UNIVERSITY OF CALIFORNIA
SANTA CRUZ

NARRATIVE INSTRUMENTS: AI-BASED PLAYABLE MEDIA FOR STORYTELLING

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTATIONAL MEDIA

by

Max Kreminski

September 2022

The Dissertation of Max Kreminski
is approved:

Prof. Noah Wardrip-Fruin, Chair

Prof. Michael Mateas

Prof. Gillian Smith

Prof. Edward Melcer

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

Narrative Instruments: AI-Based Playable Media for Storytelling

by

Max Kreminski

Creativity has been proclaimed as a grand challenge for research in both artificial intelligence (in the form of computational creativity) and human-computer interaction (in the form of creativity support tools). Storytelling is an exemplary creative domain: stories are complex creative artifacts in which many different facets—including theme, plot, character, and narration—must all be brought into alignment for the story as a whole to succeed. Consequently, the development of *intelligent narrative technologies* represents an excellent way to improve our understanding of creativity as a computational problem. In this dissertation, I discuss my work on the use of AI to provide plot-level creativity support for creative writing—particularly through the implementation of *story sifters*, which can interactively or autonomously identify sites of narrative potential within large corpuses of potentially narratable events. By crafting human-playable *narrative instruments* (systems that can be played to produce narrative, much like musical instruments can be played to produce music) based on story sifting technologies, I illustrate how AI can help players refine vague high-level plot ideas into coherent narrative throughlines—resulting in a new form of playful AI-supported co-creative writing, with design implications for AI-based creativity support tools in a wide variety of creative domains.

To the shardfolk

Acknowledgments

The research presented in this dissertation would not have been possible without support from a great many people. There's no way I could ever name everyone who helped me to complete this work in some way, but I'll endeavor to mention some of those to whom I owe special thanks here.

My friends in the Seabright Camerata are some of my favorite people in the world, and discussion with them inspired many of the ideas in this dissertation. In particular, Kate Compton was a significant influence on my decision to go to grad school, cleared the way for my focus on playful creativity, and hosted the first gatherings of what would eventually become the Camerata. Jacob Garbe was my first close collaborator in the Expressive Intelligence Studio, a major supporter of my story sifting work, *and* he gave me a cursed sword. Jason Grinblat and Cat Manning provided valuable outside perspectives on my research and were key contributors to my theoretical outlook on emergent narrative. Tamara Duplantis was a primary influence on my adoption of the narrative instruments metaphor. Barrett Anderson and Jasmine Otto hosted *so* many structured activities. Nic Junius and Isaac Karth were my housemates and close collaborators throughout my time at UCSC. And of course, Melanie Dickinson, my closest friend and collaborator, made integral contributions to basically every aspect of my research. (Now that I'm done, Mel, it's all —————————— from here.) To every one of these people I owe an unimaginable debt.

My labmates and predecessors in the Expressive Intelligence Studio played a massive role in shaping the intellectual environment within which this work was conducted, and I am grateful to have known all of them. Besides those already listed, I owe particular thanks to Chris Martens, James Ryan, Ben Samuel, and Gillian Smith, both for their support and for establishing many of the direct foundations for my work. Thanks also to the many other EISers, former EISers, and honorary EISers I've worked with or come to consider friends—including Devi Acharya, Morteza Behrooz, Alex Calderwood, Mirjam Eladhari, Cyril Focht, Kyle Gonzalez, Katie Green, April Grow, Rehaf Jammaz, Shi Johnson-Bey, Maxwell Joslyn, Jordan Magnuson, Stacey Mason, Peter Mawhorter, Alex Mayben, Stella Mazeika, John Murray, Mark Nelson, Beth Oliver, Joe Osborn, Johnathan Pagnutti, Aaron Reed, Adam Smith, D. Squinkifer, Anne Sullivan, Adam Summerville, and Henry Zhou. And a huge thanks to my advisors, Noah Wardrip-Fruin and Michael Mateas—not just for all their guidance and support over the years, but also for establishing and fostering EIS as a close-knit and welcoming research community.

Who else? So many others. Thanks to my committee members, Gillian Smith (again!) and Eddie Melcer, for guiding the direction of my dissertation work. Thanks to many other current and former members of the Computational Media community at UCSC—particularly Raquel Robinson, James Fey, Jared Pettitt, Asiiah Song, Katy Grasse, Josh McVeigh-Schulz, Katherine Isbister, and Jim Whitehead—for your friendship, support, and/or general camaraderie at key moments during my Ph.D. Thanks to my CMPM 201 cohort, who set the ideal tone for my grad school experience. Thanks to my fellow department community managers—past, present, and future—for helping to make the department what it is today. Thanks to all of my wildcat comrades, who reminded me that a better academy is possible. Thanks to my friends and allies in the broader research community—especially Lea Albaugh, Rogelio Cardona-Rivera, Mike Cook, Matthew Guzdial, Antonios Liapis, Emily Short, and Stephen Ware. Thanks to my early research mentors at USC—Heather Desurvire, Scott Fisher, Josh McVeigh-Schulz (again!), and Dennis Wixon. And thanks to the many members of the Stochastic Labs community for their support during my residency—I couldn't imagine a better environment in which to finish up my dissertation work. I'm at once certain and terrified that I'm leaving out a bunch of very important people, but thanks also to all of my friends, collaborators, mentors and comrades throughout my whole life to date—I couldn't have gotten here without you.

Finally, thanks to my family—Ed, Tina, and Adri—for their unwavering support through [gestures vaguely] *all of this.* What a time it's been, huh?

Now for the formalities. This dissertation reproduces portions of the following previously published material:

- Max Kreminski, Melanie Dickinson, and Noah Wardrip-Fruin. Felt: a simple story sifter. In *International Conference on Interactive Digital Storytelling*, 2019. [50]
- Max Kreminski, Melanie Dickinson, Michael Mateas, and Noah Wardrip-Fruin. *Why Are We Like This?*: Exploring writing mechanics for an AI-augmented storytelling game. In *Proceedings of the 2020 Conference of the Electronic Literature Organization*, 2020. [47]

- Max Kreminski, Melanie Dickinson, Michael Mateas, and Noah Wardrip-Fruin. *Why Are We Like This?*: The AI architecture of a co-creative storytelling game. In *Proceedings of the Fifteenth International Conference on the Foundations of Digital Games*, 2020. [48]

- Max Kreminski, Melanie Dickinson, and Michael Mateas. Winnow: A domain-specific language for incremental story sifting. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2021. [46]

- Max Kreminski and Michael Mateas. Toward narrative instruments. In *International Conference on Interactive Digital Storytelling*, 2021. [56]

- Max Kreminski, Melanie Dickinson, Noah Wardrip-Fruin, and Michael Mateas. Loose Ends: A mixed-initiative creative interface for playful storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2022. [51]

The co-authors Noah Wardrip-Fruin and Michael Mateas directed and supervised the research which forms the basis for this thesis. The co-author Melanie Dickinson was the co-designer and co-developer of *Why Are We Like This?*, created the *Why Are We Like This?* system diagram featured in Chapter 4, and contributed to the design of Felt, Winnow, and Loose Ends.

Chapter 1

Introduction

It is a truth universally acknowledged that creativity can be difficult. Complex creative artifacts are *overconstrained*: a single, seemingly innocuous creative decision may have effects that ripple far outward, with unforeseen and potentially disruptive consequences on apparently unrelated aspects of the artifact you're trying to create. Possibility spaces of creative artifacts are *vast*: because so many creative decisions can be made in so many different combinations, it's often intractable to systematically search the space of possible creative artifacts, meaning you have to "guess and check" to see if a set of creative decisions will ultimately pan out to produce a successful artifact. *Evaluating* creative work is a challenge in its own right: artists commonly report feeling too close to their own work to view it objectively, and unclear or conflicting priorities can make it difficult for even outsiders to judge whether a creative artifact is successful. And it's easy to become creatively *blocked* in myriad ways: you might gaze upon the blank page or canvas and find yourself unsure where to start; feel undermined by the fear that others will judge your creation negatively; arrive at an impasse, unable to reconcile two incompatible requirements on the artifact you're trying to make; feel overwhelmed by the urge to throw out what you've already done and start over from scratch; or otherwise find yourself unable to make progress on a creative task.

Due to its difficulty, as well as to the immense possible value of successful creative work, creativity has been proclaimed as a grand challenge for research in both artificial intelligence and human-computer interaction: the former in the form of *computational creativity*, or the construction of systems that behave in ways humans deem creative [19], and the latter in the form of *creativity support tools* (CSTs), or software tools that support human creative practices [112].

Yet creativity is also playful. People actively seek out opportunities to express themselves creatively: playing a musical instrument, for example, remains a popular and culturally well-regarded hobby. Meanwhile, some audiences find joy and value in even the faulty outputs of imperfect computationally creative systems. These audiences may put substantial effort into the repair of flawed or incomplete creative artifacts produced by machines, whether by manually resolving physical implausibilities in machine-generated knitting instructions [111] or by heavily editing the incongruous outputs of a large language model into coherent novel-length stories [127]. The frequent co-occurrence of creativity and play suggests a promising way forward for those who want to ease the difficulty of creative work: perhaps, if we could leverage and draw out the aspects of creativity that people find inherently enjoyable, the difficulty of creativity might fade into the background—or at least become bearable enough to unlock entire new forms of creative practice.

Storytelling, which admits both difficulty and playfulness, is an exemplary creative domain. Narrative is overconstrained: the component parts of any nontrivial narrative (including plot, theme, character, and narration) are extensively interconnected, and small changes to one aspect of a narrative can significantly alter how other, apparently unrelated aspects of the narrative are received. Narrative possibility spaces are vast: compiling even a simple narrative planning domain to a plot graph can yield a graph containing tens of millions of nodes and hundreds of millions of edges [131], and the size of the possibility space balloons even further when one factors in other possible creative decisions, such as word choice. Evaluating narrative is challenging: what makes a good story remains a subject of active debate among narratologists, and different readers prioritize different facets of narrative in their evaluation of narrative quality. And of course, writer's block is a well-known and widely feared phenomenon among creative writers, frequently discussed in writing guides [65, p. 177] [36, p. 14–15] alongside strategies for mitigating its effects.

Nevertheless, the construction of narrative is also frequently pursued for fun—for instance by the players of simulation-driven interactive emergent narrative (IEN) games, including the *Sims* series, *Dwarf Fortress*, and *Stellaris*. The storytelling play practices that have emerged around these games suggests that some players actively seek these games out for the creativity support that they can provide, with the goal of computationally supported storytelling in mind [57, 59]. Therefore, the study of these systems—and their use by players as CSTs—is likely to give insight into the conditions under which creative play is successfully enabled or supported by computationally creative systems. These systems may also function as the kind of creativity support tools that Nakakoji [85] refers to as “skis”: CSTs that, through their design, enable new kinds of creative experiences that were not previously possible.

I propose to investigate these systems as an emerging genre of creativity support software, similar to the genre of *casual creators* proposed by Compton and Mateas [21] and investigated more fully in Compton’s dissertation [22]. Though many of the systems around which these storytelling behaviors originally emerged were marketed as games, I propose that these systems can more usefully be viewed as the embryonic form of an adjacent category of playable media, which I refer to as *narrative instruments* [56]. Like musical instruments, narrative instruments require a player to operate them; afford certain expressive possibilities through their design while discouraging others; may be played more virtuosically by more practiced players; may be played solely for the player’s own enjoyment, or for a wider audience; are often played as part of a larger ensemble, in concert with other instruments; and may be modified or creatively misused by their players to achieve novel or unexpected effects.

Why instruments? I take inspiration in the use of this analogy from several other scholars who have tried to characterize what makes instruments special—distinguishing them from tools on one side and from toys on the other. Writing in the context of CSTs, Nakakoji [85] contends that a creativity support system may be more of an instrument than a tool if it is often used playfully and if its designers prioritize the creation of a particular user experience over maximal efficiency. Tanaka [123] further unpacks the distinction between instruments and tools, suggesting that musical instruments succeed not by maximizing the efficiency of musical creation, but by contributing a particular desirable “personality” or “voice” to the music they are used to create:

> The term tool implies that an apparatus takes on a specific task, utilitarian in nature, carried out in an efficient manner. A tool can be improved to be more efficient, can take on new features to help in realizing its task, and can even take on other, new tasks not part of the original design specification. In the ideal case, a tool expands the limits of what it can do. It should be easy to use, and be accessible to a wide range of naive users. Limitations or defaults are seen as aspects that can be improved upon.
>
> A musical instrument’s *raison-d’etre*, on the other hand, is not at all utilitarian. It is not meant to carry out a single well defined task in the way that a tool is. Instead, a musical instrument often changes context, withstanding changes of musical style played on it while maintaining its identity. A tool gets better as it attains perfection in realizing its tasks. The evolution of an instrument is less driven by practical concerns, and is motivated instead by the quality of sound the instrument produces. In this regard, it is not so necessary for an instrument to be perfect as much as it is important for it to display distinguishing characteristics, or “personality”. What might be considered imperfections or limitations from the perspective of tool design often contribute to a “voice” of a musical instrument.

Therefore, though research in CSTs often aims to create CSTs that are *general-purpose*, I argue that instruments of expression may actually succeed or fail on the basis of the characteristic *voice* they provide. From the narrative instruments perspective, the perceptibility of an instrument’s grain in the stories that it is used to create marks not a failure of generality, but a success of voice. Meanwhile, per Wardrip-Fruin [128], despite their playable nature, instruments are also distinct from the form of playable media known as toys. For Wardrip-Fruin, the key distinguishing feature of instruments is that they “seek a lyric engagement”: they invite expressive use, and are meant to be used for expression first and foremost. It is here that instruments cease to resemble toys, which might or might not be used for expressive purposes—whereas if you pick up an instrument, the odds are good that you have some sort of expressive use in mind.

For the remainder of this dissertation, I will base my definition of instruments on both Tanaka's and Wardrip-Fruin's. In my view, both a characteristic voice and a primarily lyric mode of engagement are key distinguishing features of instruments as playable media. I do not intend to assert that narrative instruments must necessarily be used for live performance, nor do I intend to assert that narrative instrument play must be targeted at an audience other than the players themselves—indeed, musical instruments themselves do not always need to be played in live performance or for an audience. However, I do hope that the term "instrument" also carries some of the connotations of how musical instruments are used socially: for instance, that learning to play an instrument may take some time; that instrument-play may be a deeply skilled and socially valued activity; that instruments are often played alongside other instruments; and that instruments are often modified by their players with specific expressive goals in mind.

Adopting this perspective raises three major research questions in the areas of computational creativity and creativity support tools.

- **RQ1.** In what ways do existing systems that are used as narrative instruments succeed and fail at providing their users with creativity support?

- **RQ2.** What new technical capabilities would we need to develop to address the deficiencies of existing narrative instruments?

- **RQ3.** What new human-facing interfaces would we need to construct to integrate these new technical capabilities into playful computationally supported storytelling practices?

Correspondingly, the contribution of this dissertation is threefold. First (in Chapter 2), I characterize a design space of narrative instruments, positioned between games and tools as a form of playable media that are played with the explicit goal of producing narrative—just as musical instruments are played with the explicit goal of producing music. I also introduce *overwhelm* and *directionlessness* as key deficiencies of existing systems that would require new technical capabilities to address. Second, I present two technical systems—Felt (Chapter 3) and Winnow (Chapter 5)—that advance the state of the art in *story sifting*, or the automatic computational recognition of possibilities for narrative development, with the aim of enabling solutions to the problems of overwhelm and directionlessness. And third, I present two narrative instruments—*Why Are We Like This?* (Chapter 4) and Loose Ends (Chapter 6)—that incorporate these story sifting technologies to enable a new form of playful computationally assisted creative writing, focused on the development of high-level plot structure through AI-generated plot point suggestions that are guided by player-specified storytelling goals. Because the development of new story sifting technologies influenced the design of new narrative instruments and vice versa, I interleave the presentation of these sifters and instruments to preserve the chronological order and cause-and-effect logic of their development.

Altogether, my research process is one of iterative building-to-understand [78] and research through design [133]: operating on an initial hypothesis, I first construct a system that probes a particular design space, then reflect on the results. Insights from this reflection are incorporated into the development of the next system, and this process is repeated indefinitely to arrive at a progressively better understanding of a design space. In this dissertation, the iterative cycle of system design and reflection is most evident in chapters 3 through 6, which chronicle the successive development of two pairs of systems, incorporating more and more insights from previously developed systems into the design of each new system in sequence.

Chapter 2

Background

In this chapter, to contextualize my work, I present some brief background on three areas of scholarship from which my research heavily draws. I first discuss existing scholarship on *retellings*, or narrative artifacts that players construct based on their experiences with interactive narrative systems or games, and argue that the creation of retellings represents an emerging form of computationally engaged storytelling practice that leverages interactive narrative systems for the creativity support that they can provide. I then discuss the potential use of *story sifting* technology, which automatically extracts storyful material from vast simulated storyworlds, to provide additional support for player creativity in the areas where existing interactive narrative play experiences fail to do so. Finally, I present a brief overview of *creativity support tools*—a class of software intended to support human creative practices—and highlight a relative dearth of creativity support tools that support storytelling at a structural, rather than surface, level.

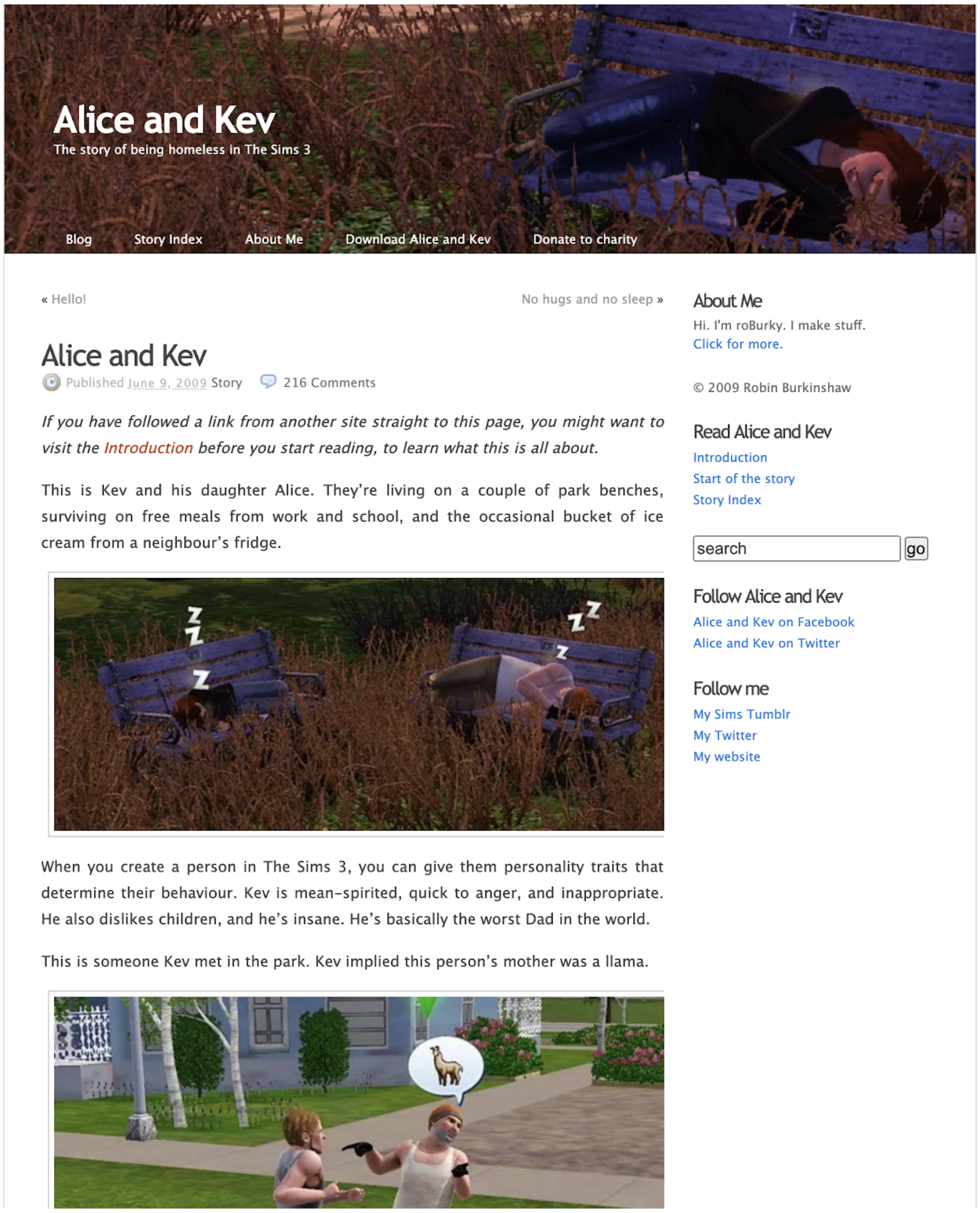
Based on this background, I then describe two key deficiencies of existing systems that have been appropriated as narrative instruments from a creativity support perspective. These deficiencies are *overwhelm* and *directionlessness*. The effort to address these deficiencies while preserving the desirable qualities of earlier appropriated narrative instruments motivates the remainder of the work presented in this dissertation.

2.1 Retellings

Retellings, as defined by Eladhari [26], are narrative artifacts created by players as recountings of their play experiences. Eladhari considers retellings to be a "fourth layer" of interactive narrative, augmenting the System-Process-Product (SPP) model proposed by Koenitz [43]—in which the player's emergent narrative experience is typically taken as the third, or Product, layer. For Eladhari, the existence of player retellings of experiences with a particular interactive narrative system may be taken as evidence that players found these experiences compelling, and—by extension—as an indicator that the system in question is successful. Moreover, Eladhari also proposes that analyzing corpora of player retellings could help researchers better understand and critique interactive narrative systems. For our purposes, however, the reason to study retellings is slightly different: we believe that they provide evidence of player practices around interactive emergent narrative games that are deliberately expressive and authorship-oriented, even when the systems from which the retellings "emerged" were not originally intended to be used as part of a larger storytelling practice. In support of this thesis, we consider several recent developments of Eladhari's initial proposal for the study of retellings, and highlight how each contributes to an understanding of at least some retellings as carefully and deliberately crafted narrative artifacts, produced by players who set out to make use of interactive narrative systems primarily as storytelling tools. (This thesis is also at least partly supported by Eladhari's initial proposal, which focuses on a particularly lengthy and virtuosic episodic *Sims 3* retelling titled *Alice and Kev* [9] (Figure 2.1) as a canonical example of a retelling that merits further study.)

Larsen [66] further clarifies the nature of retellings by distinguishing them from *afterstory*, or the residual impression of story left by a play experience in the player's mind, before it is crafted into a concrete retelling. This terminological distinction is important for clarifying exactly what constitutes a retelling, but it also calls into attention the work that players do in crafting retellings beyond simply playing and allowing a story to emerge. Much as Ryan [99] has argued, Larsen's work supports the thesis that the direct result of play is not in any meaningful way an "emergent narrative": rather, it consists of raw material—laden with narrative potential—that must be refined into a narrative through the act of (re)telling. The process of narrativization involves decisions about both story (which events are presented) and discourse (how to present them), and in retelling creation, it falls to the human player to make these key decisions, using the afterstory resulting from gameplay as a guide.

Additionally, Sych [122] argues that some retellings—labeled as *critical retellings*—represent instances of players using retelling as a means to highlight and reflect on flaws, inadequacies, or awkwardnesses in the underlying interactive narrative system. In the



Alice and Kev

Published June 9, 2009 Story 216 Comments

If you have followed a link from another site straight to this page, you might want to visit the Introduction before you start reading, to learn what this is all about.

This is Kev and his daughter Alice. They're living on a couple of park benches, surviving on free meals from work and school, and the occasional bucket of ice cream from a neighbour's fridge.

When you create a person in The Sims 3, you can give them personality traits that determine their behaviour. Kev is mean-spirited, quick to anger, and inappropriate. He also dislikes children, and he's insane. He's basically the worst Dad in the world.

This is someone Kev met in the park. Kev implied this person's mother was a llama.

About Me

Hi. I'm roBurky. I make stuff.
Click for more.

© 2009 Robin Burkinshaw

Read Alice and Kev

Introduction
Start of the story
Story Index

Follow Alice and Kev

Alice and Kev on Facebook
Alice and Kev on Twitter

Follow me

My Sims Tumblr
My Twitter
My website

Figure 2.1: A screenshot of the first installment of the episodic retelling *Alice and Kev* [9], which interleaves text with screenshots from *The Sims 3* to tell the story of a homeless family in the game.

context of other retelling studies, what stands out about critical retellings in particular is the introduction of irony as an explicit expressive goal of retelling, and the deliberate leveraging of interactive narrative systems to produce moments of humor or incongruity that the system did not intend. For critical retelling, the tellability of the player-crafted narrative derives in part from the fact that the player was able to get the interactive narrative system to produce a result that is funny, jarring, or otherwise unusual. Successful subversion of the IEN system makes for a more compelling story.

Finally, Kreminski et al. [57] and Murnane [84] both examine the process by which retellings are constructed and arrive at an understanding of *extrapolation* as essential to retelling construction. In what Kreminski et al. term *extrapolative narrativization*, players who write stories based on gameplay experiences use the events of gameplay as represented within the computational system as a source of consistent truth and do not explicitly deviate from these events in the stories they write—but they also use these events as jumping-off points for further imagination of detail, adding elements to written stories that go beyond the level of detail modeled in the game's computational systems. For instance, in retellings based on the science fiction grand strategy game *Stellaris*, players often mention random events in which leaders gain certain traits (such as the "Substance Abuser" trait) during gameplay—but they also tend to embellish these events by narrating connections between these and other gameplay events that are unrelated within the game's systems, for instance describing how an admiral's participation in a large and brutal space battle has led them to substance abuse as a means of coping with the severe destruction they witnessed. Though both gameplay events (the trait gain and the large battle) actually occurred in the game systems, the character's mental state and the associated causal connection between these events is entirely invented by the player during the process of narrativization. Grinblat et al. [38] concur with this analysis, framing the process of narrative construction by players of narrative sandbox games as a process of *repair*. Based on this understanding, Kreminski and Mateas [54] argue that the IEN systems that are most often used for retelling represent progress toward a form of *authorship play*: one of the three styles of interaction that were introduced by Louchart and Aylett [70] to characterize different approaches to interactive storytelling. Of these three modes, authorship play is the one that places the greatest amount of creative responsibility on the player.

These recent developments in the study of retellings suggest that the construction of retellings represents an emerging form of computationally engaged writing practice, making deliberate use of IEN games for the creativity support that they can provide. Despite the diversity and sophistication of existing player story construction practices around IEN games, however, these games frequently fail to support player storytelling among all but the most diehard players. These games have been successfully repurposed by these dedicated players as storytelling tools, but learning to use them for storytelling effectively can be a difficult and time-consuming process. From a creativity support perspective, *The Sims* presents users with a relatively approachable set of tools for controlling the behavior of its simulated characters, but (as Ryan [103] has noted), it still does not fully embrace authorship play as its primary goal; for instance, it does not give the player the ability to generate new desires for their Sims at will, nor does it allow players to sift past the mundanities of daily routine in order to quickly locate especially compelling bits of narrative material.

Simultaneously, many other simulation-driven IEN games (such as *Crusader Kings II*, *Stellaris*, and especially *Dwarf Fortress*) have come to resemble the equivalent of Photoshop for mixed-initiative storytelling. They present the player with a dizzying array of complex menus, offer little built-in tutorialization, and provide extensive (perhaps excessive) control over a wide variety of fine-grained options, at the cost of learnability, scaffolding, structure, and support. In essence, neither *The Sims* nor other simulation-driven interactive emergent narrative games treat player storytelling as a first-class activity. There is no built-in support for construction of an *artifact of play*, or a concrete artifact summarizing the events of the play session [41], and few if any of the story-making mechanics that Ryan [99] would describe as *curatorial affordances*: built-in tools for bookmarking, reframing, elaborating on, or otherwise crafting a narrative artifact around storyworld events. Instead, it is left almost entirely up to the players to capture and refine the story that emerges through interaction. Even in games like *Crusader Kings II* that offer automatic capture of a *chronicle* of all storyworld events, which may assist players in recalling and filtering the events that have transpired in a play session for presentation in a story, there is still nothing much to do with this information within the game itself: narrativization of this chronicle is an activity to be pursued by the player using primarily external tools, if they elect to pursue it at all. In Tanenbaum's terms [124], these games offer little if any inbuilt support for player selection of "which events should be portrayed", and no support for player selection of "how to present the details of the narrative", in a narrative artifact capturing some facet of the emergent story.

Moreover, many simulation-driven IEN games persist in a tradition of identifying the player strongly with a *player character*: a particular storyworld entity over which the player assumes more or less direct control, and through whose perspective most or all player-facing displays of information about the storyworld are filtered. This is appropriate for what Ryan [104] refers to as *ludus* play, in which the player pursues a goal given to them by the system, but is excessively limiting for *paidia* play, in which the player is primarily concerned with exploring the space of possibilities in an autotelic or non-goal-directed way. And, as Compton and Mateas [21] have argued, the "intrinsically pleasurable" mode of creativity that defines creativity-oriented play experiences is characterized in large part by its autotelic nature.

