Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2023)

DendryScope: Narrative Designer Support via Symbolic Analysis

Jasmine Otto[1], **Autumn Chen**[2], **Adam M. Smith**[1]
[1]University of California Santa Cruz
[2]Independent
jtotto@ucsc.edu, cchen.intfic@gmail.com, amsmith@ucsc.edu

Abstract

Quality-based narratives (QBN) are hypertexts with extensive implicit linked structure. The observation of one passage can have non-obvious long-range implications for the reachability of other passages, which poses an authoring challenge. To help narrative designers address this issue, we produced an interface which visually summarizes all possible playtraces. Our interface leverages answer set programming to produce a query language over possible playtraces, allowing narrative designers to drill down to interesting scenarios. We introduce this interface through the DendryScope tool, which accepts most QBNs written in the Dendry language. We evaluated DendryScope by interviewing four narrative designers as they used the tool to explore *Bee*, a notable QBN written by Emily Short.

Introduction

Narrative designers are interested in creating works of interactive digital narrative (IDN) that can be enjoyed across multiple playthroughs by many readers, who are given some agency in shaping the arrangement of scenes they will encounter, and even led to believe they can control the outcome of the story.

Extensive prior work in IDN discusses the authoring challenges inherent to this medium Millard and Hargood (2021). Calls for tool assistance in IDEs (Bernstein, Millard, and Weal 2002) have thus far been taken up within the IDN community, especially by narrative designers with strongly computer science backgrounds. At times, the specific needs of playtesting been discussed in the AI literature, which we discuss further in Background. This paper argues that *quality-based narrative* (QBN) is a particularly robust subset of IDN, and demonstrates the power of symbolic AI tools for addressing practicing narrative designers' actual needs.

In this work, we define *skeins of playtraces* in terms of answer set programming queries (ASP queries) that describe properties of certain playtraces. We have implemented a novel *creative interface* (Deterding et al. (2017)) for a subset of ASP queries, designed to empower narrative designers to reason about complex works of IDN. Building on the design space approach advocated by Smith and Mateas (2011), DendryScope enables narrative designers to manipulate the design space of skeins in a given work of IDN. Our research implementation deals specifically with *quality-based narratives* written in the Dendry language (Millington and Chen 2015).

This paper introduces the four following research contributions, which each build on each other, and apply symbolic artificial intelligence to key challenges in the production of interactive digital narratives (IDN).

- A transpiler from Dendry to the ASP domain. Each space of solutions in the domain of a given IDN, given a certain query, is a *skein of playtraces*.
- The DendryScope query language, where each query is a set of ASP constraints authored by the narrative designer. We foresee query sharing between designers as enabling continuous integration for IDN, and other novel forms of testing and collaboration.
- An inventory of seven tasks performed by narrative designers who are playtesting an IDN, especially in the context of sculptural hypertext (with discrete passages and implicit links).
- The heatmap interface to a skein, which is a direct manipulation interface for novel DendryScope queries. This interface allows narrative designers without prior ASP knowledge to 'drill down' through the families of possible playtraces in a Dendry game.

Figure 1 summarizes these contributions in the context of a DendryScope-driven playtesting workflow, using the acclaimed IDN *Bee* (Short 2012) as the object of study. We conducted four expert interviews to investigate this proposed workflow. The designer tasks we gathered explicate how DendryScope queries address the challenge of reasoning about hundreds of thousands of possible playtraces.

Background

20 years ago, the authors of *Card Shark* (Bernstein, Millard, and Weal 2002) called for tighter loops between authoring and playtesting in sculptural hypertext. Our playtesting tool supports narrative designers' existing authoring strategies for IDN, which allow them to skillfully reason about player retellings, replay stories (Mitchell (2022)), and other forms of play in and around the *story volume* (Karth, Junius, and Kreminski (2022)).

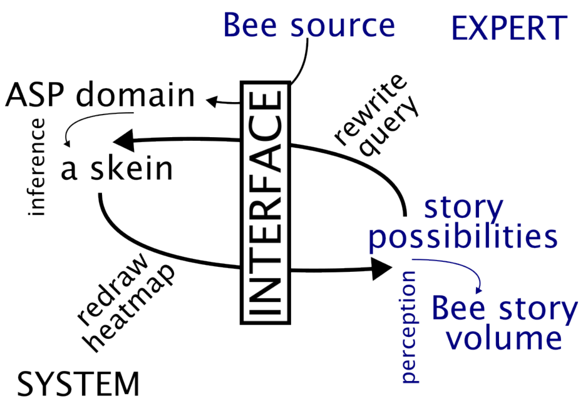

Figure 1: Diagram of a typical DendryScope workflow. Experts may use DendryScope to ingest the source of their QBN (in this case, *Bee*) and help them to envision the story volume accessible to players. Components shown in black are contributions of this paper, including: a Dendry to ASP compiler, an isomorphism between ASP queries and 'skeins' of playtraces, and the heatmap representation of a skein.

Grinblat describes the story volume of a given IDN as constructed in the player's mind through 'reparative play' in the course of one or more playthroughs Grinblat, Manning, and Kreminski (2021). The narrative designer, in the course of playtesting their IDN, is not only looking for bugs (i.e. obvious narrative failures), but also to emulate this process of story volume construction (Fig. 1). In this section, we will discuss the authoring formalism of quality-based narrative (QBN), contexts in which QBN is used, and how it enables automated symbolic playtesting that can assist with story volume construction.

Situating Quality-Based Narrative

Sculptural hypertexts are IDNs constructed from discrete passages, which are implicitly linked by preconditions and postconditions (Figure 2). A QBN is a sculptural hypertext whose world model is a string-int dictionary; this helps limit spurious complexity in the preconditions of each *storylet* (as discussed by Kreminski and Wardrip-Fruin (2018)). Practitioners of QBN call these dictionary keys *qualities*; each has an integer *value* that changes over time, as the player traverses passages with relevant postconditions. The value of a quality defaults to 0, and is often glossed by a string.

Quality-based narratives form a rich narrative substrate from simple ludological underpinnings: every goal can be expressed in terms of some qualities that have attained certain values. This design constraint greatly limits 'programming scope', and can prevent it from taking over an IDN project (Jones 2022).

Unlike traditional hypertexts with explicit link structures, connectivity between passages in a sculptural hypertext is unknown until playtime, because it is dynamically produced by the combination of game state (in terms of qualities) and the content selection procedure - i.e. which available passages from the current 'deck' (as a function of world state) are presented to the player in any given 'hand' of choices.

