Curveship's Automatic Narrative Style

Nick Montfort
Massachusetts Institute of Technology
77 Massachusetts Ave, 14N-233
Cambridge, MA 02139
1.617.324.1429
nickm@nickm.com

ABSTRACT

Curveship, a Python framework for developing interactive fiction (IF) with narrative style, is described. The system simulates a world with locations, characters, and objects, providing the typical facilities of an IF development system. To these it adds the ability to generate text and to change the telling of events and description of items using high-level narrative parameters, so that, for instance, different actors can be focalized and events can be told out of order. By assigning a character to be narrator or moving the narrator in time, the system can determine grammatical specifics and render the text in a new narrative style. Curveship offers those interested in narrative systems a way to experiment with changes in the narrative discourse; for interactive fiction authors and those who wish to use of the system as a component of their own, it is a way to create powerful new types of narrative experiences. The templates used for language generation in Curveship, the string-with-slots representation, shows that there is a compromise between highly flexible but extremely difficult-to-author abstract syntax representations and simple strings, which are easy to write but extremely inflexible. The development of the system has suggested ways to refine narrative theory, offering new understandings of how narrative distance can be understood as being composed of lower-level changes in narrative and how the order of events is better represented as an ordered tree than a simple sequence.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *discourse, language generation.*

General Terms

Interactive storytelling

Keywords

Electronic literature, interactive fiction, text adventure games, interactive storytelling, narrating, narrative.

1. INTRODUCTION

Curveship was created to combine the successes of interactive fiction with the long literary tradition of varied narrative style. A very succinct description of the system is that it takes the contributions of the classic game *Adventure* by Will Crowther and Don Woods [1], adds to those the variation in style that is exemplified by Raymond Queneau's *Exercises de Style* [2], and enlarges the possibilities of interactive narrative. *Adventure* is the canonical first interactive fiction, simulating a cave with treasures and puzzles. Queneau's book contains 99 different tellings of the same story, which is rather boring in and of itself but comes to life through the play of narrative and other literary styles. One of the example fiction files that was released with the system, *Adventure in Style,* is a more or less direct combination of these two provocative works.

FDG'11, June 29-July 1, Bordeaux, France.

To understand the motivation behind Curveship, then, it is important to discuss the reason from creating an interactive fiction system and the reason for the focus on narrative variation.

1.1 Why Interactive Fiction?

Interactive fiction has been an important part of the landscape of compute culture since the min-1970s. It was one of the most popular diversions on time-sharing systems. Then, thanks mainly to Infocom, but also to the contributions of several other companies, it was once a dominant form of entertainment software in the early 1980s. Today, IF does not have much of a direct presence in the commercial game marketplace, but innovation in the form continues, thanks to the efforts of individuals and the availability of free development systems.

An interactive fiction is a type of virtual reality, or simulated world, presented in a textual interface. Although IF conventionally uses the otherwise unusual pronoun, "you," it is not constructed or operated like a Choose-Your-Own-Adventure book. Instead, players type short natural language commands, the result of each action is simulated, and the new situation is described in text. [3]

Interactive fiction aspires to have human-like dialogue in natural language, not command-line interaction. Rather than being just a riff on a largely outmoded interface, necessary for us to learn the textual interfaces of 1980s home computers, it is a model for how to interact with computers in a natural, semantically rich way.

Interactive fiction has developed good abstractions of the world that are effective for text-based exchange. It models aspects of the world that are important to exploration, figuring out the way a strange environment or system functions, and demonstrating an understanding of such unusual workings. With regard to narrative variation, Curveship was developed to take advantage of an opportunity that was not realized by existing interactive fiction systems. But in other regards, Curveship builds on the success of interactive fiction systems. Specifically, Curveship builds upon the useful interactive fiction representations that have been developed over the past 35 years, particularly relying on the way they were articulated by Graham Nelson in his documentation of his IF system Inform 6 [4].

The standard world model in interactive fiction, developed over thirty-five years in academic, commercial, and independent

contexts, is exemplary. There are a few particular problems that remain with simulating a world in IF. Good representations of rope, fire, and liquids (some of the same objects that happen to be non-trivial to render visually using computer graphics) are difficult to develop because rope can span locations, fire interacts with flammable objects, and liquids can be subdivided, poured on things to wet them, and used in other ways that have odd effects. More can be done to improve the way IF simulates, but the future of interactive fiction is not really being held back by the difficulty in simulating rope. The worlds that can be modeled effectively right now are rich and interesting enough to supply us with many different, powerful playing and reading experiences. While world simulation can be improved, it is not the bottleneck of IF development. Existing IF provides many good examples of how to represent different aspects of the world in different, appropriate levels of detail.

Similarly, the standard IF "parser" is far from perfect at understanding natural language, but, thanks in part to the grounding of IF in a simulated world and in part to convention, it accepts an effective subset of natural language. In the early 1980s, millions managed to figure out (with the help of manuals, friends, and trial and error) how to interact with IF. Just as Curveship's focus is not on improving the simulated world, it is also not on improving the parser.

Interactive fiction's existing capabilities for simulating a world and accepting natural-language-like input provide a good basis for this project, which seeks to add a new capability.

1.2 Why Narrative Variation?

The development of Curveship has focused on the level of narrative discourse for two reasons: First, because the level of narrating is essential to the power and effectiveness of stories; second, because modeling the story level has already been done effectively by interactive fiction.

