Intentionally Generating Choices in Interactive Narratives

Michael Mateas and **Peter Mawhorter** and **Noah Wardrip-Fruin**
Computer Science Department
University of California Santa Cruz
Santa Cruz, CA 95064 USA
{michaelm, pmawhorter, nwf}@soe.ucsc.edu

Abstract

Interactive stories face a famous "authorial bottleneck." Two existing approaches to this problem are story management systems, such as drama managers, and interactive narrative generators. Existing work leverages well-understood qualities of linear narrative such as suspense to generate content, but interactivity brings new capacities, like the ability to make a player experience regret. These interactive poetics arise from the player's ability to make choices, and depend heavily on the structure of the choices that are presented to the player. This system description paper presents a system that creates choices by reasoning about their structure, and describes the architecture that enables it to do so.

Introduction

Since the 1970's, researchers in artificial intelligence have been making systems that can creatively generate stories (Klein et al. 1971). With the rise of digital games, and in particular, interactive narratives[1], this research has found a new application: generating and managing the complexities of interactive narratives. One approach to this problem is to manage players' experiences. A managed experience lets authors create a diverse array of content while letting players experience a coherent narrative that includes different parts of the content depending on their choices. Another approach is to create systems that generate content, letting authors work at a more abstract level (perhaps writing re-combinable actions or events) which the system can then use to generate a wide variety of possible stories. Both approaches are proposed solutions to the fact that the work necessary to create a truly open world is overwhelming for human authors (Orland 2011). Existing systems have demonstrated the viability of reasoning about traditional narrative qualities for both experience management and story generation. Qualities unique to interactive narratives have not yet been widely used for reasoning in such systems, however. For example, the ability to make a player regret their own actions is unique to interactive contexts, and it depends on aspects of the narrative (such as which actions the player intended, and which outcomes were consequences of player actions) that go beyond traditional narrative qualities.

Interactive narrative systems thus stand to gain by reasoning about interactive as well as traditional poetics. Presented here is a system called *Dunyazad* that attempts just that: It dynamically builds choices with the goal of achieving specific poetic effects. *Dunyazad* focuses on choice poetics as a subset of interactive poetics, attempting to structure the choices that it gives the player so that they evoke feelings like safety or confusion (Mawhorter et al. 2014). As an operationalization of choice poetics, *Dunyazad*'s successes and failures can also inform the theory that drives it.

Liapis, Yannakakis, and Togelius recently stated that games were an ideal domain for computational creativity, and listed interactive narrative as an important part of that domain (Liapis, Yannakakis, and Togelius 2014). Human authors are now exploring the full potential of interactive narrative: Many independent games have earned praise for their stories, and communities that produce innovative interactive narratives have formed around tools like *Inform 7* (`http://inform7.com/`) and *Twine* (`http://twinery.org/`).[2] If generative narrative systems want to leverage the potential of interactive narrative, they will need to reason about interactive poetics, and in particular, how the choices they present to players are perceived.

Prior Work

In computational narrative systems, there has been a recent trend towards explicit poetics. Szilas' 2003 *IDTension* first proposed the idea of creating an interactive narrative by "simulating the laws of narrative" (Szilas 2003), much as one can produce a wide range of gameplay by simulating the laws of physics. This direction of work naturally proceeds by identifying the mechanism of specific poetic effects and building computational systems to produce those effects. El-Nasr's 2007 *Mirage* (El-Nasr 2007) is another example of this approach; it attempts to apply a range of dramatic techniques to increase engagement in an interactive narrative. In contrast, systems such as *Suspenser* (Cheong and Young 2006) and *Prevoyant* (Bae and Young 2008) have focused on specific poetic effects (suspense and surprise respectively).

Dunyazad as described here can be viewed as continuing this line of research because it reasons explicitly about poet-

[1] The authors are aware that interactive narratives predate digital games in several forms, but digital games have popularized interactive narrative as a medium.

[2] These are both examples of tools not explicitly designed to encourage creativity which nonetheless support it by making authoring faster and easier.

ics; it emphasizes interactive poetics, and in particular, the poetics of discrete choices. Whereas *IDTension* and *Mirage* incorporate traditional poetics into interactive experiences, *Dunyazad* focuses on interactive poetics, leveraging choice structures to create affect. In some respects *Dunyazad* is designed as much to illuminate interactive poetics as to exploit them: because it uses declarative code to construct poetic choices, its successes and failures can be traced to concrete parts of its theory, and that theory can thus be informed by the system's performance.