Many full-scale commercial games, such as *Fallen London*, *Hades* (per People Make Games (2020)), and *I Was A Teenage Exocolonist*, have been written primarily in QBN. Yet dedicated languages of QBN like Dendry are unusual; both QBN and sculptural hypertext are often embedded inside of other scripting languages, as Jones describes. For example, *Hades* is scripted in Lua, and *Facade* is scripted in ABL (Mateas and Stern 2005).

IDN Authoring Tools

A recent survey of IDN authoring tools (Green, Hargood, and Charles 2021) has found that narrative designers prefer tools that visualize the structure and connectivity of a story volume. They are also interested in tools that enable focused playtesting sessions, and in tools that encourage novice designers to experiment with branching.

In a broader sense, IDN authoring tools are a kind of IDE used by artists and teachers to explore an interactive medium. McNutt, Outkine, and Chugh (2023) have tested IDE features ranging from simple code linting, to inline code evaluation, to bidirectional manipulation of a generated artifact. DendryScope uses an intermediate heatmap visualization as the interactive surface for structured user input, such as placing 'pins' to refine a query.

Narrative designers must reason both as authors and as game designers. Cardona-Rivera, Zagal, and Debus (2020) describe narrative games in terms of two parallel goal hierarchies: narrative goals forming a descending chain of reasoning, and ludological goals forming an ascending chain of causality. For example, I might (narratively) want to open the door, and therefore need the key; at the same time, I would (ludologically) enable the 'open door' verb by getting my 'key' quality to 1 instead of 0. The narrative designer can cleverly align their player's narrative goals with the logic of which qualities they intend to track, as we will see in *Bee*.

Automated Playtesting in Hypertext and IF

The goal of playtesting is often to characterize the possible playtraces of a given sculptural hypertext. Each such playtrace is a sequence of passages (Figure 3) that a player could visit, determined by the preconditions and postconditions of each passage, and assuming the content selection procedure presents each passage at the relevant step.

We do not claim that automated playtesting strategies incorporate any knowledge of narrative goals at all. Rather, they are designed to help narrative designers diagnose issues with a set of ludological goals intended to parallel those narrative goals. To this end, we review existing playtesting strategies used by narrative designers.

Traversal Statistics An automated playtesting strategy can produce thousands of playtraces in a short time. As mass-generated playtraces tend to repeat large sections of play, to the point that reviewing them one-by-one is tedious, they should be aggregated or visualized. Indeed, statistical summaries of thousands of playtraces are commonly used by authors using ChoiceScript, Dendry, and custom narra-

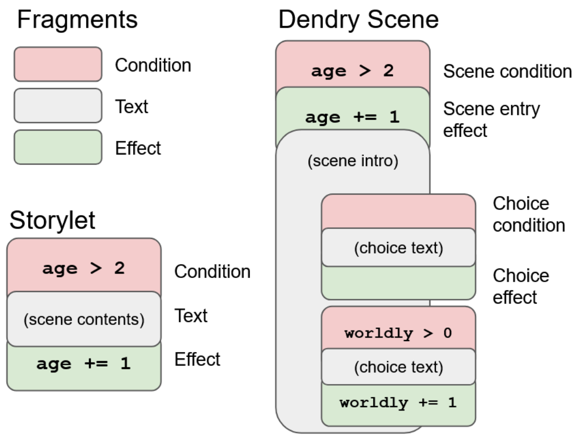

Figure 2: In sculptural hypertext, a storylet consists of three authored fragments: preconditions to check against state, postconditions to apply over state, and the passage of text that the player sees. In Dendry, a scene is typically a directed acyclic graph of linked storylets.

tive systems; an example of the latter is documented by the authors of *Ice-Bound* (Garbe et al. 2014).

A conventional traversal summary strategy is to count the percentage of sampled playtraces that saw each passage. In extending this idea to their own system of sculptural hypertext, the *Ice-Bound* authors understand their playtesting strategy as a 'level profiler' which locates combinations of 'symbols' (i.e. those qualities a player could have or acquire in a given level) that are insufficiently 'fleshed out' by passages corresponding to that combination. In other words, their histogram visualization aggregates many playtraces over time and instance according to state - in this case, the availability of various combinations of symbols.

Traversal Diagrams Recent work by Veloso and Prada (2021) produced the *Story Validator* IDE for Twine games, which produces a skein-like view of playtraces by automatically generating the graph of possible links between passages. Their enumeration strategy relies on a greedy graph search heuristic, based on weights the designer assigns to each quality. They visualize the resulting playtraces in a flow chart, which aggregates all the possible orderings into a sequence of passages over time. Prior work in IDN authoring tools for Twine have explored this concept using heuristic traversal strategies (Shibolet, Knoller, and Koenitz 2018; Oliver and Smith 2018). Similarly, the Inform 7 skein (Nelson 2006) is a flow chart produced from world state in an IF game, i.e. an IDN where passages are dynamically composed together according to complex world state. Inform's skein is constructed manually, as the author supplies each possible (partial) playtrace by playing through their own game. This makes it suitable for asking questions about choice points that ensue after the first few beats of the game, but not about skeins in the mid-game or late-game.

A symbolic approach, quite similar to the one we use in DendryScope, to enumerating Twine playtraces was demonstrated by Oliver and Smith (2018), who summarized these in terms of values that variables could attain, and `end`-tagged passages that could be reached.

Symbolic Playtesting Narrative design tasks involving the story volume can be facilitated by using modern answer set programming (ASP) systems to implement playtesting strategies. ASP is applicable when the possible values of game state variables can be modeled as a finite set, the length of interesting playtraces can be bounded, and the rules for how the state should be updated in response to each available player choice can be expressed with rules in symbolic logic. For example, there are solver-based authoring tools that support level designers creating difficulty progression in educational puzzle games (Butler et al. 2013). In the IDN context, every solution of the answer set program is a distinct playtrace of the IDN.

Player behavior can be modeled as certain constraints, such as 'I will reach this particular ending'. This playtesting strategy has been demonstrated in *PlotEx* (Plotkin 2012), a tool created by the author of *Hadean Lands* to guarantee that all four endings in his Inform 7 code were reachable. Each door in the game requires specific resources to open, and certain resources are consumed thereby, so it is not apparent which resources should be located behind which doors. If an ending did prove to be unreachable, then Plotkin would need to make specific resources available earlier or in greater quantity at certain points in the game; or otherwise, make certain doors barring the endgame less demanding to open.