The reason people find narratives powerful (whether they are movies, oral stories, novels, or in some other medium) is bound up in the way they are told. This may seem particularly clear in modernist fiction, such as William Faulkner's *The Sound and the Fury.* In novels like these, the unusual telling of the story is foregrounded and is often much more remarkable than the events themselves. But even in ancient stories such as *the Odyssey*, the way the story unfolds — with Odysseus hearing his own story sung to him by a bard, weeping and concealing his tears, being recognized, and then finally being coaxed to continue to tale himself — is extremely important to the effect of the narrative. Concern for the interesting qualities of the narrative discourse is not only seen in centuries-old classics or modernist and avant-garde writing. It can also be seen in Stephanie Meyer's initial drafts of chapters that retell the *Twilight* series from Edward Cullen's perspective.

A very direct demonstration of how interesting it can be to change the narrative discourse while leaving the underlying story unchanged is seen in Raymond Queneau's 1947 *Exercices de Style* (which was translated into English by Barbara Wright). In this extraordinary book, the same unremarkable story, about a minor confrontation on a bus and catching sight of one of the people involved later, is told in ninety-nine different ways. The book was the inspiration for Matt Madden's 2006 comic *99 Ways to Tell a Story: Exercises in Style,* which is based on a different but still rather uninteresting sequence of events, in which a character walks into a room, replies to someone upstairs by saying what time it in, and stares into the refrigerator wondering what he was looking for. Madden's book uses variations specific to the comic medium (for instance, telling the story in one panel and in 30 panels) in addition to incorporating changes in narrative style. These two books, by showing many variations side by side, make a clear case for how vital and expressive the narrative discourse can be.

As discussed in the previous section, there are already good ways to model that which exists (the "existents") in a simulated world. Developers and programmer/authors have worked to improve this world model further. For instance, in interactive fiction, as in gaming generally, much work has been done on creating better computer-controlled characters, who act within the story world. Almost no work, however, has been done on creating computer-controlled narrators, who relate the events within the story world in the narrative discourse. Authors have done a great deal to create particular interactive fiction works that embody excellent, interesting one-off narrators (Dan Shiovitz's *Bad Machine*, Admiral Jota's *Lost Pig*, and Jeremy Freese's *Violet* are examples), but until now, none of the general flexibility and power of the world model has been brought to narrating.

Curveship was created not to improve the way that characters act, not to facilitate more believable, lifelike or dramatically engaged characters, but to provide for the first time an array of expressive computer-controlled *narrators.* As is the case in the novel, narrators can be characters within the story (e.g., Marlowe in *Heart of Darkness,* Ishmael in *Moby Dick*) or not (as in, e.g., the unnamed narrators of *The Odyssey* and *Blood Meridian.*) The narrating and particular narrators constitute a different, orthogonal dimension from that occupied by characters. This emphasis on narrative variation is the major aspect distinguishing Curveship from other state-of-the-art IF systems such as Inform 7 and TADS 3, which have been developed with other directions (such as natural-language programming, rule-based programming, and multimedia support) in mind.

2. CURVESHIP ESSENTIALS

In creating a new Curveship game/fiction, an author's main task is to define what exists in the simulated world – the "existents" in the terminology of narrative theory. This means creating a fiction file, which has as its main components an items list and game-specific subclasses of the different subclasses of Item. The items list is converted on startup into a tree; each item except the root has its parent listed and the resulting graph is checked at runtime and set up if it is acyclic. Initial actions can also be specified in the fiction file. So can a "spin," a way of telling the story. It is also possible to apply a spin from a separate file.

Curveship was released under a free software license on February 2, 2011 (at http://curveship.com, where some of this information is available in a different form) with several example fiction files of different lengths and complexities. They range from *Artmaking,* a one-page fiction file that defines a world with one actor, one room, and two things to *Adventure in Style,* a complete implementation of the classic game *Adventure* with 96 different possible narrative styles that can be applied based on the in-game actions of the player character.

2.1 An Example of Interaction

In *Adventure in Style,* the player more or less directly controls the narrative style by turning a dial and flipping switches on the lamp. Generally, there is no restriction on what can change the narrative style in Curveship. In the example game *Lost One,* the player's commands influence the narrative style more indirectly. In the very simple *Artmaking*, there is no way to change the style except by selecting a spin file on startup or with debugging directives. The goal of Curveship is to effectively translate a narrative specification into a grammatical specification and then into realized text. Curveship does not dictate how that narrative specification is determined; that is up to the author/programmer.

Here is an example of interaction (the ">" texts were typed by the player) with *Adventure in Style:*

>get the lamp
You pick the lamp up.

>turn on the lamp
I turned the lamp on.

>examine the lamp
I took a look at the lamp.
A brass carbide lamp, the kind often used for illuminating caves. It was shiny and brightly glowing. It displayed the word MEMOIR and had three switches: a "HESITANT" switch, a "SURPRISE" switch, and a "VALLEY GIRL" switch. The lamp also featured a dial which can range from 1 to 12 and was set to 2.