More recently, several studies have attempted to formally investigate and model poetic effects in interactive narrative contexts, with a focus on choices. In 2011, Thue, Bulitko, Spetch, and Romanuik measured players' perceptions of agency and found that they often differed from what one might expect based on the choices available to the player (Thue et al. 2011). Their system did manipulate an interactive narrative to achieve a poetic effect (give the player a sense of agency), but it focused on manipulating events in a way that was invisible to the player, rather than on changing a player's perceived options at any particular choice. In a study of agency which did not involve a generative system, Fendt, Harrison, Ware, Cardona-Rivera, and Roberts were able to create an illusion of agency, albeit in the context of an extremely simple interactive narrative (Fendt et al. 2012). A follow-up to the Fendt et al. study by Cardona-Rivera, Robertson, Ware, Harrison, Roberts, and Young linked players' perceptions of differences between outcomes to their perceptions of agency (Cardona-Rivera et al. 2014). In another paper focusing on choices in interactive narratives, Yu and Riedl were able to predict player choices using collaborative filtering (Yu and Riedl 2013).

This active research surrounding choices in interactive narratives shows that authors are interested in the poetic effects of choices. However, systems that actually reason about the poetics of the choices they generate are scarce– most existing systems reason about different options and outcomes independently. Barber and Kudenko's 2007 work on dilemma-based interactive narrative is a notable exception (Barber and Kudenko 2007). Their work focuses on a single type of choice, generating interactive experiences where each choice is a dilemma.

Ideally, a system that took choice poetics into account would dynamically construct each choice that it offers the player for maximum poetic impact. Of course, just as *IDTension* and *Mirage* reason about a range of classical poetics, such a system could take into account a range of interactive poetics (including aspects beyond choice poetics). But even a system that only considers choice poetics is a step in the right direction.

Choice Poetics

The theory of choice poetics described by Mawhorter, Mateas, Wardrip-Fruin, and Jhala in (Mawhorter et al. 2014) provides a framework for reasoning about choices, which is crucial for an interactive narrative system which must generate them. When analyzing the poetics of a choice, the first consideration is the player's mode of engagement: how is the player approaching the game, and what do they hope to achieve through their play? Common modes of engagement include power play (playing to achieve ludic goals like scoring points), avatar play (playing by projecting yourself into the game and making the choices you would make in a character's situation) and role-play (playing to express a particular role through the actions of one or more characters you control). There are also other less common modes of engagement like critical play, and players can (and usually do) engage with multiple modes at once. Taking modes of engagement into account does not require reading the player's mind, however: just as with any other element of a game, designers can make decisions based on their intuitions about how players will play, and they can refine their designs through playtesting.

Dunyazad directly encourages avatar play, and assumes that this will be players' primary mode of engagement when it constructs choices. Role play is also supported to some extent, but because there are only minimal game mechanics, *Dunyazad*'s stories do not lend themselves to power play. The game mechanics that do exist (skills which affect outcomes) are deployed in such a way that favorable outcomes from an avatar play perspective (which are favorable for the diegetic protagonist) are aligned with favorable gameplay outcomes (those in which the action attempted is successful, generally leading to successful endings).

Once a choice is considered in terms of a particular mode of engagement, it may fall into one of several classes of recognizable choice idioms, such as the dilemma or the false choice. Recognizing these idioms is based on an analysis of the framing, options, and outcomes of a choice (for example, a classic dilemma must have exactly two options, and the options' outcomes should each thwart a different player goal). Besides classifying choices as examples of choice idioms, (Mawhorter et al. 2014) does not say much about how to construct choices, although it does list some aspects of player experience that can be manipulated through the use of different choice structures. More analysis of existing interactive fictions within the framework of choice poetics would likely yield more specific methods for choice construction, however, and there is some existing advice on choice construction in the form of authoring advice for human authors of interactive narrators (e.g., (Choice of Games LLC 2010)). The choice construction methods in *Dunyazad* are currently based on this latter body of work, as described in the Choice Generation section on page 5.

Dunyazad

Although not yet complete, *Dunyazad* is a novel story generation system that is intended to generate interactive narratives in the style of *Choose Your Own Adventure* books using second-person narration and explicit choices. *Dunyazad* treats choices as first-class objects, and reasons about their structures. In particular, it has rules for constructing a variety of choice types based on the player's estimated expectations and evaluations in its choice structure module.

Dunyazad ultimately produces natural language narratives. Each story consists of sections of text followed by choices, where each choice leads to another section of text or to an ending. *Dunyazad* is not interactive, but instead generates

entire interactive narratives that players can interact with separately (importantly, this allows players to re-play parts of the narrative).

As you journey onwards, a leviathan rises majestically up from the ocean, tentacles curling. It is threatening you.