In PlotEx, sets of constraints are understood as a kind of unit test, which can be run against the latest development build of an IDN in order to spot regressions. In Ludocore (Smith, Nelson, and Mateas 2010), these are called speculative assumptions, because they represent constraints on play styles that apply to some scenarios and not others. In the context of DendryScope, we will call these *queries*. All three of the above systems may be construed as story planners in the domain of a particular IDN or gameplay model, which produce (all possible) examples from that story volume.

Methods

We now identify the elementary passages in Dendry's authoring language, and how they relate to the changing value of qualities over time in a given playtrace. Our symbolic analysis strategy is to enumerate all playtraces which satisfy a given query, which we will call a *skein*. The empty query corresponds to the universal skein, containing all possible playthroughs of the given QBN.

The *Dendry* Authoring Language

We chose to build the DendryScope prototype on top of the Dendry language (Fig. 2): it is narrowly scoped around QBN, its compiler is open-source, and the complete source of *Bee* is available. However, any language of IDN that can be expressed in terms of ASP rules is amenable to our overall strategy, and any QBN can be represented in the DendryScope interface.

The Dendry language includes three kinds of storylet:

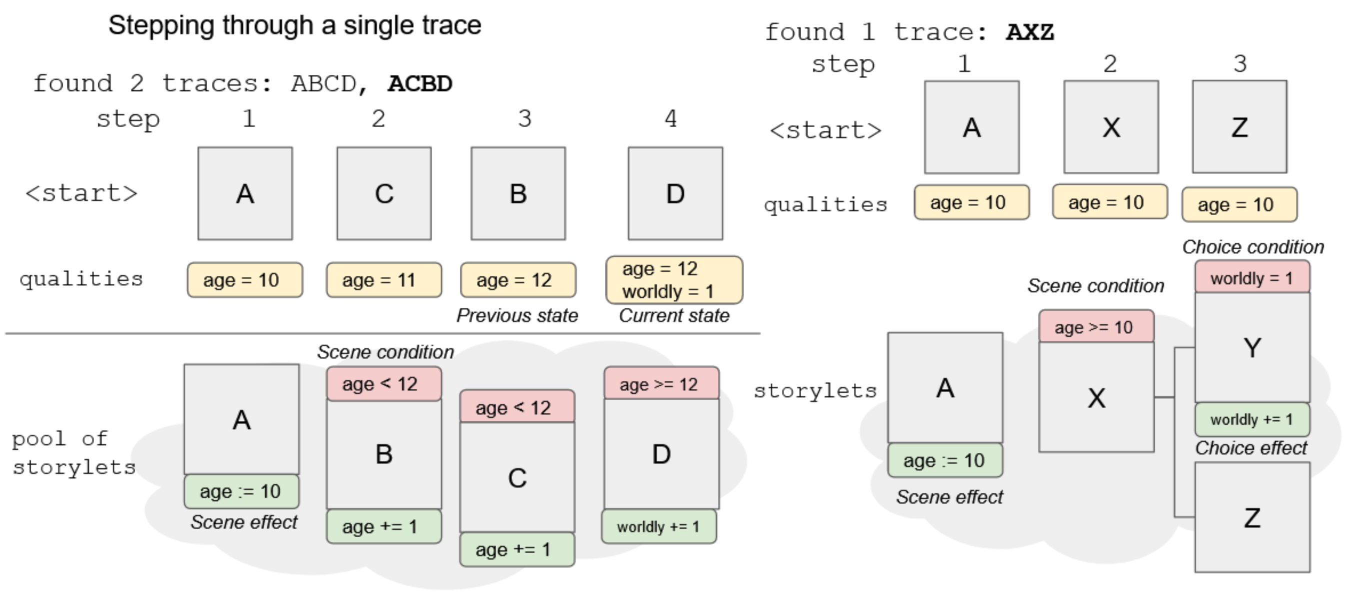

Figure 3: Each playtrace in the skein of a sculptural hypertext (in this example, a QBN) is a possible sequence of storylets. **Left:** A skein with two playtraces: passages B and C can be seen in either order, but both must be seen before D. **Right:** A skein with one playtrace: Either Y or Z must follow X. The precondition of Y is unsatisfiable, so AXZ is the only possible playtrace.

- **Top-level scenes**, which are accessible from the player's hand of choices.
- **Linked choices**, which either link to further choices or exit to the player's hand.
- **The player's hand**, which contains a small number of top-level scenes whose preconditions are met. The player decides which to pursue next.

The postconditions of every Dendry storylet describe how to transform the state of qualities from one timestep to the next, influencing which storylets' preconditions can be satisfied on the next timestep. In *Bee*, none of the qualities attain values outside of a small integer range (from -1 to +24); in general, knowing the range of each quality is useful to speed up ASP grounding.

Every Dendry scene file consists of scene and choice storylets, and has a Markdown-like representation[1]. These correspond to scenes in the sculptural hypertext system Lume (Mason, Stagg, and Wardrip-Fruin 2019), whose authors likewise chose to embed branch-and-bottleneck choice structures (i.e. DAGs) inside of content selection, as a sensible unit of authoring.

Reducing *Dendry* Traversals to ASP

In the literature of answer set programming (ASP), it is common to sketch an ASP formulation in terms of which logical facts the solver is allowed to nondeterminstically *guess*, which facts the solver must *deduce* from others, and which potential facts the solver must *forbid* from appearing in solutions to be enumerated.

In our formalization of Dendry, we say the solver must **guess** exactly one passage to select for display among those passages that have their preconditions satisfied by the current quality values. From the sequence of selected passages, we **deduce** the evolution of quality values over time, influencing which passages are available.

Finally, we **forbid** solutions that do not respect the user's query conditions: we reject solutions that do not satisfy all *goals* (scenes that need to play at least once), *poisons* (scenes that must never play), and *pins* (scenes that must play at least once on the specified timestep).

DendryScope is implemented using the `clingo` ASP solver (Gebser et al. 2019) through its WebAssembly port (Moritz 2022). Rather than asking the solver for just one satisfying solution, we systematically enumerate a few hundred solutions. These concrete examples allow the designer to observe differences in state between playtraces.

Applying `clingo`'s brave enumeration (a form of projected enumeration that efficiently computes the union of all of a program's answer sets), we also obtain the complete set of which passages could be seen at which timesteps while abiding by the query conditions, without directly constructing every playtrace.

Full implementation details, including our exact traversal strategy and our method of tracking integer-valued quality variables over time, can be found in our system's source code at: github.com/JazzTap/DendryScope.