>turn dial to 6
You select "RETROGRADE."
Just beforehand, you turned the dial to 6.
Just beforehand, you took a look at the lamp.
A brass carbide lamp, the kind often used for illuminating caves. It was shiny and brightly glowing. It displayed the word MEMOIR and had three switches: a "HESITANT" switch, a "SURPRISE" switch, and a "VALLEY GIRL" switch.
The lamp also featured a dial which can range from 1 to 12 and was set to 2.
That was after you turned the lamp on.
Earlier, you picked the lamp up.
That was after you took a look at the building's interior.
You were inside the building, the well house for the large spring.
You saw the keys, the food, the bottle, and the lamp. Water was in the bottle.

Initially, the narrative style is that of a typical interactive fiction and that of the original *Adventure*: The story is being told to the player character (the "you") and the narrator is speaking as if there while the events are transpiring (present tense). When the lamp is turned on, the narrative style switches to "MEMIOR." Turning the dial to 6 switches it to "RETROGRADE," so that recent events are related (with appropriate grammatical adjustments) in reverse order. The changes made by turning the dial do not affect anything else in the underlying, simulated storyworld – nothing except for the position of the dial. If an author/programmer wishes to have some actions affect both the narrative style and the simulation, that is easily done. But they are abstracted from one another in Curveship to allow for independent manipulation.

2.2 World & Concept

The **world** is the main, simulated universe of the interactive fiction, which defines the "reality" within which all of the characters live. One of the main things that distinguishes interactive fiction from hypertexts, conversational characters, and story and poetry generators is that they have a simulated world. So, this is a feature Curveship has in common with other IF systems, not one that distinguishes it from them.

A **concept** is a particular actor's theory of the world, based on knowledge (as initially represented in a fiction file) and perceptions (as experienced as the actor moves around and looks at things). Concepts are almost never complete, and they may be wrong. They allow the telling of actions and the description of items to be focalized, that is, to be restricted to what a particular actor knows and sees.

2.3 What Exists: Items

Item is the Curveship term for "existents" or "objects" – everything that exists in the simulated world. They must be in one of five categories, so that they are either Actors, Doors, Rooms, Substances, or Things.

An **Actor** is an item that can take action on its own, due to either code that an IF author has written or a script the author has dropped in. Any Item can react (that is, any Item can have its own react method defined) but only an Actor can initiate action (equivalently, only an Actor or subclass of Actor can have an act method). Each Actor has a concept; when an Actor acts, the action is put together using this concept. The player character is an Actor (or an instance of a subclass of Actor). Although particular fictions may allow or disallow it, any Actor has the potential to be commanded and focalized, becoming the player character.

A **Door** may actually be a door in the usual sense, or it may be a passageway or other portal that connects exactly two rooms. Doors, like Rooms, are all on the first level of the item tree, directly below the root node. Therefore, they are not children of either room which they connect, although the logic of visibility and accessibility assures that they are among the items that can be seen and accessed from both rooms. A Door can be understood as a Room that one can only go through, not remain in.

A **Room** is a discrete location which can have exits leading to other rooms or to doors. Rooms are all on the first level of the item tree, directly below the root node.

A **Thing** is typically an item that is not a location and is more or less inert. Any item that isn't a room and doesn't need to act or have its own concept is a thing. Things can react when something is done to them or done in the same room, so that pressing a switch on a lamp can cause the lamp to react by increasing its glow.

A **Substance** is something like a powder or liquid that can be poured into a vessel but can't really be carried around otherwise. Sources provide an endless supply of substances; vessels are used to contain them. It's not necessary to define any of the particular amounts of a substance as items; after defining an overall substance item and designating other items as sources or vessels, the rest is done automatically during setup.

The special Actor that is the root of the item tree is **Cosmos**. This Item is responsible for earthquakes, power outages, and any occurrence where the author doesn't want to model the cause as its own Actor. The Cosmos can also change the spin, which allows for a connection between the simulated world and the way the telling is done.

Curveship has different types of **parent-child relationships.** This is the relationship between an item and an item a level under it, and connected by a link, in the tree. When an actor walks into a room, he or she becomes the child of the room. If the room moves (perhaps because it is an elevator) the actor will move with it. If the lights in the room become brighter, the actor and everything else in the room will be better illuminated. An apple placed in a sack similarly becomes the child of the sack and, for instance, is itself stolen if the sack is stolen.

The item tree's edges are called **link**. Each one connects an item to another item and indicates a specific type of parent-child relationship. Links are labeled, with the label indicating more about the relationship. If a person is holding a sack with an apple inside and wearing a cloak, the sack and cloak are the children of the person and the apple is the child of the sack. Furthermore, the link between the sack and the person is labeled "of" (indicating a possession) and the link between the cloak and the person is "on"

(indicating something being worn). The link between the apple and the sack is "in" (indicating containment). This means it is straightforward to model a desk that you can put things *on* (on top of) and *in* (in a drawer): It is simply an Item that allows (via its allow method) some other Items to be in both sorts of relationships with it, using both types of links.

Typing **world tree** immediately after starting the Curveship version of the standard IF example *Cloak of Darkness* shows all of the items in this tiny simulated world and described how they are arranged:

```
@cosmos: nature []
    @bar: bar [of]
        @message: message [part_of]
    @cloakroom: cloakroom [of]
        @hook: hook [part_of]
    @foyer: foyer [of]
        @person: operagoer [in]
            @cloak: cloak [on]
```

Typing **concept @person tree** shows the operagoer's concept, which contains only those items that this player character, the only actor in this simple fiction, knows about:

```
@cosmos: nature []
    @foyer: foyer [of]
        @person: operagoer [in]
            @cloak: cloak [on]
```

In this case, it is a simple subset (or subtree), but actors can also be loaded with knowledge about the world that is incorrect. To see the item tree change, one can type "remove cloak" and then check either the world's or the operagoer's concept's item tree again. The cloak will be "of" the person (indicating possession) rather than "on" the person (indicating a garment being worn).

