Bardiche: An Interactive Online Narrative Generator

Bachelor Thesis AI, University of Utrecht

Geerten Vink, 3471233
g.j.a.vink@students.uu.nl

26-06-2015

Introduction

Bardiche is an interactive online narrative generator. That means it's a system to make narratives, it interacts with a user to collaborate in creating these narratives, and it plans and replans fast enough for the user to wait while it is doing so. Of course, a bard is a storyteller. The algorithm is based on the Glaive system (Ware & Young, 2014), and a bardiche is a type of glaive, so the name is a nod to the algorithm at its base.

Definition 1. Narrative is "the recounting of one or more real or fictitious events communicated by one, two, or several narrators to one, two, or several narratees" (Prince, 2003).

Narrative can be separated into story, which is the sequence of events recounted in the narrative, and discourse: the way in which the story is presented. Bardiche focuses exclusively on the story and makes no special effort at the presentation, although it is somewhat human-readable and a user only slightly familiar with the system will be able to understand what's going on in the story.

An example from the "door" domain which is used for examples throughout the text would be that `(give princess key knight hallway)` is generated. This is an event where the princess gives a key to the knight in the hallway. Discourse for this action might go something like: *With a flourish, princess Amarinth presented an ornate key. Sir Geoffrey knew this was the key to the outer gate, which was guarded by the evil knight Lorain. Although the villain might still stop him, at least now he had a way to open the gate and escape from Castle Dread if he got the chance.*

Although at some point in the future it may be interesting for narrative planners to generate their own domains and what kinds of objects, agents and actions are possible as such, Bardiche does not attempt this, and simply uses pre-made domains in which stories can take place. It aims to create suspenseful stories in any reasonable domain provided to it, in cooperation with a user who controls one of the characters. It does so online, i.e. in real time as a user is interfacing with it, without making the user wait for response overly long.

Bardiche is not a game (although it could be part of one.) It generates a story based in part on the actions of the user, but it's aim is to create a good story, not necessarily a fun or challenging experience. A good analogy for Bardiche's function is an improvisational theater group, with the user as a guest who gets to contribute to a play with a specific theme. No theater group can make a good play out of it if the guest is uncooperative, but it should be able to adapt to make the story interesting if the guest is not a great actor but cooperating to the best of his ability.

This thesis seeks to answer if this aim can be accomplished by dynamically choosing story endings to plan towards considering user actions. Bardiche uses the Glaive intentional narrative generation algorithm to determine a possible story flow, then replan if the user takes actions which don't follow this story. For example, if the system assumes the character will accept princess Amarinth's key from the earlier example, but he refuses to take it, it may replan for her to "lose" it, so the knight may pick it up, or for the knight to retrieve an axe from the armory and break down the door that way.

Related Work

Narrative generation systems can be divided into *simulative* and *deliberative* categories (Roberts, Riedl, & Isbell, 2009). Simulative systems generate a story world and the characters in them, and let the characters take actions which are appropriate to their surroundings. Deliberative systems “solve” a story by picking appropriate actions for the characters to reach a goal, within certain constraints (Riedl & Young, 2010). Since a deliberative system plans out the actions of all the characters that figure in its narratives, the characters tend to not have true agency, since the planner determines all their actions.

Although Simulative approaches, with characters that are individual agents, seem interesting because we ascribe a certain agency to any character we encounter in a story, the problem with such systems is that there is no good way to ensure character actions are chosen with global story structure in mind. On the other hand, a system in which the characters are simply pawns to move towards a goal does not strike us as particularly attractive either.

The Glaive (Ware & Young, 2014) system is a good middle road. It is a deliberative system, but plans according to the intentions of its characters. It is based on Hoffman and Nebel’s Fast-Forward planner (Hoffmann & Nebel, 2001), and on the IPOCL (Riedl & Young, 2010) and CPOCL (Ware, 2011) algorithms. IPOCL and CPOCL solve the same intentional planning problems as Glaive, but do so with partial order planning in plan space and are much slower. Speed is important in the creation of an online interactive narrative generator, which makes Glaive excellently suited to serve as a base for Bardiche. It will be extensively explained in the next section.

Glaive itself is not an interactive system though, unlike Façade (Mateas & Stern, 2003), Scheherazade (Li & Riedl, 2015), or a recent adaptation of the IRIS system (Fendt & Young, 2014) by Matthew William Fendt. We believe Bardiche will have advantages over all of these systems.

