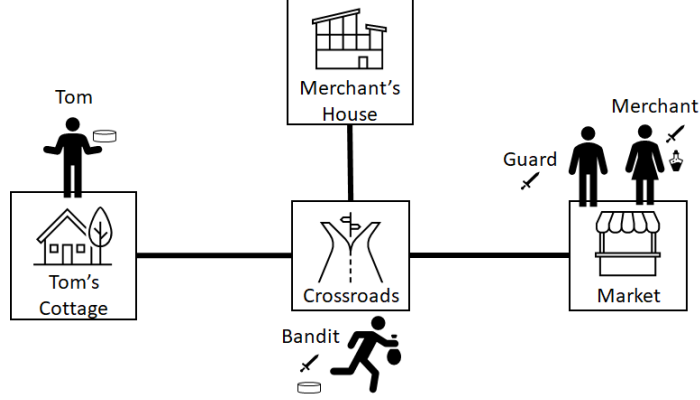


Figure 3.1: A depiction of the initial state in *Grammalot*

3.2 Grammalot Domain

The experiments and examples in this chapter use a story domain called *Grammalot*, which was originally developed to test players’ experience of interactive narratives using different methods for controlling non-player characters (Ware *et al.*, 2022). The domain has been modified slightly from its original version.

There are four characters including the player character, named Tom. Tom begins at his cottage where he is given the task of acquiring a potion for his grandmother, and a coin that can be used to buy an item. A merchant is in the market selling the potion in question, as well as a sword. There are four locations: Tom’s cottage, the market, the merchant’s house, and a crossroads that connects them all (see Figure 3.1). The town guard is in the market watching for criminals, and a bandit lurks in the crossroads. The bandit has one coin and wants to acquire other items of value—coins and potions.

Seven actions are available. Characters can *walk* from one place to another if connected by a path. Characters can *buy* items from the merchant in exchange for a coin. If a character is armed, they can *rob* an unarmed character (steal an item from them). An armed character can *attack* and kill another character. Characters can *take* items from characters who are sleeping or dead. Tom can *wait* for night, causing the guard to disappear and the merchant to go to sleep at her house (instantly, and with all her belongings). Finally, the guard can *arrest* a character who has committed a crime (robbed, attacked, or taken an item that did not belong to them). The story ends when Tom returns to his cottage carrying the potion, or is arrested or killed.

3.3 Saliency Vectors

As I have described in Section 3.1, the EIM defines the situational indices in a binary fashion, recording simply whether or not two events are linked along each index. Prior work has modeled the saliency of *events*, but the present work concerns the saliency of the *entities* responsible for linking the events. Each time an entity is involved in the latest event, it becomes salient—i.e. it is present in the current situation model—and

its salience decays by some amount each time an event passes that does not involve it. The salience of all story entities after the story has occurred can then serve to summarize the reader’s memory of the story.

For all but the causality index there is a specific type of entity whose involvement in the two events establishes the linkage between them. The protagonist index is defined in terms of which characters are involved in both events, so the set of entities that can become salient through the protagonist index is the set of characters in the domain. Likewise, the sets of locations and time frames (typed parameters in the domain) are the entities for the space and time indices, respectively. The intentionality index is based on character goals, which intentional planners require domains to explicitly define, although their representations differ. In Sabre, a character goal is any condition of a character utility function other than the constant \top .

The causality index is unique in that it is defined not in terms of the properties of the events, but of the events themselves and how they occur in the specific context of the story. Sharing the causality index means that one event is a causal ancestor of the other—i.e. there is a causal chain connecting some effect of the earlier event to some precondition of the later event. There is no clearly identifiable finite set of entities that become salient through this index. One option is the set of grounded *actions* in the domain: We can say that a specific event becomes salient due to either being the current event itself or being a past event that enabled it. Alternatively, we could use the set of all grounded *fluents* or even all possible *assignments* of some value to some (non-numeric) fluent: We can say that these properties become salient when they are used in the causal ancestry of the current event. The section below gives definitions for all three of these variations of causality, and all are included in the evaluation in Section 3.5.

Definitions

A story is summarized as a set of five fixed-length numeric vectors, called the *salience vectors* because they represent the relative salience of the entities in the story at a given time. Each vector reflects one of the five situational dimensions previously discussed. This section defines a function for each dimension that determines whether an entity of a certain type becomes salient upon an action occurring. Variations of some definitions are included in an effort to generalize them to other types of story systems.

Let $S = \{S_1, \dots, S_n\}$ be a sequence of n explained actions. The function $\text{SALIENT}(e, i, S)$ returns a Boolean value indicating whether or not an entity e becomes salient after step i in S , and is defined based on the entity type as follows.

Protagonist The *protagonist* dimension links events that involve the same important characters. Three alternative definitions are given here. The first makes use of intentional planners’ distinction between characters who are responsible for taking the action (its actors), and those who are involved in it but not willfully (e.g. a recipient of the action). The second definition does not make this distinction, instead

treating all characters involved in the action equally. The third adheres closer to the original EIM description of the protagonist index as linking events that involve the story’s protagonist, thus a predefined story protagonist is required for this definition. In the *Grammalot* domain, the protagonist is Tom.

For a character c :

Definition 1a. $\text{SALIENT}(c, i, S) \iff c \in \text{ACT}(S_i)$

Definition 1b. $\text{SALIENT}(c, i, S) \iff c \in \text{args}(S_i)$

Let c_{protag} be the story protagonist.

Definition 1c. $\text{SALIENT}(c, i, S) \iff c = c_{\text{protag}} \wedge c \in \text{args}(S_i)$

Time The *time* dimension links events that occur within the same time frame. The planning domain should explicitly provide the time frame of each action in its parameters.

For a time frame t :

Definition 2. $\text{SALIENT}(t, i, S) \iff t \in \text{args}(S_i)$

Space The *space* dimension links events that occur in the same location. As with time frames, locations must be provided as arguments of each action.

For a location l :

Definition 3. $\text{SALIENT}(l, i, S) \iff l \in \text{args}(S_i)$

Future work may improve the definitions of time and space using a hierarchical representation, since both can be organized hierarchically in memory (Magliano *et al.*, 2001; Zacks *et al.*, 2009). However, the domains available at present are too small to make use of such a representation, and this discrete symbol-based definition is sufficient to demonstrate useful properties for now.

Intentionality The *intentionality* dimension links events that occur for the same reasons. Intentional planners differ in their representations of character goals and how they decide which goal an action is contributing to—e.g. intention frames (Riedl and Young, 2010), relevant actions (Teutenberg and Porteous, 2013), or explanations (Ware and Young, 2014). Sabre tracks explanations for every action—the plans that justify each actor’s participation in it. Each explanation satisfies some condition of a character’s utility function; these conditions represent the reasons why the action occurred.

This is not perfect. Some planners may not distinguish between *which* of a character’s goals an action is in service of, only that the action is justified. Even when this information is modeled by the planner, it is still possible that there are multiple valid explanations for the character’s action, or that the audience could interpret

the behavior in a completely different way. This limited definition is still useful, but future work may improve upon it, e.g. through goal recognition techniques.²

Sabre tracks one explanation per actor in an action. Let $explanations(S_i)$ be the set of explanations found during search for the action S_i . For some explanation π , let $goal(\pi)$ be the character goal that is achieved (i.e. the condition of the character’s utility function that holds) at the end of the sequence.

For a goal g :

Definition 4a. $SALIENT(g, i, S) \iff \exists \pi : \pi \in explanations(S_i) \wedge goal(\pi) = g$

Since related work has emphasized the importance of goal motivations for comparing stories Amos-Binks *et al.* (2016, 2017), I include an alternative definition for intentionality that considers not just the goals that explain why the action occurred, but also any goals that are *motivated* as a result of the action. An action is said to motivate a goal when it changes the goal condition from being satisfied (true) to unsatisfied (false). For example, in *Grammalot* the guard wants to punish criminals: He has a goal for each of the other characters that they are either not a criminal, or have been punished. Initially this condition is true for Tom: He is not a criminal. If Tom commits a crime, it becomes false, so this motivates the guard’s goal to punish Tom.

Recall that $\alpha(S, s_0)$ denotes the state reached by executing the sequence S from the initial state s_0 .

Definition 4b. $SALIENT(g, i, S) \iff (\exists \pi : \pi \in explanations(S_i) \wedge goal(\pi) = g) \vee (\alpha(S_{1:i-1}, s_0) \models g \wedge \alpha(S_{1:i}, s_0) \models \neg g)$

Causality The *causality* dimension links events that are causally related, meaning the earlier action enables the later action. Planners can readily access this information through causal links or equivalent structures. Informally, two actions are causally linked when the earlier action has some effect that enables the later action, or enables some other action that in turn enables it, and so on. Below I give a formal definition, but the exact implementation is not important. What matters is that causal relationships between events are identified in some way.³

In Sabre, event effects are always represented as conditional effects in the form $p \rightarrow g$. We use a method based on UCPOP (Penberthy and Weld, 1992) to account for causal links through conditional effects: Any effect condition that holds true when the event occurs is treated as a precondition of that event for the purpose of identifying causal links.

A causal link $\langle S_i, S_j, p \rangle$ links parent S_i to child S_j (for $j > i$) via precondition p iff:

²Modeling belief changes the goal recognition task, as I have observed in previous work (Farrell and Ware, 2020). Since characters’ beliefs affect their plans and behavior, and are not directly observable, beliefs and goals must be inferred together. In other words, there are even more possible explanations for actions when characters are allowed to have wrong beliefs.

³For example, the current definition does not include causal links when an event sets a property that is already true. In some situations, such as when using numeric fluents, it may be better to include these (Farrell and Ware, 2017).

$$\alpha(S_{1:i-1}, s_0) \models \neg p \wedge \alpha(S_{1:i}, s_0) \models p \wedge \text{PRE}(S_j) \models p \wedge \neg \exists k : i < k < j \wedge \alpha(S_{1:k}, s_0) \models \neg p$$

Let $\text{ancestors}(S_i)$ be the set of causal parents of S_i and all actions in the transitive closure of this relation.

As discussed previously, three alternative versions of the causality definition are given, each of which uses a different type of entity (actions, fluents, and assignments). For an action a :

Definition 5a. $\text{SALIENT}(a, i, S) \iff a = S_i \vee a \in \text{ancestors}(S_i)$

Let $\text{fluent}(p)$ denote the fluent for which the literal p states some relation. For a fluent f :

Definition 5b. $\text{SALIENT}(f, i, S) \iff \exists k : (k = i \vee S_k \in \text{ancestors}(S_i)) \wedge (\exists p : \text{PRE}(S_k) \models p \wedge \text{fluent}(p) = f)$

For an assignment $f=v$:

Definition 5c. $\text{SALIENT}(f=v, i, S) \iff \exists k : (k = i \vee S_k \in \text{ancestors}(S_i)) \wedge (\exists p : \text{PRE}(S_k) \models p \wedge \text{fluent}(p) = f \wedge \alpha(S_{1:k-1}, s_0) \models f=v)$

Algorithm

Algorithm 2 summarizes a given story S with a set of five vectors representing the salience of all entities at the end of the story. Since the task is to compare complete stories, it captures summaries at the end. Summarizing incomplete stories could be accomplished simply by halting this algorithm at the desired step. The function $\text{SALIENT}(\text{entity}, \text{step}, \text{story})$ returns whether an entity is made salient by a particular action, based on its type, according to the definitions given above. By default, Definition 1a is used for protagonist, 4b for intentionality, and 5a for causality.

Let d be a constant decay factor between 0 and 1 (using 0.5 as a default). Let C be the set of characters in the domain, T the set of time frames, L the set of locations, G the set of character goals, and A the set of grounded actions. The set of entities E comprises these five sets. The algorithm begins by initializing the set of salience vectors V with five numeric vectors, each having length equal to the number of elements in the corresponding set in E .

Entities are assigned a zero salience value initially, representing no salience at all. The procedure then steps through the actions in the story, assigning the maximum salience value (1) to each entity involved in the current action, and decreasing the salience of all entities that are not involved. The resulting set of vectors V represents the salience of each entity in E at the end of the story.

For an example comparison, consider the two stories displayed in Table 3.1, which appear to be fairly similar—Tom either waits for night (story Y) or he doesn't (story X), but otherwise the same events occur: he walks to the crossroads and gets attacked by the bandit. Due to these similarities there should be a small distance value between them. Notice, however, that they share no identical action signatures. Many

Algorithm 2 Create salience vectors for a given story

```
procedure SUMMARIZE( $S, d$ )
2:    $E \leftarrow \{C, T, L, G, A\}$   $\triangleright$  the set of sets of entities
    $V \leftarrow \{v_1, v_2, v_3, v_4, v_5\}$   $\triangleright$  such that  $|v_i| = |E_i|$ 
4:   Initialize all values in  $v_{1...5}$  to 0.
   for  $i$  in  $1...|S|$  do
6:     for  $j$  in  $1...5$  do
       for  $k$  in  $1...|E_j|$  do
8:         if SALIENT( $E_{jk}, i, S$ ) then
            $V_{jk} \leftarrow 1$ 
10:        else
            $V_{jk} \leftarrow V_{jk} * d$ 
```

Story	Action Signature	Label
X	walk(Tom, Cottage, Crossroads, Day)	a_{1X}
	attack(Bandit, Tom, Crossroads, Day)	a_{2X}
Y	wait(Tom, Cottage, Day)	a_{1Y}
	walk(Tom, Cottage, Crossroads, Night)	a_{2Y}
	attack(Bandit, Tom, Crossroads, Night)	a_{3Y}

Table 3.1: Example stories X and Y

Step	v_1		v_2		v_3		v_4		v_5				
	T	B	Day	$Night$	$Cottage$	$Xroads$	g_T	g_B	a_{1X}	a_{2X}	a_{1Y}	a_{2Y}	a_{3Y}
1	1	0	1	0	1	1	1	0	1	0	0	0	0
2	.5	1	1	0	.5	1	.5	1	1	1	0	0	0

Table 3.2: Salience vectors after each step in story X

metrics rely on comparing sets of common actions, including ISIF distance, and would therefore deem these stories to be highly different.