Players in games like *Stellaris* have repeatedly expressed a need for tools to support player storytelling. For instance, at various times, players have demanded a summary screen that can be used to view major accomplishments in the life of an in-game character[1]; described the use of external tools to keep track of in-game events as essential to their enjoyment of the game[2]; requested tools for building custom start-game scenarios, with story creation in mind[3]; and even created elaborate custom mock-ups of in-game screens chronicling character biographies[4]. The frequent recurrence

[1]https://www.reddit.com/r/Stellaris/comments/avd1uz/we_should_see_a_summary_screen_on_leader_death

[2]https://www.reddit.com/r/Stellaris/comments/aygl4j/stellaris_is_fantastic_for_storytelling

[3]https://www.reddit.com/r/Stellaris/comments/azbd5k/could_we_get_a_galaxy_editor_so_we_can_make_our

[4]https://www.reddit.com/r/Stellaris/comments/bq5e9j/suggestion_leader_biographies_

of these requests on player forums indicates a real demand for curatorial and other narrativization-oriented affordances in IEN play. And even virtuosic retelling creators like Robin Burkinshaw (who made *Alice and Kev*, the key case study discussed in Eladhari's original retellings paper and probably the most well-known *Sims* retelling) have felt the need to mod the games they use in order to make them better at supporting storytelling. Burkinshaw's *Meaningful Stories* mod pack for *The Sims 4* [10] makes a variety of changes to the game's simulation, with particular focus so far on making the simulation's handling of character emotion "smarter, subtler, and more varied", and with future changes planned to the handling of character autonomy, memory, and whims. This essentially represents a case of a skilled player of an instrument deliberately modifying the instrument in order to change its voice.

2.2 Story Sifting

What can we do to make simulation-driven IEN play experiences more approachable, usable, and supportive as narrative instruments, especially to casually creative users? There are many possible approaches, but one particularly promising strategy centers on *story sifting*, an approach introduced by Ryan as part of his broader "curationist" approach to the development of emergent narrative systems. Though Ryan's recent work on curationism (especially the generative podcast *Sheldon County*) has focused primarily on the automatic generation of stories without a human operator in the loop, narrative instruments suggest an alternative purpose for curationist techniques:

`histories`

namely, to provide players with sifting-based tools that help them make narrative sense of and locate narratively potent material within a rich simulated storyworld, without requiring simulation designers to compromise the richness of the simulation (and thus risk compromising emergence as well) for the sake of legibility.

Ryan et al.'s 2015 paper "Open Design Challenges for Interactive Emergent Narrative" [101], although it did not use the term "story sifting" directly, represents a key milestone in the development of story sifting as an approach. In this paper, Ryan poses four major challenges for future IEN research, including two that are directly pertinent to story sifting technologies: "story recognition", dealing with the automatic identification and extraction of narratively potent "storylike event sequences" from simulated storyworlds, and "story support", dealing with the use of these sequences once they are recognized—for instance by presenting them to the user directly, or "nudging" the simulation to extend these sequences in particular ways, as is done in *The Sims 2* [8, 87].

Ryan's dissertation [99] further develops his ideas on the subject, including by introducing the term "story sifting" and by directly addressing Mateas's critique of emergent narrative as "just one damn thing after another". Sifting for Ryan represents an important part of the answer to this critique, because—in Ryan's view—the things produced directly by emergent narrative systems are not meant to be understood as full-fledged narratives in their own right. Rather, these outputs represent corpuses of *narrative material*, which must be narrativized through a process of sifting (to extract narratively potent event sequences) and narration (which assembles discrete chunks of

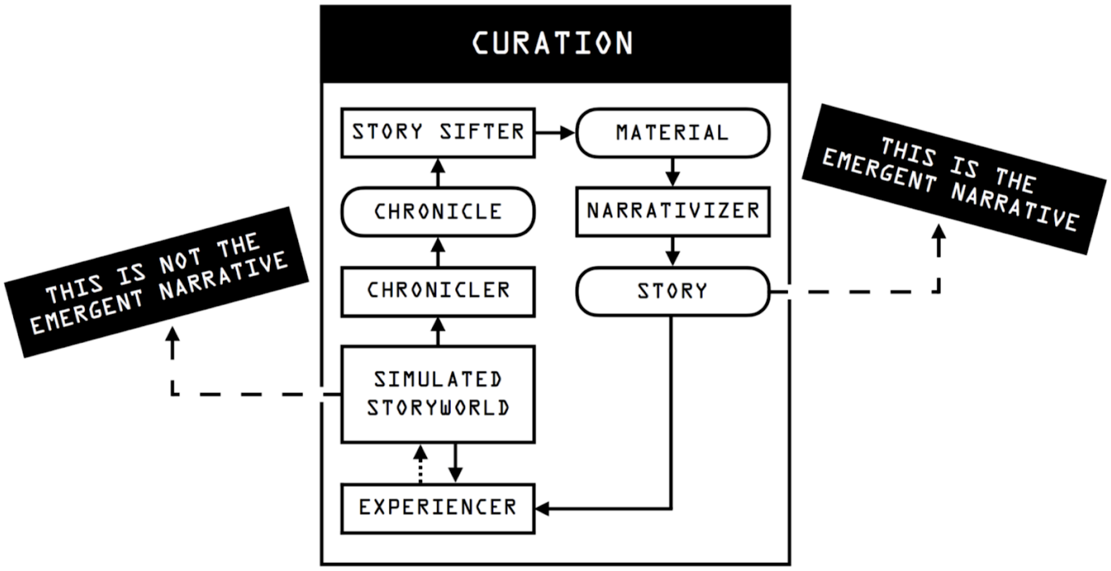
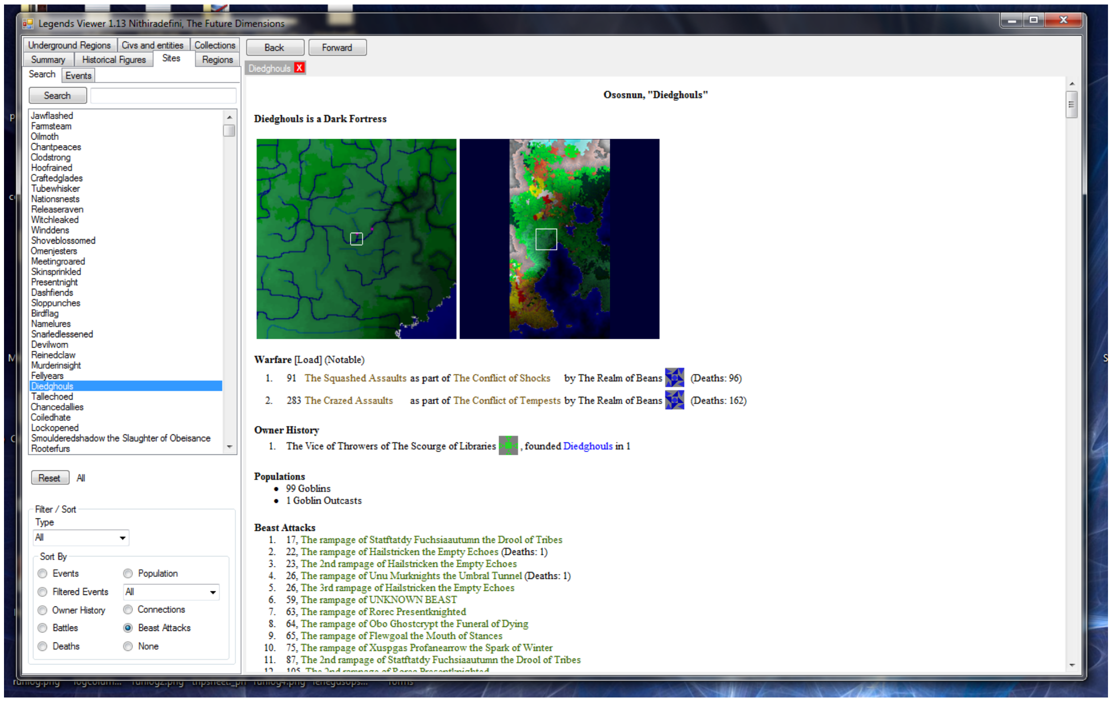


Figure 2.2: A diagram of Ryan's proposed curationist architecture for interactive emergent narrative systems, showing the different roles that users or computational systems could play within the story-making process and highlighting the distinction between simulated storyworld and emergent narrative. Source: Ryan [99, p. 243].

narrative material into a particular telling). This process consists of multiple distinct stages, and each of these stages can be performed by either a computational system or a human. For this reason, the "curationist architecture" Ryan proposes (Figure 2.2) represents a compelling meta-blueprint for the creation of narrative instruments, which can be designed either to assist human operators or to perform tasks autonomously at each stage of this process.

Sifting tools operated by human users have been used as narrative instruments in several different contexts. The third-party *Dwarf Fortress* tool *Legends Viewer* [61] and its *Dwarf Grandpa* extension [32] both present their users with story sifting affordances, as does the "wizard console" component of the *Bad News* assemblage [109]. To date, the approachability of these tools has generally been limited; the *Bad News*



Jawflashed
Farmsteam
Oilmoth
Chantpeaces
Clodstrong
Hoofrained
Craftedglades
Tubewhisker
Nationsnests
Releaseraven
Witchleaked
Winddens
Shoveblossomed
Omenjesters
Meetingroared
Skinsprinkled
Presentnight
Dashfiends
Sloppunches
Birdflag
Namelures
Snarledlessened
Devilworn
Reinedclaw
Murderinsight
Fellyears
Diedghouls
Tallechoed
Chancedallies
Coiledhate
Lockopened
Smoulderedshadow the Slaughter of Obeisance
Rooterfurs

Reset All

Type: All

Sort By: Events, Filtered Events, Owner History, Battles, Deaths, Population, All, Connections, [✓ Beast Attacks], None

Ososnun, "Diedghouls"

Diedghouls is a Dark Fortress

Warfare [Load] (Notable)

1. 91 The Squashed Assaults as part of The Conflict of Shocks by The Realm of Beans (Deaths: 96)
2. 283 The Crazed Assaults as part of The Conflict of Tempests by The Realm of Beans (Deaths: 162)

Owner History

1. The Vice of Throwers of The Scourge of Libraries , founded Diedghouls in 1

Populations

- 99 Goblins
- 1 Goblin Outcasts

Beast Attacks

1. 17, The rampage of Statftatdy Fuchsiaautumn the Drool of Tribes
2. 22, The rampage of Hailstricken the Empty Echoes (Deaths: 1)
3. 23, The 2nd rampage of Hailstricken the Empty Echoes
4. 26, The rampage of Unu Murknights the Umbral Tunnel (Deaths: 1)
5. 26, The 3rd rampage of Hailstricken the Empty Echoes
6. 59, The rampage of UNKNOWN BEAST
7. 63, The rampage of Rorec Presentknighted
8. 64, The rampage of Obo Ghostcrypt the Funeral of Dying
9. 65, The rampage of Flewgoal the Mouth of Stances
10. 75, The rampage of Xuspgas Profanearrow the Spark of Winter
11. 87, The 2nd rampage of Statftatdy Fuchsiaautumn the Drool of Tribes

Figure 2.3: A screenshot of the *Legends Viewer* interface showing information about a particular "site" in a player's imported storyworld, including a log of events (filtered by event type) that took place there. Source: cartographersguild.com forum post by user "nitus".

wizard console requires the user to write Python code, while the *Legends Viewer* interface (Figure 2.3) presents the user with a dizzying array of tabs and filtering options for inspecting different aspects of the storyworld state. Further iteration on design is required to make sifting approachable to non-hardcore users.

Story sifting is just one possible technical design choice for supporting player creativity in a story-making context. However, it seems like a good match for the challenges facing players who enjoy constructing retellings based on IEN gameplay today, as documented in (for instance) Reddit threads requesting additional curatorial affordances in games like *Crusader Kings II* and *Stellaris*. Moreover, notable player-authors like Tim Denee (creator of the popular webcomic-style Dwarf Fortress retellings *Bronzemurder* and *Oilfurnace*) have described sifting as an important part of their storytelling process [77]. As a result, the further development of sifting technology seems like a promising direction to explore with the goal of creating a wider range of narrative instruments in mind.

2.3 Creativity Support Tools

Existing scholarship on creativity support tools (CSTs), or computational tools intended to support human creative practices [112], may have potential utility for the design of narrative instruments. For instance, Resnick et al. [96] have proposed a set of design principles for CSTs, including "low threshold, high ceiling, and wide walls", which are likely to be desirable properties for narrative instruments as well; Carroll et al. [14] have proposed a standard evaluation index for CSTs, the use of which is further explored by Cherry and Latulipe [15]; and both Frisch et al. [31] and Chung et al. [16] have surveyed a number of existing CSTs, including some targeting the domain of storytelling, that may provide inspiration for the development of narrative instruments. Of particular note within this literature, due to the emergence of many notable narrative instruments from the game AI research community, may be scholarship on *mixed-initiative co-creative* processes involving AI systems and humans working together toward a shared creative goal [24, 68]. Although this body of literature contains valuable insights into the nature of creativity and how tools can be designed to support it, it also tends to assume that creativity is fundamentally goal-directed in nature, prioritizing professional users' ability to use a CST to efficiently produce desirable outputs over the experiential aspects of using the tool or the characteristic grain (if any) of artifacts that the tool is used to create. As a result, existing knowledge about the design of CSTs may not always be directly applicable to the development of narrative instruments, which often prioritize creating a certain user experience or producing a certain voice over productive efficiency in the conventional sense.

As a partial corrective to the tendency to assume goal- and output-directedness in the CSTs literature, Compton and Mateas [21] introduce *casual creators* as a category of creativity support tools that specifically aim to support autotelic and playful use. Compton also further develops the idea of casual creators in her dissertation [22]. Casual creators prioritize creating a subjective sense of ease, pleasure, and expressiveness in the user over providing users with maximally precise creative control of the output artifacts they produce. In playful storytelling contexts, experience may be more important than output; for this reason, casual creator design patterns may be of particular use when designing narrative instruments intended to produce a specific user experience through their design. However, not all of the specific experience goals associated with casual creators need be embraced by all narrative instruments: for example, narrative instruments intended to facilitate especially deep engagement might inherently require more practice to learn to use well, and therefore might legitimately reject creating a sense of ease in the user as a design goal. *Reflective creators* [55], which retain the process-over-product focus of casual creators but prioritize the development of what Schön [110] calls *reflective practice* in the user over creating a sense of ease and pleasure, may represent one alternative source of design inspiration for these more depth-oriented narrative instruments.

Most existing creativity support tools for creative writing can be divided into those that primarily aim to support the authoring of narrative *structure* and those that primarily aim to support the authoring of *surface text*. In the former category, Samuel et al.'s *Writing Buddy* system [107] presents players with a beat-based authoring environment that allows them to move back and forth between constructing a plot outline made of dramatic exchanges between characters in a simulated storyworld (tagged with the high-level actions they accomplish, from a library of system-defined action types) and passages of prose narrating the characters' actions. Actions are gated on characters' motivation to perform these actions, and may change characters' motivations when they are performed; character motivation is tracked and updated by the system to ensure characters do not perform actions they would have no motivation to perform. *Writing Buddy* also provides players with optional writing prompts and goals that they might want to achieve, the latter of which can be recognized as fulfilled by the system when the plot outline contains an appropriate set of beats. Additionally, several planning-based tools—including Stefnisson and Thue's Mimisbrunnur [119]—attempt to provide creativity support for the authoring of story outlines by providing a mixed-initiative graphical user interface to a narrative generation planning algorithm.

Tools that support the authoring of surface text, meanwhile, typically make use of language models. One of the earliest tools based on this approach, Swanson and

Gordon's *Say Anything* [121], provides a turn-taking-based co-creative writing interface in which the user can type a sentence, then choose one of a number of system-suggested next sentences (drawn from a large corpus of sentences based on similarity to the most recent user-written sentence) to continue the story. The *Creative Help* system [98] extends the *Say Anything* model of human/computer creative writing collaboration by enabling users to choose where in the story to insert system-suggested sentences (breaking from the strict turn-based model of *Say Anything*) and allowing users to modify the system-provided text. Botnik's *Predictive Writer* [6] uses a finer-grained language model to generate predictions at the word level (rather than at the level of complete sentences) and allows the user to provide their own corpus of input, allowing them to influence what kinds of suggestions the tool will provide. Sloan [115] and Manjavacas et al. [72] report on the use of fine-grained character- or word-based language models for creativity support in the creative writing process, while Chung et al. [17] and Singh et al. [114] present systems that integrate these language models into more sophisticated mixed-initiative writing interfaces that offer affordances beyond mere continuation of user-written prompt text. In all of these cases, language models tend to be useful for introducing unexpected new plot directions or evocative bits of language, but the generated text usually needs to be edited by a human author—often at a fairly fine-grained level—to maintain coherence. Similar results were also found in Calderwood et al.'s small user study of a language model-based creativity support tool for creative writing [11]. Per Samuel et al. [107] and Kybartas and Bidarra [62], this weakness in language model-based systems stems from the fact that even the most sophisticated language models do not construct a higher-level semantic representation of the story under construction, and therefore cannot understand the larger-scale implications of local creative decisions—leaving it up to the human operator to reconcile inconsistencies in the generated text.

Altogether, both structure-focused and surface text-focused CSTs for creative writing are effective at supporting users in coming up with new ideas for narrative threads, whether through the suggestion of discretely modeled character actions (in structure-focused tools) or of unstructured text completions (in surface text-focused tools). Historically, neither category of tools has been able to effectively support users in keeping track of existing threads, though structure-focused tools that make use of social simulation and story sifting to track and surface character motivations have recently made considerable progress in this direction. Meanwhile, neither category of tools currently provides much in the way of support for tying up narrative threads and bringing a story to a satisfying conclusion. Additionally, due to fundamental technological limitations in the current generation of language models, there is no clear route forward for surface text-focused tools that aspire to provide this form of support. Therefore, my primary interest for short-term future work is in the development of structure-focused storytelling technologies that provide support for resolving stories in a satisfying way.

2.4 Deficiencies of Appropriated Instruments

Having surveyed past work in several areas relevant to narrative instruments, we can now identify two key weaknesses of existing systems that have been appropriated as narrative instruments in the past.

2.4.1 Overwhelm

The first of these weaknesses is *overwhelm.* Interactive emergent narrative games such as *Dwarf Fortress* and *Crusader Kings II* are notorious for presenting players with overwhelming amounts of information, making it difficult to locate the most compelling possibilities for narrative development. Simultaneously, existing language model-based narrative instruments use a model of interaction in which the user can "say anything" [121] and the range of AI responses is similarly open-ended, creating difficulties similar to the fear of the blank canvas [59]: when anything can be written, what *should* you write? Users of these latter systems may therefore sometimes experience limited *agency*, in the sense of the word described by Wardrip-Fruin et al. [130]: though the set of available options for what to write next is very large, users are not necessarily reliably "entice[d ...] to desires" for what should happen next in the story by the system's continuations.

Story sifting aims to mitigate the difficulty of overwhelm by assisting players in locating the sites of greatest narrative potential within vast chronicles of simulation or gameplay events. However, it has not yet been made approachable to casual storytellers, such as the players who write stories about their experiences in emergent narrative games. For instance, in the simulation-driven interactive theater experience *Bad News* [109], story sifting is carried out by the "wizard": a skilled human operator of the underlying Talk of the Town simulation engine, who is familiar enough with the simulation engine's datastructures and affordances to compose Python code that inspects the state of the simulation to locate interesting narrative material in real time. Moreover, the wizard is only one of two human operators required to make *Bad News* function as an experience, suggesting that sifting in the context of *Bad News* is such an all-consuming task as to require a *dedicated* "wizard" operator working in partnership with the player-facing "actor" role. It would thus be unrealistic to expect most casual storytellers to put in the time and energy necessary to become proficient with a sifting toolset akin to the *Bad News* wizard's, especially when the goal of introducing sifting into the storytelling process is to *mitigate* overwhelm.

Additionally, early approaches to story sifting are limited in their scalability. In particular, sifting as an approach is currently reliant on libraries of human-written *story sifting patterns* that describe groups of interrelated events which tend to make for compelling narrative building blocks. When the goal is to construct a story sifter that can recognize a wide variety of potentially interesting emergent microstories (and thus to get as much storytelling value as possible from the open-endedness of emergent narrative simulation), human authors thus need to create large numbers of sifting patterns—one pattern for each unique *form* of emergent microstory that the sifter aims to recognize. Authoring these patterns can be time-consuming and error-prone, especially when they are specified in terms of a general-purpose programming language like Python. Moreover, when these patterns make direct use of low-level simulation engine APIs, they run the risk of breaking when the simulation engine is changed in some way, creating a significant maintenance burden for story sifters that target a simulation engine which is under active development. Consequently, there are considerable difficulties associated with applying naïve forms of story sifting to the rich emergent narrative simulation engines that are most often used for the construction of retellings.

2.4.2 Directionlessness

Beyond overwhelm, systems that are currently used as narrative instruments also tend to suffer from the weakness of *directionlessness*. This is because, broadly speaking, even systems that provide help in finding and pursuing interesting short-term suggestions offer little assistance in crafting a coherent plot at a high level. Consequently, players of games like *The Sims* (a conventional simulation-driven interactive emergent narrative game) and *AI Dungeon* (a storytelling game backed by a large language model) must exert considerable effort to keep the story moving in a single consistent direction: in *The Sims*, the events of play rarely if ever cohere into a recognizable high-level plot structure from a traditional narratological perspective [105], while in *AI Dungeon*, the language model that provides suggestions on how to continue the story tends to drive the plot in obviously “surreal” directions unless constantly checked by the player [40]. Thus, despite the pleasure of “coaxing a good story out of the system” [105] in these games, the experience of storytelling-oriented play in these existing systems tends to be one of constantly teetering on the edge of total incoherence—an experience which can swiftly prove exhausting if sustained for any length of time. Ultimately, it is the directionlessness of these systems that leads to the products of emergent narrative games being viewed by some as “just one damn thing after another” [99, p. 4].

To the extent that existing systems have been able to avoid this difficulty, they have generally done so by deliberately designing for a descent into emergent chaos to mark the natural endpoint of a player story. In *Dwarf Fortress*, for instance, the details of individual player stories can vary substantially, but most retellings share a common and highly recognizable arc: a fortress is established, faces several minor setbacks, nevertheless gradually grows in size and ambition, and suddenly collapses when a minor error or overreach provokes a spiral of catastrophe. This prototypical arc lends *Dwarf Fortress* retellings a characteristic texture, and when *Dwarf Fortress* is viewed as a narrative instrument, the common destiny of most fortresses seems to be treated by players as a desirable aspect of the instrument’s voice. However, this acceptance of the inevitability of emergent chaos also limits the range of stories that *Dwarf Fortress* players can use their instrument to tell, and many stories constructed *without* the aid of emergent narrative games follow very different narrative templates. Consequently, if our aim is to develop a wide range of narrative instruments, each with their own characteristic voice, we will need some solution to the problem of directionlessness besides just allowing system-generated twists to accumulate until the story as a whole collapses under their weight.

In the quest to develop AI systems that can help human storytellers overcome directionlessness, story sifting again seems like a potentially promising approach. A system that can reason about all of the events that have taken place in a story *so far* seems especially well-placed to discover latent, incompletely developed high-level narrative structures (such as plot threads, character arcs, and themes) that could guide the longer-term development of the story if brought to the storyteller's attention. However, early sifters are limited in their ability to support the development of high-level structure by their purely *retrospective* nature: because they can only reason about events that have already taken place, they are not generally able to recognize complex *partial* narrative structures that merit future development, nor can they be directly leveraged to suggest *future* events which would advance or complete latent narrative structures. In *Bad News*, this limitation was worked around by framing the player's role in terms of the retrospective *discovery* of microstories that had already played out to completion (centered around a deceased character and their relationships with their contemporaries in a small American town). However, in most emergent narrative contexts, the *responsiveness* of the simulation engine to player actions in the middle of a story is key to assisting the development of unexpected stories that still feel rooted in the player's sensibilities. Therefore, in developing a mode of mixed-initiative storytelling that draws inspiration from retelling construction around emergent narrative games, it seems necessary to augment early approaches to story sifting with some ability to reason about how past events are related to future possibilities.

2.4.3 Toward Solutions

What can be done to address these deficiencies in future narrative instruments? The remainder of this dissertation represents my answer to this question. Specifically, I present two pairs of projects, each of which aims to mitigate one of the two key weaknesses we have identified here. First I present Felt and *Why Are We Like This?* (Chapters 3 and 4), which aim to mitigate overwhelm while leaving the issue of directionlessness to future work. Second I present Winnow and Loose Ends (Chapters 5 and 6), which extend the technical capabilities and user interface affordances introduced by Felt and *Why Are We Like This?* to mitigate directionlessness as well.

Chapter 3

Felt

The problem of *story sifting* involves the selection of events that constitute a compelling story from a larger chronicle of events. Often this chronicle is generated through the computational simulation of a storyworld, whose output consists of a profusion of events, many of which are relatively uninteresting as narrative building blocks. The challenge, then, is to sift the wheat from the chaff, identifying event sequences that seem to be of particular narrative interest or significance and bringing them to the attention of a human player or interactor.

Ryan, who introduced the term "story sifting" [99]—as well as its predecessor, *story recognition*[1]—has identified story sifting as one of four major challenges [101] currently facing work in the domain of interactive emergent narrative. Emergent narrative, which Ryan characterizes as the approach taken by many of both the greatest successes and failures in the area of story generation, remains an area of interest for interac-

[1]As distinct from the natural language understanding term "story recognition", which refers to the identification of embedded story content in natural language text.

tive narrative design [64, 71] and narrative generation [1, 63] communities. Despite this ongoing interest in emergent narrative approaches, however, story sifting has received relatively little attention to date.

There has also been a recent wave of interest in *retellings* [26, 57, 59]—the stories players tell based on their play experiences in interactive narrative games—and in how design elements of games can facilitate and frustrate the player's creative process. From this perspective, story sifters could be viewed as mixed-initiative creativity support tools [68] that help players narrativize their play experiences by surfacing sites of potential narrative interest as they emerge.

One goal of the *Bad News* project [109] was to learn lessons about story sifting needs that could be applied to the design of a computational system that performs story sifting. Unfortunately, a computational story sifter that incorporates the learnings from *Bad News* has yet to materialize. At the same time, our own recent work has involved the design and development of several interactive emergent narrative projects, and we have increasingly found ourselves making use of approaches that resemble story sifting, especially in designing interactive narrative systems that position the human interactor as a narrative co-author. For this reason, we developed a simple story sifter geared primarily toward use in a mixed-initiative context—a system that assists players in the process of narrativizing their play experiences by helping them locate sites of potential narrative interest in a larger simulated storyworld.

Our system, Felt, implements a variation of one of the approaches to story sifting discussed by Ryan, namely that involving the human specification of interesting event sequences. In order to ensure that our human-specified event sequences are generalizable, we implement them not as literal sequences that must be matched exactly, but as *sifting patterns*: queries that seek out ordered sets of events matching certain criteria, with the possibility that other events may be interspersed between the events that are matched. In the remainder of this chapter, we discuss related work in story sifting and adjacent areas; elaborate on the design of Felt; present three design case studies of in-development interactive narrative projects that make use of Felt; and discuss what we have learned from the design, development and application of Felt about story sifting in general.

Many of the design decisions that went into Felt are naïve. This is by design: at each turn, we attempted to do the simplest possible thing that had a reasonable chance of realizing our design intent. It is our hope that Felt functions as a *computational caricature* [117] of a query language-based story sifter, oversimplifying where necessary to ease development while still containing fully realized versions of the key features that are needed for the system to serve as an effective argument for the value of our approach.

3.1 Related Work

Ryan's original chapter introducing the term "story recognition" [101] provides a partial list of existing systems that do something similar to story sifting, including *The Sims 2* [80], which recognizes sequences of events that match the early steps of pre-authored "story trees" and nudges the simulation engine to promote the completion of the story tree [8, 87]. Also of note is the Playspecs [91] system, which applies regular expressions to the analysis of game play traces. Samuel et al. have made some use of Playspecs in a narrative-focused context in *Writing Buddy* [107] and in the analysis of *Prom Week* playtraces [106].

Several systems discussed in Ryan's dissertation [99] also make use of story sifting. Foremost among these is *Sheldon County*, a generative podcast set in a listener-specific simulated American county. In *Sheldon County*, a sifter called *Sheldon* operates over a chronicle produced by the *Hennepin* simulation engine to recognize, extract and narrativize (in the form of podcast episodes) sequences of events that match certain human-defined sifting patterns. These patterns are defined as chunks of procedural Python code that search for candidate events and then bind relevant aspects of these events (such as the perpetrator of a crime) to pattern-specific variables. This approach is similar to the approach we use in Felt. In *Sheldon*, however, authoring a sifting pattern requires knowledge of both the Python programming language and the specific data structures used within the *Hennepin* engine, and even simple pattern definitions are often lengthy due to the verbosity of the procedural code used to implement them.

The "wizard console" in *Bad News* [109] provides an expert human interactor (the "wizard") with a view into the underlying simulation of a small American town. Behind the scenes of the main performance, the wizard uses the console to seek out narratively potent information about the state of the storyworld, and—in real time—relays this information to a human actor who is performing as one of the town's simulated inhabitants. The wizard console is essentially a Python interpreter that enables the wizard to examine the state of the simulation data structures. As such, it provides little computational support for story sifting, although the wizard may make use of a set of helper functions intended to make common sifting tasks easier. The wizard console makes no attempt to realize sifted stories as prose, leaving it largely up to the human actor to decide how to leverage the information gathered through sifting, and—like *Sheldon*—requires familiarity with both Python programming and the particulars of the underlying *Talk of the Town* simulation engine [100] to use effectively.

Dwarf Grandpa [32], an extension to the Legends Viewer interface for browsing *Dwarf Fortress* [5] world data, makes use of story sifting to extract and narrativize the lives of certain notable characters from the game world. Dwarf Grandpa performs story sifting exclusively in a backwards-looking manner, rather than attempting to sift in real time as the simulation runs, and performs only fully automatic sifting, without a human in the loop. Unlike many existing sifters, Dwarf Grandpa also performs the natural language generation needed to automatically present sifted stories as human-readable prose.

Caves of Qud's [30] biography generation system for notable historical characters [37] also makes use of story sifting. Biographies are generated by selecting a sequence of random actions for a character to perform, then running sifting patterns over these random events to retroactively justify each action with an in-world reason. Where no pre-existing reason for an action can be located, new facts about the world are generated on the fly to produce a working rationalization. Like Dwarf Grandpa, *Qud*'s biography generator operates fully automatically and realizes sifted stories as prose.