→ *You try to flee from it.*
– *You attempt to pacify it with music.*

You flee from it and escape. You travel onwards.

Figure 1: A minimal example vignette

Because *Dunyazad* is focused on operationalizing choice poetics, its default domain is simple travel/adventure stories in a fantasy setting, made up of sequences of relatively independent "vignettes" or scenes. Each vignette is made up of a setup plus a few basic actions, some of which may be player-initiated choices ("choice" nodes) and some of which may be events dictated by the system ("event" nodes). Each story node thus represents a single event or choice, including a context, one or more actions that might happen, and any outcomes of those actions. Although most world state is reset at the beginning of each vignette, the state of the player's party is not, allowing for some overall continuity. Figure 1 shows the full text of a minimal example vignette composed of a single choice (at which the player chose to flee), and a single event (the default vignette-ending event "travel onwards"). This vignette also introduces some new context at its choice node (an attacking monster) which is described in the text. Of course, when presented to the player, the text stops at the choice until the player has selected an option.

Although more complex vignettes are possible, the system is designed to create a stream of simple, direct choices, imitating the game *Spent* (`http://playspent.org/html/`) (McKinney 2011). By limiting the complexity of vignettes and refreshing most of the world state between vignettes, the user experience is directed towards shallow and playful interaction, and at the same time, the system has fewer opportunities to accidentally create plot holes. This also creates an environment in which the poetics of individual choices (e.g., was the last choice relaxing?) are important to the feel of the story overall, as opposed to merely supporting a dramatic arc defined by traditional narrative elements.[3] The system accordingly assumes that players will mainly engage in avatar play, and perhaps also light role play, and attempts to provide choices that enable these modes of engagement.[4]

From a technology standpoint, *Dunyazad* combines imperative Python code with declarative answer set programs to iteratively grow a branching story. The imperative code manages the iteration, at each step filling in single story node and adding new blank child nodes for each new option created. Filling in a node is accomplished by using the Potassco Labs tools `gringo` and `clingo` to ground and solve an answer set program (Gebser et al. 2011). The answer set program for each node includes predicates that represent the entire current story state, but facts from the solution to the program are only used to modify the currently-focused node.

After a complete story structure is created, *Dunyazad*'s imperative code uses a set of text templates to render the story into natural language. This module takes care of verb conjugation and pronominalization where necessary, and the text templates form a generative grammar which adds extra variation to the story. This variation doesn't change the underlying sequence of events, and mostly consists of word choice and sentence-structure variation that reduces literal repetition when similar events are described multiple times.

While *Dunyazad*'s hybrid iterative/declarative approach does limit the kinds of constraints that the system can easily place on multi-node story structures, it is necessary to keep the answer set problems tractable: Asking the solver to produce a complete story with hundreds of nodes in a single step is not a task that many modern computers could handle (if any), whereas just solving a single node can be accomplished in seconds. At the same time, being able to use answer set programming for the creation of individual nodes provides two benefits. First, answer set programming reasons simultaneously about all of its constraints, which means that building some logic which detects a certain condition also allows direct control over that condition (by e.g., prohibiting it or requiring that it hold). This means that there is little distinction between writing code which *recognizes* a phenomenon and writing code which *produces* it: the answer set solver does the hard work of figuring out what has to happen in order for the phenomenon to occur.

The second main benefit of using answer set solving is that it *directly* encodes constraints. *Dunyazad* as a project aims to apply choice poetics to the generation of interactive narrative, but it should also be able to push back on choice poetics when constructing choices based on theory fails to produce the expected results. Setting aside the difficult issue of blame assignment between the system and the theory, using answer set programming enables the system to better inform the theory because the constraints responsible for producing behavior can be directly translated into theoretical statements. For example, a rule like `regret(Choice) :- consequence(Choice, Outcome), bad_for_player(Outcome)` translates directly to a theoretical statement "When the player chooses an outcome that leads to something which is bad for them, they will feel regret." If testing reveals that players do not feel regret when the system thinks they should, the rule can be refined, and because it is a direct encoding of the theory, such refinement can directly inform the theory.

Representation

Although the output of *Dunyazad* is natural language, it has an underlying predicate representation of the stories it generates. Each story node describes either a choice or an event, and the structure of the two is the same, the only difference being events have only one option. Story nodes have a rich predicate representation of their initial state, which can encode arbitrary properties of and relations between story elements, including characters and items. Story nodes also

[3] In this case, the overarching plot of a journey to an exotic destination constrains little in terms of tension, narrative developments, etc.