[1]For example, scenes in *Bee* are represented by `.scene.dry` files found in its repository: https://github.com/aucchen/bee/blob/master/source/scenes/church.scene.dry

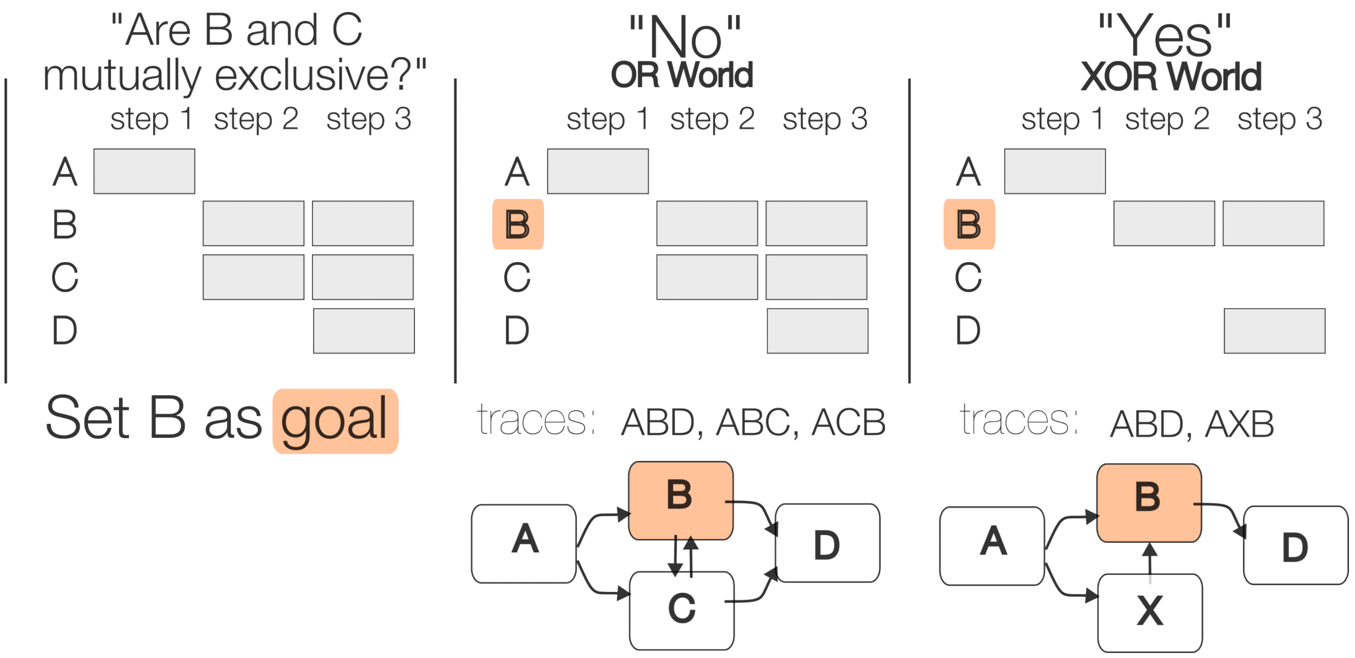

Figure 4: Worked example of reading a skein, and using a goal to discover implicit structure from subskeins. Four rows correspond to four Dendry passages. Each grey cell shows that some traces hit that passage at that timestep. We set 'B' as a query goal, and recalculate the skein (which now contains only playtraces visiting 'B'). Now if the playtraces 'ABC' and 'ACB' exist, then B and C are not mutually exclusive ('OR World'). Conversely, if C disappears from the skein, then B and C must exclude each other ('XOR World'), and (in this case) some offscreen passage X explains the trace that sees B at step 3.

Visualization in DendryScope

In addition to formalizing the QBN domain, we created a frontend for query design via direct annotation of playtesting information in the form of a skein, i.e. the set of playtraces satisfying a given query. The visual skein interface makes it easier to read and write DendryScope queries than working from scratch. We foresee the narrative designer loading their own Dendry story, but for evaluation purposes, we precompiled *Bee* from Dendry to ASP.

Visualizing ASP Queries as a Skein

Upon loading their story in DendryScope, the designer will see a *skein* corresponding to their QBN, similar to Fig. 4. In our demo, this skein contains the first 20 steps of playtesting results for the empty query (i.e. 'What passages can I see?'). From here, they may relax the time horizon (as we chose an arbitrary value, to avoid timing out on long QBNs), or refine the query by directly manipulating storylets and steps in the heatmap.

The skein is represented by a heatmap whose cells correspond to the timestep vs. current storylet. Each cell counts the number of concrete playtraces that have seen it, and more playtraces corresponds to a darker blue (Fig. 5). This number can be zero, because cells may be observed by brave enumeration without corresponding to a concrete playtrace.

Because scenes (and all storylets therein) are automatically sorted by the first time they are observed, the skein appears linear where the player is certainly in a given scene at a given time (i.e. where the story volume is constricted), and appears diffuse where there are many possible passages the player could be in.

The initial skein corresponds to the empty query, returning all possible playtraces. Adding terms to the query produces subskeins (Fig. 6), corresponding to a more specific scenario. Each DendryScope query produces a fixed skein, up to four tuning parameters: a timestep horizon, a maximum number of constructed playtraces, a maximum iteration count on bravely-enumerated playtraces, and a timeout.

DendryScope includes graphical tools for browsing scenes by their tags, and for viewing complete scene trees. In the scene tree widget, storylets whose title is highlighted in blue were visited by a concrete playtrace in the current query; storylets highlighted in grey are possible and have been enumerated, but not constructed. Storylets that are not highlighted cannot be reached given the current query.

Visualizations of State across Related Playtraces

Each cell in the skein corresponds to a certain passage at a certain timestep in a set of visiting playtraces, but each of these playtraces has its own world state. In the skein, we broke out the 'current passage' as the most salient element of world state, but many more visualizations of this dataset are possible. For example, the narrative designers we interviewed wanted to know what values could be attained by specific qualities at various cells in the skein.

The preliminary state visualization widget in DendryScope is a heatmap of quality versus value, describing the final observed state of each playtrace that touches the given cell. Although it is clearly more intuitive to produce the state of each playtrace as observed at the given timestep, the time cost of writing all state at all timesteps of all playtraces was excessive in our naive implementation. Besides, the AST of the IDN provides enough information to reconstruct (up to an arbitrary timestep) the state of every playtrace from its ordering of passages, although we did not implement this function.