Rules-based simulationist narrative generation systems often provide some way for events to be directly dependent on or make direct reference to past events, and therefore have some similarity to story sifting. Here we include systems such as Comme il Faut [82], the rules-based "social physics" system that underlies the social simulation game *Prom Week* [81], and Versu [28]. In both of these systems, characters may act in ways that are directly dependent on the presence or absence of a set of past events that meet a set of specified criteria—essentially a sifting pattern. This can arguably be viewed as a form of "internal story sifting": these systems recognize patterns of relevant past events, but only for internal use, and without surfacing the fact that a given pattern was recognized to the audience directly. In Ryan's terms, these systems lack *story support*: the presentation of system-recognized stories to an audience.

Prom Week in particular complicates this evaluation somewhat by presenting players with a list of all of the "social facts" that contribute to a given character's evaluation of the present social situation. Social facts are often directly tied to past events that have played out within the simulation. Arguably, the surfacing of these relevant past events to the player could be considered to be a form of story support, especially if the player is viewed as a co-author alongside the system rather than a mere experiencer of a totally system-curated story. However, in this case, narrativization of the sifted events does not occur within the system; it occurs totally within the player's head, if it occurs at all.

More generally, to describe an approach as making use of story sifting, it is arguably necessary for the underlying plot generation technique to be one that produces a profusion of events, including many mundane events about which the audience does not necessarily care. Many approaches to story generation that allow events to directly reference past events have some similarities to sifting-based approaches, but aim to exclusively produce narratively interesting events that are worthy of being surfaced to the audience. This includes many planning-based techniques [93, 132]. Similarly, certain *plan recognition* techniques bear some resemblance to story sifting and have been applied to an interactive narrative context [12]; however, plan recognition tends to focus specifically on determining the goals of a particular storyworld agent, while story sifting has a much wider range of associated aims. We recognize that these approaches may have much to offer the developers of sifting-based systems, but we do not include them under the label of "story sifting" here.

3.2 System Description

Felt is a query language-based story sifter coupled with a rules-based simulation engine. Events that have transpired in the storyworld are stored in the database as entities, and users of the system write queries—which we, following Ryan, refer to as *sifting patterns*—to identify scenarios and sequences of past events that might make for good narrative material. A sifting pattern is defined in terms of a set of *logic variables* to bind—effectively "slots" or "roles" into which certain database entities, such as events or characters, can be substituted—and a set of relations between these logic variables, which constrain the values that each variable is allowed to take. A sifting pattern could specify, for instance, that `eventA` must be an instance of the `betray` event type; that `eventA` must have taken place before `eventB`; that both events must have the same protagonist, a character `char`; and that `char` must have the `impulsive` trait. The system will then consult the database and return a list of all possible combinations of variable bindings for the pattern as a whole.

In designing a Felt storyworld, users combine sifting patterns with several other features to define *actions*. The structure of actions is directly inspired by the structure of rules in Ceptre [73], a linear logic programming language for specifying interactive simulationist storyworlds. An action consists of a sifting pattern; an optional weighting function that decides how likely it is that this action should be performed, given a set of bindings for the logic variables defined in the sifting pattern; and a function that constructs an *event object* representing this action, which will be added to the database if this action is chosen to be performed. A minimal event object contains an autogenerated *timestamp*, which can be compared with the timestamps of other events to determine which happened first; a short string identifying its *event type*; and a *template string* into which the values of bound logic variables are substituted to produce a human-readable description of the event. It may also contain zero or more *effects*, which describe any other updates that must be made to the database if this event is accepted as part of the history of the storyworld, and possibly other properties on a case-by-case basis, such as the ID of an earlier event that was a direct cause of this event. Because actions are added to the database as events, Felt's story sifting features can be used to run sifting patterns over the history of everything that the simulated characters have said

```
(eventSequence ?eventA ?eventB)
[?eventA "eventType" "betray"] [?eventA "actor" ?char]
[?eventB "eventType" "betray"] [?eventB "actor" ?char]
[?char "trait" "impulsive"]
(not-join [?char ?eventA ?eventB]
  (eventSequence ?eventA ?eventMid ?eventB) [?eventMid "actor" ?char])
```

Figure 3.1: A moderately complicated Felt sifting pattern that will match a sequence of two betrayals perpetrated by the same impulsive character, with no other actions perpetrated by the same character (but arbitrarily many other events) in between.

and done.

By convention, in the projects we describe here as case studies, actions come in two flavors. *Internal* (or *reflection*) actions describe a character reflecting on past events. These actions typically generate "intent tokens" or "motive tokens", which represent a character's intent to act on a particular interpretation of these events in the future. *External* actions describe a character acting on a previously formed intent. These actions typically consume intent tokens and update the state of the world in some outwardly visible way. This separation ensures that intent tokens can be both produced and consumed in multiple different ways: many possible actions that produce the same type of intent token can serve as the motivation for many possible actions that all consume the same type of intent token, opening up the space of possible cause/effect relationships between events. Additionally, in an emergent narrative system where actions are the player's primary window into what is happening in the simulated world, separate reflection actions help make it clear to the player that sifting patterns are at work behind the scenes, and that character behavior is meaningfully influenced by the history of past simulation events—sometimes in complex or sophisticated ways.

This is one way in which we hope to address another of Ryan's four design challenges for interactive emergent narrative [101], namely that of *story support*: once a storyful sequence of events has been recognized, how should this be surfaced to the player? It also helps to ensure that we do not fall victim to the *Tale-Spin effect* [129] by failing to surface the interesting technical capabilities of our interactive narrative system to the player in a compelling way.

Internally, Felt uses the DataScript library [94] to store and query simulation state, including the history of events that have transpired within the simulated world. Felt sifting patterns translate directly into queries against a DataScript database, and are written in a minimal query language that desugars to a subset of Datalog, a simple logic programming language. DataScript provides facilities for storing, updating, and querying state as a set of simple facts of the form `[e a v]`; each fact represents an assertion that the database entity with integer ID `e` has an attribute named `a` with value `v`. A DataScript database is an immutable value: all operations that "update" the database in fact create a fresh copy of the database with the desired modifications, leaving previously stored versions of the database intact and unchanged. This property can be leveraged to snapshot the complete Felt simulation state and run queries against these snapshots while allowing the main copy of the simulation state to continue evolving, which we have found helpful during debugging. It has also enabled us to implement several features, described in section 4.2, that rely on the ability to perform actions in a speculative mode and easily undo them if they lead to unwanted consequences.

DataScript also provides several other useful features that assist with the authoring of sifting patterns. `not-join` query clauses enable testing for the non-existence of an entity that meets a certain set of criteria; this feature is frequently used to specify sifting patterns in which two target events must not be separated by any interceding events involving the same protagonist. Rules bundle groups of query clauses that are commonly used together under a common name; for instance, an `(eventSequence ?eventA ?eventB)` rule may simultaneously specify that both `eventA` and `eventB` refer to event entities and mandate that `eventA` must precede `eventB`. Rules may also be recursive, allowing for the implementation of a `(causalRelationship ?eventA ?eventB)` rule that will match not just direct causes but also indirect causes (separated by one or more intermediate stages of causation) of `eventB`.

3.3 Case Studies

3.3.1 *Starfreighter*

Starfreighter [44] is a small prototype of a procedural narrative game in which the player captains a small starship in a procedurally generated galaxy, completing odd jobs to make a living while managing the needs of a small crew. The primary intent of this game was to test whether *parametrized storylets* [58]—atomic units of narrative content that, like Felt actions, are equipped with slots, preconditions, and effects—could be used to produce compelling emergent story arcs for procedurally generated characters.

It was in the context of this game that we began to develop the earliest version of Felt. Like Felt, *Starfreighter* stores a chronicle of past events (framed as a sequence of "memories" accessible to the characters who participated in each event) and provides features for architecting storylets that refer directly to sequences of past events that meet certain criteria. As a result, *Starfreighter* storylets can contain instances of characters reflecting on sequences of past events, such as the circumstances that led them to leave their home planet or the evolution of their ongoing relationship with another character. Whenever the player completes a storylet, *Starfreighter* evaluates the sifting patterns of all other storylets to identify which ones it would currently make sense to present to the player, then chooses from this pool via simple weighted random selection—essentially using story sifting to implement a form of what Short terms *salience-based narrative* [113].

The early version of Felt used in *Starfreighter* differs significantly from the version we present in this chapter. Most importantly, sifting patterns in this early version of Felt were not authored in terms of a true query language, but in terms of an ad-hoc collection of functions that retrieved entities from the game state in specific predefined ways. One notable consequence of this design decision was that, although storylets were equipped with sifting patterns that could bind a set of logic variables to appropriate values, the system would make no attempt to *unify* these variables with one another, meaning that there was no guarantee of being able to find all of the possible instantiations of a sifting pattern at any given time. Additionally, the authoring of new content became bottlenecked on the development of new functions that enabled the authors of sifting patterns to ask specific questions about the game state, forcing content authors to either learn how to write these often-complicated functions themselves (requiring deep knowledge of how the game state was structured) or else wait for the game's lead developer to implement the functions they had requested. Finally, because there was no straightforward way to get all of the possible instantiations of a sifting pattern in the context of the current game state, debugging was consistently difficult; in particular, if a sifting pattern was repeatedly failing to match a set of values for which it ought to succeed, the nondeterministic nature of sifting pattern resolution made it difficult to determine why. These difficulties led us directly toward the development of a more sophisticated version of Felt, which has helped to alleviate each of these issues in later projects.

3.3.2 *Cozy Mystery Construction Kit*

Cozy Mystery Construction Kit (*CMCK*) [45] is a prototype AI-supported collaborative storytelling play experience (inspired by collaborative storytelling tabletop games like *Microscope* [97] and *The Quiet Year* [4]) in which two players collaborate with a computational system to write a mystery story about a small cast of simulated characters. *CMCK* was the first prototype in the line of systems that would later produce *Why Are We Like This?* and Loose Ends, which are described in chapters 4 and 6 respectively. *CMCK* uses Felt as a simulation engine for characters that sometimes perform actions autonomously and sometimes are directed to perform certain specific actions by players. It also uses Felt to help players locate and build on sites of narrative interest, such as a growing jealousy or resentment between two characters or a building conflict between two values—for instance, comfort and survival.

Of the case studies presented here, *CMCK* is the most explicitly focused on using story sifting to provide creativity support by recognizing emerging story structures as they unfold and suggesting elaborations on emergent patterns and themes. Several Felt features are especially useful in this context. Clear separation of actions that produce and consume "intent tokens" or "motivation tokens" enables players to ask the system questions about character motivation: for instance, "Who had a motive to harm this character?", or "What motives might this character currently want to act on?" This can be particularly useful when writing mystery stories. Because DataScript query evaluation is highly optimized, many Felt sifting patterns can be run over the database at once to provide players with a wide variety of suggestions as to what characters might reasonably do next. Additionally, because sifting patterns provide explicit slots for the characters and events they concern, *CMCK* can give players an interface that lets them filter action suggestions by specifying the values of one or more variables in advance. This enables players to (for instance) get a list of actions that a particular character might currently want to perform, or a list of actions that any character might want to perform in response to a particular past event.

Since the DataScript database in which Felt stores simulation state is an immutable value, *CMCK* can allow players to perform actions in a speculative mode, run queries against the updated database to decide whether they like the effects of these actions, and easily roll back to a previous database state if they do not. Immutability may also be leveraged to facilitate a sort of planning or goal-directed search over actions:

because Felt actions (like planning operators) are defined in terms of preconditions and effects, search over Felt actions can be used to locate speculative future worlds where some specified set of conditions holds true. This can then be used to present users with an interface in which they specify a scenario they would like to bring about in the storyworld, and the system searches for a sequence of actions that might realize this scenario. We did not implement this feature in *CMCK*, but it would not be difficult to do so in the future.

Another potential Felt-enabled feature that we discussed in the context of *CMCK* involves the automatic surfacing of "almost actions": dramatic actions that are *almost* possible, but currently invalid due to a small number of unmet preconditions. This feature concept was directly inspired by the feature with the same name in *Writing Buddy* [107], and would have leveraged Felt's per-action weighting functions to judge how dramatically significant a given action would be if performed. However, in practice, our initial naïve implementation plan for this feature (involving the procedural generation and parallel checking of many variants of each action's precondition set, with different preconditions omitted in each variant to locate a wide variety of nearly-possible instances of each action) turned out to be too computationally intensive to be feasible in Felt alone. It was only our later work on Winnow, discussed in chapter 5, that would make this feature feasible.

CMCK is also notable for its use of story sifting to highlight character personality and subjectivity through sifting-driven reinterpretation of events. Each *CMCK* character holds several randomly selected *values* drawn from a pool of eight possible values, and these values are used in sifting patterns to influence how characters will interpret certain event sequences. Consider, for instance, a sequence of events in which a character forbids anyone from using the kitchen until a crime that took place there has been thoroughly investigated. A character who values comfort above all else may evaluate this sequence of events very differently than a character who values safety. Much as *Terminal Time* [79] narratively spins historical events to cater to the audience's ideological biases, and *Caves of Qud*'s biography generation system [37] retroactively decides how to interpret the motivations behind a character's randomly generated actions, *CMCK* characters engage in retroactive interpretation of events through the sifting of their own stories, one another's stories, and the stories of the world around them. Moreover, in *CMCK*, the differential interpretations that result from this process of sifting serve as the main driver of character conflict. In this sense, *CMCK* could be viewed as an instance of AI-based game design [25, 126] in which the AI process at the heart of the play experience is a story sifting engine.

3.3.3 *Diarytown*

Diarytown is a prototype game in which players craft diary entries about their real life and watch as the described experiences are creatively enacted, extrapolated on, and respun through the lens of a simulated, personalized town.

In *Diarytown*, Felt is primarily used to recognize story patterns in a player's diary entries over time, allowing the game to surface to the player different possible interpretations of things that have happened within their life. This underpins one of the project's primary design goals, of facilitating playful, generative reflection on one's life. Whenever a story pattern is recognized, *Diarytown* surfaces it to the player as a new scenery object within the simulated town. The player can then interact with this scenery object to view the recognized story, and can choose to edit the object's appearance and placement, or even to remove it from the town entirely if they reject the interpretation of their life's events that it represents. Multiple recognized story patterns that share many of the same attributes (for instance, a common focal character) might be collapsed over time into a single, larger scenery object, and these objects may thus gradually take on the role of symbols of larger patterns within the player's life (for instance, a monument to the player's ongoing relationship with a particular friend).

Players craft diary entries by composing terms from a symbolic action library consisting of actions, connectors, and modifiers designed to reflect common actions in a person's everyday life. Some are optionally parametrized with character names, places, and other reference nouns, which the player defines during play. The parametrized nature of Felt actions make it ideal for representing elements of these complex diary entries as simulation actions.

Felt is also being used to simulate autonomous town activities and background characters that are partially conditioned on player-entered actions and character definitions. This integration of player-defined and autonomous actions allows us to playfully extrapolate on a player's account of their daily life, and leverage the expressive affordances of emergent narrative (which generally requires a large number of events to sift through) even when there are relatively few player diary entries.

In the context of the *Diarytown* project, Felt was introduced to four high-school-aged research interns, three of whom had some prior programming experience (primarily in Java) and one of whom had none. At the end of a single day of instruction, all four interns were able to author new actions (including sifting patterns) on their own. Within a week, they had authored 85 new actions without expert intervention.

3.4 Discussion

3.4.1 Authoring Sifting Patterns

When adopting an approach to interactive narrative that makes integral use of story sifting, the design and development of sifting patterns becomes part of the content authoring pipeline. As such, we made it one of our design goals for Felt to make the authoring of sifting patterns as easy and approachable as possible. As a result of this focus on approachability, we initially intended to provide sifting pattern authors with a large library of preauthored functions for accessing the database in certain specific ways, and thereby to avoid creating a situation in which sifting pattern authors had to learn how to interact directly with the complicated network of relationships between game entities.

In practice, however, we soon found that it was very difficult to anticipate in advance the full range of questions that a sifting pattern author might want to be able to ask about the game state. This made it near-impossible for us to create an adequate library of preauthored functions. As a result, we found ourselves turning instead toward the path of giving sifting pattern authors access to a “real” query language. Query languages are designed for flexibility, enabling the user to ask a wide variety of questions about the game state on an as-needed basis—including questions that no one specifically anticipated ahead of time.

It may, at first glance, seem counterintuitive that authoring can be made more approachable by presenting content authors with a query language they must learn. However, as argued by Nardi [86] and evidenced by the widespread success of the Tracery language [20] among users with little or no prior programming experience, people are generally quite good at learning simple formal languages when the language is tied to a task they want to perform. This is especially the case when a gentle on-ramp to query authoring is available: novice content authors may start off using pre-composed sifting patterns without modification, graduate to making slight modifications of these pre-composed patterns, and eventually gain sufficient facility with the query language to author their own sifting patterns from scratch. Our success with having high-school-aged research interns on the *Diarytown* project write sifting patterns with little training supports the hypothesis that users can learn to write sifting patterns in a simple query language fairly quickly when they are provided with a robust library of examples to copy, paste, and modify.

3.4.2 Debugging Story Sifters

Another advantage of using a database with a full query language to store game state is that it greatly simplifies the process of debugging, enabling developers and content authors to write and run queries against the live database at any point. This stands in sharp contrast to the debugging experience in *Starfreighter*, where the opacity of the ad-hoc game state data structures made it difficult to explore the game state when trying to track down the reason for a sifting pattern's failure or misbehavior—especially for content authors, who had particular difficulty learning how different parts of the game state related to one another.

DataScript query evaluation is computationally inexpensive. This makes it tractable to get a list of all sifting patterns that are currently succeeding, including all possible sets of variable bindings that they could use, simply by running all of the available sifting patterns against the database in quick succession. This can significantly speed up debugging by making it visible at a glance whether or not a particular instantiation of a sifting pattern is currently possible, saving a substantial amount of time that a developer might otherwise have to spend manually testing sifting patterns they are attempting to debug.

Moreover, the DataScript queries that underlie Felt sifting patterns are partitioned into distinct clauses, which can be evaluated against the database individually or in subgroups as well as in the context of a complete query. We took advantage of the structured nature of our sifting patterns to implement a debugging helper function we refer to as `whyNot`. This function takes a sifting pattern as an argument, and can optionally also be supplied with a partial set of variable bindings for the pattern's logic variables. It then tests each clause of the sifting pattern in isolation, then each subgroup of clauses, until it identifies the set of clauses that are currently causing the pattern to fail. This information can then be reported to the pattern's developer, potentially saving them the work of manually stepping through the pattern line by line to identify why it is not succeeding when it ought to be.

3.4.3 Coupling Sifting and Simulation

Felt was developed as a sifting engine coupled with a simulation engine. The primary reason for this was convenience. At the time of Felt's development, we were largely attempting to sift the outputs of storyworld simulations in which character actions were gated on complex precondition checks against simulation state. Because sets of preconditions that enable actions, like Felt sifting patterns, can easily be represented as small logic programs, it made sense to incorporate Felt's existing pattern-matching capabilities into our simulation designs as the language for defining action preconditions as well. This also made it easy for us to write character actions that are preconditioned on sequences of past events—actions which can represent characters reflecting on, interpreting, and responding to events that have transpired in the past. The ease of writing these reflective actions directly influenced the design direction of our later narrative instruments, especially *Why Are We Like This?* (as discussed in chapter 4).

Strictly speaking, it is possible to make full use of Felt's story sifting features without making any use of its simulation capabilities. Sequences of events can be generated by an external process and then added to the database in a Felt-compatible form, enabling the authoring of sifting patterns that operate over these externally generated events—and in fact, several of our later projects (such as Loose Ends, as discussed in chapter 6) generate event sequences to sift without also using Felt as a simulation engine. As our sifting work increasingly moved toward the sifting of action sequences generated by simpler simulation engines, we largely stopped using the simulation capabilities of Felt, and developed a preference for simulation engines and sifters to be defined separately. This preference is reflected in our later work on Winnow (as discussed in chapter 5), which is a standalone sifter without built-in capabilities for simulation.

3.5 Conclusions and Future Work

By making sifting patterns easier to author quickly, in large numbers, and for novice authors, Felt enables the development of larger and more sophisticated sifting-based computational narrative systems. This, in turn, allows us to create simulation-driven narrative instruments that incorporate sifting as a solution to the problem of overwhelm. In chapter 4, we present *Why Are We Like This?*, a narrative instrument that incorporates Felt for this purpose.

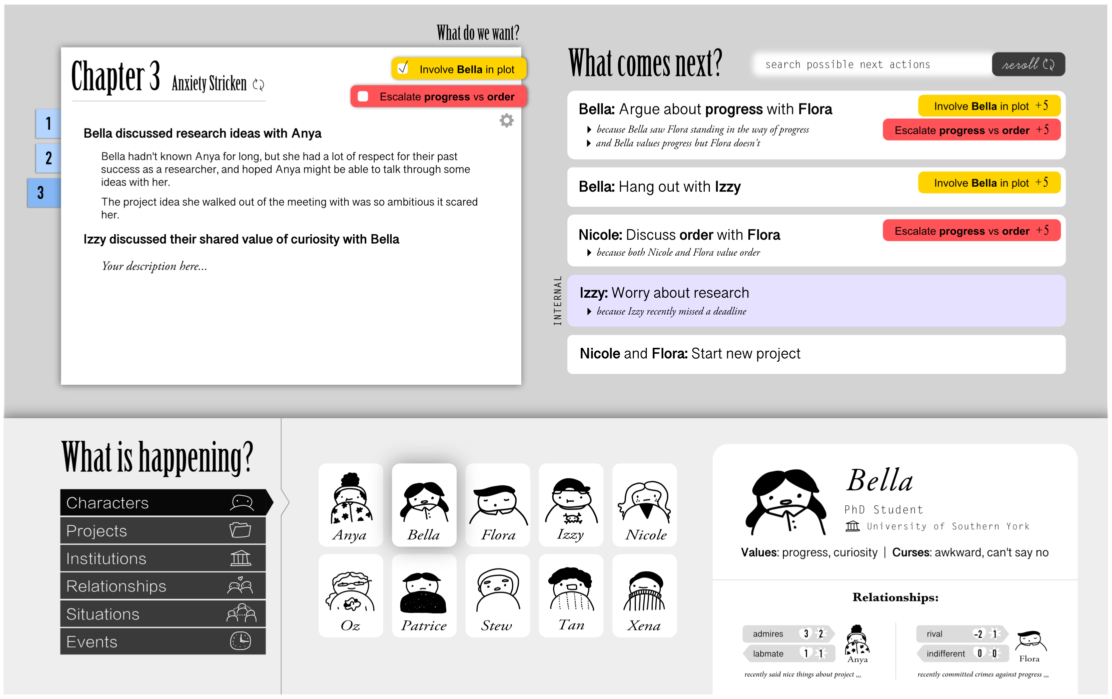
More generally, it is our hope that, by presenting this system, we will encourage the development of a wide variety of approaches to story sifting. The query language-based approach we explore here is only one of many possible approaches, and we have only presented a first step toward the realization of our own preferred approach. We also hope that the existence of a "reference" story sifter will inspire the design of new kinds of interactive narrative experiences based on story sifting technology—particularly experiences that use sifting to provide creativity support for the human interactor in a collaborative storytelling context.

Chapter 4

Why Are We Like This?

Why Are We Like This? (abbreviated *WAWLT*) is an AI-supported digital storytelling game, previously reported on in [47] and [48], and based on the Felt system discussed in chapter 3. In *WAWLT*, one or more players (ideally two) work to construct a story in a pastiche of the cozy mystery genre, supported by an AI system that serves to provide players with inspiration and keep the story moving forward, even when players are unsure what should happen next.

WAWLT's setting is grounded in the familiar context of academic research, dealing with a small community of researchers who are temporarily stranded during a symposium at a remote venue. Over the course of a play session, because different characters have access to different *story sifting patterns* that they use to make narrative sense of the world, characters end up telling themselves different stories about the events that have transpired in the world so far—and, therefore, end up acting in conflicting ways based on their conflicting evaluations of the same evidence. The closed environment of



Chapter 3 Anxiety Stricken

Bella discussed research ideas with Anya

Bella hadn't known Anya for long, but she had a lot of respect for their past success as a researcher, and hoped Anya might be able to talk through some ideas with her.

The project idea she walked out of the meeting with was so ambitious it scared her.

Izzy discussed their shared value of curiosity with Bella

Your description here...

What comes next?

Bella: Argue about **progress** with **Flora** — Involve **Bella** in plot +5 — Escalate **progress** vs **order** +5
- *because Bella saw Flora standing in the way of progress*
- *and Bella values progress but Flora doesn't*

Bella: Hang out with **Izzy** — Involve **Bella** in plot +5

Nicole: Discuss **order** with **Flora** — Escalate **progress** vs **order** +5
- *because both Nicole and Flora value order*

Izzy: Worry about research
- *because Izzy recently missed a deadline*

Nicole and **Flora:** Start new project

Bella
PhD Student
University of Southern York

Values: progress, curiosity | **Curses**: awkward, can't say no

Relationships:

admires	3	2	Anya
labmate	1	1	
rival	-2	1	Flora
indifferent	0	0	

recently said nice things about project ...

recently committed crimes against progress ...

Figure 4.1: The main *WAWLT* interface, with the running transcript of the story so far in the upper left, action suggestions in the upper right, and the storyworld investigator on the bottom, focusing on a specific simulated character.

the symposium venue acts as a pressure cooker, exacerbating initially minor tensions between characters until a variety of plausible motivations exist for characters to commit severe crimes.

WAWLT represents an example of *AI-based game design* [25] inspired by the study of existing player storytelling practices [26, 57] in simulation-driven games like *Dwarf Fortress* [5], *The Sims 2* [80], and *Crusader Kings II* [92]. We particularly set out to provide creativity support features that would help players overcome four major barriers to creativity documented in [59]: fear of the blank canvas; fear of judgment; writer's block; and perfectionism. Further design inspiration was drawn from tabletop storytelling games like *Microscope* [97] and *The Quiet Year* [4], and from the AI-augmented improvisational theater experience *Bad News* [109], of which we wanted to create a "home version" that a small number of amateur players could set up and play through on their own without any special training.

The main contribution of this chapter is a *computational caricature* [117] of an AI architecture that enables a mixed-initiative co-creative storytelling play experience. Like other caricatures, our architecture is intended first and foremost to clearly present a central claim about the design space of co-creative storytelling games: that machines should support player storytelling practices by providing players with intelligent plot direction suggestions, drawn from an ongoing social simulation and guided by player utterances in a machine-understandable *intent language*. Secondarily, we also aim to provide readers with a small number of reusable abstractions that might be generally applicable in other, similar systems.

In the remainder of this chapter, we first describe our AI architecture at a high level, through the lens of an idealized vision of a typical *WAWLT* play experience. We then describe the key subsystems in our overall AI architecture and the roles of these subsystems in producing the desired player experience of co-creative storytelling. Throughout these sections, for illustrative purposes, we draw comparisons between our AI architecture and several other simulation-based storytelling architectures, including Ensemble [108], Versu [28], Ceptre [73], and Tale-Spin [83]. Finally, we discuss the results of early playtesting and possible directions for future development.

4.1 Related Work

The current iteration of *WAWLT* was conceived first and foremost as an analogue to tabletop storytelling games, which attempt to provide scaffolding for a collaborative storytelling process between a group of human players. *Microscope* [97] in particular offers a wide variety of creativity support features, particularly the *palette*—a way for players to collectively negotiate what they do and do not want to see in the story—and a recursive story structure that enables players to "dive deeper into" any part of the story that they would like to further flesh out. *The Quiet Year* [4] involves the collective production through play of a physical artifact, namely a map of the world that the players have created, which players can take with them after play as a reminder of the play session. And *Archives of the Sky* [95] provides mechanisms for structuring character conflict around *values* held both by individual player characters and the larger society in which they exist. All of these features have directly inspired design elements in the current version of *WAWLT*.

WAWLT is a *mixed-initiative co-creative* system [68], and can be viewed as a *casual creator* [21] for cozy mystery stories set in a particular context. Other casual creators for storytelling, such as *Writing Buddy* [107], and other mixed-initiative co-creative systems intended to be used in a storytelling context, such as *Mimisbrunnur* [119], have provided valuable design inspiration for *WAWLT*, but have not fully embraced the use of a fine-grained simulated storyworld in the way that we aim to here.

The same is true of co-creative writing processes driven by language models, such as the Botnik *Predictive Writer* app [6], the *Creative Help* system [98], and the writing practices described by Manjavacas et al. [72] and Sloan [115]. Moreover, language model-based systems are particularly flawed from a creativity support perspective due to their lack of an explicit world model. Because of this lack, they frequently go off track or make suggestions that clearly contradict previous statements, forcing users to spend time and energy repeatedly reminding them of established facts via prompting techniques. This distracts from the useful creativity support features they provide (suggestions about "what happens next") and exacerbates the problem of maintaining consistency, which even experienced authors may already find difficult on their own.

WAWLT is built around *story sifting* in both its implementation and its design, making central use of the story sifting and simulation engine Felt [50]. Story sifting approaches to emergent narrative attempt to address the challenges of narrative generation through curation: by allowing a simulated storyworld to run, generating a massive chronicle of mostly-uninteresting simulated events, and then searching within this chronicle for patterns of narratively compelling events, it is possible to provide players with compelling stories or microstories without baking knowledge of how to tell a compelling story directly into the simulation engine itself. Story sifting, originally known as "story recognition", was first proposed as an open design challenge for interactive emergent narrative by Ryan et al. [101], and further refined by Ryan in his dissertation [99].