[4] For more details about of modes of engagement, refer to (Mawhorter et al. 2014)

1. `st(root, inst(actor, monster_76)).`
2. `st(root, property(name, inst(actor, monster_76), "leviathan")).`
3. `st(root, relation(threatening, inst(actor, monster_76), inst(actor, you))).`
4. `at(root, action(option(1), flee)).`
5. `at(root, outcome(option(1), o(success, escape))).`
6. `at(root, outcome(option(1), o(get_injured, safe))).`
7. `at(root, arg(option(1), fearful, inst(actor, you))).`
8. `at(root, arg(option(1), from, inst(actor, monster_76))).`
9. `at(root_1, action(option(1), travel_onwards)).`

Figure 2: Some example predicates describing parts of fig. 1

have some number of options, each of which has an action associated with it, along with argument bindings for that action. Figure 2 shows some of the predicates that describe the example vignette in fig. 1.

Story states are sets of state predicates each of which takes one of four forms:

1. `st(root, inst(Type, ID)).`
   Declares the existence of a particular instance, which has a `Type` of either `actor` or `item`.
2. `st(root, state(State, Inst)).`
   Assigns a unary state such as `injured` to an instance.
3. `st(root, property(Prop, Inst, Value)).`
   Associates a property with an instance and specifies its value. For example, an actor can have the `has_skill` property with a value of `music` indicating that they possess the `music` skill. Properties can be multi-valued.
4. `st(root, relation(Rel, From, To)).`
   Asserts a relation between two instances. For example, an actor can have the `has_item` relation with an item. Some constraints (like exclusivity of the `has_item` relation) are enforced.

Frame axioms dictate that state changes only occur when specified by actions. Actions are defined by arguments, outcome variables, skill links, preconditions, and post-conditions as follows:

1. `argument(Action, Arg, Type).`
   Specifies an argument `Arg` which must bind an instance of type `Type` in the current state.
2. `outcome_val(Action, Var, Val).`
   Specifies that outcome variable `Var` can take on value `Val`. Each variable has multiple possible values.
3. `skill_link(Skill, Type, NeedsTool, Action, Arg, o(OutVar, OutVal)).`
   Skill links specify how character skills influence action outcomes. The four link types are `required`, `promotes`, `avoids`, and `contest`. These indicate player expectations. For example, the `healing` skill is linked to the `healed` value of the `success` outcome variable for the `treat_injury` action via a `required` link that also specifies that a tool is needed. Thus if the player lacks the `healing` skill and an option for them to take the `treat_injury` action is presented, the system assumes that the player will expect the action to fail.
4. Pre- and post-conditions. These have no fixed form, but instead are arbitrary logical constraints. For example, it is an error for the `treat_injury` action to be performed on a patient who is not injured. Most depend on outcome variables having specific values. Another example: if the `success` variable of a `treat_injury` action has a value of `healed`, then the `injured` state is removed from the patient, but if the `success` variable is either `still_injured` or `killed` this doesn't happen.

As an example, the text "You try to flee from it," in fig. 1 is a rendering of an action "flee" with the player character and the monster as arguments. The flee action has two outcome variables: `success` which has values `escape` and `failure`, and `get_injured`, which has values `injured` and `safe`. In fig. 2, facts 4 to 8 describe the action, outcome, and arguments of this option (the node that it is part of is `root`, and it is the first option at that node). Of course, each option at one node leads to another story node, and any consequences of the outcome associated with that option are reflected in the starting world state of the linked node. In fig. 1, the initial choice story node links to two successor nodes, only one of which is displayed.[5] In this case, the consequence of the successful flee action is that you have escaped the threat of the monster (which was part of the initial world state). The "travel onward" action's initial world state thus does not include the threat of the monster attack, which is actually a precondition for the "travel onwards" action–it requires an absence of "problems."

"Problems" and more generally "potentials" (which are either "problems" or "opportunities") are an important part of how the system builds stories. *Dunyazad* represents a "setup" as a partial world state which is added to the current world state when a new vignette begins. In fig. 1, the "monster attack" setup is used, which introduces a monster (in this case a leviathan) which is threatening the player-character.

[5] From a developer's perspective, all of the linked nodes are part of the same vignette, but of course without re-play, a player will only see one of them.

The fact that the player is being threatened (which is encoded as a relation) is explicitly recognized by the system as a "problem"–in fact all instances of the "threatening" relation are considered "problems," even when the player is not the target. This explicit representation of both problems and opportunities drives the basic consideration of what actions are appropriate in a given situation, and also plays into the rules about choice structures.