2.4 What Happens: Actions

Action indicates a specific, usually intentional action taken by an actor. A command (such as "get lamp") usually corresponds to a single action. As described later, an Action can succeed, be refused by the actor, or fail. Authors can easily make up new actions. The representation of actions is the basis of narrative. There are four categories of actions: Behave, Configure, Modify, and Sense.

A **Behave** Action which has no direct effect on the world, but which an item may react to and which may be narrated. For instance an actor waving hello would most straightforwardly be represented with a Behave Action.

A **Configure** Action changes the position of an item in the item tree and/or changes its relationship with its parent. So, wearing a cloak that one is holding is a configure action; so is taking a lamp.

A **Modify** Action changes the state of an item. For instance, an actor may turn on a lamp or open a door; those actions are best represented as modify actions.

A **Sense** Action is used to represent apprehending an item with a particular sense, most commonly sight, although sensing using five sense modalities is supported. As with Behave, an Action of this type does not by itself change the item tree (which is only changed by configure) or change any state of any item (which is only changed by modify).

A concept, like world, contains a list of actions in addition to a tree of items. To focalize a particular actor, the Teller module simply narrates based on the corresponding concept. To be "omniscient," the Teller uses the concept of the Cosmos, which has everything in the world in it.

2.5 Successful Actions, Failure & Refusal

Actions have preconditions and may have a postcondition, indicating that something changes in the world as a result of this action happening. After an action succeeds, it can be viewed using the **world actions** debugging directive and will be displayed like this:

```
/ / / has_feature @grate open
/ / / modify_to_different @grate open True
/ / / has_value @grate open False
/ / / can_access_direct @adventurer ['@grate']
/ / / has_value @grate locked False
:19: OPEN (modify) agent=@adventurer direct=@grate
force=0.2 feature=open
 old_value=False new_value=True cause="OPEN_UP
@grate" start=18
\ \ \ has_value @grate open True
```

This is action 19, an OPEN action which is of the modify type. Specifically, it is an action undertaken by the adventurer (agent=@adventurer) on the grate (direct=@grate) to change its "open" feature from False to True. Actions can be caused by act methods (representing the decisions to act made autonomously by actors), and by react methods (representing quick responses that any item can produce), and by being entailed by other actions, but the cause of this one is a command that was typed in and understood as "OPEN_UP @grate". The amount of force used is the default; the action starts at tick 18 of the world's clock.

There are five preconditions, indicated by the "/ / /" lines. In order, they state that the grate has to have the open feature (it has to be an openable item), that this has to be an attempt to modify its open feature to a different value (so that trying to open an already-open grate will fail), that the open feature is False to begin with (meaning that the grate is closed to begin with), that the adventurer has access to the grate, and that the grate is not locked. If the grate didn't have the locked feature – if it were an openable item that was not lockable – the system wouldn't add this last precondition. The second and third preconditions turn out to be redundant in this case and whenever the feature is True/False or otherwise two-valued. It would be nice to fix this, but except for a tiny bit of time spent checking it, there is no harm to having the extra precondition there.

The one postcondition, indicated by "\ \ \", is that the grate's open feature has a new value, True. That is, after the action has completed, the grate is open.

An action fails either because some precondition is not met or because some item in the vicinity prevents it from succeeding. For instance, "open the grate" fails if the grate is already open. A message is produced: "You are unable to open the grate because the grate is open to begin with." And inspecting the action by typing **world actions** shows that preconditions 2 and 3 fail and that the postcondition does not obtain:

```
/ / / has_feature @grate open
#####> modify_to_different @grate open True
#####> has_value @grate open False
/ / / can_access_direct @adventurer ['@grate']
/ / / has_value @grate locked False
:20: Failed OPEN (modify) agent=@adventurer
direct=@grate force=0.2 feature=open
```


FDG'11, June 29-July 1, Bordeaux, France.

```
 old_value=False new_value=True cause="OPEN_UP
@grate" start=19
 ##### has_value @grate open True
```

Even if the grate were closed, the action would fail if some guardian was in the area and (by means of the another method, a prevent method) prevented any gate-opening actions.

An action is refused by the player character under a small number of pre-defined circumstances. Namely, if a player types a command to go in a certain direction (for instance, "go east") and there is no obvious exit in that direction, the actor will refuse. Beyond this, an actor may have special cases for refusal defined and associated with its "refuses" keyword. In the Curveship version of *Cloak of Darkness,* the operagoer refuses to drop the cloak anywhere, although it can be hung on the hook in the cloakroom. This produces the message "You decide not to set the cloak down because the floor is not the best place to leave a smart cloak lying around." Typing **world actions** shows this representation of the refusal:

```
:3: Refused DROP (configure) agent=@person
direct=@cloak force=0.2 new_link=in
 new_parent=@foyer cause="DROP @cloak" start=2
 ##### parent_is @cloak in @foyer
```

Unlike a failure, a refusal doesn't actually represent any physical action happening in the simulated world. In a failed attempt to open the grate, the agent probably is exerting some energy and moving around. Something else may happen as a consequence. (In *Cloak of Darkness,* for instance, the message is unintentionally rewritten by the operagoer if that actor blunders around in the dark, trying to go in directions where there are no exits and failing.) But refusals do not correspond to the same sorts of attempts at action as do failures.