Several existing play experiences make use of story sifting technology in some way, but none of them attempt to center story sifting as a player-facing game mechanic as we aspire to in *WAWLT*. *Dwarf Grandpa* [32] runs sifting retroactively on the history of a Dwarf Fortress world to pull out and highlight the stories of certain kinds of vampires. *Bad News* [109], an interactive theater experience with both human and computational components, involves a process of live sifting in which a human "wizard" (one of the performers, rather than a member of the audience) interacts with a Python console to pull out interesting information about the simulated storyworld in which the story is set and feeds this information in real time to the human actor portraying the simulated characters. *Cozy Mystery Construction Kit* [45], which we discussed in chapter 3, also makes use of story sifting, but its design was less sifting-centric than that of *WAWLT*.

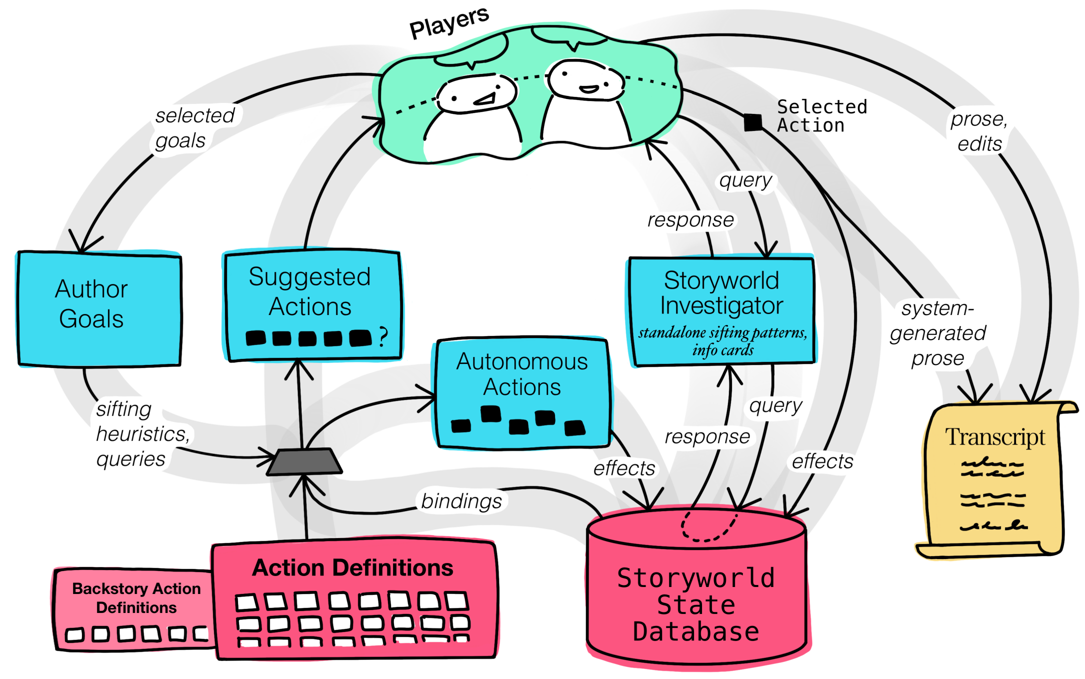


Figure 4.2: An overall system diagram of *WAWLT*, showing the important modules and data flows. Pink subsystems (action definitions and the storyworld state database) consist of inert data; blue subsystems (author goals, suggested actions, autonomous actions, and the storyworld investigator) act on this data; and the transcript emerges from player actions over the course of play. Subsystems depicted in this diagram are discussed in greater detail in sections 3.3.1-3.3.7.

4.2 Architecture

To give a high-level overview of the *WAWLT* AI architecture and how the various subsystems fit together, we will first present a brief walkthrough of an idealized *WAWLT* play session. We will then elaborate on each of the indivdual subsystems in sections 3.3.1-3.3.7.

At the start of the play session, two players sit down and generate a new *WAWLT* scenario. The game generates a storyworld state database containing an initial cast of characters and institutions and some basic relationships between them. It then performs *backstory simulation* to quickly generate a history for the community as a whole. This setup process is described in section 3.3.1.

Control is then handed off to the players, who are prompted to choose some *author goals* (discussed in section 3.3.3) and a subset of all the characters at the symposium to participate in the first *scene*. The players do not yet know anything about the history of the storyworld and the preexisting relationships between the characters, so they pick a couple of characters with interesting-sounding names and traits to participate in the first scene. They also select a single author goal: "cast suspicion on Vincent", one of the two characters they selected.

The system prompts the players with five different *action suggestions* (discussed in section 3.3.4), many of which seem likely to motivate Vincent to pursue revenge against another character. This is because the suggestions are guided by the author goals that the players have selected. The players deliberate for a bit about which of these actions to perform, and eventually select an action in which Mikayla, another character in the scene, makes a disparaging comment about Vincent's research. A terse system-generated description of this action is added to the *transcript* (discussed in section 3.3.7), with an editable text box below it in which the players are free to write their own more detailed description of this action. The action's *effects* (discussed in section 3.3.2) are also run, updating the storyworld state database.

The players decide they want to find out more about Mikayla and Vincent, so they use the *characters* tab of the *storyworld investigator* (discussed in section 3.3.6) to look up Mikayla's and Vincent's *information cards.* They discover that Mikayla and Vincent are both doctoral students in the same lab, advised by a third character, Lea. They also discover that Mikayla and Vincent had previously worked on a major project together, but eventually both left the project over personal differences. This information helps the players develop a clearer picture of the relationship between Mikayla and Vincent up until this point, and allows them to write dialogue in their description of the insult action that makes reference to these past events.

After selecting several more system-suggested actions for Mikayla and Vincent to perform within this scene, the players decide that they have accomplished their current author goal of casting suspicion on Vincent. They end the scene, and are prompted to pick characters and author goals for a new scene.

In the meantime, other characters not participating in the first scene have also performed a number of *autonomous actions* (discussed in section 3.3.5), guided partly by the players' author goals, but with a greater degree of randomness involved in action selection. The players use the storyworld investigator to see what other events have happened recently, and use this to guide their selection of participants and author goals for the second scene.

The players also disagree briefly about whether the next scene should focus on establishing a conflict between two *characters* (Alex and Rashida) or between two *values* (comfort and safety). Eventually, they choose to compromise, selecting three author goals: one goal that explicitly focuses action suggestions on the establishment of a values conflict, and two goals that instruct the system to involve both Alex and Rashida in the plot as much as possible.

Generally, as the players proceed from the start to the end of the play session, they gradually shift from spending most of their time investigating the history of the storyworld and opening up new loose ends to spending most of their time trying to pull threads together and bring the story to a satisfying conclusion. The choice of author goals from scene to scene closely follows this arc: early in play, many of the players' goals focus on escalating tension and introducing new points of contention between characters, but in the last few scenes, the players select goals that steer action suggestions toward reconciliation between characters instead.

The following subsections describe individual subsystems of the overall architecture in greater detail.

4.2.1 Storyworld State Database

The state of the *WAWLT* storyworld is stored in a DataScript [94] database managed by the Felt [50] story sifting and simulation engine. Storyworld entities represented in the database include:

- **Characters.** A character has a name; a *role*, representing their job (e.g., “PhD student” or “janitor”); two *values* drawn at random from a pool of ten; and several other personality traits, which restrict the actions this character is willing and able to perform.
- **Relationships.** A relationship entity stores one character’s view of another, including both *impressions* formed of that character’s actions and *role relationship* information (for instance, in the case of an advisor/student relationship or a marriage). It also contains a numeric *charge* value representing the source character’s overall attitude toward the target, with positive values reflecting a positive attitude and vice versa. There are two relationship entities for every pair of characters, one pointing in each direction.
- **Impressions.** An impression represents a source character’s evaluation of a target character based on a particular introspection event. It has a *score*, which is summed together with the scores of other impressions between the same two characters to produce an overall running evaluation of the target character by the source. It also has a pointer back to its *cause*: the event that led to its creation. A single relationship may only be defined by up to three positive and three negative

impressions simultaneously. Stronger impressions tend to displace weaker ones over time.

- **Projects.** A project is an ongoing effort by one or more characters. It has a set of *contributors*, a name, a numerical *drama level* representing its troubledness, and some other detail information.

- **Institutions.** An institution is an organization with which characters can be affiliated. These are mostly used in generation of character backstories during backstory simulation.

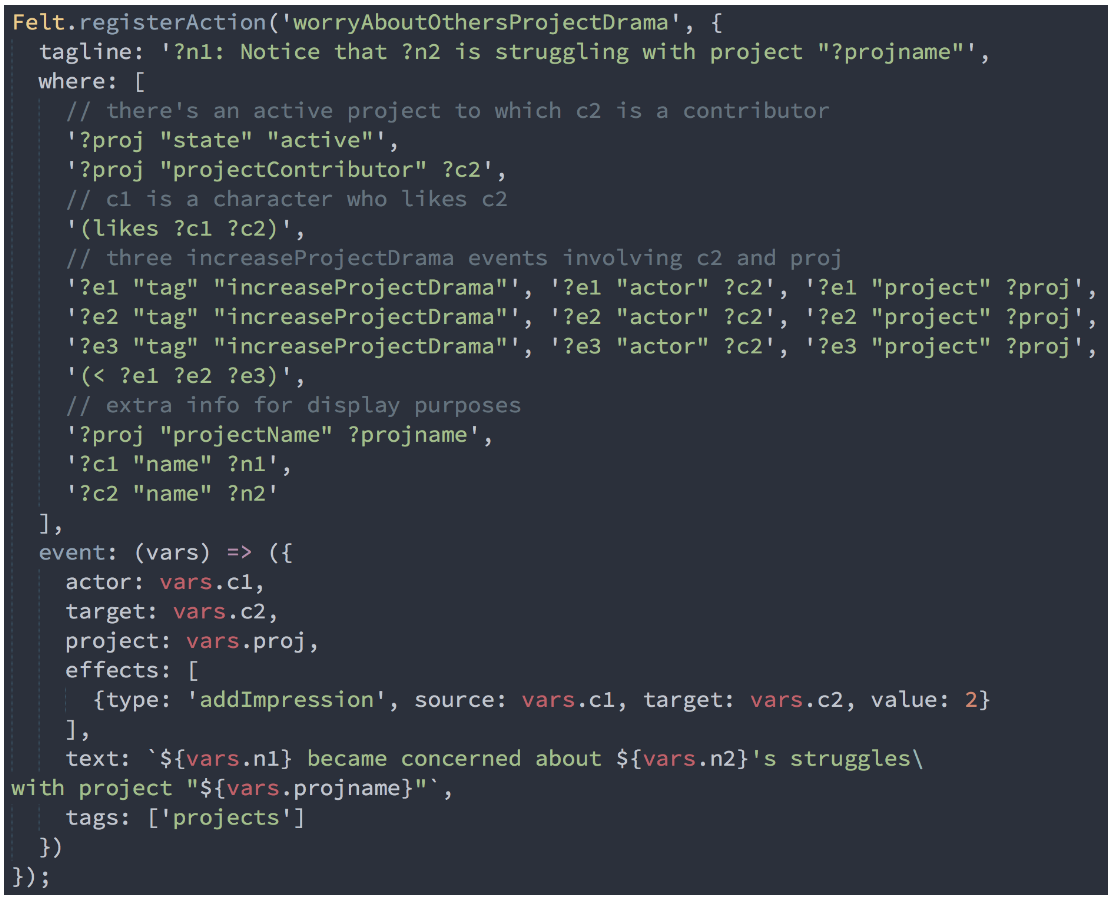
- **Events.** An event is a record of an action performed by a character. It has at minimum an *event type* drawn from the action definition; an *actor*, the character who performed it; a short textual *description*; and a *timestamp*. It may also have a number of *tags*.

At the beginning of each play session, a cast of 10 characters is initialized with random names, roles, values, and personality traits. Each pair of characters is assigned a random relationship charge value in both directions, for a fully connected character affect graph. The established character roles are then used to probabilistically set role relationships between particular characters where appropriate (“works for”, “advises”, etc).

Backstory simulation using higher-level, lower-resolution action definitions is then run to flesh out character life history and the shared intellectual and social history of the community for several years prior to meeting at the symposium. This backstory simulation draws from a pool of special actions, not available during gameplay, that operate at a larger time scale. Examples of these actions include “graduate”, “join institution”, “write paper together”, “publish a controversial book”, “take industry position”, and “take leave to parent”. Such coarse-grained actions don’t make as much sense in the context of a scenario playing out over the course of a few days, but they help bootstrap rich social context quickly, and make the characters feel more real.

Backstory simulation is intended to mitigate the *fear of the blank canvas* [59] in players by providing an interesting situation from the start, with many sites of potential narrative interest for players to explore together and begin building their story from. The generated backstory can also be referenced and queried at any time via the storyworld investigator. Browsing backstory can be useful as inspiration for creating a shared understanding of a persistent character, for sources of possible motivation for surprising character actions, or for finding and fleshing out a character who is new to the story so far.

After initial setup, all subsequent simulation actions (both chosen by players and autonomously running in the background) draw from a finer-grained pool of action definitions that make sense to happen in the course of a few days, on location at the symposium. These actions operate on the same state database initialized by the backstory simulation, gradually evolving it over the course of play.



```
Felt.registerAction('worryAboutOthersProjectDrama', {
  tagline: '?n1: Notice that ?n2 is struggling with project "?projname"',
  where: [
    // there's an active project to which c2 is a contributor
    '?proj "state" "active"',
    '?proj "projectContributor" ?c2',
    // c1 is a character who likes c2
    '(likes ?c1 ?c2)',
    // three increaseProjectDrama events involving c2 and proj
    '?e1 "tag" "increaseProjectDrama"', '?e1 "actor" ?c2', '?e1 "project" ?proj',
    '?e2 "tag" "increaseProjectDrama"', '?e2 "actor" ?c2', '?e2 "project" ?proj',
    '?e3 "tag" "increaseProjectDrama"', '?e3 "actor" ?c2', '?e3 "project" ?proj',
    '(< ?e1 ?e2 ?e3)',
    // extra info for display purposes
    '?proj "projectName" ?projname',
    '?c1 "name" ?n1',
    '?c2 "name" ?n2'
  ],
  event: (vars) => ({
    actor: vars.c1,
    target: vars.c2,
    project: vars.proj,
    effects: [
      {type: 'addImpression', source: vars.c1, target: vars.c2, value: 2}
    ],
    text: `${vars.n1} became concerned about ${vars.n2}'s struggles\
with project "${vars.projname}"`,
    tags: ['projects']
  })
});
```

Figure 4.3: An example *WAWLT* action definition.

4.2.2 Action Definitions

Character actions in *WAWLT* are defined as Felt actions. They have preconditions, which take the form of Felt sifting patterns, and effects, which update the database when the action is performed. When an action definition is registered, Felt precompiles its sifting pattern to a Datalog query, which can later be run against the storyworld state database to return all possible sets of parameter bindings for this action.

We draw a distinction between *external actions*, which involve characters acting on the world outside of themselves, and *introspection actions*, which involve characters reflecting on past events through the lens of a particular narrative frame. External actions, for instance, might include actions like "discuss a shared value with Rashida", "insult Vincent", or "sabotage Alex's experiment"; introspection actions might include actions like "speculate that Lea dislikes me" or "conclude that Mikayla is a mean person." Introspection actions often produce *impressions* that influence the introspecting character's attitude toward another character. By separating out character reasoning into a category of first-class actions that players can observe and perform directly, and that are added to the transcript for players to elaborate on within their stories, we hope to expose character reasoning to the players explicitly—and, thereby, to mitigate the *Tale-Spin effect* [129], in which a system is mistaken by players as less intelligent than it actually is due to insufficient exposure of the internal processes.

Introspection actions bring *story sifting*—the process of matching and narrativizing sequences of past simulation events—into the storyworld as a diegetic component of how simulated characters make sense of the world around them. In this way, story sifting plays a central role in *WAWLT*'s implementation of character subjectivity.

Most introspection actions require that the first matched event in a sequence of events took place within a certain window of recency. This helps ensure that characters don't suddenly decide to spend their time thinking about actions that took place a long time ago, unless something (such as a special-case introspection action, without recency-gating on matched events) specifically prompts a reevaluation of those past events in particular.

Impressions define characters' perceptions of other characters. One character's perception of another can be influenced by up to three positive and three negative impressions simultaneously. Impressions are formed through introspection actions, and have numerical strength values indicating how strongly (and in what direction) they influence the holder's perception of the target. The holder's overall numerical "charge", or liking, toward the target is the sum of the scores of all the holder's currently held impressions of the target. Stronger impressions displace weaker ones upon formation, so a character's perception of another is generally defined by their strongest positive and negative impressions of that character. Impressions can be communicated from one character to another, albeit generally in weakened form, through gossip actions.

Impressions are *WAWLT*'s answer to character knowledge modeling, which is a common element of similarly simulation-driven narrative systems. Tale-Spin [83], Talk of the Town [100], and Versu [28], for instance, all model character knowledge phenomena at a fine-grained level, tracking per-character awareness of individual facts about the world and allowing character knowledge to change substantially over time. In *WAWLT*, however, we avoid modeling character knowledge phenomena directly due to complexity. Instead, we assume that all characters know about every event as soon as it transpires, but that most characters don't care about most events until these events somehow become directly relevant to them. Character *knowledge* is thus replaced in our system by subjective impressions, which are substantially fewer in number and more narratively consequential than granular facts about the world.

From a social simulation perspective, *WAWLT* also differs substantially from previous similar systems in its handling of character motivation. Ensemble [108], for instance, treats character motivations—*volitions* in Ensemble terminology—as transient and implicitly derived, due to the system's automatic re-computation of volitions after every action. Our motivation model, in contrast, is more like Ceptre [73]'s "tokens": character motivations modeled as consumable "resources" in a linear logic framework, produced by actions and persistent in the world state until explicitly consumed by another action. Thus, motivations in *WAWLT* are both less transient and less implicit (due to their production by explicit character actions, generally introspection actions) than volitions in Ensemble and other, similar social simulation engines.

4.2.3 Author Goals

Author goals are intended to provide players with a compositional *player intent language* [74] that they can use to explicitly communicate their current creative intent to the system. This intent language is one of the main novel features in the *WAWLT*

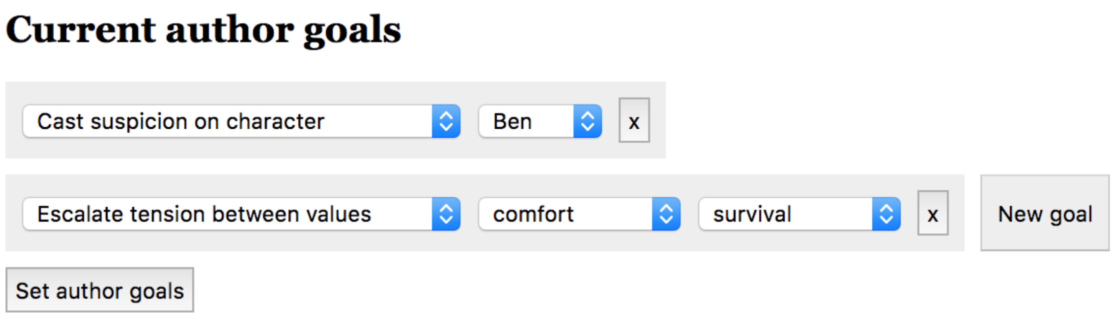


Figure 4.4: The *WAWLT* author goal selection interface.

architecture, and—we argue—an important feature for mixed-initiative co-creative architectures in general to support, due to its central role in enabling the system to respond intelligently to changes in player intent on a moment-to-moment basis.

Author goals currently include the following:

- Involve *character* in plot as *role*
- Cast suspicion on *character*
- Dispel suspicion on *character*
- Escalate tension between *character* and *character*
- Defuse tension between *character* and *character*
- Escalate tension between *value* and *value*
- Defuse tension between *value* and *value*
- Introduce false lead
- Dismiss false lead

- Custom author goal: *textual search query* (*weight*)

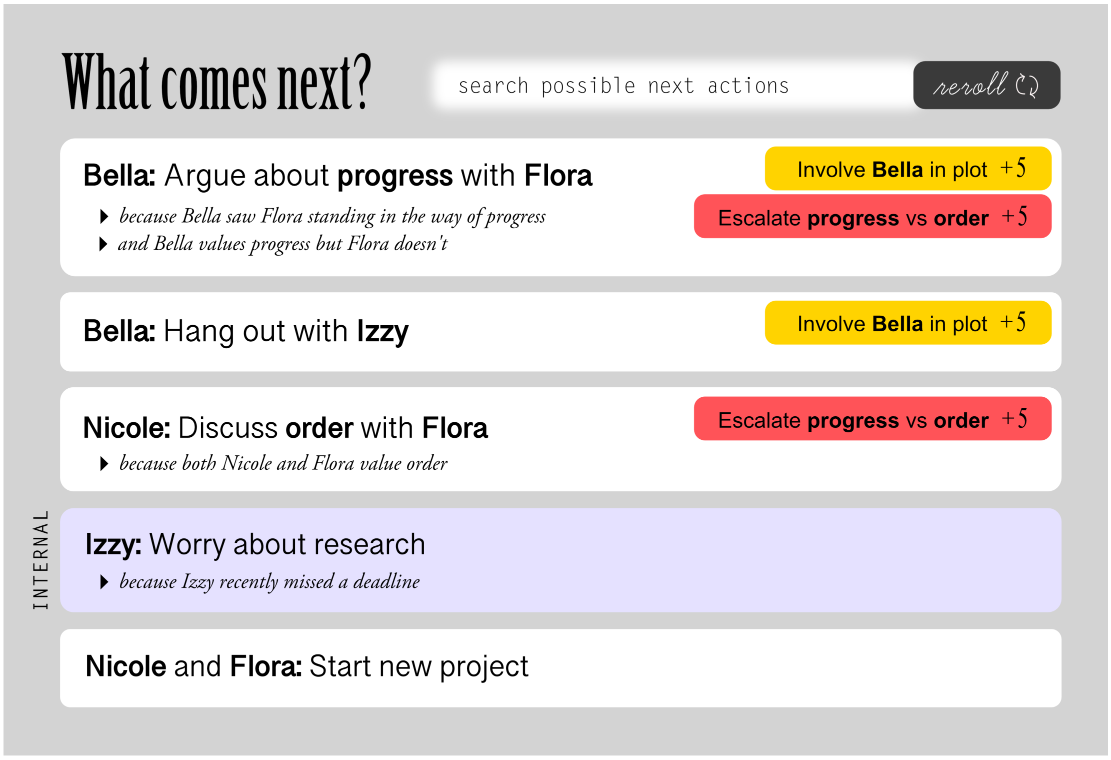
Italics in goal names indicate parameter slots. For instance, if players want to focus on actions involving a particular character, they can add this character to a *character* slot in any goal that provides one. Alternatively, players can also leave this slot empty, in which case the system will treat it as a wildcard that stands for “any character.”

The *custom author goal* option in particular allows players to specify finer-grained constraints on action suggestions than would otherwise be possible. When specifying a custom author goal, players can use a lightweight textual query language to restrict action suggestions to actions that contain certain substrings in their taglines; actions belonging to certain categories, such as “introspection actions” or “actions that involve projects”; and to filter action suggestions in a variety of other ways. Players can also specify the numerical weight that this custom author goal will contribute to any matched actions. This textual search mechanism greatly expands the expressiveness of the author goal language, but also requires the players to know what they’re looking for and how it might be described in this system. Therefore, we expect that custom author goals will be used more frequently later in gameplay sessions, and by players who have a better sense of the space of all possible actions within the simulation.

Author goals are used to rank all currently possible actions in order to provide players with action suggestions. Each author goal is associated with a heuristic function that takes a possible action as an argument and returns a numerical score representing the relevance of this possible action to this author goal. Whenever the system needs to provide players with action suggestions, it first generates a list of all currently possible actions, then evaluates these possible actions against the set of currently active author goals. Every active author goal contributes a score to each possible action, and the list of possible actions is sorted by the total combined score from all active author goals, so that the most goal-relevant actions appear closest to the top of the suggestions list. This provides players with support in navigating the space of currently possible actions, which may contain hundreds of possible actions at any given point.

In addition to ranking action suggestions for characters participating in the current scene, the system also uses author goals to guide the autonomous actions of characters who are currently "in the background." The process by which autonomous actions are selected and performed is described in greater detail in section 3.3.5.

Author goals are also intended to help players negotiate their intent with one another by making this intent explicit. Because players have to agree on the author goals that they are entering into the system, and because they are prompted to adjust their author goals at the start of every new scene, the system encourages players to regularly discuss their intent with one another, and players may have to explicitly argue for the things they want to have happen in the story. In this way, author goals can function similarly to the "palette" mechanism in tabletop storytelling games like *Microscope* [97], which requires players to explicitly discuss what they do and do not want to include in the story they are creating together.



What comes next?

search possible next actions

reroll

Bella: Argue about **progress** with **Flora**

- *because Bella saw Flora standing in the way of progress*
- *and Bella values progress but Flora doesn't*

Involve **Bella** in plot +5

Escalate **progress** vs **order** +5

Bella: Hang out with **Izzy**

Involve **Bella** in plot +5

Nicole: Discuss **order** with **Flora**

- *because both Nicole and Flora value order*

Escalate **progress** vs **order** +5

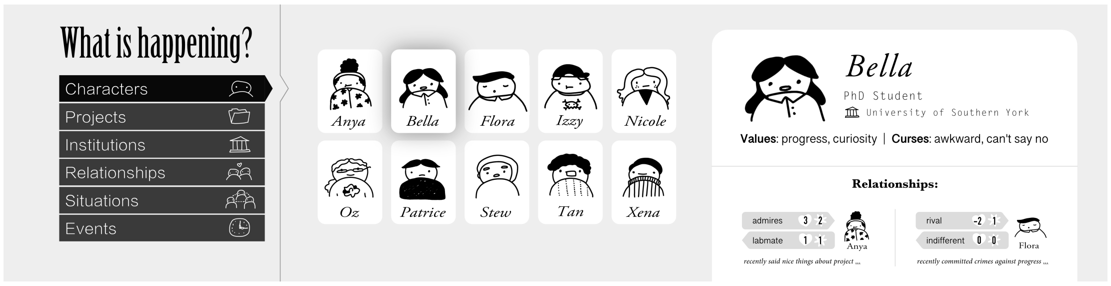
INTERNAL

Izzy: Worry about research

- *because Izzy recently missed a deadline*

Nicole and **Flora:** Start new project

Figure 4.5: The *WAWLT* action suggestion interface, showing the current top five next possible actions for players to choose among to enact and add to their transcript. Possible actions are scored according to the current, player-set author goals (e.g., "involve character (Bella) in plot," "escalate tension between value (progress) and value (order).")



What is happening?

Bella

PhD Student

University of Southern York

Values: progress, curiosity | **Curses**: awkward, can't say no

Relationships:

admires 3 2
labmate 1 1
Anya
recently said nice things about project ...

rival -2 1
indifferent 0 0
Flora
recently committed crimes against progress ...

Figure 4.6: The *WAWLT* storyworld investigator interface, showing a portion of a character information card.

4.2.4 Action Suggestions

The action suggestion interface displays the five most highly ranked possible actions for characters who are involved in the current scene. Whenever the storyworld state database is updated, *WAWLT* automatically recalculates which actions are currently possible. For each registered action definition, Felt executes the action's compiled Datalog query against the database, returning all possible sets of bindings for this action, and instantiates a "possible action" for each set of bindings. This complete list of possible actions is then used to drive both action suggestions for the players and autonomous actions performed by characters in the background.

Action rankings are based primarily on the current author goals, each of which contributes a positive score to possible actions that can be read as advancing this author goal in some way—and, possibly, a negative score to possible actions that may be seen as detracting from the realization of this author goal. Action rankings also take into account the base weight specified in each action definition as an indicator of the action's base narrative significance, since players may find it confusing if relatively mundane actions (e.g., "gossip about weather") are ranked above more inherently dramatic actions (e.g., "accuse of murder"). This base weight is multiplied together with the total score from author goals to produce an overall score for each action, which is then used to sort the list of possible actions and display the top five.

In the future, we may also expand the calculation of action scores to consider other factors. For instance, in order to prevent the same few actions from appearing as highly ranked suggestions over and over again, we may apply a ranking penalty to instances of the last few actions that were performed. In the context of introspection actions that involve a character ruminating about or reflecting on a particular set of past events, we may also take into account the subjective significance of the matched events to the character who is doing the rumination. Event significance calculation may consider such factors as the recency of the matched event, the strength of the relationships between the ruminating character and the event's participants, and perhaps some of the other factors suggested by the Indexter [13] model of event salience. These event salience heuristics would likely begin to resemble generic *sifting heuristics* [99] as proposed by Ryan, due to their basis in general-purpose models of why a sequence of events might be perceived as an interesting story.

Possible actions in the action suggestion interface are displayed alongside some details about why this action is currently possible. This includes a display of what active author goals contribute to the surfacing of this action suggestion, and with what strength; a short description of all the previous actions that partially caused this action, with the ability to expand the causality trace backward (to view the causes of the causes, and so on); and a short description of any other preconditions for this action, such as the presence of certain character traits on the action's protagonist, if any such preconditions exist. Introspection actions are also clearly marked as such, to distinguish them from external actions.

A search bar at the top of the action suggestion interface allows players to use the same lightweight textual query language used in custom author goals to rapidly filter action suggestions without having to add or update any author goals. If multiple top action suggestions have equal overall scores, actions with the same score are shuffled when suggestions are retrieved; we therefore also provide players with a “reroll” button, which allows them to re-randomize the sort order of actions with equal weights, possibly changing which actions are displayed in the suggestion interface.

4.2.5 Autonomous Actions

Action suggestions concern possible actions for characters who are active in the current scene. However, characters who aren’t involved in the current scene can also perform actions autonomously in the background. Like player-visible action suggestions, autonomous actions are influenced by author goals. Autonomous actions, however, are chosen via a weighted random selection process over some of the higher-scoring possible actions for offscreen characters, rather than selected by the player directly. For efficiency, autonomous actions reuse the set of calculated possible actions that are used to provide players with action suggestions.