Designer Tasks

We conducted a data-first design study (Oppermann and Munzner 2020) to identify narrative designers' existing playtesting strategies, taking *Bee* itself as our dataset (shown in Figure 5). *Bee* is a QBN written by Emily Short in 2012, which follows the life of a home-schooled girl who is training for a spelling bee. Designers used the prototype DendryScope interface (Fig. 1) to read skeins and write queries according to their interest.

We recruited five narrative designers for these interviews, using a snowball sampling technique. All five narrative designers (including co-authors) were familiar with reading sculptural hypertext in videogames such as Fallen London and Hades. Each designer has written games in languages that implement sculptural hypertext, including ChoiceScript, Inform 7, and languages of their own design. Five of the six designers were familiar with *Fallen London* (Failbetter Games 2009), a long-running work of QBN which Emily Short has worked on. Two of the designers were familiar with *Bee* itself prior to the study. A mixture of three academic programs and three industry backgrounds are represented across the five designers.

One designer became a co-author on the paper, so their interview was omitted from coding, although their insights remain in this paper. By coding the remaining four interviews, we identified seven tasks that narrative designers perform while writing or revising IDN.

Narrative Designer Tasks:

T1: Interpret the game's source code as a *story volume*.

T2: Answer story progression questions about the story volume. Is this passage reachable?

T3: Model player behavior as it influences the story volume. Are there rare or difficult-to-reach passages that might become player goals?

T4: Identify subsets of playtraces that correspond to certain player knowledge, or lack thereof.

T5: Understand how subsets of playtraces correspond to the query, and rewrite the query if it does not match the intended scenario.

T6: Discover how long it can or must take the player to reach a certain passage, including passages near the end of long stories.

T7: Develop flexible and powerful authoring patterns, such as *scenes* and other branching structures; or *menaces* and other kinds of qualities.

Evaluation

Our semi-structured interview process was designed to gather information about the four narrative designers' experience working with DendryScope for the first time. We wanted to understand whether the underlying ASP solver could be controlled by designers without prior solver-specific experience, through their own domain knowledge and our data-driven interface.

The first 10 minutes of each interview introduced *Bee*. The interviewer then asked the designer to describe structural features of *Bee* in Dendryscope's skein view (Figure 5), up to and including endings of the game. The interview takeaways below are organized according to the seven tasks we identified from the interviews.

T1 - Reading the Story Volume

The DendryScope skein visualization is "like an extra sense," said one designer. Another designer reported, "The tool might be able to surface combinations of things much more easily than a human could, because of the sheer number of combinations possible. The exhaustive approach that a machine can take allows you to see the possibility space way easier than just playtesting it."

In this paper, we have treated traditional playtesting as a form of search over skeins. In practice, experienced designers described playtesting using language about the player experience. "If I get here, what are the qualities I need to have had?" We found that, regardless of what combinatorial framework or experiential goal is being interrogated by a particular narrative designer, the skein allowed us to discuss both specific passages in the context of the game, and possible ways in which players could have reached those moments.

T2 - Answering Story Progression Questions

We found that the designers were not only anticipating specific routes through the game, but also that they were interested in supporting variation along any given route. "I almost want to be playing [...] whatever the normal way to do it is, and then having this as some kind of 'what could have happened' off to the side?"

One of the designers recounted their process in establishing the consequences of early player choices. They understood one such mechanic as a quality that can build past a certain threshold, and were interested in discovering what values that quality can possibly attain at certain points in the story.

> "Let's imagine you were writing something where the player was able to take on a role, but there were a couple of different roles possible. If you had challenges which you could solve by punching your way through or talking way through. Then, seeing it's not actually possible for my punching score to be very high in this area. I didn't know that [either] I shouldn't have had this choice here, or I should have given more opportunities to raise [punching] earlier."

This designer is interested in not wasting authoring effort: if a scene in the story responds to a quality past a certain

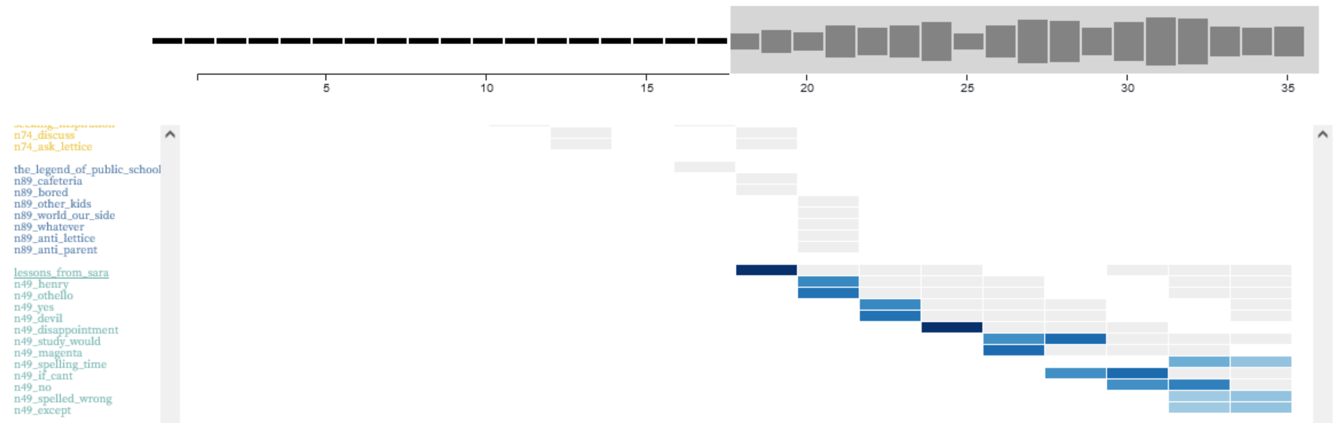

Figure 5: Screenshot of the DendryScope visualization of a skein in Emily Short's *Bee*. The user has clicked on the passage `lessons_with_sara`, setting it as the *goal*: it must be seen in each playtrace matching this query. This skein has been zoomed into steps 18 – 30 using the histogram (counting distinct passages at each timestep) on top. Gray cells show all possible passages at each step across all playtraces. Cells highlighted in blue have been observed in a concrete playtrace, with a deeper blue for more playtraces hitting that cell. The expressive range (Smith and Whitehead 2010) of a family of playtraces hitting a certain cell can be inspected by hovering over it.