The salience vectors after each step in story X are shown in Table 3.2 (using $d = 0.5$). The character goals are abbreviated as g_T (Tom’s goal to have the potion and be at the cottage) and g_B (the bandit’s goal to have Tom’s coin), and the action labels reference Table 3.1. Entities whose values remain zero across all vectors are omitted for space.

After step 1 (Table 3.2), the entities *Tom*, *Day*, *Cottage*, *Crossroads*, Tom’s goal, and the action a_{1X} itself—*walk(Tom, Cottage, Crossroads, Day)*—are salient. After step 2, the *Bandit*, the bandit’s goal, and the new action a_{2X} have become salient. Entities that are still involved remain salient: *Day*, *Crossroads*, and a_{1X} (since it is a causal ancestor of a_{2X}). Uninvolved entities have decayed: *Tom* (since he was not an actor in the attack), *Cottage*, and Tom’s goal.

Step	v_1		v_2		v_3		v_4		v_5				
	T	B	Day	$Night$	$Cottage$	$Xroads$	g_T	g_B	a_{1X}	a_{2X}	a_{1Y}	a_{2Y}	a_{3Y}
1	1	0	1	0	1	0	1	0	0	0	1	0	0
2	1	0	.5	1	1	1	1	0	0	0	1	1	0
3	.5	1	.25	1	0.5	1	.5	1	0	0	1	1	1

Table 3.3: Saliency vectors after each step in story Y

The final row of this table (representing the saliency vectors at the end of Algorithm 2), encodes important information about the story: that it ends with the bandit acting on his goal at the crossroads; that it never involves night or the other locations and characters (not shown); and which actions are most important.

Table 3.3 shows the vectors for the second story (Y), revealing that this story also ends with the bandit acting on his goal at the crossroads, but at night and with different actions.

3.4 Saliency Distance

The saliency distance SD between two stories x and y is defined as a linear combination of the distances between their corresponding saliency vectors of each type:

$$SD(x, y) = \sum_{i=1}^5 w_i * NSE(v_{ix}, v_{iy}) \quad (3.1)$$

where w_i are relative weights for each dimension, which sum to 1 (by default we use equal weights, $w_* = 0.2$), and $NSE(u, v)$ calculates the normalized squared Euclidean distance between a pair of vectors (their squared Euclidean distance after scaling their lengths to have unit norm). This is an appropriate function for this case because the magnitudes of the vectors are not important; only their directions. This accounts for the unbalanced sizes of the different vectors (e.g. the causality vector is likely to be much larger than the other four). The normalized squared Euclidean distance is always between 0 and 1, so the final saliency distance value is also bound between 0 and 1.

The saliency distance between stories X and Y from Table 3.1 is calculated using Equation 1 with $v_{1...5X}$ from the bottom row of Table 3.2, and $v_{1...5Y}$ from the bottom row of Table 3.3. Since the final v_1 , v_3 , and v_4 happen to be identical for these two stories (all characters, locations, and goals are equally salient at the end), these contributions to the equation will be zero. The differences come entirely from the time and causality vectors, i.e.:

$$\begin{aligned} SD(X, Y) = & 0.2 * NSE([1, 0], [0.25, 1]) \\ & + 0.2 * NSE([1, 1, 0, 0, 0, \dots], [0, 0, 1, 1, 1, \dots]) \end{aligned}$$

The result is a distance value of 0.296, which is relatively small, as intuition suggests it should be. The metric captures the similarity of some elements while recognizing differences in others. This example demonstrates a key benefit of this method: not only can it automatically calculate a useful distance metric to compare stories, but the vectors also reveal where their differences come from.

3.5 Distance Metric Evaluation

We compared the accuracy of salience distance to several other metrics, including edit distance, action set distance, ISIF, and several machine-learning based methods. This section describes first the dataset we collected from humans, then the various distance metrics included in the comparison, and finally the analysis and results.

Story Distance Dataset

To evaluate the accuracy of the salience distance metric we collected a dataset of human annotations of story similarity in the *Grammalot* domain (Section 3.2). Distance values from different metrics cannot be directly compared due to their differing scales (e.g. a salience distance of 0.2 and an ISIF distance of 0.2 do not have the same meaning). Therefore this dataset focuses on *comparisons* between two distance values. A comparison is a triplet of solutions $\langle REF, A, B \rangle$ where *REF* is a reference story and *A* and *B* are two other stories, and is interpreted as the question, “Which story is more similar to the reference: *A*, or *B*?” This question can be answered by any distance metric: If the distance between *REF* and *A* is less than the distance between *REF* and *B*, then its answer is *A*. If the two distances are equal then its answer is undefined; otherwise it is *B*. Human answers to these kinds of comparisons were collected as ground truth so that accuracy scores for the distance metrics could be obtained.

Subjects were recruited using the online crowd-sourcing platform Prolific. They read a description of the domain, then completed 8 tasks in which they read a reference story followed by a pair of stories (shown in random order). Subjects were randomly assigned a question type: whether they are asked which story is “more similar to” or “more different from” the reference (inverting their answers accordingly). To limit cognitive load, the reference story remained the same for a given participant throughout the whole task, as did the question type. Equality was not an option; *A* and *B* were the only available answers.

A breadth-first search of the *Grammalot* problem to depth 6 using Sabre produced 58 solutions. The stories used for comparisons were randomly selected from these, constrained only on the requirement that each participant would view only one reference story. First the solutions were randomly sampled for 8 different stories to use as references, then for each reference, 8 pairs of different stories for *A* and *B* were randomly sampled. Stories shown to participants were first translated into basic English sentences, and ranged from 2 to 6 sentences long. Data was collected from 64 participants, each answering 8 questions, for a total of 512 answers.

Metrics

This analysis includes the following distance metrics adapted from the literature. For each of these metrics we produced a distance matrix, containing pairwise distance values for all 58 of the *Grammalot* solutions used in the above dataset. For the methods that require text input, we used the same translations that were shown to participants.

Action Distance measures the Jaccard distance between the sets of grounded actions in each plan (Srivastava *et al.*, 2007). The Jaccard distance between two sets x and y is defined as: $J(x, y) = 1 - \frac{|x \cap y|}{|x \cup y|}$.

ISIF (Amos-Binks *et al.*, 2016) measures a combined Jaccard distance between two sets of grounded actions taken from each plan. The *important steps* (IS) of a story are those with the most causal connections to other steps. The *intention frame summaries* (IF) of a story are the motivating and satisfying steps of the plans characters enact in pursuit of their goals. The ISIF distance between two stories x and y is then defined as:

$$ISIF(x, y) = 1 - \frac{1}{2} \left(\frac{|IS_x \cap IS_y|}{|IS_x \cup IS_y|} + \frac{|IF_x \cap IF_y|}{|IF_x \cup IF_y|} \right).$$

Edit Distance counts the number of insertions, deletions, or substitutions required to convert one plan into the other (Levenshtein, 1966). Three variations are used: *Edit Action* applies these operations to whole action signatures; *Edit Symbol* applies them at the symbol level (the action name and parameters); and *Edit Word* applies them at the word level using the English translation of the story set.

BLEU is an automatic evaluation metric for summary comparison that evaluates textual similarity as co-occurrence of subsequences between a target text and a reference text (Papineni *et al.*, 2002).⁴ BLEU considers a combined score from multiple lengths of n-grams; all possible equally-weighted groupings of these n-gram lengths were tested. To present the range, the lowest-scoring BLEU method (using only unigrams or unigrams and bigrams) and the highest score (e.g. using only trigrams—most combinations tied for this score) are reported.

Rouge is another text comparison metric similar to BLEU, but considers a single n-gram length or the longest common subsequence (Lin, 2004). Rouge was tested using n-grams 1 through 5, and Rouge-L (longest common subsequence). The lowest Rouge score was a tie between Rouge-1 and Rouge-L; the highest between Rouge-3, Rouge-4, and Rouge-5.

BERT (Devlin *et al.*, 2019) is an English language text transformer which encodes similarities of input texts by placing them in a latent space. Each story was processed as a sequence of word-level tokens, with special tokens to indicate

⁴*REF* is used as the reference to which *A* and *B* are compared.

where sentence separations occur and where the story starts and ends. The final predicted position of each story in latent space was extracted, and distance measurements were calculated as the cosine distance between two of these positions. This was tested both with an unmodified instance of the base transformer and with five tuned instances. For these, the text stories were processed into two datasets for BERT’s Next Sentence Prediction and Masked Language Modeling tasks. The five instances were trained for these tasks for some number of training epochs (1...5). Note that this does not fine-tune BERT on the task of measuring distance between stories, but on generating text more like these stories. The lowest score (using the unmodified BERT), and the highest score (the instance with one training epoch) are reported.

All 18 possible combinations of the definitions presented in Section 3.3 are included in this analysis. Section 3.5 shows results for the default variation (using Definitions 1a, 4b, and 5a), along with the best and worst scores among all 18 variations (labeled best variation and worst variation). The best score was a tie between the two variations that used Definition 1a for protagonist and 5b for causality (using either 4a or 4b for intentionality). The worst variation used Definitions 1c, 4b, and 5c. Results for all variations are included in Appendix A.

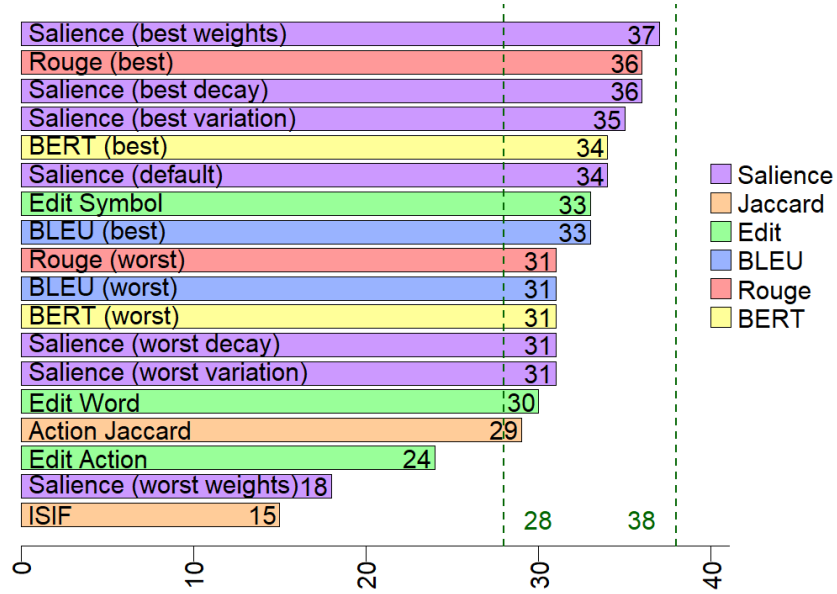
Different index weights were tested using the default vector definitions and the default decay rate $d = 0.5$. This included all possible combinations of the five weights $w_{1...5}$ ranging from 0 to 1 in increments of 0.1. The results in Section 3.5 show the best and worst scores achieved using modified weights (labeled best weights and worst weights). The weights that achieved the highest scores are shown in Appendix B. They were different between the two analyses and there were many ties, but one clear trend is that time should be weighted low or zero for the most accurate performance in this domain.

Likewise, different decay rates were also tested ($0.05 \leq d \leq 0.95$, in increments of 0.05), using the default weights and vector definitions. Scores for the best and worst decay are reported in Section 3.5 (best ≤ 0.25 , worst = 0.95).

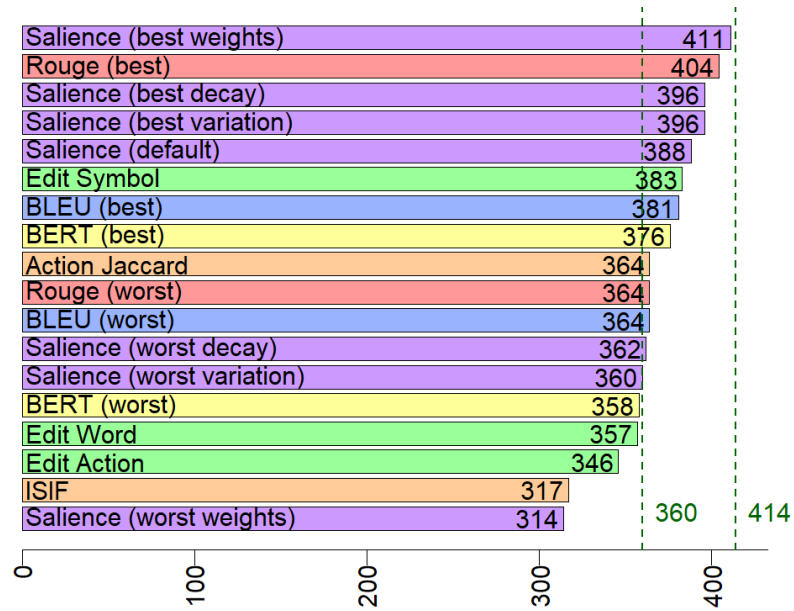
Results

The metrics were compared in two ways; first using only the questions where people agreed on an answer, and second using all the questions. Participants significantly agreed if at least 7 out of 8 provided the same answer (binomial test, $p = .039$), which occurred for 37 of the 64 questions. Figure 3.2a shows the results of the analysis using these questions, in which a metric scores one point for each question it answers the same as the majority of humans. Undefined answers (equal distances) are counted as incorrect.

A binomial test indicates that a score of 24 or higher is statistically better than chance. Most metrics exceed this threshold. The default variation of Saliency answers 34 out of 37 questions correctly (92% accuracy), which is higher than all other metrics



(a) Using the 37 questions humans agreed on



(b) Using all questions

Figure 3.2: Distance metric accuracy

except the best BERT (with which it is tied) and the best Rouge. A z-test of two proportions was used to assess whether any of these differences are significant. The green dotted lines indicate the thresholds where the test becomes significant: A score of 28 or below is significantly worse than 34, and only 38 or above, which is impossible, would be significantly higher. Saliency distance performed very well in this evaluation. It was significantly more accurate than ISIF and Edit Action, and scored higher than almost all of the other metrics, though not significantly higher.