In a particular room, there will simply not be exits in certain directions. In many cases, it may be clear to any actor that exiting in those directions is impossible. To indicate a special reason why there is no exit in a particular direction, one can simply place a template in the "exits" dictionary of the room. This is the dictionary that is used in other cases to specify adjacent doors or rooms. This template will produce a sentence complation such as "the crack is far too small to follow" that explains why *any* actor can't go in a particular direction.

3. CAPABILITIES OF CURVESHIP

The architecture of Curveship draws on well-established techniques for simulating an IF world, separating these from the subsystem for narrating, which includes a standard three-stage natural language generation pipeline. The system includes concepts (the representations of the knowledge and perspectives of focalizers) that are separate from the world model.

The world is the base, authoritative model within the interactive fiction, the "actual world" from the standpoint of interactive fiction actors, following the terminology of Marie-Laure Ryan [5]. This model is one of several, however; there is also a concept, based on perceptions and experiences, for each actor. Each concept represents one actor's theory about the "reality" that the world encodes. The Simulator and world model have been developed to represent things such as the physical movement of objects and the configuration of a space in a flexible way. They are not to richly model emotional and mental qualities, although they can be used to do this to some extent.

One of the innovations of Curveship is the abstraction of the Simulator module from the module that does narrating, the Teller. This separation means that the workings of the world (whether a door opens or not, where an actor is located, etc.) are abstracted from the telling of the story. In other interactive fiction systems, a representation of action is generated when a state changes, as the simulation is being done. In Curveship, the Simulator produces and processes first-order representations of actions; then the Teller determines what (if anything) to narrate depending upon the spin. Each action is represented as completely as is each item in the world. Without such a strong, persistent representation of action, there would be no easy way to accomplish a re-ordering of events in the telling to produce flashback, flashforward, retrograde narration, sylleptic (by category) narration, other anachronies.

The Simulator is completely responsible for maintaining the state of the world and determining whether or not actions (whether they result from the player's commands or from code the author/programmer has written) succeed. The Teller is completely responsible for what is narrated – what is described and what actions are represented. There are other modules in Curveship that deal with game-level directives (saving the game, restarting, quitting), with clarifying input that is ambiguous, with recognizing player input, and so on, but the Simulator and Teller embody the important distinction that Curveship makes between the story or content level and the discourse or expression level. The idea of separating these levels is fundamental to modern narrative theory. Curveship is based on narratology as developed beginning in the 1970s. In particular, the narratology of Gérard Genette [6, 7] was a starting point in development of the system. Ideas developed by Gerald Prince [8] and others have also been incorporated.

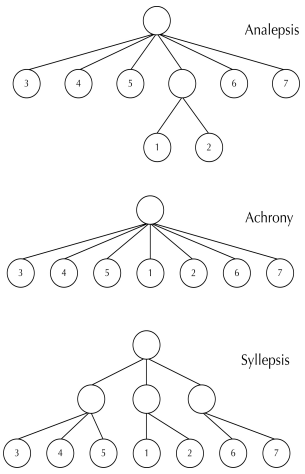

Figure 1. Ordered tree representations to allow narrating in the same sequential order, but for different purposes.

The Teller module has its own internal architecture. Its organization is based on a standard three-stage pipeline for natural language generation. The first stage is the reply planner, where the high-level arrangement of expressions is managed. The output of this stage is an ordered tree that, among other things, indicates the sequence in which these expressions will finally appear. In the next stage, the microplanner, the grammatical specifics corresponding to this structure, including tense, aspect, and number, are determined. Finally, the last stage, the realizer, produces the particular strings to be formatted and output.

The narrator is capable of varying several aspects of the narrating. Among these are Genette's categories of variation in narrative tense: order, frequency, and speed. It is also possible to vary focalization (a type of narrative mood) and time of narrating (a type of narrative voice). In the remainder of this section, I describe the system's ability to vary order and the time of narrating – two types of variation that are actually closely linked

from the standpoint of text generation [9][1]. To have a computer generate narrative, it is necessary to define not just different sequences of events that fall into the categories described by Genette, but also the particular processes that characteristically generate these sequences. In other words, formally defining an analepsis (or flashback) is not enough for narrative generation; it is also necessary to specify an algorithm that can generate an analepsis – preferably one that is flexible enough to specify most or all analepses. That is, the task of generating narrative demands that we have not only formal models for narrative, but also formal models for narrating. Characterizations of some of these formal models, algorithms for re-ordering events, are as follows:

Chronicle: Sort a set of events into chronological order. Saying that events are arranged chronologically may not be enough to specify a unique order, because some events may be simultaneous.

Retrograde: Sort a set of events into reverse chronological order.

Zigzag: This is the process of interleaving sections from period 1 (the "now") with those from period 2 (the "once") while narrating chronologically within each. A passage from Marcel Proust's *Jean Santeuil* provides an example [6]. The "now" and "once" must be designated along with a rule for moving between sequences. This could be as simple as "narrate a single event before switching," or it could involve specifying that all the events in a single physical location are narrated in the "now," then the corresponding events in the "then," and then similarly with the next physical location.