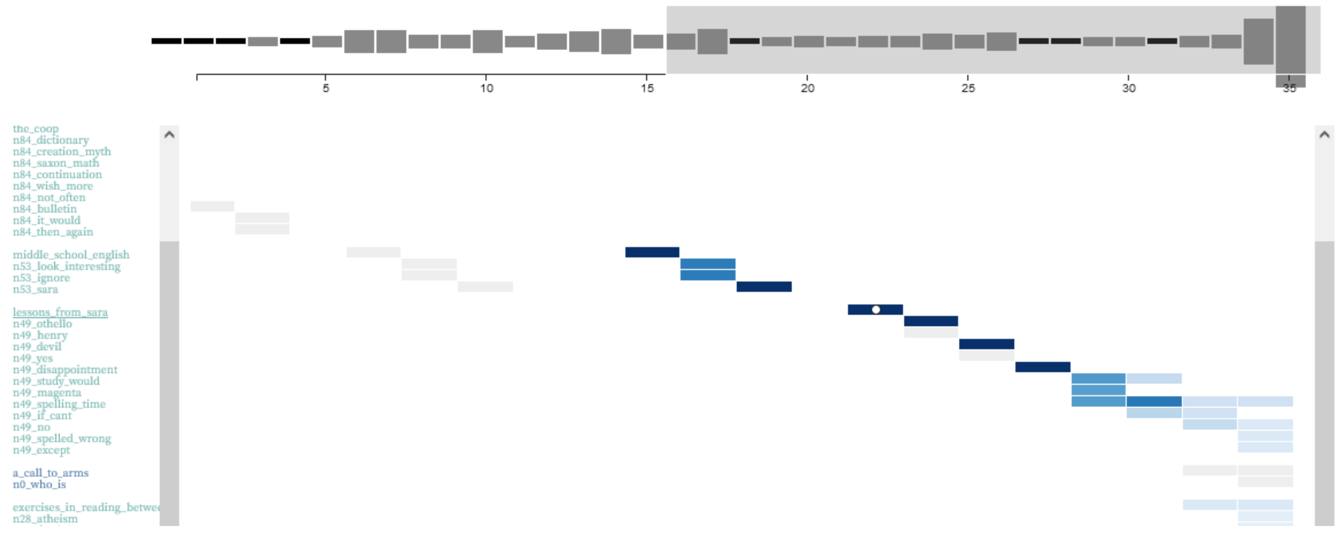

Figure 6: The result of refining the previous query (Fig. 5) by *pinning* the `lessons_with_sara` visit to step 28 (rather than step 27, the earliest possible moment). The other two scenes visible in this part of the skein have changed. Moreover, the 'cloud' of every other step at which we could enter the scene has now vanished. We confirm that `lessons_with_sara` begins linearly, and branches near its end. The skein fans in toward `lessons`, and fans out from it.

threshold value, then they want to make sure that threshold is actually attainable, so that the branch is reachable. They said later, "It's kind of a mystery at any given point in the game, what [the range of values for this quality] might be. Being able to actually visualize that looks wonderful."

T3 - Modelling Player Behavior

The designers understood that DendryScope queries are not themselves knowledgeable about player intent, unlike statistical models of players. One designer asked us, "There's going to be a disconnect between what the tool is modeling, and how players actually would play through, right?"

The purpose of the DendryScope interface, therefore, is to re-incorporate the narrative designer's awareness of what goals players might seek or be guided toward in the story volume into their DendryScope query. We do not claim that visual interfaces must be used in every context that involves a query and its skein; but in the context of an IDE, they are a useful way to orient the designer in the story volume.

T4 - Discovering Structure in Skeins

DendryScope supports the designer by visualizing the skein of a query. Sculptural hypertexts can contain both branching structure and complex multicursal structures, similarly to IF games like *Hadean Lands* - and scenes can vary over time, due to the complex world model. One designer explained, "You can imagine a game where there was more of a puzzle element to it and you could loop around a bunch of times before figure out the option to move forward."

Another designer described their reasoning while creating a query: "And there are two opportunities, three, actually. [...] So if I pin something extremely late - it's going to give me just a little bit of variation."

"I think it really works like a cloud of possibilities. I said that that was quite explicit here, where what we saw before was quite a lot of different possible nodes. But now once we've reduced the [scope to] what we actually wanted to see, it linearized a lot."

T5 - Discovering Missing Playtraces

Designers were not only able to reason about the presence of playtraces which they did not anticipate, but also about the absence of playtraces which they did expect to find.

Describing the `doll` quality in *Bee*, one designer talked through writing a query for a certain absent playtrace, which the DendryScope skein is designed to prevent (because proving nonexistence is expensive, and yields little information):

> "I was thinking about when we did the doll thing earlier if I could pin, or like customize a pin, say I want [the doll] to happen at the 12th time step. And then try to run that. Right now I can't click it because [it] didn't do it. But okay, we knew that doll could happen earlier. So, if it just isn't possible, it would just give me, that's not possible. This is a deliberate interface constraint, right?"

Another designer described a hypothetical scenario in which an abundance of opportunities to increase some quality caused some content to be seen too often. They would then want to write queries that show the passage is rare:

> "Let's say you have a choice which is supposed to be possible, but only really possible if you've pushed your choices in a certain way beforehand. But let's say you actually just balanced it wrong, and everyone can reach that, or 75% of the time you can reach that."

T6 - Dealing with Long Stories

Although DendryScope solutions are complete, in practice, large story volumes with 30+ steps of lookahead slow down the solver. As queries reach 10+ seconds of runtime, the designer's process of exploration is clearly impeded. We tried to mitigate this issue with timeouts, but these produce truncated sets of solutions, which can be misleading.

We found out that some queries take too long to run, and interrupting runaway ASP is an important step in debugging. While this is a natural action in command-line IDEs, we did not have adequate support in the DendryScope interface during the pilot. When a query contains a contradiction, the interface should eventually time out; but the ideal threshold might be a minute, or it might be ten seconds, depending on the author's debugging strategy. For longer response times, we expect that both a fast approximate result and a time estimate for the full result would be useful.

One designer asked, "Is there any way to get real-time feedback from the ASP, so that you could be populating the view [as the solver runs]?" Because combinatorial explosion is intrinsic to sculptural hypertext, as a medium of orderings, we do not anticipate that all performance issues with large DendryScope queries can be optimized away. Therefore, it is important to learn from designers how they reason about and use 'poison' in queries to block off irrelevant kinds of playtrace.

T7 - Developing Authoring Patterns

In the course of understanding what qualities are used by Short in *Bee*, our designers identified certain qualities with mechanics from tabletop storygames that gate certain forms of progress. "It's Blades in the Dark that has [Countdowns], it's like the concept of [the spelling quality]. [...] A number of different storylets could progress that clock [...] when that clock hits 12 it unlocks new possibilities."