Reasoning

Dunyazad uses answer set programming to create the individual events and choices that make up a story. It is thus governed by a set of logical constraints which dictate what event configurations are acceptable. As already mentioned, solving for dozens of story nodes simultaneously is infeasible, because solving time is exponential in the number of nodes considered. As a compromise, *Dunyazad* iteratively solves individual story nodes.

Thus *Dunyazad*'s reasoning revolves around the construction of a single event or choice node. The rules governing node construction can be divided into three categories:

- Constructive rules–rules that help create the basic structure of facts, such as the rule that stipulates that each option has an action associated with it.
- Sense rules–rules that disallow nonsensical story structures, such as the rule that says that no choice should have two identical options or the rule that disallows trading items with oneself.
- Content rules–rules that discard some valid stories as uninteresting or otherwise undesirable, such as the rule that requires successive vignettes to use different setups.

Without constructive rules, core facts like those that assign values to arguments would be missing from result answer sets, and the system would crash. Without sense rules, all of the basic components of a story would be there, and the story could be rendered to natural language by the text generation system, but the result would be at best surreal and at worst gibberish. Without content rules, the result would be an understandable sequence of events, but it would probably not be an interesting story.

Because of the nature of answer set programming, *Dunyazad* effectively chooses an arbitrary permutation of an event among all possibilities that satisfy its rules (consult (Gebser et al. 2011) for more background on how answer set programming works). Each rule represents a constraint on the generative space of story nodes, which makes it both easy to prune the generative space, and easy to see how an individual constraint effects the generative space. The following variables determine the space of possible choice structures:

- The number of options (minimum 2 for a "choice" node; maximum 4 for performance reasons).
- The action for each option (there are 13 actions in the current domain model).
- The possible argument bindings for each action (most actions have 2-4 arguments, and each argument generally has 5-7 type-appropriate bindings at a given story state).
- The values for each outcome variable of each action (most have 1-2 outcome variables with 2 values each).

Unsurprisingly, there are a staggering number of possibilities under the constructive rules, but this space is reduced drastically by the sense and content rules.

Choice Generation

Dunyazad's design is in part based on concrete human advice for writing choice-based narratives offered by Choice of Games (an interactive narrative publisher) in several online articles (Choice of Games LLC 2010). In particular, *Dunyazad* focuses primarily on expectations and outcomes, which factor prominently in an article about the fundamentals of choice design titled "5 Rules for Writing Interesting Choices in Multiple-Choice Games," (Fabulich 2010).

Dunyazad's choice structure subsystem is devoted to estimating and managing the poetics of the choices it generates. Abstractly, this subsystem reasons about choices in terms of expectations and outcomes, using estimates of player perception. This same structure for representing and reasoning about choices could be used by other systems that wanted to generate choices intentionally.

The most basic structure of *Dunyazad*'s choice representation has already been described: a choice consists of *context*, *options*, and *outcomes*. *Context* in this case is a world state, *options* correspond to discrete, fully-specified actions that the player-character can take, and *outcomes* are the changes in world state that result from a particular action. To actually reason about the poetics of a choice, however, the system needs to make some assumptions about the player's experience, which gives rise to three more entities: *player goals*, *player expectations* and *perceived outcomes*.

Player goals are the basis for reasoning about how players might perceive choices. Some basic player goals can be predicted by the author, and to the extent that players actually pursue these goals, an author can design choice poetics. For example, an author might presume that players will want to keep their character alive and healthy, and that players will also want to maintain the health of their allies. A choice where the player is forced to sacrifice either their character's health or the health of an allied character could then be constructed with the goal of adding to the player's sense of tension. If players do in fact value their character's health and that of their allies, the choice should be a tense one (other details of its construction notwithstanding). For players who don't value one of these goals, the choice will lack the tension that the author intended, but that doesn't mean that the author's strategy for creating a tense moment was invalid. The author also has methods for encouraging the pursuit of various goals, such as using standard narrative techniques to try to promote empathy with the characters.

Dunyazad relies on the same strategy as this hypothetical author to create choice poetics: through its fixed introduction segment and according to genre conventions, it encourages players to pursue certain goals. It then estimates the poetic effects of the choices it creates assuming that the player will be invested in those goals. *Dunyazad* assumes are that the player will pursue the following goals:

- Avoid injury to themselves and their allies (high priority).
- Avoid threats to themselves and other non-aggressive characters (high priority).
- Have all actions they take be successful (low priority).
- Acquire and retain tools for their skills (low priority).