Autonomous actions are intended to help mitigate writer’s block by ensuring that the storyworld will always continue to develop in significant ways, even if the players can’t think of anything interesting to do in a particular scene. We are inspired here by the design property of *incrementality* [59], which was found to be helpful in supporting player storytelling in some existing simulation-driven games. After a scene, players can use the storyworld investigator to check in on what the characters not involved in the scene have been up to in the background, which might inspire new directions for the story.

However, in practice, our playtesting (discussed in greater detail in the Playtesting section of this chapter) found that autonomous actions were not especially effective as a means of keeping the story moving forward in a coherent direction, based on the frequency with which playtesters of *WAWLT* self-reported directionlessness. Consequently, and in order to simplify the overall architecture of our systems, we removed autonomous actions from Loose Ends, the successor system to *WAWLT* discussed in chapter 6. The AI system in Loose Ends has an alternative means of asserting initiative over the direction of the storytelling process—namely through the autonomous suggestion of new storytelling goals.

4.2.6 Storyworld Investigator

Playtesting of previous versions of *WAWLT* revealed that players needed a way to proactively browse the full history of the simulated storyworld while writing their stories. In response, we created the *storyworld investigator*, which provides players with fine-grained tools for investigating the history and current state of the storyworld. The investigator is divided into several tabs, each of which displays a complete list of all instances of a certain type of storyworld entity—such as characters, relationships, projects, institutions, and events—and lets the player view *information cards* containing more detailed information about these entities.

Information cards are linked together with hyperlinks to enable rapid exploration of the web of storyworld entities. For instance, while viewing the information card for a particular character, links under the “Relationships” section allow rapid navigation to information cards containing more detailed information about this character’s relationships with other characters, including the events and impressions that played a role in shaping each character’s perception of the other.

The storyworld investigator also provides a *situations* tab, which allows players to make proactive use of story sifting to discover potential sites of narrative interest in the storyworld. This tab is equipped with a number of premade parametrized sifting patterns designed to help players locate emergent situations that might complicate the story in interesting ways: for instance, relationships in which the two involved characters have strongly incompatible assessments of one another, escalating cycles of revenge between characters, or long-standing instances of jealousy. As with author goals, players can leave these sifting patterns unconstrained or constrain them by specifying parameter values, for instance to view only instances of jealousy that involve a particular character.

In general, we expected that players would make especially extensive use of the storyworld investigator toward the start of a play session to familiarize themselves with the world’s backstory, including the existing relationships between characters. We also expected that players would frequently use the investigator when deliberating between scenes about what situations they intend to explore next—primarily in order to discover untapped or neglected sites of narrative potential (especially through the *situations* tab), to explore autonomous actions performed by “out-of-focus” characters during the most recently completed scene (especially through the *events* tab), and to learn more about characters and other entities they might want to spotlight in the future.

Our expectations for investigator use cases were largely borne out in playtesting, but the storyworld investigator was not used as extensively as we anticipated, despite being one of the more work-intensive components of the overall *WAWLT* design from both a design and engineering perspective. Additionally, the investigator did little to limit the sense of directionlessness that playtesters of *WAWLT* experienced. Consequently, and because Loose Ends features much less complicated simulation state than *WAWLT*, we removed most features of the storyworld investigator from Loose Ends. We retained a limited version of the characters tab (which shows only the basic traits of each character, and does not show updates to character goals or relationship states in real time) but folded it into the top of the transcript UI. We also folded the situations tab into the author goals area via system-generated author goal suggestions: when the AI system in Loose Ends detects a latent narratively potent situation, it proactively suggests one or more new author goals related to the development of this situation, which the player can either accept or reject.

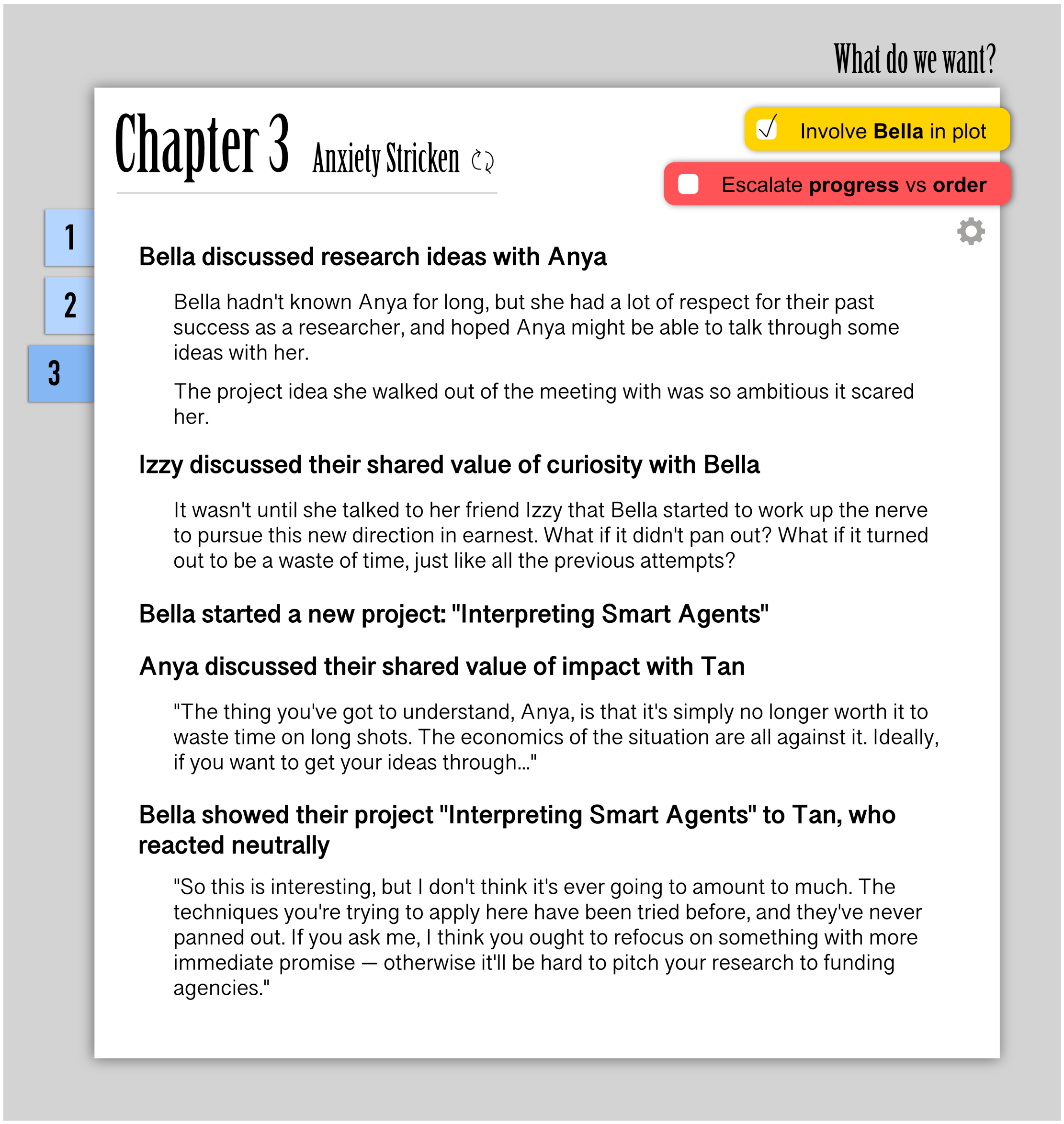
4.2.7 Transcript

The transcript holds a running record of “the story so far”: short system-generated summaries of all player-accepted action suggestions since the start of the play session, interleaved with more detailed player-generated prose descriptions of these actions. By the end of the play session, the transcript will constitute an “artifact of play” [41] summarizing the events of play, much like the map in *The Quiet Year* [4] or the board in *Threadsteading* [3].

By giving players a way to annotate events with their own descriptions, we aim to provide support for *extrapolative narrativization* [57]: a player storytelling behavior in which players seize on and elaborate minor details in the stories they tell about their play experiences, regardless of whether these details are explicitly modeled in the simulation. Free-text descriptions of events give players a place to decide for themselves what aspects of an event are most important, and to establish and reference recurring story elements that are not modeled in the computational system.

Player-generated prose descriptions of events are not read, interpreted or reasoned over by the system in any way. This allows players to flexibly "retcon", or revise their descriptions of past events to match an updated understanding of where the story is going. Event description text remains editable indefinitely and doesn't impact forward simulation, so future actions cannot be invalidated by edits to the descriptions of previous events. Allowing the system to read and respond to player-generated text would compromise this flexibility: if player-generated text was incorporated into the simulation directly, subsequent edits to a block of player-generated text might compromise the system's interpretation of an event that was used as the foundation of a running chain of causality between several additional events in the meantime.

By treating player-generated prose as opaque to the system, we also aim to distinguish *WAWLT* from co-creative writing systems based on language models, such as *Creative Help* [98]. These systems treat prose-level suggestions as first-class while making no attempt to model causality at the level of discrete events or overall plot structure. We believe that these systems, although useful for injecting moment-to-moment "what



Bella discussed research ideas with Anya

Bella hadn't known Anya for long, but she had a lot of respect for their past success as a researcher, and hoped Anya might be able to talk through some ideas with her.

The project idea she walked out of the meeting with was so ambitious it scared her.

Izzy discussed their shared value of curiosity with Bella

It wasn't until she talked to her friend Izzy that Bella started to work up the nerve to pursue this new direction in earnest. What if it didn't pan out? What if it turned out to be a waste of time, just like all the previous attempts?

Bella started a new project: "Interpreting Smart Agents"

Anya discussed their shared value of impact with Tan

"The thing you've got to understand, Anya, is that it's simply no longer worth it to waste time on long shots. The economics of the situation are all against it. Ideally, if you want to get your ideas through..."

Bella showed their project "Interpreting Smart Agents" to Tan, who reacted neutrally

"So this is interesting, but I don't think it's ever going to amount to much. The techniques you're trying to apply here have been tried before, and they've never panned out. If you ask me, I think you ought to refocus on something with more immediate promise — otherwise it'll be hard to pitch your research to funding agencies."

Figure 4.7: The *WAWLT* transcript editing interface. Bold text in the transcript is system-generated, non-bold text is authored by the players.

happens next" suggestions that keep players from becoming completely stuck, fail to provide other, more important forms of creative support. In particular, because they lack an explicit world model, these systems frequently issue suggestions that contradict previously stated facts about the world, exacerbating the existing problem of maintaining consistency within a fictional world and distracting from the creativity support features they do provide. We prefer to offload the difficult problem of maintaining consistency to the computer, which can excel at this task given the right kind of architecture, and free up the humans to focus on authoring prose. For the sake of clarity, we thus focus the *WAWLT* AI architecture primarily on the provision of plot-structural or event-level suggestions, grounded in an explicit world model whose consistency is maintained by the system.

4.3 Playtesting

We conducted playtests with three solo players and two groups of paired players. Initially, each playtester was given a brief introduction to the project and the different parts of the user interface. Playtesters were then instructed to think aloud during their interaction with the game for 5-15 minutes at the player's discretion. Paired players engaged with a single instance of *WAWLT* simultaneously on a single computer, under the same conditions as the individual playtesters.

Broadly speaking, we found that the current version of *WAWLT* already supports player creativity in some of the intended ways, and is capable of producing an enjoyable play experience. Playtesters had little difficulty making use of the game's primary mechanics once they were introduced. All playtesters, even those who initially struggled to make the pieces of their story fit together into a larger storyline, eventually found themselves excited or curious to discover what would happen next in the story. All playtesters also expressed overall enjoyment of the play process. Six of seven playtesters (including all four paired playtesters and two of three solo playtesters) reported some sense of ownership over the story they produced through play. Moreover, the paired playtesters in particular expressed a great degree of enjoyment of the play experience; desire to continue working on the story (to such an extent that they were vocally disappointed that they could not continue at the conclusion of the playtest session); and feeling that what happened in the storyworld was somehow "real".

Nevertheless, there were also some significant points of confusion among players. Five players (including three of the four paired players) reported a sense of directionlessness at least once during the play process, suggesting that the system's action recommendations were not always sufficient to provide players with a sense of narrative structure. In one paired-playtesters group, both players initially assumed that author goals were intended primarily to be used by the system to filter and prioritize action suggestions, without realizing that they were also intended to be used as a way to encourage multiple simultaneous human players to negotiate intended story directions. Debriefing after the playtest also indicated that three players (including both of the paired players in one group) at some point forgot that the author goals existed, although the paired players "rediscovered" the author goals when a minor creative conflict briefly emerged between them.

The success of the paired playtesters in particular suggests to us that the creativity support features we provide in *WAWLT* are, like their counterparts in tabletop storytelling games, perhaps useful for individual players, but especially transformative when helping to scaffold and structure co-creativity in a multiplayer context, where negotiation between players regarding the content and direction of the story they intend to tell becomes a central part of play. As a result, we have since begun to consider *WAWLT* a multiplayer game primarily, while still aiming to support solo play as a viable mode of use.

4.4 Discussion

4.4.1 Story Sifting

At an implementation level, story sifting in *WAWLT* is functionally identical to precondition matching where some of the preconditions happen to involve sets of related storyworld events. However, from a design perspective, story sifting is not just precondition matching, but matching *and applying a particular subjective narrative frame* to the matched events. The execution of sifting patterns produces a variety of valid possible “readings” or interpretations of the same set of events, and even though these interpretations may be mutually incompatible, it is up to another subsystem—or to the player—to determine which readings to build a story around going forward. This distinction between sifting and precondition matching is made clear in *WAWLT*’s use of sifting patterns to implement subjective character reasoning, but is also present in more conventional sifting-based systems, such as Ryan's Sheldon system [99]: many valid sifting pattern matches are never promoted to a "real" part of the story, and a number of aesthetic judgment calls are made by the system as to which ones will be included in the story that gets told. Sifting patterns therefore bear some similarity to "rhetorical devices" in *Terminal Time* [79], producing a particular reading of a set of events that may be used or discarded depending on its suitability to the author's current rhetorical goals.

In this sense, we have found that story sifting can be viewed as a *thematics for narrating the operation* of precondition matching, in the same sense that Agre argues AI techniques in general tend to simultaneously provide "both a method for designing artifacts and a thematics for narrating [their] operation." [2] As in *WAWLT*, using the language of story sifting to discuss what is functionally precondition matching can help to inspire design approaches that might not have been considered otherwise, despite their longstanding technical possibility.

4.4.2 Simulation Design

Ryan, in his dissertation [99], proposes two key design patterns for simulations intended to support story sifting: *causal bookkeeping*, or explicit tracking of the causal relationships between storyworld events; and *contingent unlocking*, or the use of action preconditions that check for the existence of certain past events to support gradually escalating event "storylines", such as a "flirt" event between two characters unlocking a "secret tryst" event between these same characters as a later possibility.

WAWLT makes extensive use of causal bookkeeping, albeit with a small twist. In Ryan's Hennepin simulation engine, causal relationships are tracked directly between external actions taken by characters on the world, and actions may explicitly cause other subsequent actions to be queued for later performance at the moment the first action in a causal chain is performed. In *WAWLT*, however, we impose an authoring convention that external actions never cause other external actions directly: instead, external actions may cause introspection actions, which may in turn produce interpretations that cause other external actions. In this sense, causality in *WAWLT* is more frequently inferred after the fact (by characters performing introspection actions that project a particular causal interpretation onto a set of matched past events) rather than explicitly determined at the moment an action is performed (as with queuing of future actions in Hennepin.) Nevertheless, we—like Ryan—find it useful for the system to keep track of the causal relationships between actions, especially for the purposes of debugging and exposing character motivations to players.

Contingent unlocking, meanwhile, is used extensively within *WAWLT*'s simulation design, and differs from Ryan's interpretation of contingent unlocking in few ways if any. In particular, contingent unlocking enables plausibly gradual escalation of conflict in the *WAWLT* storyworld from less to more dramatic interactions between conflicting characters. Certain interpretation actions unlocked by repeated antagonism, for instance, effectively enable one *WAWLT* character to adopt the attitude that "this is the last straw", unlocking a variety of retaliation actions that allow for the further escalation of a simmering conflict between two characters who have vaguely disliked one another for a while.

Notably, the design of the *WAWLT* simulation imposes a fairly high degree of overhead on the authors of simulation actions, who must carefully compose preconditions that make actions available only under appropriate circumstances. In our later work on Loose Ends (as discussed in chapter 6), we moved away from a precondition-heavy approach to action suggestion and instead used a more sophisticated formalism for author goals to prioritize the most plausible action suggestions within a pool of all possible action type/character combinations, even those that do not make much sense from a causal perspective. This alleviates much of the burden of action authoring but also results in less sensical action suggestions when exploring possible next actions beyond a handful of top-ranked action suggestions, which players took note of in Loose Ends playtesting. We believe it is likely that the appropriate level of causal modeling for systems similar to *WAWLT* and Loose Ends falls somewhere between the levels of detail used in these two systems.

4.4.3 Author Goals

Author goals were initially implemented via speculative execution of possible actions. Under this formulation, every author goal included a Datalog query fragment which could be executed against the simulation state database to determine the extent to which the goal in question had been realized. Each possible action was executed speculatively, producing one updated version of the simulation state database for each action, and each author goal's query was then run against the current and all speculatively updated versions of the database. By taking the difference between the number of matches found for each author goal query before and after each possible action was performed, a score was derived for each possible action indicating its overall suitability to the current author goals. These scores were then used to rank the actions.

This approach provided substantial authorial leverage, as author goals implemented in this way do not need to include any knowledge about what specific events are possible in the storyworld: events can be judged as advancing or detracting from author goals based on their actual effects alone. However, although elegant, this approach proved too inefficient to be practical with nontrivial numbers of actions and author goals, so the current implementation of author goals instead uses heuristic functions to evaluate possible actions. Each author goal's heuristic function takes a possible action as an argument and returns a numerical score indicating this possible action's suitability for this author goal. This makes goal evaluation more computationally tractable, but entangles action authoring with author goal implementation in a way that substantially decreases authorial leverage, frequently forcing action authors to modify the heuristic functions of relevant author goals when they introduce a new action or make a substantial change to an existing action definition. The use of goal heuristic functions that rely on action *tags* rather than specific action names to determine goal relevance has somewhat mitigated this added authorial burden, because it is relatively easy to tag new actions in such a way that they become "visible" to author goals that are already looking for existing actions with these same tags, but this merely reduces the frequency of the problem rather than eliminating it entirely.

In Loose Ends, we moved back toward speculative execution for the improved authorial leverage that this model provides. This was made possible from a performance perspective by a more sophisticated sifter known as Winnow (discussed further in chapter 5). By maintaining a substantial amount of intermediate state about partially fulfilled author goals, Winnow makes it possible to check candidate actions against the complete pool of current author goals (to determine which author goals would be advanced or cut off by each candidate action) much more quickly.

4.4.4 Effect Handlers

A wide variety of existing generative narrative systems, including planning-based systems [93, 132] and linear logic-based systems like Ceptre [73], define character actions in terms of preconditions and effects. Action effects, in turn, are commonly defined in terms of add and delete lists, which describe facts to add and delete from a blackboard or database of logic sentences when an action is performed. Effects in Felt actions are defined similarly, but Felt provides an additional layer of abstraction—called *effect handlers*—over add and delete lists. Effect handlers are Felt's implementation of *procedural effects*, a feature of some planning-based systems (such as the system driving NPC behavior in *F.E.A.R.* [90]) that allows for the implementation of dynamic changelists. We found this feature to be particularly beneficial in our implementation of *WAWLT*.

A Felt effect handler is a function, defined by the authors of a particular Felt simulation domain, that takes in both the storyworld state database and some other parameters, then returns a list of facts to add and delete from the database. Essentially, effect handlers represent a way to make changes to the database in terms of the overall effect you want to achieve (e.g., "create an advisor/student relationship between these two characters") rather than the specific individual facts you want to add and delete from the database.

From a simulation implementation perspective, this separation between intent and implementation allows for the construction of safer and cleaner interfaces for high-level changes that frequently involve the simultaneous editing of many individual facts. Robust effect handlers can be written to change *all* relevant individual facts consistently whenever you make a specific kind of high-level change, so you don't have to manually enforce this through action authoring discipline or worry about forgetting some individual changes within a larger changelist intended to achieve a particular overall effect. This also greatly eases refactoring of how simulation updates are performed, since all changes to individual facts take place through a smaller number of cleanly defined effect handlers: actions are not permitted to change individual facts directly, they may only invoke effect handlers.

Simultaneously, from an action authoring perspective, this separation between interface and implementation allows action authors to use and reuse effect handlers without knowing how they work exactly, shielding them somewhat from the internal complexity of the database. We also found that, for action authors, a well-defined set of effect handlers can effectively serve as a rapidly skimmable catalog of the kinds of state changes that actions can make in the storyworld. This can serve to inspire the design of new character actions by hinting at the possibility of introducing more ways to flavorfully achieve each available effect.

Finally, because effect handlers are first-class functions that receive the entire database as an argument, they are substantially more flexible than hand-authored literal changelists, and are able to consider broader context when calculating the sets of specific changes that ought to be made to individual facts when trying to achieve a specific outcome in the database. For instance, consider the `addImpression` effect handler, which is used to conditionally update the set of impressions influencing one character's evaluation of another. This complicated effect handler makes several queries against the state of the database to determine whether or not it should even add the new impression at all; which (if any) existing impressions it should displace to "make room" for the new impression; and how the overall relationship between the characters should be updated as a result. This effect handler, whose behavior is core to the *WAWLT* social simulation, could not be replaced by a static changelist; the changelist of an action that adds an impression to the database while obeying the elaborate rules governing impression formation must necessarily be dynamic.

4.5 Conclusions and Future Work

In this chapter, we presented the AI architecture of *Why Are We Like This?*, an AI-supported storytelling game intended to provide explicit support for the kinds of player storytelling practices seen in many simulation-driven games. *WAWLT* makes extensive use of story sifting, both to implement character subjectivity and to provide players with tools for investigating the history and current state of the storyworld. *WAWLT*'s architecture is also intended as an argument for a central claim: that machines should support player storytelling practices by providing players with intelligent plot direction suggestions, drawn from an ongoing social simulation and guided by player utterances in a machine-understandable intent language, realized here in the form of *author goals*.

Broadly speaking, our preliminary playtesting of *WAWLT* supported the idea that we were on the right track with regard to mitigating overwhelm. Players of *WAWLT* were able to jump right into the storytelling process and found the system both usable and fun, and the system-generated action suggestions struck an effective balance between generating suprising continuations and allowing players to retain a sense of ownership of the stories they created. However, playtesters' frequent reports of uncertainty as to where the story should go next suggested that we had not made much progress on mitigating directionlessness. Preserving the successes of Felt and *WAWLT* while also addressing their deficiencies in helping players create coherent longer-term story structure became our primary goal for the next pair of systems we developed—Winnow (discussed in chapter 5) and Loose Ends (discussed in chapter 6).

Chapter 5

Winnow

Story sifting (as implemented in our first sifter Felt, and used as infrastructure for the implementation of our first narrative instrument *Why Are We Like This?*) is demonstrably useful for mitigating the problem of *overwhelm* in simulation-driven interactive emergent narrative gameplay. However, as evident in our playtesting of *WAWLT*, Felt-style sifting does little on its own to solve the problem of *directionlessness*, in which players struggle to compose coherent longer-term story structures despite the provision of short-term action suggestions by a sifting-based AI system. Additionally, certain desirable architectural features of *WAWLT*-like narrative instruments (such as the ability to discover *almost actions*, or narrative components whose preconditions are partially but not fully met, and the ability to rank action suggestions via efficient *speculative execution* of all possible next actions) cannot be implemented performantly atop Felt. Our next sifter, Winnow, addresses both of these problems.

From the perspective of the directionlessness problem, existing sifters share one major weakness: their fundamentally *retrospective* nature. In retrospective sifters like Felt, microstories can only be recognized once they have run to completion, and not while they are still in the process of playing out. Though a purely retrospective approach to sifting enables some forms of play experiences well [109], it limits the capacity of systems based on story sifting to reason about and intelligently foreshadow future narrative possibilities. Anticipation of future narrative outcomes is essential to how humans engage with stories [69]; in narrative generally, readers who anticipate future outcomes may come to desire or dread these outcomes, and in narrative games, players may act to increase or decrease the likelihood that anticipated outcomes will occur. If a sifting-based system is only able to match events against narrative frames retrospectively, its capacity to play into anticipation by procedurally foreshadowing possible outcomes will be limited, as will its ability to make sense of player actions that were performed with anticipated outcomes in mind.

Meanwhile, from a performance perspective, Felt struggles to support features like almost action discovery and speculative execution because it fails to conserve information about partial sifting pattern matches from one pattern execution to the next. Consequently, each time Felt runs a sifting pattern, it has to inspect the entirety of the storyworld state database to locate plausible bindings for every logic variable in the pattern being executed—even if the storyworld state database has not changed at all, or changed only minimally, since the last time this pattern was executed. Since almost action discovery and speculative execution both involve the rapid execution of many very similar story sifting patterns, these features could be made much more feasible from a performance perspective if Felt was able to preserve partial information about plausible logic variable bindings across multiple executions of similar or identical patterns.

Winnow is a domain-specific language for story sifting that simultaneously addresses both of these issues by providing affordances for incremental sifting. Winnow introduces clear acceptor-based semantics around the maintenance of *partial* sifting pattern matches, which can be narrativized or exposed to a human user prior to their completion. Additionally, where incremental sifting is not required, Winnow sifting patterns can be compiled directly to Felt sifting patterns for Felt-equivalent performance on retrospective sifting tasks. This enables the use of Winnow as a more human-friendly syntax for the specification of Felt sifting patterns.

5.1 Related Work

See the Related Work section of chapter 3 for background on story sifting.

5.2 Motivation

Broadly speaking, prior attempts at story sifting share a major weakness when applied to the challenge of *prospective* sifting in the context of a still-running simulation. Because sifting patterns are conventionally written as descriptions of *complete* microstories, existing sifters have no way to anticipate (given a complete sifting pattern) whether that pattern is likely to be fulfilled in the future or not.

Given a library of complete sifting patterns, we could perhaps enable prospective sifting by creating several partial variants of every complete pattern, each of which matches a subset of events leading up to the complete pattern's realization. However, if these partial variants are hand-authored, then substantially more authoring effort (both to create and maintain partial variants) is required per pattern. The temporally unstructured nature of Sheldon and Felt sifting patterns makes it difficult to generate partial pattern variants automatically, because the precise details of when certain constraints must hold are not necessarily clear from a complete retrospective pattern: for instance, if a sifting pattern states that the perpetrator of a particular event must have the "selfish" trait, does that character need to retain the "selfish" trait throughout the entire event sequence, or is the character's personality allowed to change before or after certain pivotal events occur? And regardless of how partial pattern variants are created, all of these variants must be re-run repeatedly to detect new partial matches every time a new event occurs—necessitating the computationally expensive repeat evaluation of numerous pattern constraints.

Winnow represents our solution to these difficulties. By imposing a temporally divisible structure on sifting patterns, Winnow requires its users to clarify when exactly in a pattern's evaluation each constraint must hold. As a result, whenever a new event occurs, Winnow only needs to check whether the event *initiates* any new partial pattern matches (by matching the first temporal stage of a complete sifting pattern), *advances* any already-tracked partial pattern matches (by matching the *next* temporal stage of the relevant pattern), or *invalidates* any of these partial matches (by matching a negative constraint in the relevant pattern). This dramatically reduces the amount of computational work that needs to be performed per event, since only a small subset of each pattern's constraints must be evaluated when a new event occurs. Additionally, it fully relieves users from the authoring burden of creating and maintaining partial variants of complete sifting patterns by hand.

5.3 System Description

Winnow is an open source[1] domain-specific declarative query language for incremental story sifting. It is used to write sifting patterns. Each Winnow sifting pattern describes a sequence of interrelated events that constitute a potentially interesting microstory and enables the computer to detect instances of this microstory as they emerge.

Here is an example Winnow sifting pattern, modeled after the "violation of hospitality" Felt sifting pattern presented by Kreminski et al. [60]:

```
(pattern breakHospitality
  (event ?e1 where
    eventType: enterTown,
    actor: ?guest)
  (event ?e2 where
    eventType: showHospitality,
    actor: ?host,
    target: ?guest,
    ?host.value: communalism)
  (event ?e3 where
    tag: harm,
    actor: ?host,
    target: ?guest)
  (unless-event between ?e1 ?e3 where
    eventType: leaveTown,
    actor: ?guest))
```

[1]https://github.com/mkremins/winnow

This pattern attempts to match a sequence of events in which a `?guest` character enters a town; is shown hospitality by a `?host` character, who values communalism; but then is harmed in some way by the `?host` before the `?guest` has a chance to leave town.

As in Felt, identifiers prefixed by the `?` character represent *logic variables* that will be bound as the pattern is matched, and whose values must be consistent across the match as a whole. A single complete match for the `breakHospitality` pattern would include bindings for three event variables (`?e1`, `?e2` and `?e3`) and for the characters `?guest` and `?host`.

The sequence of `event` clauses present in a sifting pattern's body constitutes the pattern's *kernel*. A pattern seeks to match an ordered sequence of events corresponding to its kernel, disregarding events that take place between kernel events unless they match one of the pattern's `unless-event` clauses (which can be used to invalidate pattern matches if an event with certain characteristics intervenes between a specified pair of kernel events).