Whilst `spelling` is a progress quality in *Bee*, Short has also included menace qualities like `parents` (which increases when the protagonist defies her parents) and `lettice` (which increases when she clashes with her sister), that can lead to alternate endings of the game. Like in tabletop games, it may be more narratively satisfying to end *Bee* in a 'loss' of this kind; not only is the protagonist not going to win the spelling bee, but she is also going to grow apart from her family.

Short also includes thematic qualities such as `worldliness`, which increases as the protagonist gains experience with places outside of homeschooling (like the hair salon); and `poverty`, which increases as the protagonist discovers it is unusual to, say, barter homemade jam in exchange for pantry items. These qualities do not generally impact the reachability of any scenes (only certain choices), but instead tend to reward and motivate reflective choices which are thematically linked.

Findings

We divide the research contributions of this paper thus: with implications for close readings of IDN on the one hand, and implications for authors of IDN on the other. Either way, the contribution of symbolic analysis is to support the narrative designer by visualizing the skein. Once visualized, this structure becomes more tangible to novices, and easier for experts to compare with their own intuitions. Ultimately, we view queries on QBN (and other forms of IDN) as a promising domain-specific language for future development.

The medium of ordering Ultimately, the passages in a sculptural hypertext are fixed; what varies is which ones the player encounters, and the order in which they appear. Ordering can constitute either diegetic or extra-diegetic choice, in the terms of Peter Mawhorter's choice poetics (Mawhorter et al. 2014); which one can be ambiguous. For example, Aaron Reed recounts (Reed 2017) his playthrough of Yoon Ha Lee's sculptural hypertext *Winterstrike* (Lee 2012) in terms of progressing the `Ice` quality to reach a certain tragic ending:

> "I will need to build my own story of how I became the person who had enough Ice to reach this particular goal, out of the various opportunities in the story world [...] Perhaps if I keep investigating, I'll find better ways to increase this stat — better, for my own personal sense of my character and their morality."

Junius et al. call for research into the ordering of scenes as a mechanism of dramatic agency (Junius, Mateas, and Wardrip-Fruin 2019), describing many techniques used by theater practitioners to modify a fixed script into a dramatic experience for the performer - one that is rich with awareness of the moment, and all the meanings it could have. Like theater performers, IDN players do not have the power of an author over the story, but rather the power of interpretation.

We propose that an IDN is 'reactive' (Mason, Stagg, and Wardrip-Fruin 2019) when the player takes part in the juxtaposition of dramatic elements. Narrative designers do not only reason about IDN in terms of state and reachability, although these are powerful tools afforded by sculptural hypertext formalisms, but simultaneously about narrative causality and player agency.

For example, in *Bee* the player character is introduced to an English tutor named `sara`, after much debate by their `parents`. One of the first things they can ask Sara is whether she is really a feminist, which Sara takes care to explain is not actually a bad thing. Later, if the protagonist should run away from home (after tensions with their `parents` reach a breaking point), one of the characters to whom they can go is `sara`. This bespoke scene can occur only if the player has met her (i.e. the value of the `sara` quality is nonzero), and if they think this is the right choice for the playthrough.

In-situ playtesting DendryScope is designed to assist authors in maintaining their mental map of a story volume, given the source code of their game. A future, dedicated Dendry IDE should include features like syntax highlighting, scene and choice stub generation, and line highlighting upon compiler error, as well as access to the the queryable DendryScope skein.

Storing queries We were asked to implement query saving features by one of the narrative designers. Without saved queries, there is no way to revert from a failed query (i.e. which produces an unexpected contradiction) to a good one. Moreover, query saving would double up as query sharing, as DendryScope queries (like raw ASP predicates) can be encoded as URL argument strings.

Enhancing queries We envision pinning (Figure 6) as a strategy for guiding sampling within the story volume, producing colorful 'stains' of the skein corresponding to storylines within the IDN. Multiple colors of pin could represent divergence within a single skein. Also, pinning could be implemented at the level of scenes (rather than passages), allowing the relative ordering of scenes to vary without excluding alternative scenes of different length.

Conclusion

Our work demonstrates how symbolic artificial intelligence can be applied to the story volume of works of IDN, and help narrative designers address their authoring problems. Conversely, we see this work as connecting narrative design expertise to the AI research tradition of plot generation, through the medium of sculptural hypertext.

DendryScope translates the designer's intent to examine a portion of the story volume, through direct annotation of the skein representation, into a suitable ASP query over playtraces. We have described 'goals', 'poison', and 'pin' constraints in this paper. In the future, other types of queries can be developed in terms of ASP to extract more nuanced information from a given QBN.

From a practitioner's perspective, we aim to raise awareness among narrative designers of symbolic analysis as a debugging technique. It is easy to start writing queries in DendryScope: all five narrative designers we interviewed could develop simple queries after exploring the interface for about 30 minutes.

We hope to develop DendryScope further into a practical tool with integrated script authoring, built-in detection of unreachable scenes and critical-path scenes, and useful debugging feedback on complex queries. We believe that designers would readily adopt an IDE for sculptural hypertext with a playtesting capability that is as accessible as Twine and as powerful as PlotEx.

Acknowledgements

We thank Bjarke A. Larsen, Leonardo Abate, Joey D. Jones, and Maxwell Joslyn for contributing their time and expertise as narrative designers.

References

Bernstein, M.; Millard, D. E.; and Weal, M. J. 2002. On writing sculptural Hypertext. In *Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*, HYPERTEXT '02, 65–66. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-58113-477-3.

Butler, E.; Smith, A. M.; Liu, Y.-E.; and Popovic, Z. 2013. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, 377–386. St. Andrews Scotland, United Kingdom: ACM. ISBN 978-1-4503-2268-3.

Cardona-Rivera, R. E.; Zagal, J. P.; and Debus, M. S. 2020. Narrative Goals in Games: A Novel Nexus of Story and Gameplay. In *International Conference on the Foundations of Digital Games*, 1–4. Bugibba Malta: ACM. ISBN 978-1-4503-8807-8.

Deterding, S.; Hook, J.; Fiebrink, R.; Gillies, M.; Gow, J.; Akten, M.; Smith, G.; Liapis, A.; and Compton, K. 2017. Mixed-Initiative Creative Interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 628–635. Denver Colorado USA: ACM. ISBN 978-1-4503-4656-6.