Analepsis: Also called flashback or retroversion, this indicates an anachronism inserted into a main sequence that is presumably chronological to begin with. For this process to work, both a main sequence and the point of insertion of the analepsis need to be designated. From the standpoint of the analysis of narrative, measures such as Genette's *reach* and *extent* are useful, but when generating an analepsis, those measures, which represent the difference in time and the overall duration of the analepsis, are not the most useful ones to specify. It is better to specify what should be included in the analepsis based on features of events. For instance, "select the most salient event from the first time the focalizer encountered this character," or "select the most salient events that the focalizer has seen happen in this room in the past, up to three of them." Of course, to make the latter rule useful, a rule for determining the salience of events must also be precisely specified. Given the main sequence, the point of insertion, and a fully specified rule for selecting events from the past, the process of ordering events so as to include an analepsis is straightforward.

Prolepsis: To insert a prolepsis, also known as flashforward or anticipation, the same three inputs are needed: a main sequence, a point of insertion, and a rule for selecting events from the future. When some newly simulated events are being narrated for the first time, there will not be a supply of simulated events waiting in the future. However, there are still circumstances under which a prolepsis can occur. An IF author can prepare "inevitable" events with future timestamps, representing things like the sun going down or nuclear missiles arriving. Also, there will be plenty of times in which the main sequence of events being recounted is from the past, so that future events relative to that span of time will be available.

Syllepsis: This is the organization of events into categories. Beyond the original set of events, only a sequence of categories seems essential for specifying sylleptic narrating. For instance, such a sequence might have these three categories of events in it: "the adventurer entering a new area," "the adventurer defeating a monster," and "the adventurer acquiring a treasure." If all events are in exactly one category, the categorization will be unique. The narrator can move through each of the categories and, within each category, can represent each of the events chronologically. There is no reason to restrict a sylleptic narration to chronological order within categories, though. Any principle for ordering based on time alone (chronicle, retrograde, achrony) can be specified for ordering the narrative within categories.

Achrony: Ordering events at random seems a suitable way to produce the type of order needed for achrony.

This describes how events can be rearranged from a chronological sequence into a narrative one that may not be chronological. Reordering has been characterized as producing a sequence, but there are problems with this view, because much structural information is lost in the flattened representation of the sequence. An analepsis, for instance, is not well represented by the sequence 3 4 5 1 2 6 7. The sequence of events that is in the past, relative to the main sequence – the "1 2" in this case – is actually embedded in the main sequence, which is "paused" while the telling returns to the past. There is no evidence of this, however, when seven numbers are presented in a row. The information about the embedding of "1 2" will usually be necessary to generate a narrative. When the main sequence is being told in the present tense, the "1 2" will almost certainly be told in the past. If the main sequence is already being told in the past tense, there will almost certainly be some cue that "1 2" occurs at a much earlier time: a phrase such as "before that," an explicit reference to the earlier date, some statement about habitual occurrences in the past, or a statement in the perfect leading into the analepsis. Without knowing that "1 2" is embedded, it is difficult to figure out how to shift the tense appropriately or add such a cue. Furthermore, using a simple sequence, there is no way to distinguish this analeptic case from an achronic jumble or from a sylleptic narration in which "3 4 5" are in the first category, "1 2" is in the second, and "6 7" is in the third. The representation used in Curveship distinguishes these three cases, as shown in figure 1.

Even without attempting to generate all of these sorts of transitions, there is clearly a need to designate more about the order of events than a simple sequence encodes. Such a representation should not force the tense of the analepsis to be different, but it should allow for this difference. It should also integrate the times at which events occurred into the decision about tense. Simply associating an arbitrary tense with the main sequence and another arbitrary tense with the analepsis would not accomplish this. The grammatical tense should be a result of the position of the simulated events in time and other essential parameters. To accomplish this, an ordered tree representation is used. For this particular analepsis, the tree will have a root node at the top level, 3, 4, 5, a special node, 6, and 7 at the level below, as children of the root, and then 1 and 2 at the lowest level, as children of the special, internal node. This sort of tree is called a reply structure in Curveship; it is provided within the Teller by the first-stage reply planner to the microplanner, the second stage of the pipeline.

The temporal position of the narrating has a special status: "I can very well tell a story without specifying the place where it

[1] The basic mechanism for changing order is the same is as described in the paper cited; Note, however, that many terms used in the system, along with the name of the system itself, have changed since that publication and other previous publications about the system. The terms used in this paper are those used in the Curveship code and documentation at the time of the system's initial release.

happens, and whether this place is more or less distant than the place where I am telling it; nevertheless, it is almost impossible for me not to locate the story in time with respect to my narrating act, since I must necessarily tell my story in present, past, or future tense" [6, p. 215]. These tenses lead to the "three major possibilities" for the temporal position of the narrating relative to the narrated: posterior, anterior, and simultaneous narration [8, p. 27]. While Genette deals with this in his category voice rather than in order, from the standpoint of generating narrative and determining grammatical tense, the temporal relationship of the narrator to events is as important as the temporal relationship of events to one another. Both must be dealt with jointly.