Accuracy was also measured using all 512 answers collected, scoring one point for each individual human answer the metric agrees with. This effectively weighs the score for each question by how strongly people agreed on it. Half a point was awarded for undefined answers, since in this case people did not necessarily agree that one answer is correct. This means that providing a wrong answer is worse than providing no answer; and that when people were evenly split (4 out of 8 subjects answered A, and the other 4 answered B—this happened for 5 out of 64 questions), all three possible answers (A, B, and undefined) are worth the same 4 points. The highest possible score is 420, which represents agreement with the majority on every question.

Figure 3.2b shows the results of this analysis. The default Saliency scores 388, and again the low and high z-test thresholds are shown, 360 and 414. Note that no p -value correction is being applied, so one might reasonably consider comparisons on the thresholds to be insignificant. Here the default Saliency outperforms all other metrics except the best Rouge, and significantly outperforms the unmodified BERT and Edit Word in addition to Edit Action and ISIF.

Discussion

Both analyses suggest that intelligently choosing values for w_* and d may improve accuracy, although the improvements here were not significant. Equal weighting appears to be a good default, but better weights may be informed by domain knowledge (e.g. dimensions or entities that are more or less important). The ideal decay value is also likely to vary by domain, but may be related to measurable properties like average story length.

None of the 18 variations of Saliency were significantly better or worse than the default, with accuracy ranging from 84-95%. This suggests that the technique does not strictly depend on one set of definitions, and may be adaptable to other kinds of story systems. The selection of default definitions here is based on two factors. When possible I prefer those that utilize more of the planning model; Definition 1a considers all characters and distinguishes between actors and non-actors, so it is more nuanced than 1b and 1c. Definition 4b for intentionality is more nuanced than 4a because it considers goal motivations. In the case of causality, I prefer the definition that yields the most human-readable feature names, i.e. Definition 1a using actions. This is important for the work described in the next section; actions make more intuitive cluster labels than fluents or assignments, and provide information in a form people are more likely to expect in this context.

While most of the metrics performed well in both analyses, ISIF is a notable exception. ISIF and Action Jaccard both have a tendency to produce distance values of 1, e.g. for stories that do not contain identical action signatures (such as the example pair discussed in Section 3.3). This results in many undefined answers, i.e. when both distance values are 1, which explains why they perform better in the second analysis than the first (undefined answers are worth half credit). Still it was surprising to see ISIF perform worse than Action Jaccard, upon which it is based. The difference is that Action Jaccard compares the full set of actions in both stories, whereas ISIF only compares actions of specific importance. These action sets may be helpful toward describing important story differences, but as a distance metric, this study shows it to be less accurate than considering the full sets of actions (at least in this domain).

The text-based methods generally performed well, with the best being Rouge-3+. This is a viable approach for measuring story distance, as long as the stories can be automatically translated into natural language. Note that these methods are sensitive to different text realizations; the translations used here were simple and repetitive, which is helpful to these metrics. Edit Symbol also performed well, and does not rely on text translations, but does require a plan-like symbolic representation. Saliency is comparable in accuracy to these approaches, does not depend on text translations, and does not necessarily depend on a planning representation either. (For example, if the characters, locations, etc. can be extracted from text stories using natural language processing, then the same method can be used to measure distance between stories that never involved a planner.) Furthermore, the primary benefit of saliency distance is that the saliency vectors encode important semantic information about the stories that can explain where their differences and similarities come from. The other methods, apart from ISIF, do not have this capability.

Saliency with best weights was the only metric to correctly answer all 37 questions on which subjects agreed; but this is not a fair metric, since we cannot assume the optimal weights are known. I include the following example in this discussion for possible insight into how this technique might be improved in the future. No fair metric answered this question correctly, despite human subjects agreeing on its answer with a 7/8 majority.⁵

(Reference) Tom walks to crossroads. Bandit walks to market. Guard walks to crossroads. Bandit walks to crossroads. Bandit attacks Tom.

(A) Tom walks to crossroads. Tom waits for night. Bandit attacks Tom.

(B) Tom walks to crossroads. Tom walks to market. Tom buys potion. Bandit walks to market. Bandit attacks Tom.

According to the majority of subjects, story *A* is more similar to the reference than story *B* is; yet almost all of the metrics answered that *B* is more similar. *B* contains more of the same actions and content as the reference compared to *A*, so why do humans think *A* is more similar?

⁵The story text has been abbreviated here for space.

In the reference, Tom goes to the crossroads and *does nothing*, while others walk in and out of the market, and then he gets attacked. It may seem like Tom bears some responsibility for this fate—if he had done something instead of nothing, he might have avoided it. Story *A* involves waiting for night, which is semantically similar to doing nothing. In *A*, Tom goes to the crossroads and *allows* himself to be attacked, while in *B*, he tries to achieve his goal but gets attacked in the process. In this light, *B* feels different, despite containing many of the same elements as the reference and being similar in structure and text.

Of course this is just one possible interpretation of why subjects chose *A* over *B*. It may be that waiting is similar to doing nothing, or perhaps this example reflects another type of similarity that the metrics are not accounting for, e.g. thematic or high-level structural similarities. Whatever the case, this example demonstrates that some semantic similarities are not captured by any of these metrics. Future work may benefit from exploring ways to account for situations like this.

Limitations

This study only used a single domain due to limited resources; it was preferable to get a lot of data from one domain than less data from multiple domains. This domain was chosen because it contains multiple characters, goals, and ways for characters to achieve their goals in a small number of steps. It models enough variety to capture some differences between metrics, but perhaps a larger domain would be able to capture more significant differences. In the future, if more resources become available, these results could be replicated in other domains and with more data to allow stronger conclusions to be drawn.

The salience distance metric was shown to be effective under varying definitions, but this was not tested exhaustively. The technique is theoretically applicable for any system that can identify some reasonable adaptation of its definitions, although there may be some situations that do not work well. The definitions themselves leave room for improvement, especially as models of intentionality continue to improve. Time and space could also be represented hierarchically, and other situational dimensions could be represented beyond the five discussed, e.g. emotions, ideas, or objects.

3.6 Clustering and Visualization

Thus far I have defined a numeric vector that summarizes a story’s content by calculating the salience of each entity involved in the story upon its ending. Distances between these vectors can model the extent to which two stories are semantically different from each other. I showed that the salience distance metric is highly accurate according to humans in an example domain and compares favorably to existing methods for measuring story distance. The salience method is unique in that the basis for the distance measurement is a summary of each story, based on a cognitive model of story comprehension, which contains information that can explain the stories’ similarities and differences in detail. In this section I utilize that information to

summarize the results of solution space clustering, and evaluate the technique as a potential domain authoring tool.

As a starting point for this tool, I use hierarchical clustering to break down the solution space into clusters and label them using relevant information from the salience vectors. This can provide domain authors the ability to explore key parts of the space with more control and better understand what kinds of stories the domain is modeling. Hierarchical clustering begins by assigning every point (in this case, a story) to its own cluster, and then merges the two most similar clusters together. This process is repeated until all points have been merged into the same cluster. The resulting graph is a dendrogram showing the order in which the points were merged—a hierarchical breakdown of the story space. The dendrogram allows us to choose any number of clusters and see how the space is best separated into that many groups.

Hierarchical clustering begins with a distance matrix. Let S be a solution set, and $D_{|S| \times |S|}$ be a matrix produced using the salience distance metric, where $D_{ij} = SD(S_i, S_j)$. Salience distance is symmetric, so $D_{ij} = D_{ji}$. A hierarchical clustering of D produces a dendrogram like the example shown in Figure 3.3. I use Ward’s linkage method (Ward, 1963) to determine distance between clusters.⁶

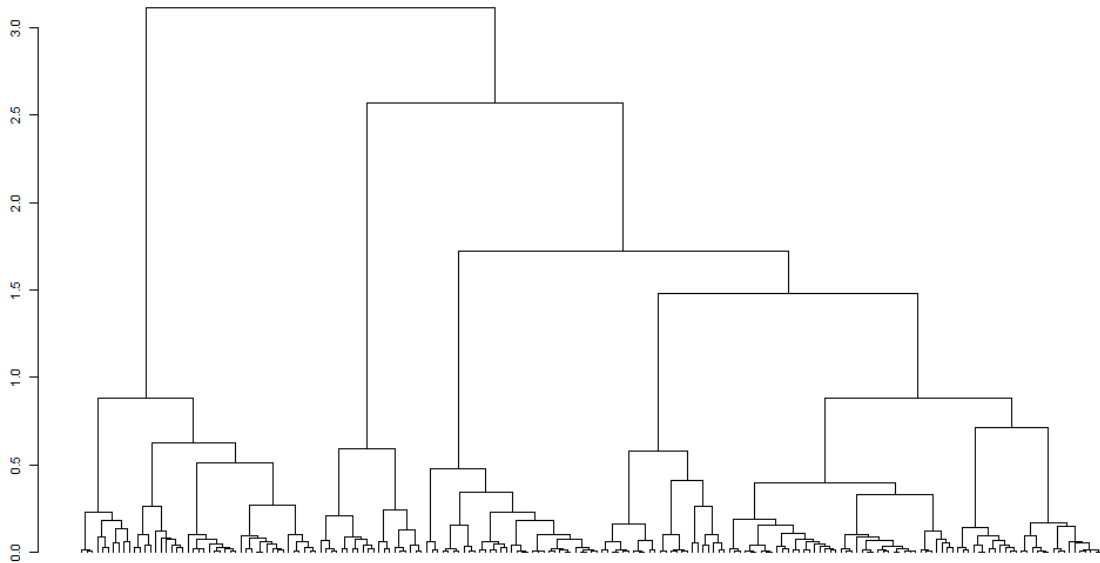


Figure 3.3: Hierarchical clustering of the 58 *Grammarlot* solutions

By itself, this graph can provide a road map that allows domain authors to save time in getting a broad sense of what is possible in the domain by reading only a select group of stories (e.g. one from each branch at a certain height). Still, this amount of reading becomes tedious when trying to understand minute differences between

⁶Linkage criteria specify which points of a cluster to compare—e.g. the center of each cluster (average linkage), or the points from each cluster that are closest together (single linkage), etc. Ward’s method measures the distance between two clusters as the increase in the error sum of squares after merging the two clusters together, similar to how differences between groups are calculated in statistical methods like analysis of variance (ANOVA).

different story spaces (e.g. after a domain modification). I propose a cluster labeling procedure that summarizes the similarities within each cluster, so that authors can quickly ascertain information about those stories without actually having to read them. This technique uses the salience vectors from which the distance matrix was derived, defined in Section 3.3, and the vector features, which are the names of the story entities (E in Algorithm 2).

The goal is to label each cluster with the name of an entity whose prominence in the target cluster distinguishes that cluster from the rest of the stories in the space. Ideally, the entity would be present in all stories in the cluster (having salience > 0 in the vectors for those stories), and absent in all other stories (salience $= 0$). However, there is no guarantee that such an entity will exist for every cluster, so it is necessary to identify the entity which comes the closest to this ideal. That is, the entity having the highest difference between its average salience in the target cluster and its average salience in the other clusters. This concept is applied in two different ways. The first method cuts the dendrogram into a set of clusters and labels them; the second method directly labels the edges of the dendrogram.

Cluster Labels

There are several approaches for determining the optimal number of clusters k for a given dataset. The examples here use the silhouette method (Zhou and Gao, 2014), but others may be equally suitable. To avoid displaying too many clusters, I constrain the value of k such that $(2 \leq k \leq 5)$. If the best value of k according to the silhouette method is not in the desired range, the second best value is used (and so on). The dendrogram is then cut at the height that produces k clusters, and the following procedure is used to label each one. This labeling method could also be used with other clustering algorithms such as k -means that simply produce k clusters in the first place.

Let V be the set of salience vectors for the stories in the target cluster, and its complement \bar{V} , the set of vectors for all stories in the solution set that are not members of this cluster. Equation 3.2 finds the feature f of these vectors for which the difference between its average salience in the target cluster and its average salience in the other clusters is the highest:

$$\arg \max_f \frac{\sum_v^V v_f}{|V|} - \frac{\sum_v^{\bar{V}} v_f}{|\bar{V}|} \quad (3.2)$$

A label can now be assigned for the target cluster using the name of the feature f . To clarify the meaning of the labels I add the words “is important” after the feature name, e.g. “The bandit is important”. Entities that are named using planning syntax (actions and goals) are translated into natural language, e.g. “The guard arresting the bandit at the market” or “The bandit wanting Tom’s coin”. To convey the relative sizes of the clusters, I depict each cluster as a circle with size scaled proportionally to the number of stories in the cluster—for this I use the open-source graph drawing tool GraphViz (Ellson *et al.*, 2001). Figure 3.4 shows the result of this procedure for

an example problem in a modified version of the *Grammalot* domain (which I will formally introduce in Section 3.7).