Failbetter Games. 2009. Fallen London. Digital game accessed 2023-09-01. Available online at: fallenlondon.com.

Garbe, J.; Reed, A. A.; Dickinson, M.; Wardrip-Fruin, N.; and Mateas, M. 2014. Author Assistance Visualizations for Ice-Bound, A Combinatorial Narrative. *Foundations of Digital Games*.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1): 27–82. Publisher: Cambridge University Press.

Green, D.; Hargood, C.; and Charles, F. 2021. Use of Tools: UX Principles for Interactive Narrative Authoring Tools. *Journal on Computing and Cultural Heritage*, 14(3): 1–25.

Grinblat, J.; Manning, C.; and Kreminski, M. 2021. Emergent Narrative and Reparative Play. In Mitchell, A.; and Vosmeer, M., eds., *Interactive Storytelling*, volume 13138, 208–216. Cham: Springer International Publishing. ISBN 978-3-030-92299-3 978-3-030-92300-6. Series Title: Lecture Notes in Computer Science.

Jones, J. D. 2022. Authorial Burden. In Hargood, C.; Millard, D. E.; Mitchell, A.; and Spierling, U., eds., *The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives*, Human–Computer Interaction Series, 47–63. Cham: Springer International Publishing. ISBN 978-3-031-05214-9.

Junius, N.; Mateas, M.; and Wardrip-Fruin, N. 2019. Towards expressive input for character dialogue in digital games. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 1–11. San Luis Obispo California USA: ACM. ISBN 978-1-4503-7217-6.

Karth, I.; Junius, N.; and Kreminski, M. 2022. Constructing a Catbox: Story Volume Poetics in Umineko no Naku Koro ni. In Vosmeer, M.; and Holloway-Attaway, L., eds., *Interactive Storytelling*, volume 13762, 455–470. Cham: Springer International Publishing. ISBN 978-3-031-22297-9 978-3-031-22298-6. Series Title: Lecture Notes in Computer Science.

Kreminski, M.; and Wardrip-Fruin, N. 2018. Sketching a Map of the Storylets Design Space. In Rouse, R.; Koenitz, H.; and Haahr, M., eds., *Interactive Storytelling*, volume 11318, 160–164. Cham: Springer International Publishing. ISBN 978-3-030-04027-7 978-3-030-04028-4. Series Title: Lecture Notes in Computer Science.

Lee, Y. H. 2012. Winterstrike. Digital game accessed 2023-09-01. Available online at: winterstrike.storynexus.com.

Mason, S.; Stagg, C.; and Wardrip-Fruin, N. 2019. Lume: a system for procedural story generation. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 1–9. San Luis Obispo California USA: ACM. ISBN 978-1-4503-7217-6.

Mateas, M.; and Stern, A. 2005. Structuring Content in the Façade Interactive Drama Architecture. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 1(1): 93–98. Number: 1.

Mawhorter, P.; Mateas, M.; Wardrip-Fruin, N.; and Jhala, A. 2014. Towards a Theory of Choice Poetics. *Foundations of Digital Games*.

McNutt, A.; Outkine, A.; and Chugh, R. 2023. A Study of Editor Features in a Creative Coding Classroom. ArXiv:2301.13302 [cs].

Millard, D. E.; and Hargood, C. 2021. Hypertext as a Lens into Interactive Digital Narrative. In Mitchell, A.; and Vosmeer, M., eds., *Interactive Storytelling*, volume 13138, 509–524. Cham: Springer International Publishing. ISBN 978-3-030-92299-3 978-3-030-92300-6. Series Title: Lecture Notes in Computer Science.

Millington, I.; and Chen, A. 2015. Dendry. Software accessed 2023-05-25. Available online at: github.com/aucchen/dendry.

Mitchell, A. 2022. Writing for Replay: Supporting the Authoring of Kaleidoscopic Interactive Narratives. In Hargood, C.; Millard, D. E.; Mitchell, A.; and Spierling, U., eds., *The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives*, Human–Computer Interaction Series, 131–145. Cham: Springer International Publishing. ISBN 978-3-031-05214-9.

Moritz, D. 2022. Hello Clingo. Software accessed 2023-05-26. Available online at: observablehq.com/@cmudig/clingo.

Nelson, G. 2006. Natural language, semantic analysis, and interactive fiction. *IF Theory Reader*, 141(99): 104.

Oliver, E.; and Smith, A. M. 2018. Twinealyzer: Static Analysis for Twine Games. Software accessed 2023-05-16. Available online at: twinealyzer.org.

Oppermann, M.; and Munzner, T. 2020. Data-First Visualization Design Studies. *arXiv:2009.01785 [cs]*. ArXiv: 2009.01785.

People Make Games. 2020. The System Behind Hades' Astounding Dialogue. Video essay accessed 2023-05-26. Available online at: youtube.com/watch?v=bwdYL0KFA_U.

Plotkin, A. 2012. PlotEx: a tool for exploring puzzle plot constraints. Software accessed 2023-05-26. Available online at: eblong.com/zarf/plotex/.

Reed, A. 2017. *Changeful Tales: Design-Driven Approaches Toward More Expressive Storygames*. Ph.D. thesis, UC Santa Cruz.

Shibolet, Y.; Knoller, N.; and Koenitz, H. 2018. A Framework for Classifying and Describing Authoring Tools for Interactive Digital Narrative. In Rouse, R.; Koenitz, H.; and Haahr, M., eds., *Interactive Storytelling*, Lecture Notes in Computer Science, 523–533. Cham: Springer International Publishing. ISBN 978-3-030-04028-4.

Short, E. 2012. Bee. Digital game accessed 2023-05-25. Available online at: ifdb.org/viewgame?id=8pe83e92v4nvabic.

Smith, A. M.; and Mateas, M. 2011. Answer Set Programming for Procedural Content Generation: A Design Space Approach. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3): 187–200. Conference Name: IEEE Transactions on Computational Intelligence and AI in Games.

Smith, A. M.; Nelson, M. J.; and Mateas, M. 2010. LUDOCORE: A logical game engine for modeling videogames. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, 91–98. Copenhagen, Denmark: IEEE. ISBN 978-1-4244-6295-7.

Smith, G.; and Whitehead, J. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 1–7. Monterey California: ACM. ISBN 978-1-4503-0023-0.

Veloso, C.; and Prada, R. 2021. Validating the plot of Interactive Narrative games. In *2021 IEEE Conference on Games (CoG)*, 01–08. ISSN: 2325-4289.