The tense of a proposed expression is necessary for realization; fortunately, this tense can be determined from three points in time (speech, reference, and event time) assigned to the proposed expression. Furthermore, these points can be defined for each specific expression using general rules that reside in the reply structure on these special, internal nodes – they do not have to each be individually prepared by the author. Tense is determined using these general rules and a theory that relates speech time, reference time, and event time to grammatical tense [10]. Three times are necessary because in a sentence such as "Peter had gone," there are three relevant points of time that are needed to explain the tense: the time at which the sentence is spoken (speech time); the time at which Peter left (event time), and another time that is being referred to, in this case, a time after the event time and before the time of speech, by saying "had gone" rather than something else, such as "went" or "was going." This last is the reference time.

Because all events in Curveship are simulated as happening at some specific time, Reichenbach's event time is always available to the Teller. Instead of requiring this that the other two times be specified manually for each event, the reply planner uses the topology of the reply structure to assign those times in a systematic way across each embedded sequence. Once all the proposed expressions have been defined with specific values for these three times, all the necessary information is in place for the next stage of the Teller to compute the tense using Reichenbach's formulas. The rules that determine speech and reference time are general (they do not require that every particular time be specified or computed by author-written code) but also flexible (for instance, every analepsis does not have to be told in a different tense from the sequence it is embedded in). Further details on how the reordering of events is accomplished are provided in [9].

4. STRING-WITH-SLOTS

The string-with-slots template representation is meant to be a contribution to interactive narrative and text generation that balances ease of authorship with narrative flexibility. This is in contrast to the more powerful but much more elaborate and difficult to author sorts of abstract syntax representations that are typically used in computational linguistics research. The string-with-slots formalism is used to generate the two main types of text in a typical interactive fiction: description and the representation of action. This section presents the first full description of the formalism as it works in the released system.

4.1 Describing Rooms

An IF author would typically write a room description by typing a string which could begin with something like "You are in ..." Some IF systems do provide for certain types of slots which are filled as text is prepared for output. In Curveship, the system is a bit more complex than is standard in IF but much less complex than a full abstract syntax representation would be. The idea behind this "string-with-slots" representation is to offer a reasonable amount of power and flexibility while still being fairly easy to compose. As a simple example, consider the description of the building in *Adventure with Style:*

```
[*/s] [are/v] inside _a_building, _a_well_house
for _a_large_spring
```

This of course produces "You are inside a building..." by default, but it can produce different strings when different spins are applied. There are three special things in this template: a slot for a subject, ending with "/s]"; a slot for a verb, ending with "/v]"; and some noun phrases that have been very lightly annotated using underscores. The "*" means "whoever is doing the sensing," and [*/s] mean "place whoever is doing the sensing here as the subject of the sentence." "[are/v]" is just the verb "to be." There is no need to say anything else about that verb, because the other piece of necessary information (the number) will come from the subject. Finally, this sentence doesn't have an object that is represented by a slot. It does have a few noun phrases that have been decorated with underscores to make them entities in the discourse, even though they are not simulated. After the adventurer enters the building and looks around, a second look will result is something like "You are inside **the** building, **the** well house for **the** large spring." (Emphasis added.) People typically shift from using the indefinite pronoun to using the definite one after they have mentioned something for the first time. Curveship has the capability to do this as well, both with simulated items and with appropriately decorated noun phrases.

The previous description was written to imitate an existing room description in a famous work of interactive fiction. Here is another example, a bit more complex and created with Curveship in mind, from *Lost One:*

```
[*'s] senses [hum/ing/2/v] as [*/s] [view/v]
[@plaza_center/o]

the morning [conclude/1/ed/v]

it [is/1/v] midday [now]
```

This defines three sentences. When writing a string-with-slots template, sentences do not have to be capitalized or punctuated - Curveship provides for that when it realizes text. Again, the "*" here is whoever is doing the sensing. "[*'s]" means the possessive form of whoever is doing the sensing, which could end up as "your," "the visitor's," "the punk's," "my," or several other things. "senses" is simply present as a plain word, written as part of the string. "[hum/ing/2/v]" represents the verb "to hum." Since the subject of the sentence isn't specified, "/2" has been added to specify that "hum" is plural. "/ing" is also added to specify that the progressive should be realized. This can produce "Your senses are humming..." along with other variations.

Then, "[*/s]" indicates that the sensor appears as a subject of the phrase coming up. "[view/v]" is the verb "to view." Nothing else needs to be specified about it. Finally, "[@plaza_center/o]" will generate a noun phrase naming the plaza center; it is an object, hence the "/o]" ending.

"[conclude/1/ed/v]" is a singular verb (hence the presence of the "/1") and is to be realized as perfect (thus, "/ed"). This provides "The morning has concluded" along with variations.

Finally, [is/1/v] is the singular verb "to be." In almost all cases, verbs should be specified by using their infinitive form with the "to" removed: [eat/v], [drink/v], [take/v], [drop/v], and so on. The system understands "is" and "are" as indicating "be," however. That allows the *Adventure in Style* room description to begin "[*/s] [are/v] inside"; it does not need to be written "[*/s] [be/v] inside."

Finally, "[now]" is a deictic word, one that refers to the situation of the telling. When one is telling a story in the present tense, narrating it as it happens, it is possible to use "now," as in "I'm here, I'm walking up the pathway now ..." If this person were to retell that story from somewhere else, later, in the past tense, using more or less the same language, he or she would want to change the "now" to "then" and the "here" to "there": "I was there, I was walking up the pathway then ..." Curveship alters "[now]" and "[here]" appropriately based on whether or not the narrator is speaking as if "here" "now" during the telling of the story.