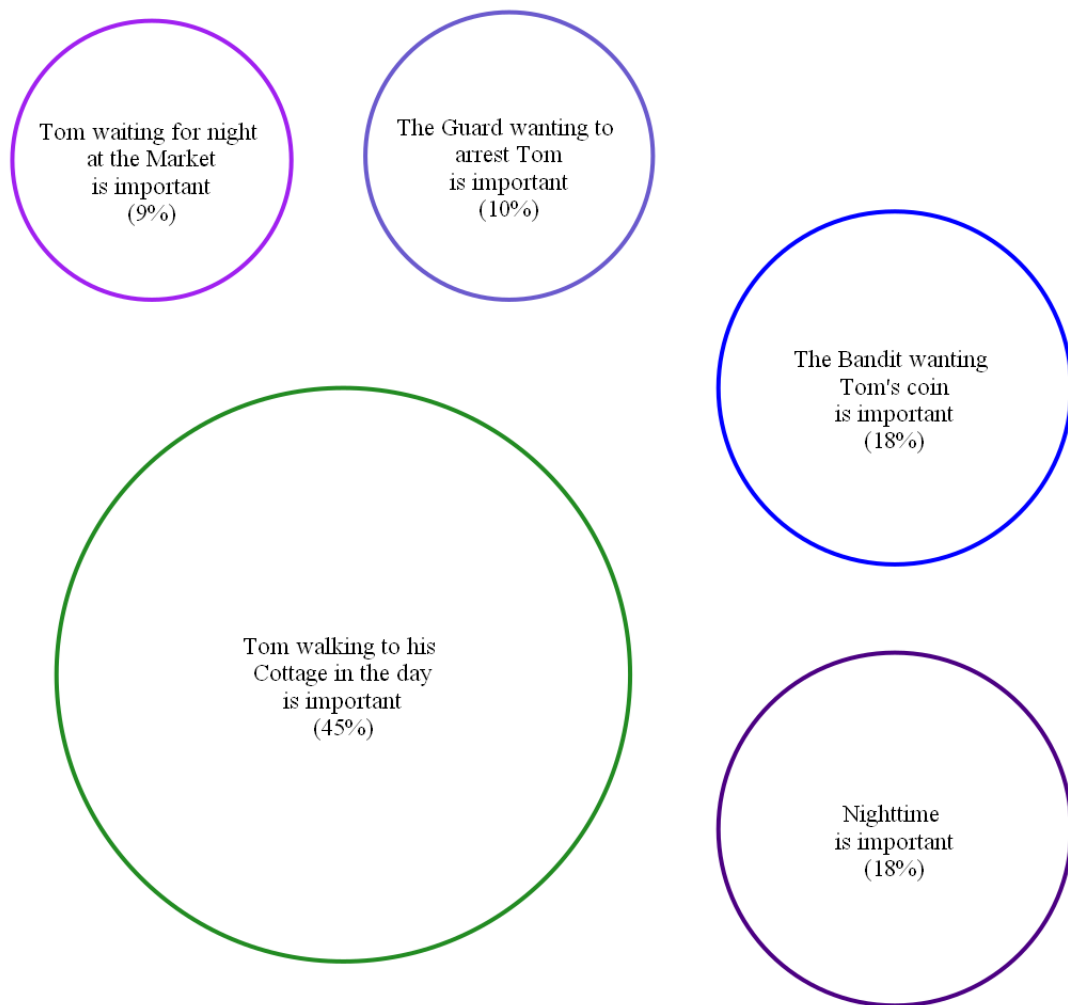


Figure 3.4: Example cluster summary for *Grammalot* problem #193

The result is a high-level summary of the solution set. Without reading any stories, domain authors can see what types of stories exist within the solution space and how many of each type. By viewing an image like this after every domain modification, the author can quickly spot where the recent changes have significantly impacted the solution set. Additional features could then allow them to examine stories within a specific cluster. I test a version of this tool in the evaluation presented in Section 3.7.

Dendrogram Labels

A similar procedure can be used to label the edges of the dendrogram, but in this case some additional factors need to be accounted for. Here, the labels should explain each “split” in the tree, i.e. distinguish specifically between a target branch and its sibling (the only other child of its parent), rather than the full complement of itself.

There is no need to consider stories that are not clustered under the parent of the target cluster, because that distinction will be described by the parent’s label.

Additionally, the label for a given cluster should be balanced between the cluster’s two children. Consider, for example, a cluster whose right child contains 90% of its stories, and the left child only 10%. A feature which is prominent in its right child, but absent in its left child, may still have a high average salience in the cluster. It is important not to label this cluster using such a feature, since the feature is not representative of the cluster as a whole (it would make a better label for the cluster’s right child). The equation below therefore considers the balance of a feature’s salience between the cluster’s two children. In summary, it seeks a feature that is (1) highly salient in this cluster, (2) minimally salient in the sibling of this cluster, and (3) evenly distributed between the two children of this cluster.

Let n be a node in the dendrogram. Let V_n be the set of vectors for all stories clustered in n , and $V_{n'}$ the set of vectors for the stories in the sibling of n . Let V_{lc} and V_{rc} be the vectors for the left and right children of n , respectively. Assign a label for node n using the feature f that maximizes this formula:

$$\arg \max_f \left(\frac{\sum_{v \in V_n} v_f}{|V_n|} - \frac{\sum_{v \in V_{n'}} v_f}{|V_{n'}|} \right) - b * \left| \frac{\sum_{v \in V_{lc}} v_f}{|V_{lc}|} - \frac{\sum_{v \in V_{rc}} v_f}{|V_{rc}|} \right| \quad (3.3)$$

where b is a constant between 0 and 1 weighing the relative importance of the balance between the children as compared to the primary calculation. For the images presented here I use $b = 0.25$. I also impose a feature significance threshold of 0.5, to account for situations where the best label found is not very descriptive of the cluster. If the maximum value of the above formula is less than this threshold, I do not use that feature (i.e. no label is returned). When multiple features tie for the maximize value, the feature with the *lowest* mean salience in n is returned.⁷

As with the previous diagram, the label comprises the feature (translated into natural language if it is an action or goal) and the words “is important”. Algorithm 3 recursively labels the dendrogram nodes in a depth-first order beginning at the root, and excluding labels that have already been used for ancestors of the current node.⁸ When no label is found, an inversion of the sibling’s label is used instead. If both siblings return no label, those branches remain unlabeled. The function `GETLABEL(node, excludeList)` returns the feature f that maximizes Equation 3.3 for this node, ignoring all features in the exclude list, or null if the maximum value did not exceed the feature significance threshold. The function `INV(label)` replaces the text “is important” in the label with “is NOT important”, or returns null if the label is null.

⁷The goal of this tie-breaker is to encourage the labels to be ordered more chronologically from root to leaves when possible. Parents are labeled before children (see Algorithm 3), and labels cannot be reused in the same branch, so choosing the less salient of the tied features first—i.e. the one which tends to be present earlier in the stories—allows the more salient (later) feature to be used later in the branch.

⁸Each node is responsible for labeling its children. The root node itself does not receive a label since it describes the whole space rather than a single cluster.

Algorithm 3 Label the nodes of a dendrogram

For the initial call, n is the root of the dendrogram, $depth = 0$, and $excludeList$ is an empty list.

```
function LABELTREE( $n, depth, excludeList$ )  
  if  $d \leq depth_{max}$  and  $size(n) \geq minClusterSize$  then  
    Let  $n_L$  and  $n_R$  be the left and right children of  $n$ , respectively.  
    Set the label of  $n_L = \text{GETLABEL}(n_L, excludeList)$   
    Set the label of  $n_R = \text{GETLABEL}(n_R, excludeList)$   
    if label of  $n_L$  is not null then  
      Append label of  $n_L$  to  $excludeList$ .  
    else  
      Set label of  $n_L = \text{INV}(\text{label of } n_R)$   
    if label of  $n_R$  is not null then  
      Append label of  $n_R$  to  $excludeList$ .  
    else  
      Set label of  $n_R = \text{INV}(\text{label of } n_L)$   
    LABELTREE( $n_L, depth + 1, excludeList$ )  
    LABELTREE( $n_R, depth + 1, excludeList$ )  
  
  return
```

I display the labeled tree using GraphViz to avoid overlapping labels while fitting the final image into a desired size.⁹ Figure 3.5 shows the labeled tree for the same example problem summarized in Figure 3.4. For this image and those used in the evaluation below, the maximum depth of the tree is limited to 4, both to constrain the image size and to avoid presenting too much information. I also impose a minimum cluster size of 5. If a cluster is more than 4 branches deep or contains fewer than 5 stories, I do not label it. All unlabeled branches are cropped from the displayed tree, and clusters with inverted labels are colored red.

The labeled tree shows more detailed information about the solution space than the k -clusters graph defined in Section 3.6. For example, it is apparent from both graphs that in 10% of the stories, the guard’s goal to arrest Tom is important. This means that Tom likely commits crimes in these stories. The circle graph stops there, but the tree graph further reveals that there are two primary ways for this to occur. Either Tom robs the potion from the merchant, or the bandit attacks the merchant (allowing Tom to take the potion from her, which is still a crime since it does not belong to him).

These two graphs are both useful, and have different strengths and weaknesses. The circle graph is less informative, but it is easier and faster to read. It also draws attention directly to the clusters that are most important, whereas the tree graph may obscure that information. Both diagrams provide a high-level summary of the

⁹This conversion retains the structure of the dendrogram but loses information about the height of each cluster split. If display size were not a constraint, it would be preferable to label the actual dendrogram and keep the height information intact.

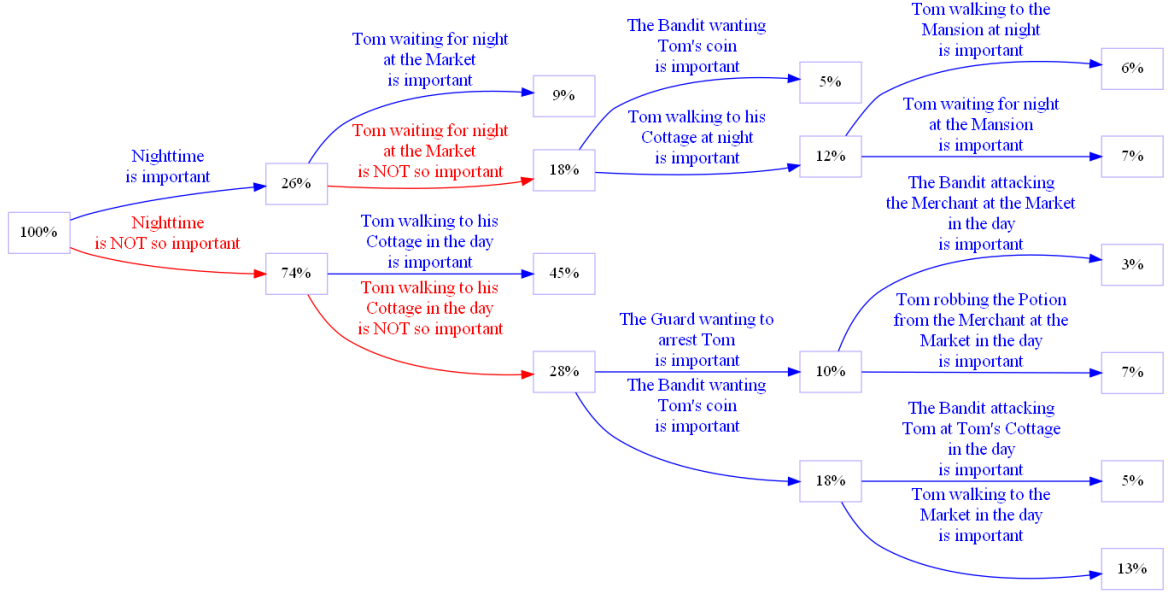


Figure 3.5: Example tree summary for *Grammalot* (problem 193)

solution space and a means of exploring the stories more intelligently (e.g. click on a specific cluster to read or sample those stories). The goal of the tree graph is to provide as much information about the clustering as possible, while the circle graph aims to show just as much as is needed to convey the high-level groupings without overwhelming the reader. In the next section, I evaluate both of these graphs as domain authoring tools and find that while they have some different effects, both graphs are able to help people better understand solution spaces.

3.7 Evaluation of Solution Space Summaries

The final evaluation demonstrates that both the tree and circle diagrams described above can improve people’s understanding of a solution space—operationalized as their ability to correctly answer questions about whether or not a specific story is part of that space. Participants were recruited and paid through Prolific to configure a story domain and then answer questions of this type about the domain they configured. They were incentivized to consider the questions carefully via possible bonus payments for answering either at least 90% or at least 50% of the questions correctly. Subjects were given one of three different tools—either a baseline tool, or a tool using one of the two graphs—to help them in these tasks.

It is important to point out that these tools are meant to help domain authors better understand the output of a creative task in which they are highly engaged. Due to limited resources it was not feasible to conduct this study using only individuals who represent potential domain authors (e.g. people with interactive narrative experience). This evaluation uses crowd-sourced survey takers, which comes with a known risk of low engagement, and this task in particular is mentally demanding. To account for this, I allowed participants to engage with the summary tools as much

or as little as they wanted throughout the task, and used metrics derived from their activity logs to classify their overall engagement level when analyzing the results. This study finds no significant effect among the whole set of participants, but this group includes those who interacted with the tools very little, if at all. Among those who interacted a lot, demonstrating high engagement, both of the summary tools significantly improved users’ accuracy over the baseline.

Domain Configurations

In a perfect world, we would have asked study participants to create their own story domain with the help of one of the tools, and then evaluated how well they understood their domain and how helpful they found the tool to be. This approach is impractical for two reasons: First, it would require training many people to do a very technical task. Second, it would have required the tools to run online, including the time-consuming step of searching for all solutions to the problem (and doing so many times while the domain is being modified). Instead, we settled for the closest reasonable approximation of this design: We asked participants to simply configure several predetermined properties of an existing domain, for which all the necessary searches could be precomputed.

For this experiment we used the *Grammalot* domain discussed in Section 3.2, but removed the crossroads location to reduce the size of the search space and allow more solutions at smaller depths. The three remaining locations are now connected directly to each other as in Figure 3.6. The bandit’s initial location is now the market since he can no longer be at the crossroads.

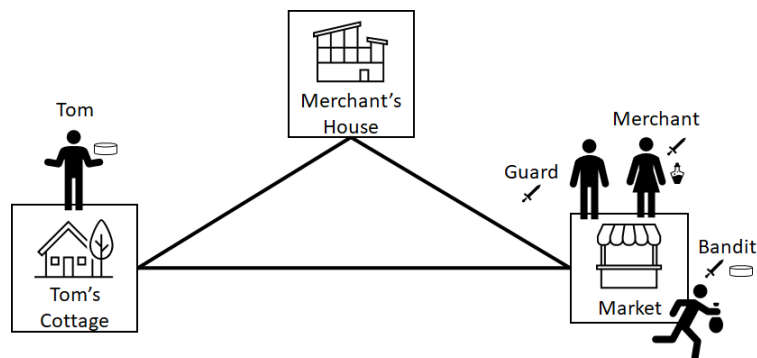


Figure 3.6: Depiction of the *Grammalot* initial state without the crossroads

Subjects each configured a version of this domain by choosing values for the eight binary configuration variables listed in Table 3.4. In total there are 256 possible domain configurations, varying several different aspects of the domain: a belief in the initial state (C_1), two preconditions (C_2 and C_3), two effects (C_4 and C_5), an observation function (C_6), and two utility functions (C_7 and C_8).