Given assumed *player goals*, a choice can be considered in terms of its *player expectations* and *perceived outcomes*. Both of these will vary from player to player, but just like with *player goals*, human authors can often estimate them. Like the *player goal* estimation, *player expectation* and *perceived outcome* estimation depends on the system author. This works via the skill link system as mentioned in the Representation section above: the author of an action specifies which skills are linked to which outcomes and how, and this information is used by the system to estimate player expectations. For example, if the player has a goal to maintain their health, but they're missing the `fighting` skill, an option allowing the player to `attack` an enemy will be marked as dangerous, because the `fighting` skill is linked to the `injured` value of the `aggressor_state` outcome variable, and that outcome would cause the player to be injured, threatening their goal.

As an example, consider the choice in fig. 1. This choice has two options, which correspond to the "flee" and "pacify" actions. Unbeknownst to the player (before they've made a decision at least), the outcome of the "flee" action in this case will be a successful escape, while the outcome of the "pacify" action (not shown) will be a failure that does not change the world state (i.e., the monster continues to threaten the player). While generating this choice, the system creates a *player expectation* for each option for each player goal, indicating how the player would expect that option to impact that goal. Taking "escape from threats" as a player goal, both options at this choice are expected to threaten that goal, because the system knows that both options could fail to achieve it. At the same time, both options are expected to enable that goal, because depending on their outcomes, either option could achieve that goal.

But is either option *likely* to succeed or fail? Assuming that the player has the "wilderness lore" skill (linked by a "contest" link to the "flee" action) but the monster does as well, the first option is indeterminate. However, based on a "required" skill link, if the player does not have the "music" skill, the second option is likely to fail.

There are thus five possible non-exclusive *player expectations* per *player goal*:

- Irrelevant–this option is irrelevant to this goal.
- Threatens–this option risks failing this goal.
- Enables–this option might achieve this goal.
- Fails–this option is expected to fail this goal.
- Achieves–this option is expected to achieve this goal.

Threatens and enables expectations are assigned based on all possible outcomes of an action, while fails and achieves are based on outcomes that the player has reason to believe are likely. Combinations of these expectations can describe a variety of situations. For example, a choice which threatens, enables, and fails a goal could be seen as a desperate gamble: it has a possibility of success, but it is expected to fail.

Similarly, there are five *perceived consequences* for each *player goal*:

- Irrelevant–this outcome does not affect this goal.
- Hinders–this outcome hinders progress towards achieving this goal, but does not actually cause it to fail.
- Advances–this outcome contributes to achieving this goal, but does not actually achieve it.
- Fails–this outcome directly fails this goal.
- Achieves–this outcome directly achieves this goal.

Unlike *player expectations*, *perceived consequences* are mutually exclusive, and while expectations only reason about the potential outcomes of an action, *perceived consequences* are assigned based on actual outcomes.

Returning to our example, the *player expectations* for the first option with regards to the goal "escape from threats" will include "threatens" and "enables," but since both the player and the monster have the associated contest skill, no stronger expectation is formed. For the second option, because the player lacks the relevant "music" skill[6], the *player expectations* will be "threatens," "enables," and "fails." The *perceived outcome* of the first option will be that it achieves the "escape from threats" goal, while the *perceived outcome* of the second option will be that it fails this goal.

This representation of player goals, expectations, and perceived outcomes enables rich reasoning about the poetics of a choice. To start with, it's easy to encode simple choice idioms (see (Mawhorter et al. 2014)). An example would be a dilemma–traditionally a choice with exactly two options which lead to two different negative consequences. A choice with exactly two options, each of which is expected to fail one of two goals and enable the other fits these criterion. The perceived outcomes here determine what kind of dilemma it is, for example a false dilemma might have identical outcomes for both options.

You come to a tavern and decide to rest for a while. A merchant is selling a music book and she is selling an oboe and a noble is bored and a peasant is bored and an innkeeper seems knowledgeable.

- *You play a song for the peasant. (+music)*
- *You gossip with him. (+elocution)*
- *You offer to trade the merchant some perfume for the music book. (no skill)*
- *You tell the noble a story. (+storytelling)*

Figure 3: A "relaxed" choice.

To give a more concrete demonstration of choice generation, consider figs. 3 and 4. These examples were generated

[6]Relevant to an actual player's expectations is whether they are *aware* of this lack. For now, *Dunyazad* ensures this by mentioning any relevant skills (or lack thereof) in parentheses after each option. These are omitted in fig. 1 to avoid confusion.

As you travel onwards, a dragon slowly approaches you. It is threatening you.

- *You attack it. (-combat)*
- *You attempt to pacify it with music. (-music)*

Figure 4: A "grim" choice.

using a single player goal based on the idea of power-gaming: "Succeed at every action." Evaluating expectations relative to that goal, fig. 3 is the result of asking for a "relaxed" choice: a choice where there are no options which the player expects to fail. Figure 4[7] is the opposite: a "grim" choice where every option is required to be expected to fail.