Façade (Mateas & Stern, 2003) tries to bridge the gap between the simulative and deliberative approaches. The user plays him- or herself, visiting a couple for a pleasant evening. Unfortunately, the marriage is dissolving, and both Trip and Grace, the hosts, try to pull the user into their quarrels. Although Façade offers a large amount of freedom to the user, the dialogue of Trip and Grace has been authored completely by hand. In effect, the user is interfacing with two very highly scripted agents behind a natural language interface, which react differently based on the “beat” the drama is currently in, as determined by a hidden drama manager agent. Although we don’t feel some authorial direction is a bad thing, Façade feels more like a directly authored story with a lot of branching than as a narration generator. In addition to this, we feel that although Façade can generate many different versions of this one story, in the end it only tells this one story, with a fixed starting point and fixed ending (Trip and Grace fight, pull you in, and one of them kicks you out of the apartment).

Scheherazade (Li & Riedl, 2015) uses specialized domain knowledge to generate *plot graphs*, which are generalized models of the topic domain. For example, in a plot graph about bank robberies, there may be a node “robber pulls out a gun” with edges to “robber points the gun at teller”, ”teller screams” and ”robber gives teller a bag” to signify which events would be appropriate to follow the pulling of the gun. Scheherazade parses these plot graphs from short stories about its topic written by human authors. We see two problems with this approach. First of all: A plot graph is too restrictive. Only those events the authors thought of in advance are possible, while plan-based systems will generally allow any character to take any action in the domain, generating plots the authors may not have thought of in advance. The second problem is that Scheherazade essentially mixes together some human-authored stories and claims that it is now original content. We are not sure we agree with that sentiment. The program did not generate a story, it generalized over existing stories authored by humans.

IRIS (Fendt & Young, 2011) is an intentional narration generator which has characters revise their intentions based on the state of the world, their beliefs, and their desires. In itself, that is an advantage over Glaive, since in Glaive the intentions of characters are static unless changes are explicitly defined in the domain (for example, we may create a *covet* action which makes an agent intend to take an item). However, Glaive is much faster in complex domains. IRIS is not interactive, but an interactive adaptation has been constructed in 2014 (Fendt & Young, 2014). This adaptation tries to create suspense in its stories, but does so by limiting the actions the user can take to ensure he or she experiences the suspense the generator tries to create. We don’t feel this approach is to be favored. The user should be free to take any action his character might take in the domain, the generator should adapt and attempt to create a “good” story no matter which actions the user chooses. This is the type of system we developed. We do, however, limit actions to those which can be explained by the intentions of the user’s character.

The Bardiche Interactive Narration Generator

Bardiche is a high level interactive narration generator which uses Glaive to work out details for it's stories.

Although there are many possible narrative structures, Bardiche creates stories with a single plot, focused on the protagonist controlled by the user. It attempts to create a story where the protagonist reaches his goal despite interference by other characters. In a game context, we would want to create suspense for the user, that is not what Bardiche does. The user has full knowledge of the state of the world, and is working in cooperation with the program to create an interesting, and suspenseful, story. This suspense should be inherent to the story, not to the experience of the user during its creation.

The intention of Bardiche is to create interesting, or 'tellable' stories that end well for the protagonist. Narratologists agree that conflict is an essential part of interesting stories (Abbott, 2008; Egri, 1960; Brooks & Warren, 1959). Conflict has been described as the thwarting of plans of intentional agents (Herman, Manfred, & Marie-Laure, 2010), so we consider a story to contain conflict when the different characters interfere with each other's plans. In addition to this, suspense contributes significantly to the enjoyment of stories (Brewer & Lichtenstein, 1982).

According to the Routledge Encyclopedia of Narrative Theory (Herman et al., 2010), suspense is inversely related to the number of reasonably computable outcomes. It comes to a climax when the outcomes become binary. Success or failure, life or death, win or lose. There is no suspense in a situation where only one outcome is possible. In the "door" example given before, we start with several options: the knight may just get the key and walk out of the castle unimpeded. The knight may try as hard as he wants, but not manage to escape. The knight may be foiled at first, but succeed in the end. There are multiple options, several possible ways for the story to end.

Our limited definition of what makes a good story, which Bardiche strives to create, is:

Definition 2. A *good story* is a suspenseful story in which some possible good endings for the protagonist are made impossible through conflict with other agents.