The *Grammalot* domain previously discussed in this document (Section 3.2) most closely matches, but is not identical to, problem #193 (11000001), in which: The bandit initially knows about Tom’s coin ($C_1 = \top$), being armed keeps characters

C_1	\top : Initially the bandit knows that Tom has a coin. \perp : Initially the bandit does not know that Tom has a coin.
C_2	\top : Armed characters are safe from being robbed. \perp : Armed characters are not safe from being robbed.
C_3	\top : Armed characters are safe from being attacked. \perp : Armed characters are not safe from being attacked.
C_4	\top : The guard arrests criminals by tying them up in the current location. \perp : The guard arrests criminals by taking them away to an unknown location.
C_5	\top : Taking an item from a sleeping character wakes them up. \perp : Taking an item from a sleeping character does not wake them up.
C_6	\top : All characters observe the effects of armed robbery. \perp : Only characters in the same location as the robbery observe its effects.
C_7	\top : The merchant may or may not commit crimes to achieve her goals. \perp : The merchant is unwilling to commit any crimes to achieve her goals.
C_8	\top : Tom may or may not commit crimes to achieve his goal. \perp : Tom is unwilling to commit any crimes to achieve his goal.

Table 3.4: *Grammalot* domain configurations

Min	1st Qu.	Med.	Mean	3rd Qu.	Max
40	89.5	173	212.9	260.8	828

Table 3.5: Number of solutions per problem

safe from being robbed ($C_2 = \top$), but not attacked ($C_3 = \perp$), characters disappear when arrested ($C_4 = \perp$), sleeping characters do not wake up when interacted with ($C_5 = \perp$), robbery is not universally observed ($C_6 = \perp$), and the merchant is not willing to commit crimes ($C_7 = \perp$), but Tom is ($C_8 = \top$). This is the problem summarized in Figures 3.4 and 3.5.

For each of these 256 problem configurations, all solutions up to 5 steps long were collected using Sabre’s breadth-first search. The number of solutions for these problems ranged from 40 to 828, as summarized in Table 3.5. Each solution set was clustered using the procedure outlined in the previous section, and both types of visual summary (circle and tree graphs) were created for each.

Procedure and Summary Tools

Participants were tasked with helping a “storytelling robot” (Figure 3.7) with its job of configuring a story world that could tell lots of different stories—specifically, stories that would fit on its small screen. They were explicitly told that this means stories could be at most 5 sentences long. To reinforce the length limit, stories were always displayed against the same backdrop, representing the robot’s screen, which clearly

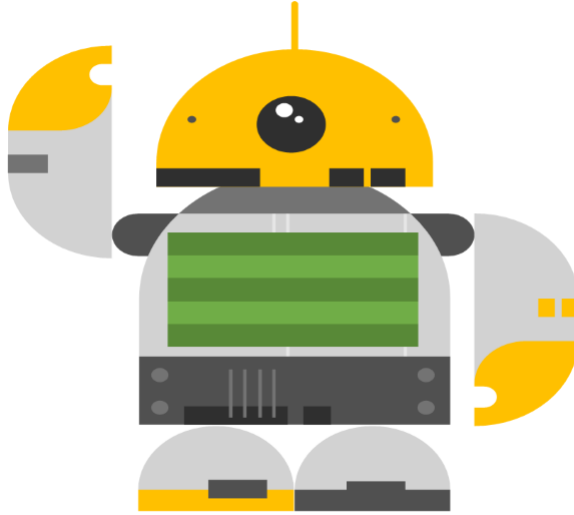


Figure 3.7: Storytelling robot

fits exactly five lines.

The task began with the robot describing what it has decided about the story world already (a description of the domain that pertains to all configurations). It then asked the participant to help answer a few remaining questions. Participants initially provided an answer to each of the eight questions representing the configurations listed in Table 3.4. They were informed that they would be allowed to change these configurations later. Once the questions were answered, an initial configuration was identified, and a summary tool determined by the participant’s grouping was displayed.

The baseline tool (Group 0) displays a random story from among the entire solution set of the currently configured problem, and allows the user to randomly sample a new solution using a button labeled “Generate Another” (Figure 3.8). The same story display box is also used for the other two tools, but where different subsets of the solution space can be sampled. The baseline tool always samples the entire solution set.

The tree graph tool (Group 1) displays the labeled tree diagram (Section 3.6) of the configured problem, with clickable nodes. When a node is clicked, the story display box appears, showing a random story from the cluster that was clicked. The “Generate Another” button is configured to continue sampling the same clicked cluster. The tree diagram remains on screen so that the user can scroll between it and the story display box. The clicked cluster on the tree remains highlighted while the story display box is configured to sample that cluster, and the cluster label is used as a title for the story display box—e.g. “Generating stories where: the Merchant is important”. (For the baseline tool, the story display box is untitled.) The user can close the story display box to view only the diagram, or click another cluster to reconfigure the display. The circle graph tool (Group 2) is identical to the tree graph tool except that it uses the circle graph (Section 3.6) instead of the tree.

Participants were asked to try to understand what is possible in the story world

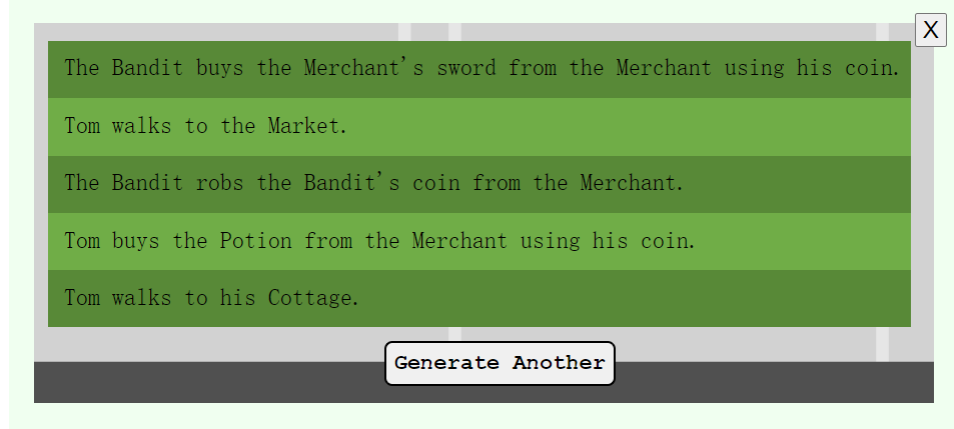


Figure 3.8: Story display box

they have just configured using the summary tool provided. They were allowed to interact with the tool as much or as little as they wanted before proceeding. On the next page eight toggles were added to the screen, one for each configuration option. When a toggle is changed, the underlying domain configuration is changed and the summary is updated accordingly; for Group 1 and 2 the summary image is replaced with that of the newly configured domain, and for Group 0 a new story is randomly sampled from the new domain. Participants were asked to configure the story world to their liking, and to try to understand what is and is not possible in that world before moving on so that they could accurately answer questions about it on the next page. On the questions page they were allowed to continue using the tool, but this was intentionally not mentioned beforehand because it would likely encourage them to skip this phase.

Questions were displayed one at a time, each using one of the 14 stories listed in Appendix C. The question asked, “Is the above story possible in your story world?”, with answer choices “Yes, it is possible” or “No, it is NOT possible”. Accuracy scores were obtained from each participant based on the number of these questions they answered correctly, as determined by the domain configuration they had selected.

In addition to directly measuring accuracy, the study also assessed how well subjects thought they understood the solution space. This was measured in two ways. First, after each story question, a followup question asked how confident the user felt about their answer on a scale from “Not confident at all” (1) to “Absolutely confident” (5). Second, the study ended with a qualitative survey targeting perceived understanding of the space using Likert scale agreement ratings with statements like “I have seen an example of every type of story that is possible in my story world.” (see the full survey in Appendix D).

Perceived understanding was measured, but no hypothesis was made for how the summary tools would affect these measurements. This is because two conflicting effects were expected. On one hand, if the tools help people understand the space better, then people who use the tool should rate their understanding of the space higher than people who did not use the tools. This assumes people are accurate in

judging their understanding. On the other hand, it is reasonable to expect a Dunning-Kruger effect (Kruger and Dunning, 1999). People whose actual understanding of the space is limited may overestimate their knowledge, whereas people with improved understanding (via the summary tools) should be able to assess their knowledge more accurately. In truth, solution spaces are very difficult to fully comprehend, so the most accurate answer is likely to be lower than the baseline.

Data was collected from 120 individuals. Two responses were discarded due to invalid logs (the participant completed the survey twice). Of the 118 valid responses, 40 were in the control group, 38 in the trees group, and 40 in the circles group.

Stories

All participants were asked about the same 14 stories, which are listed in Appendix C. The stories were selected based on their dependencies on the eight domain configurations. A story depends on a configuration if that story is only possible when that configuration is set a certain way (either true or false). For example, consider the story labeled Q_5 , which has two dependencies, $C_2 = \perp$ and $C_3 = \top$:

Tom walks to the Market.
Tom buys the Potion from the Merchant using his coin.
The Bandit robs the Merchant’s sword from the Merchant.
The Guard arrests the Bandit.
Tom walks to his Cottage.

This story is only possible if armed characters are *not* safe from being robbed ($C_2 = \perp$), since the merchant is armed in step 3 when the bandit robs her. It is also only possible if armed characters *are* safe from being attacked ($C_3 = \top$), although the reason for this is less obvious. The bandit is already armed so he has no need for the merchant’s sword; his only reason for stealing her sword is to cause her to become unarmed, thus allowing him to attack her and take Tom’s coin. The bandit’s explanation for step 3 (rob the sword, attack, take the coin) would not be valid if armed characters were not safe from being attacked, because there would be a strict subsequence of that plan that is possible and achieves the same utility (attack, take the coin). Therefore step 3 would not be explained and this would not be a solution. Provided that $C_2 = \perp$ and $C_3 = \top$, this story is always possible, i.e. it does not depend on any other domain configurations.

Table 3.6 lists the configuration dependencies for the 14 stories used for questions. The stories were selected so that all configurations are used at least once. If possible, both values of each configuration are used, though in many cases only one option can enable stories. For example, when Tom is unwilling to commit crimes ($C_8 = \perp$), this removes many stories from the solution space but does not add any. Configuration C_5 is only used once because it only enables two stories and they are very similar. Some configurations only enable stories when they are combined with certain others (most notably, C_6 requires at least 3 other configurations). No two stories depend on the exact same set of configurations.

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}
C_1	\top	\top								\top	\top	\top		
C_2	\perp	\top	\top	\perp	\perp		\perp		\perp	\perp		\top		
C_3	\top			\perp	\top	\perp							\perp	
C_4						\top	\top		\top					\top
C_5								\top						
C_6	\perp								\top	\perp				
C_7	\top			\top					\top					
C_8	\top	\top		\top		\top	\top	\top	\top	\top				\top

Table 3.6: Configuration dependencies for each story question

The number of configurations each story depends on is also important. The more dependencies it has, the less likely it is for a given participant to configure a world that enables the story. That is to say, the story is more likely to be impossible, and is easier to rule out because there are more potential reasons why it is impossible. Stories were selected to include the full range of possible numbers of configuration dependencies (1-6), but stories with fewer dependencies were preferred. Figure 3.9 shows the frequency distribution of the numbers of dependencies among the 14 stories. Six was the highest possible number, i.e. no story exists within the search depth limit (5) that depends on more than 6 of these configurations.

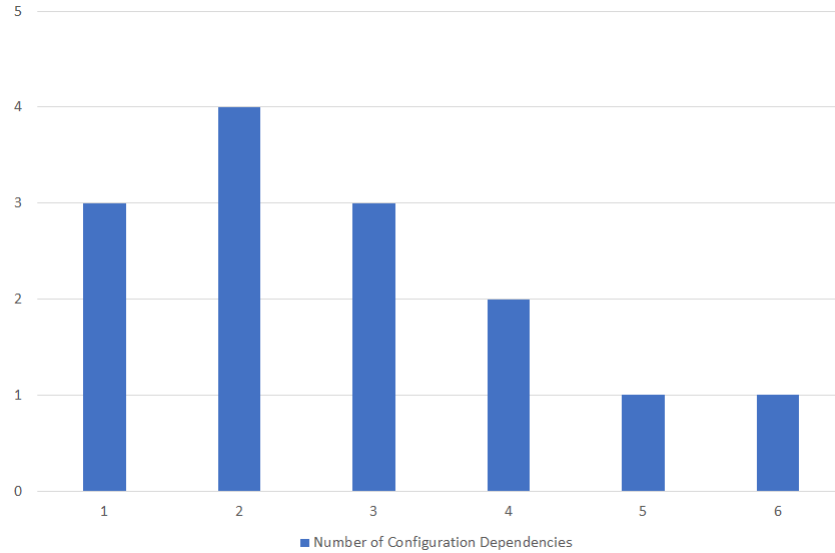
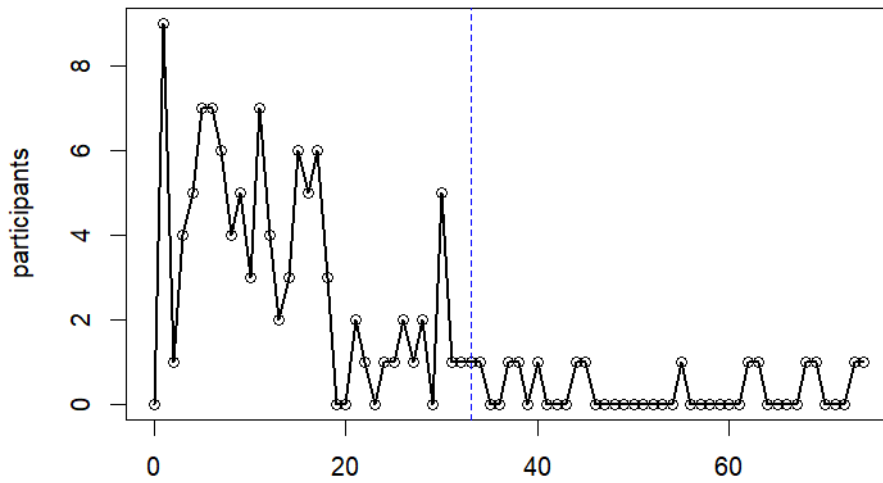