When generating individual choices, the system uses everything available to it (the choice of setups, the background including the player's starting skills, and the configuration of options and outcomes) to create choices that satisfy the given constrains, which can be expressed directly at the level of player expectations. This ability to reason about player expectations is critical in a system that wants to use choice structures to achieve poetic effects. Of course, the system isn't reasoning directly about the player's actual expectations, but merely about the system author's guess as to what those expectations will be. For human authors, this is often enough to achieve their goals, and for systems without dynamic player modelling, it will have to be enough as well.

Abstract Architecture

The underlying principles behind *Dunyazad*'s choice structure rules suggest requirements for systems that want to build choices intentionally. At a basic level, the ability to reason about all parts of a choice: the context, options, and outcomes, is required. It should be noted here that reasoning about the options individually is not sufficient: a system that wants to dynamically construct choices that create poetic effects needs to be able to reason about the range of options available at a choice and assert things like "There are no options available which the player expects will lead to positive outcomes." That brings up the next requirement: such a system needs to be able to reason about player goals, expectations, and perceptions. These things do not have to be modelled exactly as *Dunyazad* models them, but they should be represented in such a way that the system can reason about them.

In order to creatively construct a choice that gives the player a feeling of "agency" or "regret" or "power" a system needs to be able to define those things in terms of the player's view of the game. It is of course possible to give a player these feelings in an interactive narrative without representing them in any sort of system, but that just means that a human author has done the reasoning required, not that it never happened. And if a human author did that reasoning, then the system will not be able to freely generate such choices: it will be limited to generating them in situations that the human author was able to foresee.

[7] These examples were slightly edited from their original form for brevity.

So what is the next step once you have a system that reasons about all parts of a choice and the player's perspective besides? The next step is to identify the choice poetics that you want your system to create, and define them in terms of the player's perspective. For example, if you want the player to experience "agency," you might define your objective as "The system should create choices with multiple options that the player expects will lead to significantly different world states," as in (Cardona-Rivera et al. 2014). Given this concrete definition of your goal in terms of player expectations, the system should be able to construct such choices. If your goal is hard to pin down in terms of player expectations, playing some interactive narratives that create the feeling you want to create and analysing their choice structures would be the place to start.

While a computer-generated interactive narrative that successfully evoked a particular feeling using choice structures would be an achievement in its own right, even better would be a system that used multiple choice structures in service of a more complex goal. For example, if there are narrative generation mechanisms trying to achieve a desired tension level, making sure that choice structures are also contributing to this goal would be a benefit. Or if the player-character is supposed to be stumped by a mystery at some point in the plot, perhaps generating choice structures where there are no clear good or bad options could reinforce that point. The ability to craft choices towards poetic ends unlocks many new options for an interactive narrative system.

Future Work

Dunyazad is still under development, and getting it to generate full interactive narratives as opposed to individual choices is our current focus. Once it does so, it will be critical to evaluate the narratives it generates, both from a creative standpoint and to determine whether players actually perceive the effects that it is trying to create. If *Dunyazad* is able to create full interactive narratives with choices that support their stories, it will represent an important step towards narrative generators that take full advantage of interactive as well as traditional poetics. However, even if it only generates individual choices, *Dunyazad* still enables experiments that can contribute to knowledge of choice poetics.

Generating full stories will require significant authoring effort. *Dunyazad*'s current domain model has 13 actions, 6 potentials, 6 setups, and 4 player goals. In order to generate experiences even few minutes long, it would need to generate stories with dozens of nodes across perhaps 8-12 vignettes. The primary authoring effort to get to that point with a satisfying level of variety lies in creating more distinct setups, as well as adding a few more actions. Although actions, potentials, setups, and player goals are all modular from a technical standpoint and can be authored individually, practically they need to take each other into account in order to create interesting output. This is mostly a consequence of the content rules. For example, adding an action which results in a new state probably won't change the system's output by itself, because without any player goals or potentials that involve that state, the system will never consider the new action to be relevant. Actions, potentials, setups, and

goals thus form interconnected subsystems that are linked by certain key states. Although this makes authoring a bit tricky, these subsystems are at least somewhat independent of each other: the actions that involve trading items can be authored without worrying about injury and death.