Like Felt, Winnow is implemented in JavaScript and uses the DataScript library[2] as its backend for data storage and Datalog query execution. In the DataScript dialect of Datalog, facts are represented as RDF-like triples of the form `[entity attribute value]`; each triple can be read as an assertion that the `entity` on the left (usually a numeric ID) has an `attribute` whose name is given in the middle and whose `value` is given on the right. All Winnow sifting patterns ultimately operate over facts of this

[2]`https://github.com/tonsky/datascript`

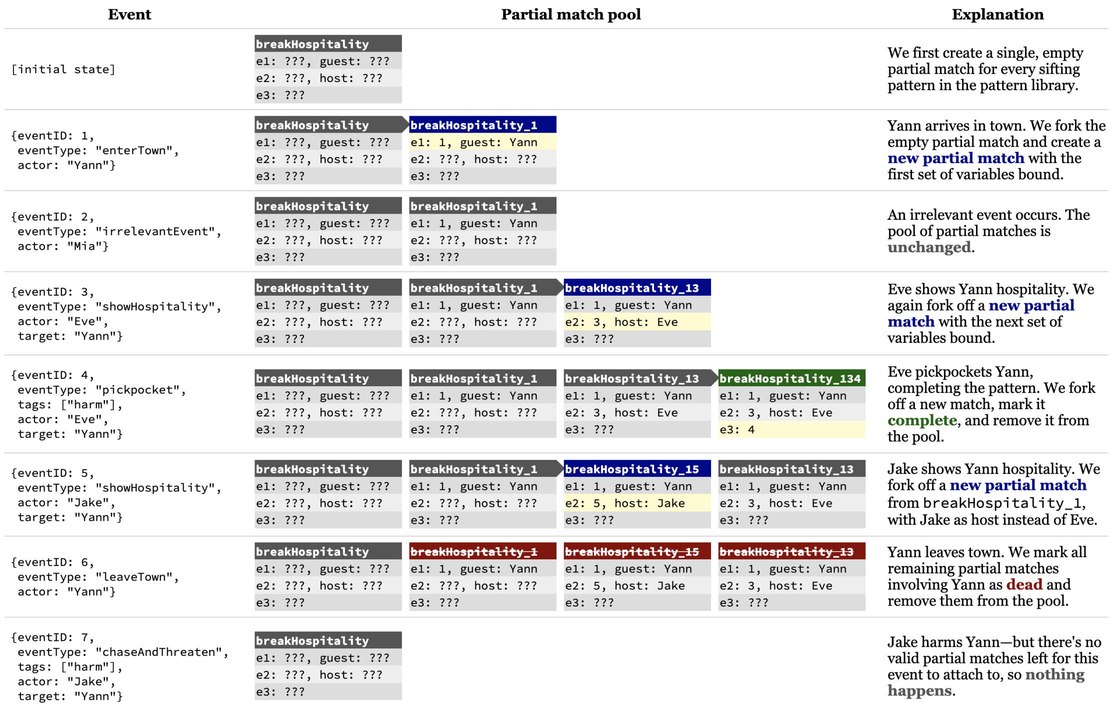
form. Many simulation engines store events and other data as graphs of related entities, with each entity structured as a JSON-like set of key/value pairs; these JSON-like data formats can often be straightforwardly translated into sets of DataScript facts, so we use these representations interchangeably in this chapter.

5.3.1 Incremental Execution

Winnow's incremental execution mode revolves around the maintenance of a pool of *partial matches*. Each Winnow sifting pattern is compiled to an acceptor that, given a partial match (i.e., a set of logic variable bindings for the first N events of this sifting pattern) and a new event to consider, performs one of several possible actions:

- **Dies** if this event matches an active `unless-event` constraint on this event sequence.
- **Forks** and **accepts** if this event matches the specification of kernel event $N + 1$ in the sifting pattern. The "parent" partial match is left unchanged (in case an alternative means of advancing this partial match is later discovered), while the "child" match has the new event (and associated logic variable bindings) pushed onto it.
- **Passes** (i.e., remains unchanged) otherwise.

When a new event occurs, it is added to the database and checked against all the active partial matches. Where applicable, these partial matches are then updated based on the new event, and dead and completed partial matches are removed from the pool.



Event	Partial match pool				Explanation
`[initial state]`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???				We first create a single, empty partial match for every sifting pattern in the pattern library.
`{eventID: 1, eventType: "enterTown", actor: "Yann"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???	**breakHospitality_1** e1: 1, guest: Yann e2: ???, host: ??? e3: ???			Yann arrives in town. We fork the empty partial match and create a **new partial match** with the first set of variables bound.
`{eventID: 2, eventType: "irrelevantEvent", actor: "Mia"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???	**breakHospitality_1** e1: 1, guest: Yann e2: ???, host: ??? e3: ???			An irrelevant event occurs. The pool of partial matches is **unchanged**.
`{eventID: 3, eventType: "showHospitality", actor: "Eve", target: "Yann"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???	**breakHospitality_1** e1: 1, guest: Yann e2: ???, host: ??? e3: ???	**breakHospitality_13** e1: 1, guest: Yann e2: 3, host: Eve e3: ???		Eve shows Yann hospitality. We again fork off a **new partial match** with the next set of variables bound.
`{eventID: 4, eventType: "pickpocket", tags: ["harm"], actor: "Eve", target: "Yann"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???	**breakHospitality_1** e1: 1, guest: Yann e2: ???, host: ??? e3: ???	**breakHospitality_13** e1: 1, guest: Yann e2: 3, host: Eve e3: ???	**breakHospitality_134** e1: 1, guest: Yann e2: 3, host: Eve e3: 4	Eve pickpockets Yann, completing the pattern. We fork off a new match, mark it **complete**, and remove it from the pool.
`{eventID: 5, eventType: "showHospitality", actor: "Jake", target: "Yann"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???	**breakHospitality_1** e1: 1, guest: Yann e2: ???, host: ??? e3: ???	**breakHospitality_15** e1: 1, guest: Yann e2: 5, host: Jake e3: ???	**breakHospitality_13** e1: 1, guest: Yann e2: 3, host: Eve e3: ???	Jake shows Yann hospitality. We fork off a **new partial match** from `breakHospitality_1`, with Jake as host instead of Eve.
`{eventID: 6, eventType: "leaveTown", actor: "Yann"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???	**~~breakHospitality_1~~** e1: 1, guest: Yann e2: ???, host: ??? e3: ???	**~~breakHospitality_15~~** e1: 1, guest: Yann e2: 5, host: Jake e3: ???	**~~breakHospitality_13~~** e1: 1, guest: Yann e2: 3, host: Eve e3: ???	Yann leaves town. We mark all remaining partial matches involving Yann as **dead** and remove them from the pool.
`{eventID: 7, eventType: "chaseAndThreaten", tags: ["harm"], actor: "Jake", target: "Yann"}`	**breakHospitality** e1: ???, guest: ??? e2: ???, host: ??? e3: ???				Jake harms Yann—but there's no valid partial matches left for this event to attach to, so **nothing happens**.

Figure 5.1: A visualization of Winnow incrementally executing the `breakHospitality` sifting pattern over a sequence of events in which a character **Yann** enters town and is first shown hospitality by, then harmed by, two other characters: **Eve** and **Jake**. As the structured events on the left are added to the database of storyworld state one by one, the pool of active partial pattern matches evolves as shown in the middle and explained on the right.

Figure 5.1 shows an example of incremental sifting pattern execution featuring the `breakHospitality` pattern defined previously. This example demonstrates all the core features of Winnow's incremental execution mode:

- Events that advance partial matches (1, 3, 4, and 5) are accepted onto forks of the matches they advance.

- Irrelevant events (2) do not change the pool of partial matches in any way.
- Events that match active `unless-event` constraints on partial matches (6) cause those partial matches to be marked as dead and removed.
- When all of a partial match's logic variables are bound (`breakHospitality_134`), the match is marked as complete and removed from the pool.
- Partial matches (e.g., `breakHospitality_1`) remain in the pool after they are advanced once (3) so that they can later be advanced again (5) in a different way.
- Events that might have advanced a partial match (7) are ignored if the partial match that they might have advanced was previously removed from the pool.

To determine whether an event matches an `event` or `unless-event` clause in the context of a particular partial match, Winnow translates the clause to a Datalog query and runs the query against the database. For instance, in Figure 5.1, to determine whether event 3 matches the `?e2` event clause in the context of partial match `breakHospitality_1`, Winnow executes the following query:

```
[3 "eventType" "showHospitality"]
[3 "actor" ?host] [3 "target" "Yann"]
[?host "value" "communalism"]
```

The results of this query are then used to establish bindings for the `?e2` and `?host` logic variables in the new partial match `breakHospitality_13`.

When maintaining a large pool of partial matches, Winnow runs many of these small queries per event—at least one per partial match, and more than one for each partial match against a sifting pattern with active `unless-event` constraints. However, the results of these queries are often very fast to compute, because the values of most involved logic variables (including the ID of the event being tested and the values of any previously-bound logic variables stored in the partial match) are already known. Therefore, only a small number of facts need to be checked to establish possible bindings for the variables that remain unbound.

5.4 Use Cases

5.4.1 Autonomous Incremental Sifting

Marie-Laure Ryan's characterization of how baseball commentators narrativize gameplay in real time [102] provides a strong motivating example for incremental sifting. In Ryan's analysis, commenting on a live baseball game involves a combination of looking back at what has already happened and looking forward at what might happen in the future, then weaving a narrative that seems coherent at the present moment while speculating about likely future outcomes to create suspense. A system that seeks to fully reproduce this kind of live gameplay commentary must therefore have some capability to speculate about the future, in addition to the capability to retrospectively interpret the past.

Consider Ryan's analysis of the FATAL ERROR theme in the baseball commentary she has selected for study. In a key moment, an outfielder forgets to flip down his sunglasses; as a result, he fails to track and catch a fly ball, allowing a runner from the opposing team to score. The broadcasters immediately narrativize this blunder as a FATAL ERROR by projecting their narration forward to a possible future in which the game is lost as a result. In Ryan's words:

> The emplotment of the game combines a retrospective interpretation with the prospective evocation of a possible outcome. A whole game is projected on the basis of the play just completed—a game in which the score stands as it is, and the future adds nothing to the story.

A system that uses purely retrospective story sifting to make sense of game events would not be able to generate this commentary live. A sifting pattern that identifies game losses resulting from blunders could be used to narrativize the blunder as a FATAL ERROR once the game has been completed; and a sifting pattern that identifies blunders would be able to recognize this event as a blunder immediately; but to link the blunder to the *possibility* of game loss before the game has actually concluded requires prospective in addition to retrospective narrative intelligence.

The following Winnow sifting pattern expresses a variant of Ryan's FATAL ERROR theme—modified slightly to work with event data adapted from the *Blaseball* API[3] via the fan-created chronicler tool Datablase[4], which allows for the easy retrieval of historical *Blaseball* simulation output.

```
(pattern fatalError
  (event ?gainLead where
    (leadChange ?gainLead),
```

[3]*Blaseball* is a surrealist baseball simulation idlegame that mixes normal baseball game events with strange occurrences (e.g., players being incinerated by rogue umpires); we use it here as a convenient source of JSON-formatted simulated baseball gameplay data. Due to details of how *Blaseball* simulates baseball gameplay, we have switched the focus of this pattern to be on a batter rather than an outfielder who commits a blunder: *Blaseball* includes a humorous "strike out thinking" blunder that batters can perform, but doesn't simulate outfielders at sufficiently high fidelity to attribute a blunder to one outfielder in particular.

[4]`https://sibr.dev/apis`

```
    (teamHasLead ?gainLead ?team))
  (event ?blunder where
    event_text: ?text,
    (includes? ?text "strikes out thinking"),
    batter_id: ?blunderer,
    batter_team_id: ?team)
  (event ?loseLead where
    (leadChange ?loseLead),
    (not (teamHasLead ?loseLead ?team)))
  (event ?loseGame where
    event_type: GAME_OVER)
  (unless-event ?e between ?gainLead ?blunder
    where (leadChange ?e))
  (unless-event ?e between ?loseLead ?loseGame
    where (leadChange ?e)))
```

This pattern matches a sequence of events in which a team gains the lead over their opponents; commits a serious blunder; subsequently loses the lead; and then loses the game without ever regaining the lead. It employs several advanced Winnow features: two custom Datalog inference rules, `(leadChange ?event)` and `(teamHasLead ?event ?team)`, are used to identify events in which a particular team gained the lead over their opponents; the DataScript built-in function `includes?` is used to determine whether a string value contains a specific substring; and the `not` keyword is used to negate a constraint.

When this pattern is executed incrementally while the game is still going on, a partial match that has advanced as far as the `?loseLead` event can safely be narrated as a *likely* fatal error. Instead of waiting for the entire event sequence to be completed before we can comment on the FATAL ERROR theme, we can comment speculatively on the likelihood that the `?blunderer`'s mistake has cost their team the win.

Additionally, if we note in the event database that we have chosen to speculatively comment on this event as a fatal error, we can make use of this in later commentary to refer back to our own past commentary as mistaken. Suppose the team that committed the blunder ends up winning the game despite our prospective evocation of the FATAL ERROR theme. In this case, the final story of the game as produced by our incremental sifting-based commentary generator can dramatize the mistaken commentary as part of a larger ERROR AND REDEMPTION theme by retrospectively invoking the moment at which it appeared that the game would be lost due to the blunder. By computationalizing the "anticipation of retrospection" that characterizes prospective narrative intelligence [7], we can produce more human-like commentary that more effectively draws out a convincing story from gameplay.

5.4.2 Interactive Incremental Sifting

Several play experiences that center on narrative coauthorship between the player and an AI system have made use of story sifting to help the player comprehend the contents of a rich simulated storyworld. These include Writing Buddy [107], *Cozy Mystery Construction Kit* [45], and *Why Are We Like This?* [47,48]. In these games, the ability to expose partial sifting pattern matches to the player introduces new potential affordances for play.

Consider a sifting pattern like the following, adapted from the `arson-revenge` Sheldon sifting pattern presented by Ryan [99]:

```
(pattern arsonRevenge
  (event ?harm where
    tag: harm,
    actor: ?victim, target: ?arsonist)
  (event ?scheme where
    eventType: hatch-revenge-scheme,
    actor: ?arsonist, target: ?victim,
    (ancestor ?harm ?scheme)),
  (event ?arson where
    eventType: set-fire,
    actor: ?arsonist, target: ?victim,
    (ancestor ?scheme ?arson)))
```

In an interactive sifting context, when a pattern like this is available and a partial match with bindings for the first two events is present, the system can make use of this partial information (that a revenge scheme has been undertaken but not yet completed) in many ways. Beyond simply informing the player that this microstory is *possible*, it can accept suggestions from the player on whether the scheme should actually be carried out or not; suggest character actions that advance the scheme in various ways (e.g., having the `?arsonist` character purchase a can of gasoline); or even suggest alternative ways that the revenge scheme could be carried out, for instance if multiple revenge-oriented patterns that all begin with a similar set of events are simultaneously present.

Additionally, a game of this nature could allow the player to choose for themselves a sifting pattern that they would like to see take place. Once chosen, the player could track the progress of this sifting pattern as new events occur; decide whether or not to accept a particular event as advancing the sifting pattern; and receive early warning from the computer when an event has a chance of disrupting the story that the player is trying to create. Such an interface could enable the computer and player to explicitly collaborate on partially realized stories in exciting new ways.

5.4.3 Retrospective Sifting

It is also possible to use Winnow as an alternative, more verbose but more human-friendly syntax for the specification of retrospective sifting patterns. For instance, Winnow can compile the previously defined `breakHospitality` sifting pattern to a roughly equivalent Felt pattern:

```
(eventSequence ?e1 ?e2 ?e3)
[?e1 eventType enterTown] [?e1 actor ?guest]
[?e2 eventType showHospitality]
[?e2 actor ?host] [?e2 target ?guest]
[?host value communalism]
[?e3 tag harm]
[?e3 actor ?host] [?e3 target ?guest]
(not-join [?e1 ?guest ?e3]
  (eventSequence ?e1 ?eMid ?e3)
  [?eMid eventType leaveTown]
  [?eMid actor ?guest])
```

One minor semantic difference between this compiled Felt pattern and the incrementally executed Winnow pattern lies in how the constraint `[?host value communalism]` (the Felt equivalent of Winnow's `?host.value: communalism`) is applied. Under retrospective execution of a Felt sifting pattern, this constraint is checked at the end of the event sequence, so the validity of the match hinges on whether the host character still holds communalism as a value after the whole sequence has played out. Under incremental execution, however, this constraint is associated specifically with the `?e2` execution stage, so we only check whether the host values communalism at the time of event `?e2`. This allows for incrementally executed patterns to make some subtle distinctions about when constraints hold that are not possible when executing sifting patterns in a purely retrospective mode.

Compilation allows for Winnow sifting patterns to be used as preconditions in Felt action definitions. It also ensures that Winnow's performance in a purely retrospective sifting context is never worse than Felt's. And even "timeless" Felt or Sheldon sifting patterns that are not formulated in terms of event sequences (e.g., a sifting pattern that finds instances of unrequited love between characters) can be expressed in Winnow as single-event patterns. Consequently, Winnow can often be used as a drop-in replacement for the sifting component of Felt.

5.5 Performance

Winnow is written in browser JavaScript, in a coding style that optimizes for clarity over performance. Nevertheless, it is fast enough to be useful for at least some real-world simulation-driven games.

To establish a performance baseline, we created an incremental sifting benchmark task involving a small simulated storyworld with 30 event types and 5 characters. We conducted several distinct runs of the benchmark; on each run, the partial match pool was initialized with a variable number of partial matches against the `breakHospitality` pattern, ranging from 10 up to 1000. One hundred random events (with an event type and tags randomly taken from the 30 available event types, and an actor and target character chosen randomly from the five available characters) were

Pool size	**Min time**	**Max time**	**Avg time**
10	9ms	40ms	13ms
50	41ms	110ms	50ms
100	78ms	227ms	93ms
500	398ms	577ms	460ms
1000	443ms	1097ms	912ms

Table 5.1: Benchmark results for various partial match pool sizes. Minimum, maximum, and average time taken to update the partial match pool per event is given for each iteration of the task.

created and added to the database one by one, and the time it took Winnow to update the partial match pool on each event addition was recorded.

The benchmark was run in Firefox 87.0, on a 2019 MacBook Pro with a 2.6 GHz 6-Core Intel Core i7 processor and 16GB of RAM. Full benchmark results are available in Table 5.1. Notably, on the most difficult version of the benchmark task (in which 1000 partial pattern matches had to be checked per event), Winnow took on average 912ms per event to update the partial match pool. This is within the one-second response window suggested by usability experts as sufficient for maintaining the user's flow of thought in an interactive context [89, Chapter 5]; therefore, Winnow's performance is likely sufficient for the provision of near-immediate feedback on player-initiated game events. Additionally, in many games, events of potential narrative significance occur much less frequently than once per second, so it is likely that Winnow will be able to keep up with a wide variety of gameplay types. (Especially frequent events, such as movement events in action games, are often of little narrative significance and unlikely to even be logged as events in a narrative-focused chronicle of gameplay.)

Low-hanging fruit for further optimization is abundant. In particular, Winnow's use of the DataScript library for Datalog query execution imposes a string parsing overhead on every query that is run; this overhead could be reduced through tighter integration with a Datalog backend. Additionally, since Winnow may frequently find itself evaluating the same Datalog expression many times when checking an event against a large pool of partial matches, some form of expression-oriented evaluation cache may help to avoid redundant computation.

5.6 Discussion

5.6.1 Pool Management Strategies

Winnow makes no attempt to remove partial pattern matches from the pool, except when they are either completed or killed by `unless-event` constraint violations. As a result, additional application-specific pool management strategies may be useful in mitigating unbounded growth of the pool over time. Collectively, these strategies bear some resemblance to the "sifting heuristics" proposed by Ryan [99]: higher-level counterparts to sifting patterns that encode more generic facets of narrative tellabilty.

One easy-to-implement and fairly generic approach to pool management involves the automatic pruning of any partial match that has not accepted any of the last K events for some reasonably large K. The exact threshold to use here is likely dependent on the texture of the simulation with which you are working, and different thresholds may even be appropriate for different sifting patterns within the same application. For instance, a partial match against a "whirlwind romance" pattern can likely be safely pruned after a relatively short period of inactivity, whereas a match against a pattern that encompasses the whole of a character's lifespan might usefully lie dormant for a much longer period.

Another strategy for mitigating pool growth involves replacing a partial match's default "fork and advance" behavior with a simpler "advance directly" behavior (i.e., a behavior that avoids growing the partial match pool) once the match is advanced past a certain point. For instance, a partial match that already has bindings for all of its non-event variables could be advanced directly without forking off duplicates. Since the non-event variables in some patterns (e.g., the identities of the `?host` and `?guest` characters in the `breakHospitality` pattern) seem much more strongly determinant of the microstory's player-perceived identity than the events themselves, a group of partial matches that are technically unique but vary only in event specifics might be perceived by the player as duplicates—a perception that would be mitigated if this strategy was employed.

Finally, narrower application-specific heuristics could be used to clean up partial matches when certain game events take place without requiring these events to be specified as `unless-event` clauses in every relevant sifting pattern. For instance, depending on the simulation domain, many emerging microstories might be invalidated by the premature death of an involved character. Therefore, instead of writing `unless-event` clauses into almost every pattern to hedge against character death, it might be easier from an authoring standpoint (and more performant from a work-minimizing standpoint) to automatically prune any partial matches involving a character that has just died—perhaps excluding matches against a smaller set of patterns that have been specifically marked as tolerant of character death.

5.6.2 Modeling Causality

As Ryan has argued [99], *causal bookkeeping*—the explicit modeling of causality relationships between simulation events—greatly aids the implementation of a curationist approach to emergent narrative. However, many of the simulation engines that are used in notable emergent narrative games today—for instance, Ryan's own Talk of the Town simulation engine [100]—do not perform explicit causal bookkeeping, instead relying on human interactors to infer or invent causality relationships between events. In order to ensure that Winnow is able to reason over the output of a wide range of simulation-driven emergent narrative games, we wanted to avoid imposing a technical requirement that Winnow sifting patterns match only causally connected sequences of events. As a result, Winnow sifting patterns are by default written in a causality-agnostic way.

When working with the output of a simulation engine that performs explicit causal bookkeeping, we expect that causal relationships between events will be available as data within the event entities themselves: for instance, every event entity might contain an explicit pointer to the previous event or events that caused it. Therefore, Winnow sifting patterns can be written to explicitly reason about this causality information when it is present—for instance, to constrain pattern matches so that all of a pattern's kernel events must be causally related—or to ignore causality information when it is either absent or irrelevant to the context in which a particular sifting pattern will be used. We believe that this added flexibility is worth the slight additional authoring burden of having to manually assert causality constraints between events when these constraints are desired.

5.6.3 Decoupling Sifting and Simulation

Unlike our earlier sifter Felt (discussed in chapter 3), Winnow does not offer any built-in support for simulation domain authoring. Instead, it is a standalone sifter. Simulation authors could choose to integrate Winnow-based story sifting into their simulation framework if they desire, but since our work on Felt, we have come to believe that sifters will most often be applied to the output of simulations (or other event-generating processes) over which the sifter's creators do not have direct control. Consequently, we believe it is cleaner to define sifters as separate libraries which simulation creators can import on a case-by-case basis if necessary.

5.6.4 Conjunction and Disjunction

At the language level, Winnow does not provide any specific support for writing sifting patterns that match the conjunction of two or more other patterns. However, conjunction of patterns can straightforwardly be implemented through modification of the event chronicle. When a match (complete or partial) against a particular sifting pattern is first detected, the fact that this match has occurred can be added to the chronicle as a new event. Higher-level sifting patterns can then be written to look for instances of these match events and advance when the appropriate conjunction of lower-level pattern matches has occurred.

Within sifting patterns, disjunction (the ability for a single event clause to match *either* an event A or an event B, where A and B have different characteristics) can be implemented via Datalog inference rules. An inference rule with multiple disjoint bodies will hold if any one set of body conditions holds true; therefore, an event clause can match disjoint events by checking whether the event satisfies an inference rule that contains a disjunction.

5.7 Conclusions and Future Work

We have introduced Winnow, a domain-specific language for incremental story sifting that improves on previous sifting technologies (particularly Felt) by enabling the implementation of *prospective* as well as retrospective narrative intelligence via sifting. Winnow is capable of expressing and incrementally executing a wide variety of realistic sifting patterns, including equivalents to existing Felt and Sheldon patterns and patterns that operate over the *Blaseball* simulation. Additionally, it is performant enough to run in an interactive context and can be used as a more human-friendly language for purely retrospective sifting as well.

At a high level, Winnow can be viewed as a narrative cognition engine that attempts to help the computer understand partial stories that might have arisen in the mind of a human spectator or user. It is from this perspective that story sifting appears to us as an especially exciting approach to narrative intelligence: if we can computationally model the way that humans make sense of emergent stories, we can build systems that are capable of understanding gameplay narratively, just as players do. We believe that future work on sifting should attempt to further explore the implications of this view.

Chapter 6

Loose Ends

Winnow's ability to reason prospectively as well as retrospectively about narrative structure (discussed in chapter 5) makes it possible to build sifting-based AI systems that can recognize and surface incomplete plot threads. Integrating this capability into an approachable narrative instrument, however, represents an additional challenge. In this chapter we present Loose Ends, a narrative instrument and mixed-initiative creative interface (MICI) [24,68] that aims to support a human user's creativity by providing them with an artificially intelligent creative partner. Specifically, by incorporating Winnow-based features into a human-facing user interface for managing plot threads, Loose Ends aims to address the problem of *directionlessness* (with which our earlier narrative instrument *Why Are We Like This?* struggled, as discussed in chapter 4) while preserving the desirable aspects of the *WAWLT* player experience.

In the domain of storytelling-oriented creative writing, most existing MICIs function by providing suggestions as to how a story might be continued, thereby injecting unexpectedness into the writing process [11] and providing an immediate answer to the question of "What happens next?" when the user would otherwise become creatively stuck [53]. These existing MICIs have shown promise in several ways. In particular, MICIs that function by providing short-term story continuations have proven effective at suggesting viable next steps for a story [98]; taking the story in unexpected directions [11, 47, 114]; and creating a sense of shared authorship [106] between the user and system [11, 47, 114].

However, these existing MICIs also exhibit several recurring problems. Most prominently, because the continuations these systems provide take only local context into account, they have a tendency to pull the story in unwanted directions [11, 98, 114] or to otherwise create a sense of long-term directionlessness [47] that inhibits the development of coherent high-level story structure.

To address these problems, we created Loose Ends, a MICI for storytelling that aims to support the development of coherent longer-term story structure. By explicitly reasoning about multiple parallel plot threads and providing a mixed-initiative interface for managing long-term storytelling goals framed in terms of these plot threads, Loose Ends aims to provide suggestions that keep the story on track with respect to the development of character arcs, conflicts, and high-level narrative themes.

The main contributions of this chapter are:

- A co-creative AI system that can reason about threaded plot structure in relation to high-level storytelling goals, proactively suggest new goals based on past plot events, and suggest character actions that advance these goals

- An approachable user interface for interacting with this AI system to create stories
- An evaluation of our approach by five experts in computationally engaged storytelling, indicating that Loose Ends succeeds at mitigating directionlessness while preserving a sense of coauthorship

In addition to these contributions, we also make the current version of Loose Ends available to be played in a web browser[1] and release its codebase as open source[2].

6.1 Related Work

Loose Ends draws inspiration from several past attempts to facilitate playful mixed-initiative storytelling, particularly *Writing Buddy* [107] and *Why Are We Like This?* [47, 48]—the latter of which is described in greater detail in Chapter 4. Both of these systems allow players to specify storytelling goals that guide the direction of the running story by influencing what story continuations the system will suggest. Both systems generate continuation suggestions in the form of structured plot events rather than prose, using a rules-based AI system rather than a language model to generate goal-relevant continuations. And both systems provide a story transcript that captures all past plot events in the form of a story outline, alongside player-written narration elaborating on the basic event descriptions generated by the system.

[1]`https://itsprobablyfine.github.io/LooseEnds`
[2]`https://github.com/ItsProbablyFine/LooseEnds`

Loose Ends follows a similar architecture, although it differs from its predecessors in two key ways. First, its storytelling goals are more sophisticated than those in either predecessor system. Unlike in *Why Are We Like This?*, storytelling goals in Loose Ends specify *sequences* of events that must be added to the story for the goal to be satisfied (rather than individual events alone)—and unlike in *Writing Buddy*, storytelling goals in Loose Ends can be *parametrized* with specific characters and additional constraints. It is important for long-term story structure that the player and AI system be able to collaborate explicitly on the development of parametrized multi-event sequences, since narrative structure often relies heavily on *reincorporation* of previously introduced elements into later parts of the story [125]. Second, the AI in Loose Ends is capable of suggesting new storytelling goals that are consistent with the story up until this point, rather than just steering action suggestions toward player-specified goals as in previous systems. Together, these changes result in a system that feels like an active writing partner while also guiding player-authored stories toward coherent longer-term structure.

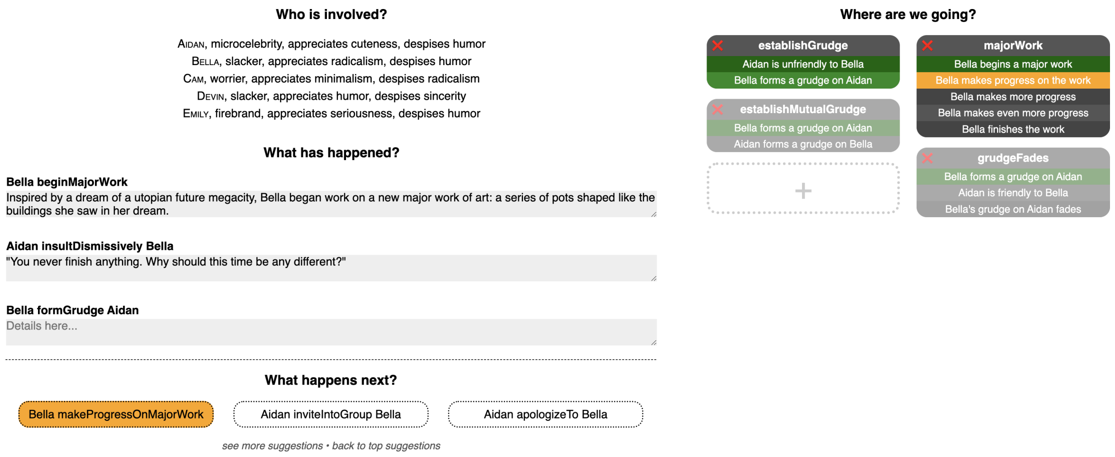
Beyond plot event-based systems such as *Writing Buddy* and *Why Are We Like This?*, a number of attempts have also been made to facilitate mixed-initative storytelling by providing continuation suggestions in the form of unstructured prose. Early examples of this approach can be found in the Say Anything [121] and Creative Help [98] systems, which use case-based reasoning to find sentences similar to the user's most recently typed sentence in a large database of preauthored stories, then suggest these sentences as continuations. More recently, textual continuations provided by language models have been used to support storytelling in a relatively unmediated way [11, 72]. Singh et al. [114] finetune a large language model on a storytelling-relevant dataset and extend its completion suggestions to include images and sound as well as text, then evaluate this approach at scale. In each of these cases, purely text-based completions have been found to be pleasantly surprising and often relevant to the immediately previous parts of the story being told, but divergent from user-intended story structure in ways that require frequent revision by the user to maintain long-term direction.