Figure 3.9: Number of stories having each number of configuration dependencies

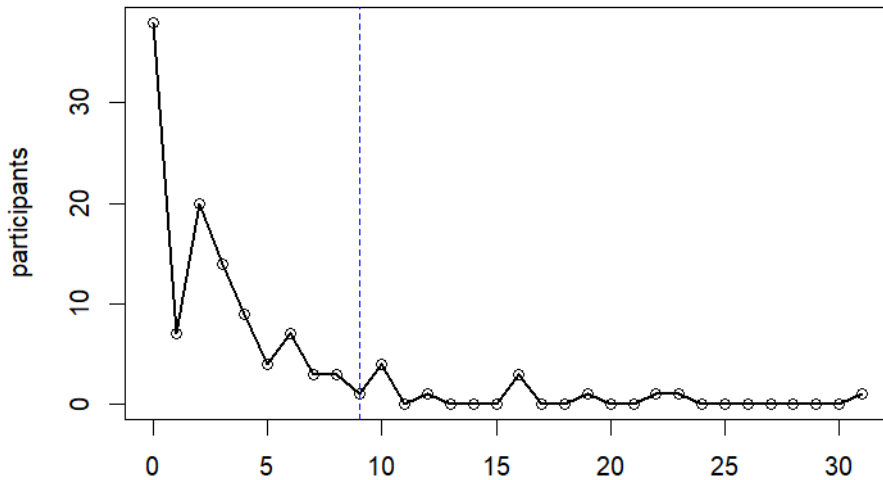
Classifying Engagement

Engagement is classified using two factors: the total number of stories the participant viewed (before and during the questioning) and the number of times they changed the domain configuration. An individual is classified as highly engaged if either of these

factors is at least one standard deviation above the mean. As expected, engagement was skewed towards the low end on both factors. As shown in Figure 3.10, the mean number of stories viewed was 17, and the high threshold was 33 (marked with a dotted vertical line). For configuration changes, the mean was 4 and the high threshold 9. The classification procedure identified 24 of the 118 participants as highly engaged (9 from the control group, 6 from the trees group, and 9 from the circles group). These numbers are relatively small, but large enough to draw some significant conclusions. This group best represents the population of potential domain authors because they willfully engaged in the task rather than skipping through it.



(a) Stories Viewed



(b) Domain Configuration Changes

Figure 3.10: Engagement factors

Results

Figure 3.11 shows the accuracy for the three groups, first among all participants (Figure 3.11a) and second, just the highly engaged participants (Figure 3.11b). A one-way anova with accuracy as the quantitative dependent variable and tool type (none, tree, circles) as the categorical independent variable reveals a significant effect of tool type on accuracy for the highly engaged participants ($p = .017$). No effect was found among the general population. A Welch’s two-sample t -test was performed to compare each tool type independently to the control for the highly engaged participants. Accuracy was significantly increased for both the tree graphs ($p = .025$) and the circle graphs ($p = .006$).

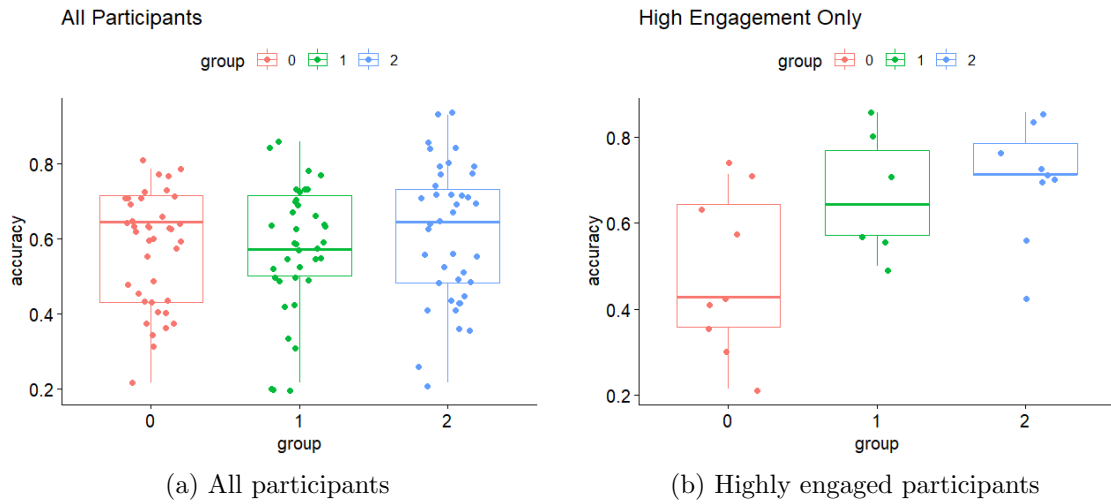


Figure 3.11: Participant accuracy

Among the participants who were most engaged in the task, the experimental tools were significantly more helpful than the baseline. This group is assumed to best reflect the population of actual domain authors, who would likely be much more engaged than the participants in this study.

All confidence ratings for all subjects in each group were combined and then compared using the Wilcoxon sum rank test. The results are displayed in Figure 3.12. The trees group had significantly lower confidence than the control ($p < .001$), both for the highly engaged participants and the whole population. Among the highly engaged, the circles group also had lower confidence than the control ($p = .012$), but this effect was not present for the general population. This is an interesting result: it means that the graphs improved accuracy while lowering confidence; likely because they did a good job of revealing how much nuance there is to the story world, and thus how difficult the questions actually are.

The survey questions were combined into a single item to measure subjects’ overall perception of their understanding of the solution space. The groups were compared using the Wilcoxon sum rank test. Among the whole population, those in the circles group felt that they had a better understanding of the solution space than those in

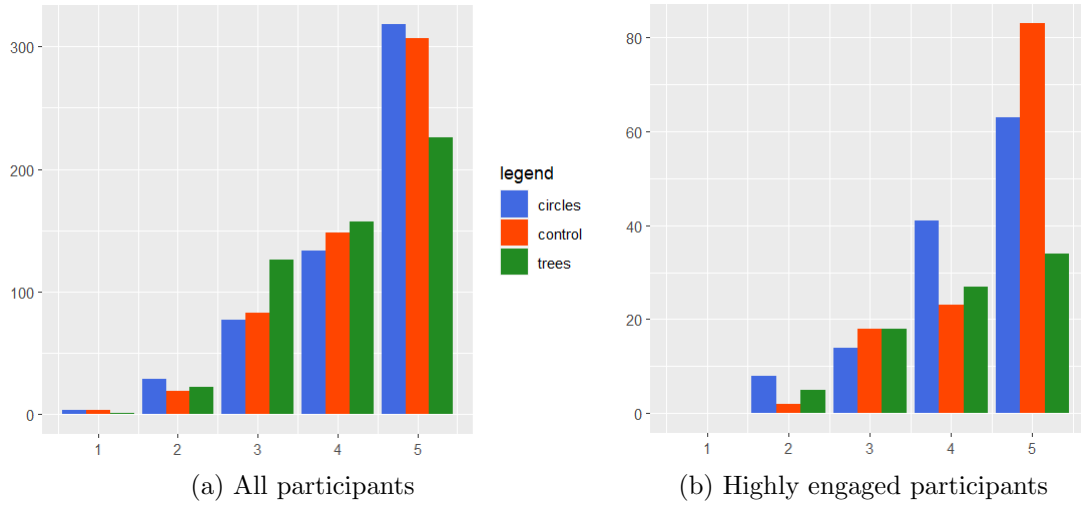


Figure 3.12: Confidence ratings

the control group did ($p = .011$), and this was marginally significant among just the highly engaged participants ($p = .088$). This result is especially encouraging because the circles group reported lower confidence than the control when answering the individual questions, but by the time they reached the end of the task, they felt that they understood the solution space better than the control group. The circle graphs likely struck a good balance between conveying the complexity of the space and summarizing its content in a comprehensible form.

On the other hand, the survey responses from the trees group were significantly *lower* than the control for both the whole population ($p = .023$) and for just the highly engaged participants ($p = .028$). That is, people using the tree tool perceived their understanding of the solution space to be significantly worse than those in the other two groups did—even though they actually understood it better than the control group according to the quantitative data. This makes sense given that the tree graphs conveyed more detail about the space than the other tools did, so people in this group were more aware of their limited knowledge.

Discussion

This is a complicated task to ask of crowd-sourced workers. These results do not show an effect in the general population, but they demonstrate how difficult the task is. The tools are intended for domain authors who are carefully considering the problem. When only the highly engaged participants are considered in the analysis, a significant effect is observed.

Participants in the control group were overly confident in their understanding of the story space—making this a difficult baseline to beat—though the accuracy results show that this was largely false confidence. They perceived their understanding of the solution space to be quite good because their tool did not give them sufficient feedback to realize how nuanced the solution space could be. This is akin to the problem a

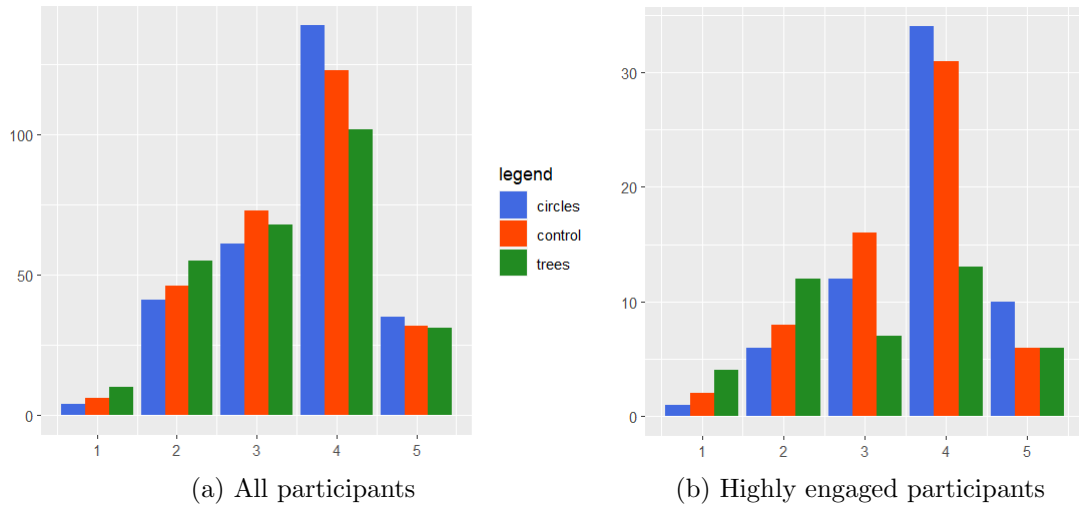


Figure 3.13: Perceived understanding of the story space (1=low, 5=high)

real domain author would experience. They believed they accurately understood the solution space, and since reading random stories is not likely to uncover the specific scenarios that reveal their misunderstanding, they were never shown any indication that they were wrong.

The tree diagrams in particular present a lot of information and may have confused some participants, which could have contributed to the low confidence effects in that group. Even though participants who used this tool felt that they understood the story space less than others, they were significantly better at answering the questions than the baseline. The circles tool improved both actual understanding and perceived understanding for engaged participants, proving that it is possible to convey similarly useful information without overburdening users.

Both diagrams had a tendency to decrease rather than increase confidence, most likely because they showed people more of the space. When people realize how complex the space is, they become less sure about their answers. Confidence was overwhelmingly high for the control group, especially among those who were highly engaged. Yet the accuracy results show the real story—that people in both experimental groups were significantly better at determining whether or not specific stories are modeled by the domain.

Limitations

This preliminary study validates the general concept, but more work is required before these techniques can be applied in a real domain authoring tool. Speed is the biggest issue at present. Generating summaries from the salience vectors is fast, but searching for solutions is not. This study was designed so that all possible domain configurations could be precomputed and searched offline. For a practical domain authoring tool, the search itself will need to be much faster.

The cluster labels used here are generally good descriptors of the cluster as a whole,

but sometimes they appear incorrect for an individual story within the cluster. For example, in problem #255, the following story appears in a cluster labeled Nighttime:

The Bandit walks to the Merchant’s House.
Tom walks to the Merchant’s House.
The Bandit attacks and kills Tom.

Nighttime does not appear in this story, but it is important in most of the other stories that are similar to this one; it is the most appropriate single-feature label to use for this cluster. Other labels like The Merchant’s house, The Bandit attacking Tom at the Merchant’s house, etc. were all considered and were deemed less descriptive of this cluster as a whole than Nighttime is. Perhaps this problem could be mitigated by looking for pairs of features rather than a single feature, although this would come at the expense of more complicated-looking labels.

As mentioned in a previous section, it would be ideal to include more than a single domain when evaluating these models, but resources to that end are again limited. One of the major goals of this work is to improve the capability of authors to create more domains so that more domains are available for evaluations like this. A future study should include more domains, and should directly target interactive narrative authors and researchers to better assess what effect the tools have on highly engaged users.

Chapter 4 Conclusion

Narrative planning is an interpretable process, but it can produce large solution spaces which are not interpretable. This is, in fact, the very reason humans can benefit from using narrative planning to generate interactive stories. The stories we want to write are too complex for us to write them by hand; but narrative planning allows us to model the story world abstractly, as we do in our minds, and let the computer automatically produce all the possible stories we did not specifically imagine. However, authors must be able to reliably evaluate the planner’s output so that they can iteratively modify the input until it produces a solution space that achieves their creative vision. The work presented in this document makes two major contributions toward this goal: Chapter 2 defined a narrative planning model that automatically tracks character beliefs and observation, allowing authors to write narrative planning domains much more intuitively; and Chapter 3 proposed a cognition-based model of story similarity that enables meaningful summarization of solution spaces.