4.2 Representing Action

The core of narration is the representation of action. After all, a text that just describes something isn't a narrative; it takes the representation of different actions to bring temporality, causality, and all the other interesting properties particular to narratives into play.

An example of the representation of action is seen in the special template for representing the "block" action. This one is particular to *Adventure in Style,* which has various guardians that block the adventurer:

```
[direct/s] [are/not/v] able to get by [agent/o]
```

This template reverses the usual object and subject: According to the action, the "agent" is blocking the "direct" (the direct object), and should be the subject. But this changes the construction so that "direct" is not able to get by "agent." The way Curveship represents sentences isn't rich enough to allow the system to automatically convert between active and passive constructions, but authors can write templates to have actions represented however they like. This template doesn't include the verb "block" – there is no requirement that the action's verb appear; the template can have what ever the author likes in it. Finally, "/not" is present in this template. It is one of the "bits" after the verb "are," indicating that this verb is to be negated.

Here is another action representation, this one one of the straightforward built-in ones, for the verb "touch" with an indirect object:

```
[agent/s] [touch/v] [direct/o] with [indirect/o]
```

With the appropriate agent, direct object, and indirect object it will produce "You touch the gelatinous cube with the ten-foot pole" and many variations. This template is extremely similar to the template for "drink" with an indirect object:

```
[agent/s] [drink/v] [direct/o] from [indirect/o]
```

The only difference is the word in the verb slot and the pronoun. Touching with a tool is best expressed using "with," while drinking from a source should be expressed using "from." By themselves, these brief representations of action may not seem very interesting. But they appear in the context of other actions and can be narrated in different orders and in different groupings, not to mention recounted within flashbacks and in other ways.

5. CONTINUING INTO NARRATIVE

So far, the process of developing of Curveship has suggested new ways to understand narrative that were prompted by the construction of a generative narrative system. Genette's sequence of events, reordered in the telling, was not adequate as a representation of how events are to be told; an ordered tree representation was effective, however. From the standpoint of generation, it was not possible to separate the consideration of the order of events from the consideration of time of narrating, since both narrative aspects are needed to determine grammatical specifics. Aditionally, although this topic has not been covered in this paper, the exploration of narrative distance undertaken in *Lost One* suggests that distance is well-understood as being composed of more primitive narrative aspects, and that changes in time of narrating, order, and narratee can increase or decrease distance.

With the release of this narrative variation system, researchers, interactive fiction author/programmers, teachers of narrative theory, and designers of narrative systems (whether games, literary works, or research projects) have the opportunity to apply Curveship in ways that interest them. For the teaching of narrative theory, for instance, Curveship offers a "narrative theory lab" in which students can interactively change the way the computer narrates, think about what will happen as a result of their change, and see the results. From the standpoint of research in expressive AI, interactive storytelling, and computational creativity, Curveship offers all the standard affordances of an IF system, the novel ability to control the narrative discourse, and implementation of the system itself and game/fiction files in a standard programming language that has rich modules available, supports interprocess communication, and is easily brought online (via Twisted). By integrating the text-generating capabilities of Curveship with computational creativity system that generates both plots and narrative style, for instance, it could become possible to realize legible textual output that can then be evaluated.

Many of the principles of narrative theory that are encoded in the system are applicable across media: In any narrative system, for instance, events are presented in a particular order which may or may not correspond to the order in which they occurred in the story world. Curveship offers the ability to quickly vary order and other narrative parameters without developing numerous assets. By providing a perspective on the narrating, the system offers the creators of narrative systems for different purposes (narrative theory education, gaming, literary development, research, etc.) a tool, a component of a larger system, and new lens through which to consider their work.

6. REFERENCES

[1] Crowther, Will and Don Woods. *Adventure.* 1976.

[2] Queneau, Raymond. *Exercises de Style.* 1947.

[3] Montfort, Nick. "Riddle Machines: The History and Nature of Interactive Fiction." *A Companion to Digital Literary Studies,* pp. 267–282. Eds. Ray Siemens and Susan Schreibman. Basil Blackwell, 2007.

[4] Nelson, Graham. *The Inform Designer's Manual,* 4th Ed. The Interactive Fiction Library, 2001.

[5] Ryan, Marie-Laure. *Possible Worlds, Artificial Intelligence, and Narrative Theory.* Bloomington UP, 1991.

[6] Genette, Gérard. *Narrative Discourse: An Essay in Method.* Trans. Jane E. Lewin. Ithaca, NY: Cornell UP, 1980.

[7] Genette, Gérard. *Narrative Discourse Revisited.* Trans. Jane E. Lewin. Ithaca, NY: Cornell UP, 1988.

[8] Prince, Gerald. *Narratology: The Form and Functioning of Narrative.* Berlin: Mouton. 1982.

[9] Montfort, Nick, "Ordering Events in Interactive Fiction Narratives." *Intelligent Narrative Technologies: Papers from the 2007 AAAI Fall Symposium,* pp. 87-94. AAAI Press, 2007.

[10] Reichenbach, Hans. *Elements of Symbolic Logic.* Random House, 1947.