One recent mixed-initiative storytelling support system that departs from the interaction paradigm of local continuation suggestion is TaleBrush [17], which instead aims to give users direct control of high-level story structure via the sketching of a visual fortune arc for the story's main character. This approach has so far only been used to generate very short stories (on the order of five sentences long), and the coherence of the generated stories is limited, but this potential alternative means of specifying high-level storytelling goals still merits mention here.

6.2 System Description

Loose Ends (Figure 6.1) is a mixed-initiative creative interface for playful storytelling. We specifically conceive of Loose Ends as an AI-based *narrative instrument* [56]: a system that can be played to produce narrative, in much the same way that a musical instrument can be played to produce music.

In the Loose Ends interaction loop, a human player repeatedly selects *action*


Who is involved?

Aidan, microcelebrity, appreciates cuteness, despises humor
Bella, slacker, appreciates radicalism, despises humor
Cam, worrier, appreciates minimalism, despises radicalism
Devin, slacker, appreciates humor, despises sincerity
Emily, firebrand, appreciates seriousness, despises humor

What has happened?

Bella beginMajorWork
Inspired by a dream of a utopian future megacity, Bella began work on a new major work of art: a series of pots shaped like the buildings she saw in her dream.

Aidan insultDismissively Bella
"You never finish anything. Why should this time be any different?"

Bella formGrudge Aidan
Details here...

What happens next?

Bella makeProgressOnMajorWork
Aidan inviteIntoGroup Bella
Aidan apologizeTo Bella
see more suggestions • back to top suggestions

Where are we going?

establishGrudge
Aidan is unfriendly to Bella
Bella forms a grudge on Aidan

establishMutualGrudge
Bella forms a grudge on Aidan
Aidan forms a grudge on Bella

majorWork
Bella begins a major work
Bella makes progress on the work
Bella makes more progress
Bella makes even more progress
Bella finishes the work

grudgeFades
Bella forms a grudge on Aidan
Aidan is friendly to Bella
Bella's grudge on Aidan fades

Figure 6.1: The Loose Ends user interface. The **Who is involved?** section displays basic information about a generated cast of five characters. The **What has happened?** section lists plot events that have taken place in the story so far, along with player-written text giving more details about these events. The **What happens next?** section shows AI-generated suggestions for what might happen next in the story. The **Where are we going?** section shows active storytelling goals, including transparent goals that have been suggested by the AI system rather than added by the player. One action suggestion (highlighted in orange in the bottom left) is being hovered over by the player; consequently, the impact this suggestion would have on the active storytelling goals if accepted (i.e, advancement of the `majorWork` goal) is also highlighted in orange on the right. Unlike *WAWLT*, Loose Ends does not offer a storyworld investigator interface. The reasons for this are discussed in more detail in chapter 4, but broadly speaking, Loose Ends maintains considerably less simulation state than *WAWLT* (making the investigator less useful), and the investigator was used less than we anticipated even in *WAWLT*, so removing it is a natural way to streamline the Loose Ends UI. Nevertheless, the most frequently used and relevant parts of the investigator's characters tab have been folded into the **Who is involved?** section (a static display of the characters in the storyworld and their individual traits), while the investigator's situations tab has been folded into the **Where are we going?** section (where system-detected situations will appear as partially satisfied storytelling goal suggestions).

suggestions furnished by the underlying AI system to continue the plot of a running story, using *storytelling goals* to steer the narrative toward player-desired long-term outcomes. Actions selected by the player are added to a running *story transcript*, and each action can be annotated with additional text by the player—for instance to narrate the action in greater detail.

Although Loose Ends as a system aims to be storyworld-agnostic, the version of Loose Ends presented here contains actions and storytelling goals that are specifically relevant to constructing stories about the development of character relationships and careers within a small community of artists. In the future, we envision that many different "playsets" for Loose Ends might be created, supporting the construction of stories set in many different kinds of storyworlds.

The AI system that powers Loose Ends consists of two major components. First is a storytelling goals tracker that updates a pool of active and possible storytelling goals as new plot events are added. Second is an action suggestion generator that generates and ranks potential suggestions for the next plot event in the story based on the currently active storytelling goals.

6.2.1 Storytelling Goals Tracker

Storytelling goals in Loose Ends are used to set and maintain the high-level direction of the story. Every goal is an instance of a *goal template*: a story sifting pattern written in the domain-specific logic programming language Winnow [46]. An example goal template is given in Appendix A.

A goal template describes a sequence of interrelated events that can be interpreted as satisfying a particular storytelling purpose or instantiating a particular kind of plot thread. For instance, the current version of Loose Ends includes templates for goals that introduce or develop character relationships (e.g., friendship or rivalry); internal conflicts (e.g., artistic or career struggles); and high-level narrative themes (e.g., moral themes related to the virtues of persistence in the face of adversity). There were 12 goal templates total in the version of Loose Ends evaluated here.

A *goal* is a partial match against a goal template, representing a sequence of past plot events that partially meet the goal template's requirements. To *advance* a goal is to locate and accept an action suggestion that continues the sequence of events that match the underlying goal template. For instance, if a `majorWork` goal involving the character Aidan has been advanced past the first event (in which the goal's main protagonist character begins work on a major art project) and a second event in which Aidan makes progress on the project is added to the story, this goal will be advanced another step.

Goals can also be *cut off* if an event that violates one of the goal's constraints is added to the story. For instance, if an `onARoll` goal involving the character Bella is active, but another character completes a major artwork before Bella manages to complete two major artworks in a row, this goal will be cut off, since a condition of the `onARoll` goal has now been violated.

Goals are *parametrized* by the characters that are involved in them, and multiple goals that are based on the same underlying goal template can be active concurrently as long as they pertain to a different configuration of characters. For instance, two `formGrudge` goals can be simultaneously active if either the character that holds the grudge, the target of the grudge, or both are different between the two goals. Additionally, if the player knows that they want a certain type of plot thread to be present in the story but does not know which characters they want to be involved, they can add a storytelling goal of the relevant type without any character parameters specified and allow the system to suggest possible ways of casting the available characters into this thread.

The Loose Ends user interface permits players to add goals manually (by selecting a goal template to instantiate as a goal, from a library of all available goal templates) and to remove goals that have already been established at any time. In addition, the AI system in Loose Ends constantly tracks and evaluates a pool of partial matches that the player has not established as goals. If one of these partial matches advances beyond a certain threshold (33% completion in the current version of Loose Ends), the system will automatically promote it to an active goal, rendered in a transparent style to indicate that this is a system-suggested goal rather than a player-added one. These goals can be removed by the player like any other (enabling the player to veto the system's suggestions of additional storytelling goals), or the player can click on them to remove the transparency effect and notionally "lock them in" as player-intended goals.

6.2.2 Action Suggestion Generator

Action suggestions in Loose Ends are drawn from two pools of actions. The *basic actions pool* contains actions that are possible for any character at any time, regardless of social state, and remains fixed at all times. The *dynamic actions pool* is recalculated whenever a new event is added to the story, and contains actions that are only possible because of active storytelling goals that are in an appropriate state. For instance, when a complete `establishGrudge` goal between the characters Cam and Devin is active, the dynamic actions pool will contain actions that Cam can only take toward Devin because of their active grudge on Devin (such as sabotaging Devin's most recent artwork). There were 32 action types total in the version of Loose Ends evaluated here: 20 basic actions and 12 dynamic actions.

Actions in general may be either *solo* (involving only a single character, the *actor* who takes the action) or *dyadic* (involving two characters, the actor who takes the action and the *target* toward whom the action is directed). Creating a minor artwork, for instance, is a solo action, while insulting another character is a dyadic action. In addition, every action has an *event type* uniquely identifying the type of action that was performed and a list of zero or more *tags* that assign the action to high-level categories (such as `release` for actions in which the actor finishes and releases an artwork, `friendly` for actions in which the actor is friendly toward the target, and `harms` for actions that harm the target).

Action suggestions are recalculated every time the set of active storytelling goals changes. When calculating action suggestions, the action suggestion generator first iterates over all possible next actions (in both the basic and dynamic action pools) and determines, for each action, which storytelling goals would be impacted (either advanced or cut off) by the addition of this action to the story. Each action is then given a priority score, which is the sum of three factors:

- The number of active storytelling goals that this action would advance
- A constant factor (0.5) if this action is from the dynamic actions pool—i.e., if it is only possible because of an active storytelling goal
- A random factor (between 0 and 0.5) to randomly permute the priority of actions with otherwise equal scores

Actions are sorted by their score and displayed in order, with the three highest-scoring actions being pulled to the top of the action suggestions list. In this way, actions that relate most strongly to the active storytelling goals are prioritized for display, with randomness ensuring a degree of alternation between suggestions that advance parallel plot threads. When the user hovers over an action suggestion to consider it, the precalculated information about which storytelling goals this action would advance or cut off is used to display the ramifications of accepting this action in the storytelling goals pane on the right side of the user interface.

Unlike in *WAWLT*, characters in Loose Ends do not perform autonomous actions in the background, and most actions are not gated by any preconditions. Autonomous actions in *WAWLT* were an early attempt at giving the AI system more capacity to take the story in unexpected directions, but did less than we hoped to address the problem of directionlessness, and have effectively been replaced in Loose Ends by system-suggested storytelling goals. Meanwhile, in Loose Ends, the more sophisticated storytelling goals are generally sufficient to pull the most relevant action suggestions to the front of the suggestions list without any explicit causal modeling, enabling the removal of explicit preconditions on most action types (which we found both time-consuming and difficult to author in our work on *WAWLT*). The main downside of this shift is that Loose Ends sometimes presents players with apparently inconsistent action suggestions when they explore beyond the first page or two of suggestions, which stood out to some players as jarring at times but did not result in a substantial degradation of player experience overall. Overall, we believe that the ideal level of causal modeling in a Loose Ends-like narrative instrument probably falls somewhere between the high level present in *WAWLT* and the low level present in Loose Ends. Both of these differences between *WAWLT* and Loose Ends are discussed in more detail in chapter 4.

6.3 Interaction Examples

In conjunction, the Loose Ends AI and user interface permit several desirable interactions that are not possible in other mixed-initiative creative interfaces for storytelling. Four especially interesting examples of novel mixed-initiative interactions enabled by Loose Ends (all of which took place organically during evaluation) are presented below.

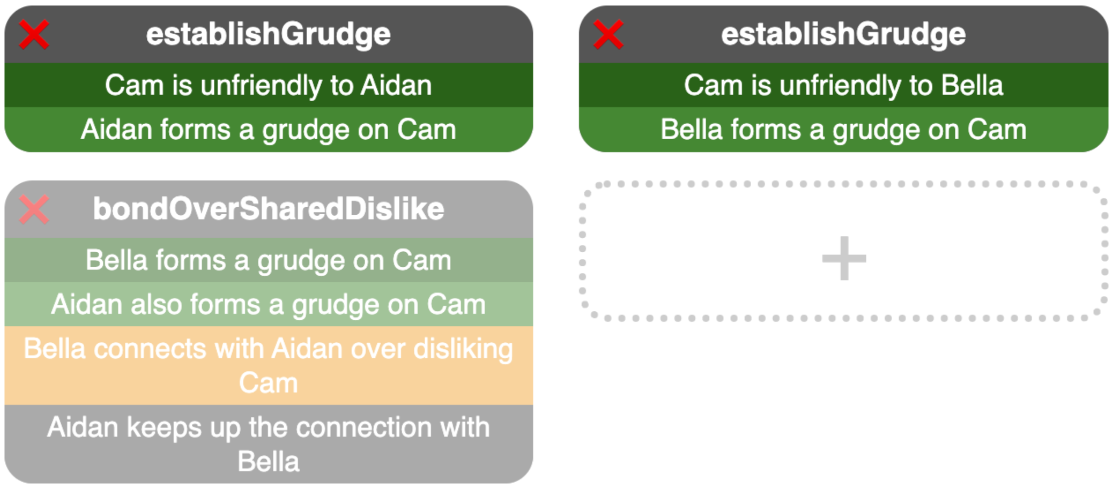

establishGrudge
Cam is unfriendly to Aidan
Aidan forms a grudge on Cam
establishGrudge
Cam is unfriendly to Bella
Bella forms a grudge on Cam
bondOverSharedDislike
Bella forms a grudge on Cam
Aidan also forms a grudge on Cam
Bella connects with Aidan over disliking Cam
Aidan keeps up the connection with Bella
+

Figure 6.2: Based on events that were added to the story to complete two `establishGrudge` goals, Loose Ends has automatically discovered and surfaced a suggestion for another author goal (the `bondOverSharedDislike` goal) to spin off a new plot thread initiated by these events.

6.3.1 Discovering New Storytelling Goals

Beyond simply suggesting action-level continuations to a running story in accordance with player-provided storytelling goals, Loose Ends can also infer new storytelling goals that are consistent with the story so far and proactively suggest these goals to the player. This often results in interactions where a player who would otherwise become uncertain of what to do next is inspired by, and begins pursuing, a system-discovered storytelling goal instead.

For instance, in Figure 6.2, the player has just completed two `establishGrudge` goals targeting the same character (Cam) have both been completed. At this point, Loose Ends automatically discovers and surfaces a successive character relationship development goal, in which Aidan and Bella (who both have grudges on Cam) bond over their shared dislike. The first two steps of this goal are already complete, because the

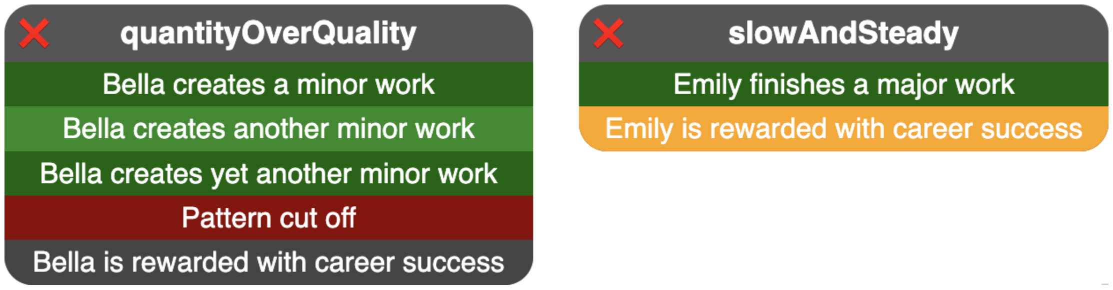

quantityOverQuality
Bella creates a minor work
Bella creates another minor work
Bella creates yet another minor work
Pattern cut off
Bella is rewarded with career success
slowAndSteady
Emily finishes a major work
Emily is rewarded with career success

Figure 6.3: As the player considers an action that would advance one of their thematic goals but undermine another, the impact of the action on both thematic goals is highlighted, making the conflict apparent.

system has been tracking the possibility of surfacing this goal in the background, but it has only just now progressed far enough to be displayed.

6.3.2 Discovering Thematic Conflicts

Loose Ends can make it apparent when a conflict has arisen between two active storytelling goals. For instance, in Figure 6.3, the player is simultaneously working toward two distinct thematic goals for the story and considering an action that will reward Emily with career success after she completes a major artwork. This would support the theme that persistent work on a single major project leads to success (`slowAndSteady`) but undermine the competing theme that the way to success is to create a rapid succession of more minor artworks (`quantityOverQuality`). When the impact of the considered action on all active author goals is visualized, the conflict between these goals is revealed to the player.

6.3.3 Resurfacing Dormant Plot Threads

Because Loose Ends can maintain a larger set of active storytelling goals than the player can hold in their head all at once, action suggestions can serve to remind players of incomplete plot threads that they would otherwise forget to revisit. For instance, long-term storytelling goals like the `tryTryAgain` thematic goal (which requires a single character to repeatedly release artworks that are poorly received, before finally releasing one that is well-received) may temporarily fade into the background as the player focuses on another subplot that weaves together a few distinct storytelling goals at once—but once this more pressing subplot is complete, actions advancing the earlier thematic goal will again rise to the top of the action suggestions pool, reminding the player to return to the previously initiated thread.

6.3.4 Interleaving Parallel Plot Threads

When multiple parallel plot threads are active and none of these threads has storytelling priority, the slight random permutation of equally ranked action suggestions means that Loose Ends by default tends to promote actions that alternately advance different threads. This can help players escape fixation [34], in which they develop a narrow and premature focus on one plot thread or set of characters and forget about the possibility of developing others.

6.4 Evaluation Procedure

To gauge the effectiveness of Loose Ends as a mixed-initiative storytelling interface, we conducted an expert evaluation. This evaluation was modeled on the evaluation of Germinate [49], an earlier mixed-initiative co-creative system.

We recruited five expert evaluators, all of whom are experienced creative writers and researcher-practitioners in intelligent narrative technologies. Four of these evaluators hold a PhD in a relevant area, while one holds multiple relevant graduate degrees and is currently enrolled in a relevant PhD program. All evaluators had past experience with mixed-initiative storytelling in general, and none had encountered Loose Ends before. Because our evaluators were familiar with the state of the art in mixed-initiative storytelling, they were readily able to compare Loose Ends to similar systems and judge what it does well or poorly in comparison.

Each evaluator participated in a single remote play session via Zoom. Each session was approximately one hour long and began with a brief (approximately 5-minute) introduction to the Loose Ends interface by one of the researchers. Subsequently, the evaluator constructed a single story using the Loose Ends interface while thinking aloud and sharing their screen. Once the story was complete, one of the researchers asked several open-ended interview questions to prompt reflection on play patterns they observed during the session. Both the think-out-loud and interview portions of the playtest sessions were recorded for later analysis. Finally, evaluators were administered a brief user experience questionnaire [67] consisting of the following questions:

Q1. What is your overall impression of the system?

Q2. How easy was it to use the system?

Q3. Were you able to use it without unnecessary effort?

Q4. Did you feel a sense of control over the story?

Q5. Was the system fun to use?

Q6. Did you feel a sense of ownership of the story?

Q7. Were you curious to see what would happen next in the story?

Q8. Did you generally know what direction you wanted the story to go next?

Q1 was open-ended and qualitative, while Q2-Q8 were quantitative, with responses ranging from 1-5 (where 5 indicates the highest level of agreement with the premise of the question). Q1-Q5 were adapted directly from the Germinate expert evaluation questionnaire [49], while Q6-Q8 were intended to elicit reflection on aspects of the co-creative storytelling experience that were frequently mentioned by playtesters of *Why Are We Like This?* [47]. A summary of evaluator responses to the quantitative questions is given in Table 6.1.

Question	E1	E2	E3	E4	E5	Avg
Q2. Usability	4	4	4	5	4	4.2
Q3. Effortlessness	4	5	5	5	5	4.8
Q4. Control	4	4	4	4	3	3.8
Q5. Fun	4	5	4	4	4	4.2
Q6. Ownership	3	4	3	3	3	3.2
Q7. Curiosity	4	5	4	3	4	4.0
Q8. Direction	4	5	4	4	3	4.0

Table 6.1: Summary of evaluators' responses to quantitative survey questions. All responses were given on a numeric scale from 1-5, where 5 is highest agreement.

6.5 Evaluation Results

6.5.1 Directionlessness Is Mitigated

Our central design goal for Loose Ends was to mitigate the sense of high-level directionlessness reported by players during playtesting of *Why Are We Like This?* [47] and assist in the development of stories that contain satisfying high-level structure. Both quantitative and qualitative evaluation responses suggest that Loose Ends successfully supports the development and maintenance of high-level narrative direction from the player's perspective.

Quantitative survey responses related to sense of storytelling direction (Q8) ranged from 3-5, indicating that all evaluators had a sense of where they wanted the story next to go at a majority of points during the storytelling process. Additionally, all but one evaluator (E5) reported a score of 4 or higher in this category.

Qualitative think-out-loud remarks and interview responses are consistent with these quantitative results. In particular, two evaluators (E4 and E5) remarked unprompted on how they never experienced writer's block or a sense of being stuck during the play process. Additionally, no evaluators explicitly reported a sense of aimlessness or insufficient medium-term direction at any point during their playthrough, in stark contrast to the prevalence of these comments during playtesting of *Why Are We Like This?*

6.5.2 Coauthorship Is Preserved

One open question for Loose Ends was whether the AI system could successfully preserve the sense of shared authorship that players experience in *Why Are We Like This?* while intervening more proactively in the storytelling process—including through the suggestion of new high-level storytelling goals. Both quantitative and qualitative evaluation responses suggest that Loose Ends succeeds in this regard.

Quantitative responses regarding sense of control over the story (Q4), sense of ownership of the story (Q6), and sense of curiosity regarding what would happen next in the story (Q7) are especially salient here. For control, all evaluators reported a score of at least 3 (indicating a moderate sense of control), and all but one (E5) reported a score of 4 (indicating a strong, but not complete, sense of control). For ownership, all evaluators reported a score of at least 3 (indicating a moderate sense of ownership), and one (E2) reported a score of 4 (indicating a strong, but not complete, sense of ownership). For curiosity, scores were distributed across the 3-5 range, indicating that all evaluators felt at least moderate curiosity, while all but one (E4) experienced either strong or very strong curiosity regarding the story's next direction. Taken together, these scores suggest that evaluators generally remained in control of the story while working with the system, but that they also created stories containing unexpected twists that they would be unlikely to invent if writing alone—to the extent that the AI system seemed to hold partial ownership of the stories that emerged.

Qualitative think-out-loud remarks and interview responses further support this interpretation. One evaluator (E5) felt that the play process reflected "a nice meeting in the middle" between player-led and system-led storytelling; another (E1) remarked that it "feels like the sweet spot for co-creativity"; and a third (E2) felt it to be a "good collaboration": "kind of the dream" for mixed-initiative co-creativity.

6.5.3 Goal Alignment Is Unexpected and Fun

Four evaluators (E2-E5) remarked unprompted on how much they enjoyed it when the system correctly anticipated where they wanted the story to go next and offered options (especially storytelling goal suggestions) for continuing the story in a relevant direction. One evaluator (E2) was particularly pleasantly surprised by how often this took place during play. This suggests that the feeling of being seen or understood by the system can be a significant source of enjoyment during mixed-initiative storytelling, perhaps related to the aesthetic of responsiveness as described by Mason [76].

6.5.4 Evaluators Found Loose Ends Easy to Use

Loose Ends was rated highly by evaluators on usability and (especially) lack of unnecessary effort involved in use, suggesting that it is considered highly usable in comparison to similar systems with which these evaluators were familiar. All evaluators reported a score of at least 4 for both Q2 (usability) and Q3 (effortlessness), and all but one evaluator (E1) reported a score of 5 for effortlessness, indicating unanimous agreement that Loose Ends is easy to use.

One caveat to this finding is that our evaluators, as experts in computationally engaged storytelling, were already familiar with several similar systems and used to putting up with unpolished interfaces. Consequently, this finding might not generalize well to other player populations.

6.5.5 Some Players Want Prose-Level Suggestions

Evaluators used the freely editable text boxes in the story transcript in very different ways. Two evaluators (E1 and E4) mostly used them to write extended narration of high-level plot events, as we originally envisioned. One (E2) ignored the text boxes almost entirely. One (E3) used the text boxes to write short notes-to-self about why they chose certain actions from a storytelling perspective—a use-case we did not envision. And one (E5) initially used the text boxes to add terse narrative details for later expansion into full narration, but then stopped using them partway through play.

In qualitative think-out-loud remarks and interview responses, two evaluators (E1 and E2) both indicated that they wanted assistance in coming up with potential details for how certain high-level actions could have been narrated. E2 in particular (who made almost no use of the text boxes) stated that they would have found this additional narration-level support especially helpful.

Altogether, under the cognitive process model of writing [29, 35], we find that

Loose Ends currently provides assistance mostly at the planning stage, specifically in the creation of plot outlines. Expansion of support to later stages of the writing process represents a potential direction for future work.

6.5.6 Storyworld Inconsistencies Stand Out

The current version of Loose Ends makes use of a stateless, naïvely random action suggestion generator rather than a full-fledged social simulation to generate candidate action suggestions. Character relationship state is not tracked anywhere besides in storytelling goals related to friendship and rivalry, and most action types can be suggested between any pair of characters regardless of these characters' current relationship state. This leads to occasional generation of action suggestions that seem nonsensical from the perspective of a player who is tracking character relationship state mentally.

Three evaluators (E3-E5) commented at least once on this perceived occasional lack of consistency as a detriment to the overall storytelling experience in Loose Ends. This finding underscores the importance of storyworld consistency maintenance features for storytelling support—as suggested by several past studies, including Kreminski et al. [57] and Calderwood et al. [11]. In the future, we intend to extend Loose Ends to use a more sophisticated suggestion generation mechanism that tracks substantially more character relationship state, hopefully alleviating this problem.

6.5.7 Common Feature Requests

Three evaluators (E1, E3, and E4) mentioned wanting to filter action suggestions to only display actions with particular characteristics, such as those of a particular event type or those involving particular characters. Three evaluators (E1, E3, and E4) mentioned a desire to express a temporary focus on a specific storytelling goal, so that the system would prioritize action suggestions that would advance this goal. Four evaluators (E1-E3 and E5) expressed a desire to minimize complete storytelling goals without removing them, in order to free up more screenspace for incomplete goals. And finally, four evaluators (E1 and E3-E5) stated that they wanted more detailed information about a particular character's traits or relationships to be immediately available while considering a suggested action involving that character. Going forward, we plan to add all of these features to Loose Ends in some form.

6.6 Conclusions and Future Work

Preliminary evaluation of Loose Ends, a novel mixed-initiative creative interface for storytelling, suggests that it preserves the desirable sense of coauthorship present in earlier systems while mitigating player-perceived narrative directionlessness. We hope that the formalization of jointly human- and machine-understandable storytelling goals presented here, and the idea of a mixed-initiative storytelling partner that can explicitly reason about and suggest high-level plot directions for a story (in addition to immediate continuations), will be taken up and further developed in the next generation of MICIs for storytelling support.

Chapter 7

Conclusion

Recall our three research questions, introduced in Chapter 1:

- **RQ1.** In what ways do existing systems that are used as narrative instruments succeed and fail at providing their users with creativity support?
- **RQ2.** What new technical capabilities would we need to develop to address the deficiencies of existing narrative instruments?
- **RQ3.** What new human-facing interfaces would we need to construct to integrate these new technical capabilities into playful computationally supported storytelling practices?

This dissertation proposes answers to all three of these questions. In Chapters 1 and 2, I introduced narrative instruments as a concept, described two high-level categories of existing systems (interactive emergent narrative games and AI-supported creative writing tools) that have been appropriated as narrative instruments, and discussed two key flaws of these existing systems (overwhelm and directionlessness) that guided and structured the remainder of my dissertation work. In Chapters 3 and 5, I presented two story sifters that enable new technical capabilities which can be used to address the issues of overwhelm and directionless respectively. And in Chapters 4 and 6, I presented two new narrative instruments that incorporate these technical capabilities, alongside preliminary evaluation of the player experience facilitated by these systems. Altogether, this dissertation suggests that the problems of overwhelm and structurelessness in narrative instruments can be mitigated through a more expressive approach to story sifting (based on logic programming) that permits partial and incremental sifting, and which enables the development of user interfaces for the explicit coordination of shared storytelling goals between players and AI systems.

That said, much work remains to be done before mixed-initiative co-creative narrative instruments can achieve their full potential. In the following section, I briefly discuss several potential high-level directions for future work based on the work presented here.

7.1 Looking Forward

7.1.1 Integrating Disparate Symbolic Models of Storytelling

The formulation of storytelling goals we adopted in Loose Ends unifies the affordances of several earlier techniques for guiding action in interactive storytelling. In particular, because Loose Ends storytelling goals can both recognize actions as advancing the state of certain microstories and provide new actions based on their internal state, they function both as story sifting patterns and as something akin to Versu's *constitutive social practices* [28], which provide characters with affordances only when certain social states are active. (For instance, a Loose Ends storytelling goal in which two characters become rivals can unlock rivalry-specific actions between those two characters only once the goal has progressed past a certain degree of completion.)

In the future, I believe that it may also be possible and useful to unify this computational model of narrative with others—in particular with planning-based approaches to guiding character action, since it would not be especially difficult to search for actions that are likely to advance storytelling goals in desirable ways beyond the immediate next timestep. The templated action sequences defined by Winnow sifting patterns, when used to guide action toward the advancement of these sequences (as in Loose Ends), already essentially function as something similar to goals in planning, albeit with only a single step of lookahead applied. Similarly, the `unless-event` constraints that define when a partial match against a Winnow sifting pattern is no longer viable somewhat resemble the open-ended and expressive *integrity constraints* that define invalid action sequences in answer set programming approaches to narrative generation [23, 116]. By introducing more fully realized forms of planning and integrity constraint specification to our unified model of computational story structure, it might be possible to make simultaneous use of affordances from numerous different story generation approaches, including "bottom-up" and "top-down" story generation techniques that are commonly seen as opposed.