Speed remains a significant limitation to this work. Generating whole solution spaces, especially for large narrative domains, is a lengthy process, and the belief model adds considerable overhead. There are a number of ways this problem can be addressed. More accurate search heuristics and pruning strategies can be developed that specifically account for beliefs and intentions; this approach has been successful for similar models in the past. The salience model may also prove useful for speeding up planning—for example, by using salience as a step cost function (Ware and Farrell, 2022). As mentioned previously, it is not strictly necessary to model infinitely nested beliefs (3-4 levels of nesting is probably sufficient for most storytelling needs), so limiting theory of mind depth can mitigate some of the belief tracking overhead. Lastly, generating the whole solution space may not actually be necessary for summarizing domains: It may be possible to generate summary trees directly by using the distance metric during search. These represent clear and important directions for future work.

The belief model has interesting implications for goal recognition, which is often used in areas such as robotics and intelligence analysis. Goal recognition can be viewed as planning in reverse: Given a sequence of observed events, infer parts of the full initial state that are not observable (i.e. the agents’ beliefs and goals). Existing techniques tend to make assumptions such as full observability and agent rationality; but by incorporating this belief model, agent behavior can be explained without making these assumptions. There are also potential applications for this in interactive narrative systems, such as audience modeling, i.e. estimating what readers are inferring about the beliefs and intentions of characters and even the author. We have proposed a basic technique for belief and goal recognition which can be improved upon (Farrell and Ware, 2020). Here too, a salience model may be useful; for example, salience could identify which plans or goals the inferrer is most likely to expect, given multiple plausible explanations.

Character believability is an important challenge for narrative planning systems, which seek to generate plots without violating audience expectations of reasonable character behavior. This work addresses one significant believability problem—that without an underlying model of character belief, the plans that justify characters’ behavior may rely on information that characters should not have access to, according to the audience. Modeling belief removes many offending stories from the solution space, but much can be done to further improve believability. Better character models, e.g. models of personality and emotion, can give authors more control over individual characters’ behavior, making it easier to build domains that produce the kinds of stories they expect. Reasoning about plan preferences may help avoid characters doing things that are inefficient or otherwise less believable due to the existence of a better alternative. The challenge here is to invalidate just the plans that cause believability problems; not all plans that are inefficient would necessarily do so. Finally, modeling uncertainty in some form could enable new kinds of character interactions; e.g. characters exploring or taking actions for the purpose of learning information. Audiences likely expect characters to be able to take such actions, and not doing so may come across as unbelievable. Although a full model of uncertainty may be intractable in this context, limited models could be developed that enable some of these interactions. Perhaps belief and goal recognition could also be used here, allowing characters to make inferences about others’ beliefs and goals rather than fully knowing them.

The story similarity work is broadly applicable beyond the specific application that is the focus of this document. The distance metric stands to improve measurements of solution diversity, which are often used to evaluate narrative planning models and features. It may also be useful in experience management, where different possible trajectories through an ongoing story are compared and evaluated. Previous studies have demonstrated that salience can be used in player modeling; e.g. to represent player agency, expectations, and preferences. Those studies used much simpler models of pairwise event salience, but could be extended to account for entire sequences of events using the richer model presented here. Salience may also be useful toward measuring story quality or related aspects, such as coherence; as well as for identifying interesting or unique stories, since cluster outliers often represent stories with unique properties.

There are clear opportunities for improving the story similarity model. First, the space and time dimensions could be represented hierarchically which may outperform the present model when tested in domains that feature nested locations and time frames—e.g. rooms within buildings, varying time of day for multiple days, etc. The metric performed best when the time index was weighted low, which makes sense given that the time dimension was relatively unimportant in the domain, but more work needs to be done to determine whether or not this representation of time as a discrete parameter of an event is effective. A possible alternative might be to simply designate certain events as segmenting time, e.g. “night falls”, rather than requiring all events to carry explicit time frame parameters. Intentionality could also be modeled more accurately. Currently we use an explanation that is mostly arbitrary (the first one found); it is a reasonable explanation, but it is not guaranteed to be

the only one, nor the most reasonable one. This could be improved in a number of ways, e.g. by checking exhaustively to find all valid explanations, or by modeling commitment to specific plans, or by inferring which explanation the audience is most likely to attribute to the action (e.g. goal recognition).

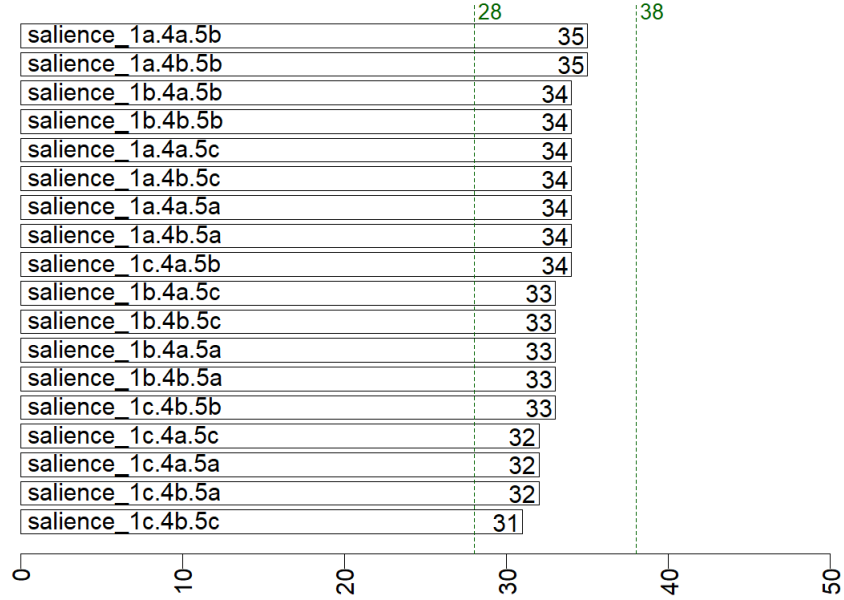
There may also be a more effective way of modeling causality, which is different from the other four indices because it does not pertain to a set of story entities that are either involved in an event or not. Causality makes previous events become salient again, and we already have a model of what happens when an event becomes salient: the entities involved in it become salient. Perhaps causality would be better represented not as its own dimension, but as another means by which entities in the other four dimensions can become salient. That is, when an event occurs, the entities involved in that event *and* those involved in the event's causal ancestors become salient. An early test of this idea showed promising results, but it was not fully explored and there are several unanswered questions, including how salient those entities should become (as salient as the entities involved in the current event, or less so?) and how this should be determined: Should salience be weighed by the recency of the ancestor event, by the number of steps in the causal ancestry, or some other way?

In this document I propose two methods of summarizing solution spaces using the salience distance metric, but these are only a starting point. They would benefit from a better labeling algorithm that does not appear to misclassify some clusters. One way to do this might be to look for pairs of features, rather than an individual feature, that together make a good descriptive label, although this still does not guarantee that a good label will always exist. There are many other ways domain authoring could be made easier. It would be interesting to build an interactive domain exploration tool that summarizes the solution space incrementally as the user chooses events, somewhat like a choice-based adventure game. Features like identifying common mistakes and pointing them out would also be helpful. Ideally there would also be a standard domain library, or some set of basic actions that authors could easily start from, building more specific domain content on top of this for their own purposes. The problem with this is that new narrative planning models continue to be built that require new types of input, so a standard library may still be far off in the future.

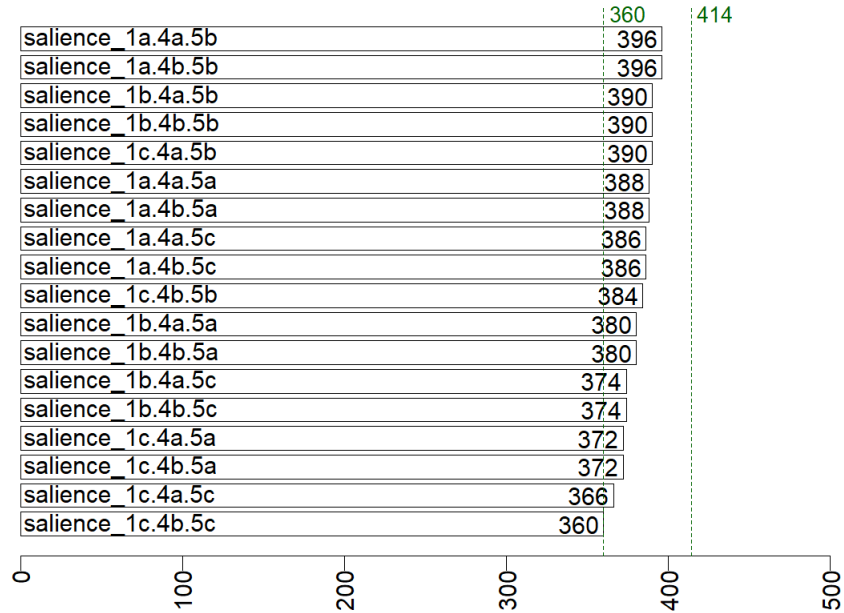
The contributions of this work are critical if narrative planning solution spaces are to be used by humans to create interactive stories. Authors cannot rely on narrative planning systems to produce high quality solution spaces unless they can both write domains and evaluate solution spaces effectively. Techniques that address important hindrances to character believability, like the belief model presented here, allow domains to be written more intuitively; but in doing so, they further complicate solution spaces and make the planner's output even harder to predict. It is therefore important to develop domain authoring tools that can provide adequate feedback, so as to facilitate ongoing research that may otherwise be stymied by these difficulties. This work represents a significant step toward a model of interactive story generation that is usable, and still retains the powerful authorial control inherent in the narrative planning paradigm.

Appendices

Appendix A: Accuracy for 18 Saliency Distance Variations



(a) Using the 37 questions humans agreed on



(b) Using all questions

Figure 4.1: Accuracy results for 18 variations of Saliency Distance

Appendix B: Best Performing Weights

Protagonist	Time	Space	Intentionality	Causality
0.0	0.0	0.1	0.3	0.6
0.0	0.0	0.1	0.4	0.5
0.0	0.0	0.1	0.5	0.4
0.0	0.0	0.1	0.6	0.3
0.0	0.0	0.2	0.3	0.5
0.0	0.0	0.2	0.4	0.4
0.0	0.1	0.2	0.4	0.3
0.0	0.1	0.2	0.5	0.2
0.0	0.1	0.3	0.4	0.2
0.0	0.1	0.3	0.5	0.1
0.0	0.1	0.3	0.6	0.0
0.0	0.1	0.4	0.5	0.0
0.1	0.0	0.1	0.2	0.6
0.1	0.0	0.1	0.3	0.5
0.1	0.0	0.1	0.4	0.4
0.1	0.0	0.1	0.5	0.3
0.1	0.0	0.2	0.2	0.5
0.1	0.0	0.2	0.3	0.4
0.1	0.1	0.2	0.3	0.3
0.1	0.1	0.2	0.4	0.2
0.1	0.1	0.2	0.5	0.1
0.1	0.1	0.2	0.6	0.0
0.1	0.1	0.3	0.4	0.1
0.1	0.1	0.3	0.5	0.0
0.2	0.0	0.1	0.2	0.5
0.2	0.0	0.1	0.3	0.4
0.2	0.0	0.1	0.4	0.3
0.2	0.0	0.2	0.2	0.4
0.2	0.1	0.2	0.3	0.2
0.2	0.1	0.2	0.4	0.1
0.2	0.1	0.2	0.5	0.0
0.2	0.1	0.3	0.4	0.0
0.3	0.0	0.1	0.2	0.4
0.3	0.0	0.1	0.3	0.3
0.3	0.1	0.2	0.4	0.0
0.3	0.1	0.3	0.3	0.0
0.4	0.0	0.1	0.2	0.3
0.4	0.1	0.2	0.3	0.0
0.4	0.1	0.3	0.2	0.0
0.5	0.1	0.2	0.2	0.0

Table 4.1: Weights scoring 37 on the first analysis

Protagonist	Time	Space	Intentionality	Causality
0.5	0.0	0.1	0.0	0.4
0.6	0.0	0.1	0.0	0.3

Table 4.2: Weights scoring 411 on the second analysis (both scored 36 on the first)

Appendix C: Stories used for Domain Authoring Study

Q_1	The Bandit robs the Potion from the Merchant. The Merchant robs the Bandit's sword from the Bandit. Tom walks to the Market. The Merchant attacks and kills Tom.
Q_2	The Bandit buys the Merchant's sword from the Merchant using his coin. Tom waits for night. Later that night, the Bandit walks to Tom's cottage. The Bandit attacks and kills Tom.
Q_3	Tom walks to the Market. Tom buys the Potion from the Merchant using his coin. The Bandit buys the Merchant's sword from the Merchant using his coin. The Bandit robs his coin from the Merchant. Tom walks to his Cottage.
Q_4	Tom walks to the Market. The Merchant attacks the Guard. Tom takes the Guard's sword from the Guard. Tom robs the Potion from the Merchant. Tom walks to his Cottage.
Q_5	Tom walks to the Market. Tom buys the Potion from the Merchant using his coin. The Bandit robs the Merchant's sword from the Merchant. The Guard arrests the Bandit. Tom walks to his Cottage.
Q_6	Tom walks to the Market. The Bandit attacks and kills the Merchant. The Guard arrests the Bandit. Tom takes the Potion from the Merchant. The Guard arrests Tom.
Q_7	Tom walks to the Market. The Guard arrests the Bandit. Tom takes the Bandit's sword from the Bandit. Tom robs the Potion from the Merchant. Tom walks to his Cottage.