Besides further work on *Dunyazad*, there are several promising research directions suggested by this work. First, the fact that an interactive narrative system is making assumptions about its players begs for a mechanism by which the system could actually measure its players. (Thue et al. 2011) is an example of exactly that, but by increasing both the complexity of choice manipulation and the detail of the player model things would become even more interesting. There is a link here to the world of intelligent tutoring system: systems like Graesser, Chipman, Haynes, and Olney's *AutoTutor* already have components that attempt to measure a student's knowledge and even emotions (Graesser et al. 2005). Adapting these to work in an interactive narrative context as opposed to a tutoring context would give the system a much better means of estimating a reader's expectations and goals than authorial guesswork.

Expanding a choice-poetics based system to deal with broader interactive poetics would be interesting too. Many games offer interactions much more complicated than discrete choices, and reasoning about these would be more difficult. Many of the same principles apply, however, and creating a system that analyzed complex interactive situations in terms of player expectations, available actions, and their consequences would allow the deliberate construction of more complicated and open-ended narratives.

Finally, a system capable of deliberate choice creation might enable new and more dynamic forms of narrative. This is an eventual aim of *Dunyazad*: to create a branching story with myriad paths where the freedom to explore a huge possibility space is enabled by collaboration between a human author and a computer system. If generative tools could enable human authors to design entire narrative possibility spaces, the resulting fictions would be the products of both human and machine creativity.

Acknowledgements

The authors would like to acknowledge the support of NSF grant IIS-1409992.

References

Bae, B.-C., and Young, R. M. 2008. A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. In *Interactive Storytelling*. Springer. 156–167.

Barber, H., and Kudenko, D. 2007. Dynamic generation of dilemma-based interactive narratives. In *3rd Artificial Intelligence and Interactive Digital Entertainment Conference*.

Cardona-Rivera, R. E.; Robertson, J.; Ware, S. G.; Harrison, B.; Roberts, D. L.; and Young, R. M. 2014. Foreseeing meaningful choices. In *10th Artificial Intelligence and Interactive Digital Entertainment Conference*.

Cheong, Y.-G., and Young, R. M. 2006. A computational model of narrative generation for suspense. In *AAAI*, 1906–1907.

Choice of Games LLC. 2010. "Game Design" posts. `http://www.choiceofgames.com/category/blog/game-design/`. Accessed 2015-2-27.

El-Nasr, M. S. 2007. Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories. *Interaction Studies* 8(2):209–240.

Fabulich, D. 2010. 5 rules for writing interesting choices in multiple choice games. `http://www.choiceofgames.com/2010/03/5-rules-for-writing-interesting-choices-in-multiple-choice-games/`. Accessed 2015-2-27.

Fendt, M. W.; Harrison, B.; Ware, S. G.; Cardona-Rivera, R. E.; and Roberts, D. L. 2012. Achieving the illusion of agency. In *Interactive Storytelling*. Springer. 114–125.

Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The potsdam answer set solving collection. *AI Comm.* 24(2):107–124.

Graesser, A. C.; Chipman, P.; Haynes, B. C.; and Olney, A. 2005. Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *Education, IEEE Transactions on* 48(4):612–618.

Klein, S.; Oakley, J. D.; Suurballe, D. J.; and Ziesemer, R. A. 1971. A program for generating reports on the status and history of stochastically modifiable semantic models of arbitrary universes. Technical Report TR142, Computer Sciences Department, The University of Wisconsin–Madison.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Computational game creativity. In *5th International Conference on Computational Creativity*, 285–292.

Mawhorter, P.; Mateas, M.; Wardrip-Fruin, N.; and Jhala, A. 2014. Towards a theory of choice poetics. In *International Conference on the Foundations of Digital Games*, FDG '14.

McKinney. 2011. Spent. Web, `http://playspent.org/html/`. Accessed 2015-2-9.

Orland, K. 2011. 'Old Republic' writer discusses '60 man-years' of work. `http://kyleorland.com/blog/2011/12/21/old-republic-writer-discusses-60-man-years-of-work/`. Accessed 2015-2-28; full article archived at `http://web.archive.org/web/20120206234922/http://ingame.msnbc.msn.com/_news/2011/12/20/9569126-old-republic-writer-discusses-60-man-years-of-work`.

Szilas, N. 2003. Idtension: A narrative engine for interactive drama. In *Technologies for Interactive Digital Storytelling and Entertainment Conference*, volume 3, 187–203.

Thue, D.; Bulitko, V.; Spetch, M.; and Romanuik, T. 2011. A computational model of perceived agency in video games. In *7th Artificial Intelligence and Interactive Digital Entertainment Conference*.

Yu, H., and Riedl, M. O. 2013. Data-driven personalized drama management. In *9th Artificial Intelligence and Interactive Digital Entertainment Conference*.