7.1.2 Developing Sifting Heuristics

Although story sifting to date has made extensive use of story sifting *patterns*—low-level specifications of event sequences that tend to make for interesting narrative material, which are matched directly against a chronicle of past storyworld events—there has been little work so far on the development of what Ryan terms story sifting *heuristics*, or higher-level abstract encodings of storyfulness that might help to guide sifting [99, p. 250]. Once a wide variety of sifting patterns targeting a particular simulation have been defined, the challenge of sifting shifts toward one of determining which of the many available pattern matches is highest-priority or most likely to function as compelling narrative material—and it is here that heuristics seem especially useful. Though it has previously been suggested that statistical improbability might serve as the basis for a good heuristic [99, p. 250], and that cognitive models of narrative perception such as the Indexter model [13] might also play a useful role in defining sifting heuristics [50], these ideas have yet to be implemented concretely in the context of a storyworld simulation as far as I am aware. At the time of this writing, I am actively working to develop a sifting heuristic that operationalizes the notion of statistical improbability among sifting pattern matches, but this work is still in its early stages, and many alternative sifting heuristics remain to be discovered and implemented.

7.1.3 Neurosymbolic Approaches to Storytelling Support

Both symbolic and neural approaches to story generation have strengths and weaknesses, especially when viewed in terms of the forms of support that they can readily provide to human storytellers. In particular, symbolic approaches enable the definition and enforcement of goals for high-level story structure, while neural approaches are capable of generating open-ended story continuations without a human author needing to manually define all the potential kinds of events that can possibly occur within a storyworld. Therefore, I have argued in the past that it would be beneficial to construct systems that are capable of combining neural and symbolic approaches to story generation in order to provide users with a wider range of support [53]. This strikes me as one of the leading priorities for future work in mixed-initiative co-creative storytelling, and I am currently pursuing this direction myself as I continue past the scope of my dissertation work.

Some work has already been done on hybrid (neurosymbolic) approaches to fully autonomous story generation, particularly by Lara Martin [75] and other current and former members of Mark Riedl's lab. However, this work has yet to be extended to the context of creativity support tools for storytelling. Many potential integrations of neural and symbolic approaches to narrative intelligence might prove useful for storytelling support, but the one that I am currently focused on is the integration of a large language model into a Loose Ends-style mixed-initiative creative interface (with explicitly, symbolically specified storytelling goals) in place of a human-authored storyworld simulation as a generator of action suggestions. Although this approach requires parsing and interpretation of open-ended text completions in order to make sense of what these completions imply from the perspective of a symbolic world model, this is a challenge that Martin's prior work has already begun to attack, and success in this approach would make it substantially easier to generate highly open-ended suggestions for story continuation that nevertheless advance (and are guided by) symbolic models of story structure.

Another, parallel line of work in neurosymbolic approaches to storytelling support might involve attempting to learn symbolic representations of story structure from open-domain corpuses of story. Elson's work on story intention graphs [27], for instance, included an attempt to manually extract generalizable story structure patterns from written fables. Many of these patterns could readily be encoded as sifting patterns in a logic language such as Felt or Winnow and used to conduct story sifting over the output of an appropriate simulation engine, and it is similarly not outside the realm of feasibility that these patterns could also be learned from written stories if appropriate natural language processing techniques are applied.

7.1.4 Learning from Players

One line of research that I have intended to pursue for several years, but that ended up falling outside the scope of this dissertation, involves the ethnographic study of players who construct retellings and the process of retelling construction around existing interactive emergent narrative games. This line of research would make use of methods such as interviews and observational studies to gather data on how players approach retelling construction, with the ultimate goal of using this information to construct a grounded theory [18] of retelling construction beyond the basic theory of extrapolative narrativization [57] that I have already advanced.

In the context of these user studies, it might also be interesting to capture and examine detailed interaction trace data, for instance by logging all of the user interface actions that users perform and analyzing this log in parallel to think-out-loud and other forms of observational data. This could enable the discovery of interaction patterns that are representative of distinct player types, perhaps analogous to the "curious user" type reported by Nelson et al. [88] in their study of a casual creator for game design.

7.1.5 Evaluation

As discussed in Chapter 1, creativity is seen as a grand challenge in computer science for several reasons—one of which is the difficulty of its evaluation. This difficulty is no less present in mixed-initiative co-creative storytelling than in other areas of creativity-oriented computer science research, and essentially every one of the projects presented in this dissertation has proven challenging to evaluate in some sense. For the time being, I chose to evaluate the new narrative instruments I designed (*Why Are We Like This?* and Loose Ends) primarily by means of small-scale think-out-loud playtesting [42]—a formative and human-centered approach to evaluation that aims to qualitatively characterize the player experience of these particular narrative instruments, without comparing them directly to alternatives (since few if any directly comparable alternatives exist) and with the goal of generating context-sensitive design insights that can be applied to future projects. Though this is a typical form of evaluation for research through design [33,133], it also limits the extent to which we can speak confidently about the effects of specific, narrowly delimited design decisions on players in general. In the future, it may be beneficial to test the effects of specific design decisions with larger user populations and to employ psychometrically validated survey instruments such as the Creativity Support Index [15] to develop a firmer sense of whether, when, and why the different features of systems like *WAWLT* and Loose Ends are useful for mixed-initiative storytelling.

When developing mixed-initiative creative interfaces, the user's or player's subjective experience of the creative process is only one of several things that might be worth evaluating. Beyond creative experience, it also makes a good deal of intuitive sense to evaluate the *outputs* of the creative process, particularly the artifacts that people create. In interactive storytelling, the field of retelling studies specifically asks researchers to consider the artifacts that people create while using interactive narrative systems (i.e., retellings) as part of the process by which systems are evaluated. I broadly agree with this call and have attempted to articulate processes by which the study of retellings can be combined with the study of player experience through more conventional self-report and interview methods in the past [57], but the details of *how* retellings should be evaluated—and how they should be compared and contrasted with measures of player experience—remain largely undefined to date.

One potentially promising approach to the analysis of retellings involves treating them as the outputs of a procedural content generator (the narrative instrument with which they were generated, potentially also including the player as a part of the overall story-generating system) and conducting an *expressive range analysis* (ERA) [118, 120] to characterize the complete space of retellings that a particular instrument is capable of producing. Tabletop role-playing games, which are frequently used as narrative instruments, have already been characterized as procedural content generators amenable to expressive range analysis in the PCG literature, so there is some degree of precedent for this proposal [39]. Additionally, some prior work has been done on the application of ERA-like analyses to emergent narrative games [64], though this work has treated sets of character goal state values (rather than complete story structures) as the output artifacts to be analyzed via an ERA-inspired approach.

Since expressive range analysis generally involves the generation of very large numbers of artifacts, it would be substantially easier to apply ERA to narrative instruments with which players have already created a wide variety of published retellings. Alternatively, an approach such as expressive range coverage analysis [52] might be used to contrast a large number of autonomously generated stories (the narrative instrument's innate expressive range) with a small number of co-created stories in order to gauge how the narrative instrument as a mixed-initiative creative interface leads players to favor or disfavor certain kinds of stories within the overall expressive range. This approach would require the narrative instrument to be capable of generating stories both fully autonomously and in a mixed-initiative mode, but the former is not guaranteed to be a capability of all narrative instruments; developing a random machine player (or a machine player that somehow imitates human players in a more sophisticated fashion) might enable the application of this evaluation method to narrative instruments that lack an innate autonomous mode. Regardless of the details, any application of expressive range analysis to narrative structures would also necessitate the development of new computationally defined metrics for story, since ERA as an approach relies on the availability of a suite of metrics that are appropriate for characterizing the artifacts under study (and ERA has not previously been applied directly to story structure, as far as I am aware).

7.2 Final Thoughts

Broadly speaking, the work presented in this dissertation can be taken as an argument for an approach to creativity research that explicitly views creation as a form of play. The idea of "narrative instruments" as playable interactive narrative systems that differ from both tools and games stems directly from a playcentric view of creativity, as does the idea of examining interactive emergent narrative gameplay from a creativity support perspective in the first place. It seems likely to me that a great deal of equally fruitful research can be done on the basis of this premise, including in expressive domains ranging well beyond narrative. Consequently, my greatest hope for this dissertation is that it will help to establish a distinct tradition of explicitly playful creativity support research, building on foundations introduced by other research traditions (especially that of games research) but not allowing itself to be constrained by these traditions' boundaries. I close with the following assertion, my highest-level takeaway from the last five years of my research: **to better support creativity, seek out inspiration from its most playful forms.**

Bibliography

[1] Tarn Adams. Emergent narrative in *Dwarf Fortress*. In *Procedural Storytelling in Game Design*, pages 149–158. AK Peters/CRC Press, 2019.

[2] Philip Agre. Toward a critical technical practice: lessons learned in trying to reform AI. In Geoffrey Bowker, Susan Leigh Star, William Turner, and Les Gasser, editors, *Bridging the Great Divide: Social Science, Technical Systems, and Cooperative Work*, pages 131–157. Mahwah, NJ: Erlbaum, 1997.

[3] Lea Albaugh, April Grow, Chenxi Liu, James McCann, Gillian Smith, and Jennifer Mankoff. Threadsteading: playful interaction for textile fabrication devices. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 285–288, 2016.

[4] Avery Alder. The Quiet Year. `https://buriedwithoutceremony.com/the-quiet-year`, 2013.

[5] Bay 12 Games. Dwarf Fortress. `https://bay12games.com/dwarves`, 2006.

[6] Botnik. Predictive Writer. `https://botnik.org/apps/writer`, 2017.

[7] Peter Brooks. *Reading for the Plot: Design and Intention in Narrative.* Harvard University Press, 1984.

[8] Matt Brown. The power of projection and mass hallucination: Practical AI in The Sims 2 and beyond. Invited talk at AIIDE 2006, 2006.

[9] Robin Burkinshaw. Alice and Kev. `https://aliceandkev.wordpress.com`, 2009.

[10] Robin Burkinshaw. Meaningful Stories for The Sims 4. `https://roburky.itch.io/sims4-meaningful-stories`, 2021.

[11] Alex Calderwood, Vivian Qiu, Katy Ilonka Gero, and Lydia B Chilton. How novelists use generative language models: An exploratory user study. In *HAI-GEN + user2agent @ IUI*, 2020.

[12] Rogelio Cardona-Rivera and Robert Young. Symbolic plan recognition in interactive narrative environments. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 11, 2015.

[13] Rogelio E Cardona-Rivera, Kara B 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*, pages 34–43, 2012.

[14] Erin A Carroll, Celine Latulipe, Richard Fung, and Michael Terry. Creativity factor evaluation: towards a standardized survey metric for creativity support. In

Proceedings of the Seventh ACM Conference on Creativity and Cognition, pages 127–136. ACM, 2009.

[15] Erin Cherry and Celine Latulipe. Quantifying the creativity support of digital tools through the Creativity Support Index. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(4), 2014.

[16] John Joon Young Chung, Shiqing He, and Eytan Adar. The intersection of users, roles, interactions, and technologies in creativity support tools. In *Designing Interactive Systems Conference 2021*, pages 1817–1833. ACM, 2021.

[17] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. TaleBrush: Sketching stories with generative pretrained language models. In *CHI Conference on Human Factors in Computing Systems*, 2022.

[18] Tom Cole and Marco Gillies. More than a bit of coding: (un-) grounded (non-) theory in HCI. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022.

[19] Simon Colton and Geraint A Wiggins. Computational creativity: The final frontier? In *ECAI 2012 - 20th European Conference on Artificial Intelligence*, volume 12, pages 21–26. IOS Press, 2012.

[20] Kate Compton, Quinn Kybartas, and Michael Mateas. Tracery: an author-focused

generative text tool. In *International Conference on Interactive Digital Storytelling*, pages 154–161. Springer, 2015.

[21] Kate Compton and Michael Mateas. Casual creators. In *International Conference on Computational Creativity*, pages 228–235, 2015.

[22] Katherine Compton. *Casual Creators: Defining a Genre of Autotelic Creativity Support Systems*. PhD thesis, University of California, Santa Cruz, 2019.

[23] Chinmaya Dabral and Chris Martens. Generating explorable narrative spaces with answer set programming. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 45–51, 2020.

[24] Sebastian Deterding, Jonathan Hook, Rebecca Fiebrink, Marco Gillies, Jeremy Gow, Memo Akten, Gillian Smith, Antonios Liapis, and Kate Compton. Mixed-initiative creative interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 628–635, 2017.

[25] Mirjam P Eladhari, Anne Sullivan, Gillian Smith, and Josh McCoy. AI-based game design: enabling new playable experiences. Technical report, UC Santa Cruz Baskin School of Engineering, 2011.

[26] Mirjam Palosaari Eladhari. Re-tellings: the fourth layer of narrative as an instrument for critique. In *International Conference on Interactive Digital Storytelling*, pages 65–78. Springer, 2018.

[27] David K Elson. Detecting story analogies from annotations of time, action and

agency. In *Proceedings of the LREC 2012 Workshop on Computational Models of Narrative, Istanbul, Turkey*, pages 91–99, 2012.

[28] Richard Evans and Emily Short. Versu—a simulationist storytelling system. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):113–130, 2013.

[29] Linda Flower and John R Hayes. A cognitive process theory of writing. *College Composition and Communication*, 32(4):365–387, 1981.

[30] Freehold Games. Caves of Qud. `https://www.cavesofqud.com`, 2022.

[31] Jonas Frich, Lindsay MacDonald Vermeulen, Christian Remy, Michael Mose Biskjaer, and Peter Dalsgaard. Mapping the landscape of creativity support tools in HCI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019.

[32] Jacob Garbe. Simulation of history and recursive narrative scaffolding. `http://project.jacobgarbe.com/simulation-of-history-and-recursive-narrative-scaffolding`, Feb 2018.

[33] William Gaver. What should we expect from research through design? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 937–946, 2012.

[34] John S Gero. Fixation and commitment while designing and its measurement. *The Journal of Creative Behavior*, 45(2):108–115, 2011.

[35] Katy Gero, Alex Calderwood, Charlotte Li, and Lydia Chilton. A design space for writing support tools using a cognitive process model of writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 11–24, 2022.

[36] Natalie Goldberg. *Writing Down the Bones: Freeing the Writer Within.* Shambhala, 2005.

[37] Jason Grinblat and C Brian Bucklew. Subverting historical cause & effect: generation of mythic biographies in Caves of Qud. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, page 76. ACM, 2017.

[38] Jason Grinblat, Cat Manning, and Max Kreminski. Emergent narrative and reparative play. In *International Conference on Interactive Digital Storytelling*, pages 208–216. Springer, 2021.

[39] Matthew Guzdial, Devi Acharya, Max Kreminski, Michael Cook, Mirjam Eladhari, Antonios Liapis, and Anne Sullivan. Tabletop roleplaying games as procedural content generators. In *International Conference on the Foundations of Digital Games*, 2020.

[40] Minh Hua and Rita Raley. Playing with unicorns: AI Dungeon and citizen NLP. *DHQ: Digital Humanities Quarterly*, 14(4), 2020.

[41] Kathryn Hymes. Developing ‘artifacts of play’ for your tabletop games. `https://`

www.gdcvault.com/play/1026745/Board-Game-Design-Summit-Developing, 2020.

[42] Tom Knoll. The think-aloud protocol. In *Games User Research*, pages 189–202. Oxford University Press, 2018.

[43] Hartmut Koenitz. Towards a theoretical framework for interactive digital narrative. In *Joint International Conference on Interactive Digital Storytelling*, pages 176–185. Springer, 2010.

[44] Max Kreminski. Procedural narrative design with parametrized storylets. https://www.gdcvault.com/play/1025699/Tech, 2019.

[45] Max Kreminski, Devi Acharya, Nick Junius, Elisabeth Oliver, Kate Compton, Melanie Dickinson, Cyril Focht, Stacey Mason, Stella Mazeika, and Noah Wardrip-Fruin. Cozy Mystery Construction Kit: Prototyping toward an AI-assisted collaborative storytelling mystery game. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019.

[46] Max Kreminski, Melanie Dickinson, and Michael Mateas. Winnow: A domain-specific language for incremental story sifting. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, pages 156–163, 2021.

[47] Max Kreminski, Melanie Dickinson, Michael Mateas, and Noah Wardrip-Fruin. Why Are We Like This?: Exploring writing mechanics for an AI-augmented sto-

rytelling game. In *Proceedings of the 2020 Conference of the Electronic Literature Organization*, 2020.

[48] Max Kreminski, Melanie Dickinson, Michael Mateas, and Noah Wardrip-Fruin. Why Are We Like This?: The AI architecture of a co-creative storytelling game. In *International Conference on the Foundations of Digital Games*, 2020.

[49] Max Kreminski, Melanie Dickinson, Joseph Osborn, Adam Summerville, Michael Mateas, and Noah Wardrip-Fruin. Germinate: A mixed-initiative casual creator for rhetorical games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 102–108, 2020.

[50] Max Kreminski, Melanie Dickinson, and Noah Wardrip-Fruin. Felt: a simple story sifter. In *International Conference on Interactive Digital Storytelling*, pages 267–281. Springer, 2019.

[51] Max Kreminski, Melanie Dickinson, Noah Wardrip-Fruin, and Michael Mateas. Loose Ends: A mixed-initiative creative interface for playful storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 2022.

[52] Max Kreminski, Isaac Karth, Michael Mateas, and Noah Wardrip-Fruin. Evaluating mixed-initiative creative interfaces via expressive range coverage analysis. In *IUI Workshops*, pages 34–45, 2022.

[53] Max Kreminski and Chris Martens. Unmet creativity support needs in compu-

tationally supported creative writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 74–82, 2022.

[54] Max Kreminski and Michael Mateas. A coauthorship-centric history of interactive emergent narrative. In *International Conference on Interactive Digital Storytelling*, pages 222–235. Springer, 2021.

[55] Max Kreminski and Michael Mateas. Reflective creators. In *International Conference on Computational Creativity*, 2021.

[56] Max Kreminski and Michael Mateas. Toward narrative instruments. In *International Conference on Interactive Digital Storytelling*, pages 499–508. Springer, 2021.

[57] Max Kreminski, Ben Samuel, Edward Melcer, and Noah Wardrip-Fruin. Evaluating AI-based games through retellings. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pages 45–51, 2019.

[58] Max Kreminski and Noah Wardrip-Fruin. Sketching a map of the storylets design space. In *International Conference on Interactive Digital Storytelling*, pages 160–164. Springer, 2018.

[59] Max Kreminski and Noah Wardrip-Fruin. Generative games as storytelling part-

ners. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019.

[60] Max Kreminski, Noah Wardrip-Fruin, and Michael Mateas. Toward example-driven program synthesis of story sifting patterns. In *Joint Proceedings of the AIIDE 2020 Workshops*, 2020.

[61] Kromtec. Legends Viewer. `https://github.com/Kromtec/LegendsViewer`, 2015.

[62] Quinn Kybartas and Rafael Bidarra. A semantic foundation for mixed-initiative computational storytelling. In *International Conference on Interactive Digital Storytelling*, pages 162–169. Springer, 2015.

[63] Quinn Kybartas and Rafael Bidarra. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):239–253, 2016.

[64] Quinn Kybartas, Clark Verbrugge, and Jonathan Lessard. Tension space analysis for emergent narrative. *IEEE Transactions on Games*, 13(2):146–159, 2020.

[65] Anne Lamott. *Bird by Bird: Some Instructions on Writing and Life*. Knopf Doubleday, 2007.

[66] Bjarke Alexander Larsen, Luis Emilio Bruni, and Henrik Schoenau-Fog. The story we cannot see: On how a retelling relates to its afterstory. In *International Conference on Interactive Digital Storytelling*, pages 190–203. Springer, 2019.

[67] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. In *Symposium of the Austrian HCI and Usability Engineering Group*, pages 63–76. Springer, 2008.

[68] Antonios Liapis, Georgios N Yannakakis, Constantine Alexopoulos, and Phil Lopes. Can computers foster human users' creativity? Theory and praxis of mixed-initiative co-creativity. *Digital Culture & Education*, 8(2):136–153, 2016.

[69] Genevieve Liveley. Anticipation and narratology. In Roberto Poli, editor, *Handbook of Anticipation: Theoretical and Applied Aspects of the Use of Future in Decision Making*. Springer, 2017.

[70] Sandy Louchart and Ruth Aylett. The emergent narrative theoretical investigation. In *The 2004 Conference on Narrative and Interactive Learning Environments*, pages 21–28, 2004.

[71] Sandy Louchart, John Truesdale, Neil Suttie, and Ruth Aylett. Emergent narrative, past, present and future of an interactive storytelling approach. In *Interactive Digital Narrative: History, Theory and Practice*, pages 185–199. Routledge, 2015.

[72] Enrique Manjavacas, Folgert Karsdorp, Ben Burtenshaw, and Mike Kestemont. Synthetic literature: Writing science fiction in a co-creative process. In *Proceedings of the Workshop on Computational Creativity in Natural Language Generation (CC-NLG 2017)*, pages 29–37, 2017.

[73] Chris Martens. Ceptre: A language for modeling generative interactive systems. In

Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference, 2015.

[74] Chris Martens and Matthew A Hammer. Languages of play: towards semantic foundations for game interfaces. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, pages 32–41. ACM, 2017.

[75] Lara Jean Martin. *Neurosymbolic Automated Story Generation.* PhD thesis, Georgia Institute of Technology, 2021.

[76] Stacey Mason. *Responsiveness in Narrative Systems.* PhD thesis, University of California, Santa Cruz, 2021.

[77] Gus Mastrapa. Kiwi comic tells tale of Dwarf Fortress failure. `https://www.wired.com/2010/09/oilfurnace`, 2010.

[78] Michael Mateas and Andrew Stern. Build it to understand it: Ludology meets narratology in game design space. In *DiGRA '05 - Proceedings of the 2005 DiGRA International Conference: Changing Views: Worlds in Play*, 2005.

[79] Michael Mateas, Paul Vanouse, and Steffi Domike. Generation of ideologically-biased historical documentaries. In *AAAI/IAAI*, pages 236–242, 2000.

[80] Maxis. The Sims 2. `https://ea.com/games/the-sims/the-sims-2`, 2004.

[81] Josh McCoy, Mike Treanor, Ben Samuel, Aaron A Reed, Noah Wardrip-Fruin, and Michael Mateas. Prom Week: designing past the game/story dilemma. In

Proceedings of the International Conference on the Foundations of Digital Games, 2013.

[82] Joshua McCoy, Mike Treanor, Ben Samuel, Aaron A Reed, Michael Mateas, and Noah Wardrip-Fruin. Social story worlds with Comme il Faut. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):97–112, 2014.

[83] James R Meehan. TALE-SPIN, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 91–98, 1977.

[84] Eric Murnane. *Emergent Narrative: Stories of Play, Playing with Stories.* PhD thesis, University of Central Florida, 2018.

[85] Kumiyo Nakakoji. Meanings of tools, support, and uses for creative design processes. In *International Design Research Symposium*, volume 6, pages 156–165, 2006.

[86] Bonnie A Nardi. *A Small Matter of Programming: Perspectives on End User Computing.* MIT Press, 1993.

[87] Mark J Nelson. Emergent narrative in The Sims 2. `https://www.kmjn.org/notes/sims2_ai.html`, 2006. Accessed: 2021-08-20.

[88] Mark J Nelson, Swen E Gaudl, Simon Colton, and Sebastian Deterding. Curious users of casual creators. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 2018.

[89] Jakob Nielsen. *Usability Engineering.* Morgan Kaufmann, 1993.

[90] Jeff Orkin. Three states and a plan: the A.I. of F.E.A.R. In *Game Developers Conference*, 2006.

[91] Joseph Osborn, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. Playspecs: Regular expressions for game play traces. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 11, 2015.

[92] Paradox Interactive. Crusader Kings II. `https://www.paradoxinteractive.com/games/crusader-kings-ii/about`, 2012.

[93] Julie Porteous. Planning technologies for interactive storytelling. In *Handbook of Digital Games and Entertainment Technologies.* Springer, 2016.

[94] Nikita Prokopov. DataScript. `https://github.com/tonsky/datascript`, 2014.

[95] Aaron Reed. Archives of the Sky. `https://archivesofthesky.textories.com`, 2018.

[96] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. Design principles for tools to support creative thinking. In *NSF Workshop Report on Creativity Support Tools*, 2005.

[97] Ben Robbins. Microscope: A Fractal Role-Playing Game of Epic Histories. `https://lamemage.com/microscope`, 2011.

[98] Melissa Roemmele and Andrew S Gordon. Creative Help: A story writing assistant. In *International Conference on Interactive Digital Storytelling*, pages 81–92. Springer, 2015.

[99] James Ryan. *Curating Simulated Storyworlds.* PhD thesis, University of California, Santa Cruz, 2018.

[100] James Ryan and Michael Mateas. Simulating character knowledge phenomena in Talk of the Town. In *Game AI Pro 360*, pages 135–150. CRC Press, 2019.

[101] James Owen Ryan, Michael Mateas, and Noah Wardrip-Fruin. Open design challenges for interactive emergent narrative. In *International Conference on Interactive Digital Storytelling*, pages 14–26. Springer, 2015.

[102] Marie-Laure Ryan. Narrative in real time: chronicle, mimesis and plot in the baseball broadcast. *Narrative*, 1(2):138–155, 1993.

[103] Marie-Laure Ryan. Narrative and the split condition of digital textuality. *Dichtung Digital*, 7(1), 2005.

[104] Marie-Laure Ryan. *Avatars of Story.* University of Minnesota Press, 2006.

[105] Marie-Laure Ryan. From narrative games to playable stories: Toward a poetics of interactive narrative. *Storyworlds: A Journal of Narrative Studies*, 1:43–59, 2009.

[106] Ben Samuel. *Crafting Stories Through Play.* PhD thesis, University of California, Santa Cruz, 2016.

[107] Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. The design of Writing Buddy: a mixed-initiative approach towards computational story collaboration. In *International Conference on Interactive Digital Storytelling*, pages 388–396. Springer, 2016.

[108] Ben Samuel, Aaron A Reed, Paul Maddaloni, Michael Mateas, and Noah Wardrip-Fruin. The Ensemble engine: next-generation social physics. In *Proceedings of the Tenth International Conference on the Foundations of Digital Games (FDG 2015)*, pages 22–25, 2015.

[109] Ben Samuel, James Ryan, Adam J Summerville, Michael Mateas, and Noah Wardrip-Fruin. Bad News: An experiment in computationally assisted performance. In *International Conference on Interactive Digital Storytelling*, pages 108–120. Springer, 2016.

[110] Donald A Schön. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books, 1983.

[111] Janelle Shane. SkyKnit: When knitters teamed up with a neural network. `https://www.aiweirdness.com/skyknit-when-knitters-teamed-up-with-18-04-19`, 2018.

[112] Ben Shneiderman. Creativity support tools: Accelerating discovery and innovation. *Communications of the ACM*, 50(12):20–32, 2007.

[113] Emily Short. Beyond branching: Quality-based, salience-based, and

waypoint narrative structures. https://emshort.blog/2016/04/12/beyond-branching-quality-based-and-salience-based-narrative-structures, 2016.

[114] Nikhil Singh, Guillermo Bernal, Daria Savchenko, and Elena L Glassman. Where to hide a stolen elephant: Leaps in creative writing with multimodal machine intelligence. *ACM Transactions on Computer-Human Interaction*, 2022.

[115] Robin Sloan. Writing with the machine. https://robinsloan.com/notes/writing-with-the-machine, May 2016.

[116] Adam M Smith and Michael Mateas. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):187–200, 2011.

[117] Adam M Smith and Michael Mateas. Computational caricatures: probing the game design process with AI. In *Workshops at the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[118] Gillian Smith and Jim Whitehead. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 workshop on Procedural Content Generation in Games*, 2010.

[119] Ingibergur Stefnisson and David Thue. Mimisbrunnur: AI-assisted authoring for interactive storytelling. In *Proceedings of the AAAI Conference on Artificial*

Intelligence and Interactive Digital Entertainment, volume 14, pages 236–242, 2018.

[120] Adam Summerville. Expanding expressive range: Evaluation methodologies for procedural content generation. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2018.

[121] Reid Swanson and Andrew S Gordon. Say Anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3), 2012.

[122] Steven Sych. When the fourth layer meets the fourth wall: the case for critical game retellings. In *International Conference on Interactive Digital Storytelling*, pages 203–211. Springer, 2020.

[123] Atau Tanaka. Interaction, experience and the future of music. In *Consuming Music Together*, pages 267–288. Springer, 2006.

[124] Theresa Jean Tanenbaum and Angela Tomizu. Narrative meaning creation in interactive storytelling. *International Journal of Computational Science*, 2(1):3–20, 2008.

[125] Zach Tomaszewski. On the use of reincorporation in interactive drama. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 7, pages 84–91, 2011.

[126] Mike Treanor, Alexander Zook, Mirjam P Eladhari, Julian Togelius, Gillian Smith,

Michael Cook, Tommy Thompson, Brian Magerko, John Levine, and Adam Smith. AI-based game design patterns. In *Proceedings of the Tenth International Conference on the Foundations of Digital Games (FDG 2015)*, 2015.

[127] u/RireMakar. This single adventure is consuming my life. 159 pages to finally slay the good ol' Dragon of Larion. My family misses me... `https://www.reddit.com/r/AIDungeon/comments/hwr0sf/this_single_adventure_is_consuming_my_life_159`, 2020.

[128] Noah Wardrip-Fruin. Playable media and textual instruments. *Dichtung Digital*, 34:211–253, 2005.

[129] Noah Wardrip-Fruin. *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*, chapter The Tale-Spin Effect, pages 115–168. MIT Press, 2009.

[130] Noah Wardrip-Fruin, Michael Mateas, Steven Dow, and Serdar Sali. Agency reconsidered. In *DiGRA Conference*, 2009.

[131] 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.

[132] R Michael Young, Stephen G Ware, Kara B 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 Daten-*

verarbeitung, Special Issue on Formal and Computational Models of Narrative, 37(1-2):41–64, 2013.

[133] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 493–502, 2007.