Q_8	Tom waits for night. Later that night, Tom walks to the Mansion. j— Mansion Tom takes the Merchant’s sword from the Merchant. Tom robs the Potion from the Merchant. Tom walks to his Cottage.
Q_9	The Guard walks to Tom’s Cottage. The Merchant robs the Bandit’s coin from the Bandit. The Merchant walks to Tom’s Cottage. The Guard arrests the Merchant. Tom takes the Potion from the Merchant.
Q_{10}	The Bandit robs the Potion from the Merchant. Tom waits for night. Later that night, the Bandit walks to Tom’s Cottage. The Bandit attacks and kills Tom.
Q_{11}	The Bandit walks to Tom’s Cottage. The Bandit attacks and kills Tom.
Q_{12}	The Bandit buys the Merchant’s sword from the Merchant using his coin. Tom walks to the Market. The Bandit robs the Bandit’s coin from the Merchant. The Bandit attacks and kills Tom.
Q_{13}	Tom walks to the Market. Tom buys the Potion from the Merchant using his coin. The Bandit attacks and kills the Merchant. Tom walks to his Cottage.
Q_{14}	Tom walks to the Market. The Guard arrests the Bandit. Tom takes the Bandit’s coin from the Bandit. Tom buys the Potion from the Merchant using the Bandit’s coin. Tom walks to his Cottage.

Appendix D: Qualitative Survey

Inverted questions are marked (-).

1. The questions were easy to answer.
2. The questions asked me about stories I had not already considered. (-)
3. I have seen an example of every type of story that is possible in my story world.
4. I could tell which story world options had greater impact on what was possible in the story world.

5. It was easy to notice when changing a story world option eliminated stories or introduced new ones.
6. The tool was helpful for answering the questions.
7. It was hard to find specific stories I was looking for. (-)

Bibliography

- Adam Amos-Binks, David L Roberts, and R Michael Young. Summarizing and comparing story plans. In *Proceedings of the 7th Workshop on Computational Models of Narrative (CMN)*, 2016.
- Adam Amos-Binks, Colin Potts, and R. Michael Young. Planning graphs for efficient generation of desirable narrative trajectories. In *Proceedings of the 13th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, volume 13, 2017.
- Byung-Chull Bae and R. Michael Young. A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. In *Joint International Conference on Interactive Digital Storytelling*, pages 156–167, 2008.
- Rens Bod, Bernhard Fisseni, Aadil Kurji, and Benedikt Löwe. Objectivity and reproducibility of proppian annotations. In *Proceedings of the 3rd Workshop on Computational Models of Narrative (CMN)*, pages 15–19, 2012.
- Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single-and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.
- Hans ten Brinke, Jeroen Linssen, and Mariët Theune. Hide and Sneak: story generation with characters that perceive and assume. In *Proceedings of the 2014 International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 174–180, 2014.
- Rogelio Cardona-Rivera and Robert Young. A knowledge representation that models memory in narrative comprehension. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, volume 28, 2014.
- Rogelio E. Cardona-Rivera, Bradley A. Cassell, Stephen G. Ware, and R. Michael Young. Indexter: a computational model of the event-indexing situation model for characterizing narratives. In *Proceedings of the 3rd Workshop on Computational Models of Narrative (CMN)*, pages 34–43, 2012.
- Rogelio E. Cardona-Rivera, Justus Robertson, Stephen G. Ware, Brent Harrison, David L. Roberts, and R. Michael Young. Foreseeing meaningful choices. In *Proceedings of the 10th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 9–15, 2014.
- Marc Cavazza, Fred Charles, and Steven J. Mead. Character-based interactive storytelling. *IEEE Intelligent Systems Special Issue on AI in Interactive Entertainment*, 17(4):17–24, 2002.

- Yun-Gyung Cheong and R. Michael Young. Suspenser: a story generation system for suspense. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games (TCIAIG)*, 7(1):39–52, 2015.
- David Christian and R. Michael Young. Strategic deception in agents. In *Proceedings of the 2004 International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 218–226, 2004.
- Fiorella De Rosis, Valeria Carofiglio, Giuseppe Grassano, and Cristiano Castelfranchi. Can computers deliberately deceive? a simulation tool and its application to Turing’s imitation game. *Computational Intelligence*, 19(3):235–263, 2003.
- Natalie Dehn. Story generation after TALE-SPIN. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 81, pages 16–18, 1981.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186. Association for Computational Linguistics, 2019.
- Markus Eger. Murder mysteries: the white whale of narrative generation? In *Proceedings of the AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 210–216, 2020.
- John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz—open source graph drawing tools. In *International Symposium on Graph Drawing*, pages 483–484, 2001.
- Rachelyn Farrell and Stephen G. Ware. Causal link semantics for narrative planning using numeric fluents. In *Proceedings of the 13th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 193–199, 2017.
- Rachelyn Farrell and Stephen Ware. Narrative planning for belief and intention recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 52–58, 2020.
- Rachelyn Farrell, Stephen G. Ware, and Lewis J. Baker. Manipulating narrative salience in interactive stories using indexter’s pairwise event salience hypothesis. *IEEE Transactions on Games*, 12(1):74–85, 2020.
- Mark A. Finlayson. ProppLearner: deeply annotating a corpus of russian folktales to enable the machine learning of a russian formalist theory. *Digital Scholarship in the Humanities*, 32(2):284–300, 2017.
- Bernhard Fisseni and Benedikt Löwe. Which dimensions of narrative are relevant for human judgments of story equivalence? In *Proceedings of the 3rd Workshop on Computational Models of Narrative (CMN)*, 2012.

- Uta Frith and Chris Frith. The biological basis of social interaction. *Current Directions in Psychological Science*, 10(5):151–155, 2001.
- Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. The belief-desire-intention model of agency. In *International workshop on agent theories, architectures, and languages*, pages 1–10, 1998.
- Richard J Gerrig. *Experiencing narrative worlds: On the psychological activities of reading*. Yale University Press, 1993.
- Barbara Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
- Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
- Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
- Jörg Hoffmann. The Metric-FF planning system: translating “ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.
- Glena H Iten, Sharon T Steinemann, and Klaus Opwis. Choosing to help monsters: A mixed-method examination of meaningful choices in narrative-rich games and interactive narratives. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.
- Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Number 6. Harvard University Press, 1983.
- Christopher Kives, Stephen G. Ware, and Lewis J. Baker. Evaluating the pairwise event salience hypothesis in Indexter. In *Proceedings of the 11th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 30–36, 2015.
- Justin Kruger and David Dunning. Unskilled and unaware of it: how difficulties in recognizing one’s own incompetence lead to inflated self-assessments. *Journal of personality and social psychology*, 77(6):1121, 1999.
- Ben Kybartas and Rafael Bidarra. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games (TCIAIG)*, 2016.
- Elektra Kypridemou and Loizos Michael. Narrative similarity as common summary. In *Proceedings of the 4th Workshop on Computational Models of Narrative (CMN)*, 2013.

- Michael Lebowitz. Story telling as planning and learning. *Poetics*, 14(6):483–502, 1985.
- Wendy G. Lehnert. Plot units and narrative summarization. *Cognitive Science*, 5(4):293–331, 1981.
- Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.
- Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. In *Proceedings of the 2013 International Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, volume 27, 2013.
- Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81. Association for Computational Linguistics, 2004.
- Benedikt Löwe. Methodological remarks about comparing formal frameworks for narratives. In *Proceedings of the 3rd Workshop in the Philosophy of Information*, 2011.
- Joseph P Magliano, Jason Miller, and Rolf A Zwaan. Indexing space and time in film understanding. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 15(5):533–545, 2001.
- Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. Event representations for automated story generation with deep neural nets. In *Proceedings of the 32nd International Conference of the Association for the Advancement of Artificial Intelligence*, pages 868–875, 2018.
- Neil McIntyre and Mirella Lapata. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 217–225, 2009.
- James R. Meehan. TALE-SPIN, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 91–98, 1977.
- Henry Mohr, Markus Eger, and Chris Martens. Eliminating the impossible: a procedurally generated murder mystery. In *Proceedings of the Workshop on Experimental Artificial Intelligence in Games*, 2018.
- Janet H Murray. *Hamlet on the holodeck: The future of narrative in cyberspace*. 1997.
- Darren Newtonson. Attribution and the unit of perception of ongoing behavior. *Journal of Personality and Social Psychology*, 28(1):28, 1973.

- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.
- Edwin PD Pednault. Adl: exploring the middle ground between strips and the situation calculus. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.
- J. Scott Penberthy and Daniel S. Weld. UCPOP: a sound, complete, partial order planner for ADL. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, volume 92, pages 103–114, 1992.
- Martha E. Pollack. A model of plan inference that distinguishes between the beliefs of actors and observers. pages 207–214, 1986.
- Julie Porteous, Marc Cavazza, and Fred Charles. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology*, 1(2):1–21, 2010.
- Gabriel A Radvansky and Jeffrey M Zacks. Event perception. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(6):608–620, 2011.
- Nils Reiter. *Discovering Structural Similarities in Narrative Texts using Event Alignment Algorithms*. PhD thesis, 2014.
- Mark O. Riedl and Vadim Bulitko. Interactive narrative: an intelligent systems approach. *AI Magazine*, 34(1):67–77, 2013.
- Mark O. Riedl and R. Michael Young. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research (JAIR)*, 39(1):217–268, 2010.
- Justus Robertson and R. Michael Young. Interactive narrative intervention alibis through domain revision. In *Proceedings of the Workshop on Intelligent Narrative Technologies*, 2015.
- James Owen Ryan, Adam Summerville, Michael Mateas, and Noah Wardrip-Fruin. Toward characters who observe, tell, misremember, and lie. 2015.
- Marie-Laure Ryan. Space. *The living handbook of narratology*. Hamburg University, 2012.
- Rushit Sanghrajka, R Michael Young, and Brandon Thorne. Headspace: Incorporating action failure and character beliefs into narrative planning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, pages 171–178, 2022.
- Alireza Shirvani and Stephen G. Ware. A plan-based personality model for story characters. In *Proceedings of the 15th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 188–194, 2019.

- Alireza Shirvani and Stephen G. Ware. A formalization of emotional planning for strong-story systems. In *Proceedings of the 16th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 116–122, 2020.
- Alireza Shirvani, Stephen G. Ware, and Rachelyn Farrell. A possible worlds model of belief for state-space narrative planning. In *Proceedings of the 13th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 101–107, 2017.
- Alireza Shirvani, Rachelyn Farrell, and Stephen G. Ware. Combining intentionality and belief: Revisiting believable character plans. In *Proceedings of the 14th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 222–228, 2018.
- Mei Si and Stacy C. Marsella. Encoding theory of mind in character design for pedagogical interactive narrative. *Advances in Human-Computer Interaction*, 2014, 2014.
- Biplav Srivastava, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do, and Ivan Serina. Domain independent approaches for finding diverse plans. In *Proceedings of the 2007 International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2016–2022, 2007.
- Jonathan Teutenberg and Julie Porteous. Efficient intent-based narrative generation using multiple planning agents. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 603–610, 2013.
- Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *Artificial Intelligence*, 168(1-2):38–69, 2005.
- Zach Tomaszewski. On the use of reincorporation in interactive drama. In *Proceedings of the 2011 International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, volume 7, 2011.
- Tom Trabasso and Linda L Sperry. Causal relatedness and importance of story events. *Journal of Memory and Language*, 24(5):595–611, 1985.
- Teun Adrianus Van Dijk, Walter Kintsch, et al. Strategies of discourse comprehension. 1983.
- Joe H. Jr. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.
- Noah Wardrip-Fruin, Michael Mateas, Steven Dow, and Serdar Sali. Agency reconsidered. In *Proceedings of the 2009 Digital Games Research Association Conference*, 2009.
- Stephen G. Ware and Rachelyn Farrell. Salience as a narrative planning step cost function. In *Proceedings of the IEEE Conference on Games*, 2022. (forthcoming).

- Stephen G. Ware and Cory Siler. Sabre: A narrative planner supporting intention and deep theory of mind. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 99–106, 2021.
- Stephen G. Ware and R. Michael Young. CPOCL: a narrative planner supporting conflict. In *Proceedings of the 7th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 97–102, 2011.
- Stephen G. Ware and R. Michael Young. Glaive: a state-space narrative planner supporting intentionality and conflict. In *Proceedings of the 10th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 80–86, 2014.
- Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. A computational model of plan-based narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games (TCIAIG)*, 6(3):271–288, 2014.
- Stephen G. Ware, Edward Garcia, Mira Fisher, Alireza Shirvani, and Rachelyn Farrell. Multi-agent narrative experience management as story graph pruning. *IEEE Transactions on Games*, 2022. (forthcoming).
- Daniel S. Weld. An introduction to least commitment planning. *AI magazine*, 15(4):27–61, 1994.
- Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. Plan-and-Write: towards better automatic storytelling. In *Proceedings of the 33rd International Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 7378–7385, 2019.
- R. Michael Young, Stephen G. Ware, Bradley A. Cassell, and Justus Robertson. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung Special Issue on Formal and Computational Models of Narrative*, 37(1-2):41–64, 2013.
- R. Michael Young. Notes on the use of plan structures in the creation of interactive plot. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*, pages 164–167, 1999.
- Jeffrey M Zacks, Nicole K Speer, and Jeremy R Reynolds. Segmentation in reading and film comprehension. *Journal of Experimental Psychology: General*, 138(2):307, 2009.
- Hong Bo Zhou and Jun Tao Gao. Automatic method for determining cluster number based on silhouette coefficient. In *Advanced materials research*, volume 951, pages 227–230, 2014.
- Lisa Zunshine. *The Secret Life of Literature*. MIT Press, 2022.

- Rolf A. Zwaan and Gabriel A. Radvansky. Situation models in language comprehension and memory. *Psychological Bulletin*, 123(2):162, 1998.
- Rolf A Zwaan, Mark C Langston, and Arthur C Graesser. The construction of situation models in narrative comprehension: An event-indexing model. *Psychological Science*, 6(5):292–297, 1995.

Vita

Rachelyn Farrell (Rachel) was born in Durham, North Carolina. She grew up in Oxford, Mississippi, and earned a Bachelor of Science in Computer Science from the University of Mississippi in 2012. In 2014, she entered the University of New Orleans and worked as a Research Assistant in the Greater New Orleans Center for Information Assurance. In 2015, she joined the Narrative Intelligence Lab, which began at the University of New Orleans and moved to the University of Kentucky in 2019. Rachel earned a Master of Science in Computer Science from the University of New Orleans in 2017.