Bardiche should generate *good stories* whenever possible.

Glaive

Glaive is an intentional narrative generator. A valid Glaive plan is a plan which achieves the author's goals, but in which each step is explained by the intentions of the characters in the story. It is based on the IPOCL (Riedl & Young, 2010) and CPOCL (Ware, 2011) algorithms which solve the same type of problem, but are very slow. Glaive is based on the fast-forward planner by Hoffman and Nebel (Hoffmann & Nebel, 2001) and combines its speed with the intentional planning paradigm introduced by IPOCL and CPOCL.

Problem and Domain

Glaive takes an intentional planning domain and an intentional planning problem as its inputs. A domain consists of sets of types, constants, parameterized actions, and axioms. A problem gives the tokens of each type in the world, the initial state, and the authorial goal which the planner will attempt to achieve. The difference between a regular domain and an intentional domain lies in the fact that each action has a set of agents which must consent to the action for it to be executed. An intentional problem may contain intentions for the characters defined by it.

Figure 1 gives a small example of an intentional domain, which will be used for most of the explanations of concepts in the rest of the thesis. Except for *give*, the actions have been abbreviated to keep the size of the figure manageable. By default, Glaive takes these inputs as text files in the Planning Domain Definition Language.

Combined, the "door" domain and problem describe a situation where a knight is trapped in a castle by an evil knight. A door blocks his way out, which he can either unlock with a key or break down with an axe. The evil knight intends for the knight to not have the key or axe while the door is still closed. The knight intends to open the door. The princess who lives in the castle has the key, and intends for the knight to have it. We would like to ensure success for the knight after some conflict, so the author goal is for the door to be open but with the evil knight in possession of either the key or the axe.

Looking at the domain, we can see there are three types of entity in this domain: items, characters, and rooms. There are three constants, which are tokens of the entity types which exist in every instance of

```
types: item, character, room     actions: (move ?char ?from ?to) ...
                                          (pickup ?char ?item ?room) ...
constants: key, axe - item                (give ?giver ?item ?receiver ?room)
           gatehouse - room                 precondition:  (and(not (= ?giver ?receiver))
                                                                (at ?giver ?room)
predicates: (has ?char ?item)                                   (at ?receiver ?room)
            (at ?char ?room)                                    (has ?giver ?item)
            (in ?item ?room)                                    (not (had ?receiver ?item))
            (adjacent ?room ?room)          effect:        (and(not (has ?giver ?item))
            (doorOpen)                                          (has ?receiver ?item))
            (had ?char ?item)               agents:        (?giver ?receiver)
                                           (take ?taker ?item ?victim ?room) ...
                                           (open ?char) ...

axiom: context: (has ?char ?item)    axiom: context: (adjacent ?room ?neighbor)
       implies: (had ?char ?item)           implies: (adjacent ?neighbor ?room)
```

Figure 1: GLAIVE: The door domain

```
objects: knight evil_knight princess - character
         bedroom armory hallway cell - room

init: (at knight cell)                  (adjacent hallway cell)
      (at evil_knight gatehouse)        (adjacent hallway gatehouse)
      (at princess bedroom)             (adjacent hallway bedroom)
      (has princess key)                (adjacent gatehouse armory)
      (intends knight (doorOpen))       (in axe armory)
      (intends evil_knight (and (not (doorOpen))
                                (not (has knight key))
                                (not (has knight axe))))
      (intends princess (has knight key))

goal: (and (doorOpen) (or (has evil_knight key) (has evil_knight axe)))
```

Figure 2: GLAIVE: A door problem

```
 1. (move princess bedroom hallway)          The princess moves into the hallway
 2. (move knight cell hallway)               The knight meets her there
 3. (give princess key knight hallway)       The princess gives the key to the knight
 4. (move knight hallway gatehouse)          The knight moves to the gatehouse
 5. (non-executed (open knight))             The knight planned to open the door
 6. (take evil_knight key knight gatehouse)  But the evil knight takes his key away
 7. (move knight gatehouse armory)           The knight moves to the armory
 8. (pickup knight axe armory)               And picks up an axe
 9. (move knight armory gatehouse)           He moves back to the gatehouse
10. (open knight)                            And breaks down the door to escape
```

Figure 3: GLAIVE: A solution for the door problem in Figure 2

the domain. The key and axe to open the door exist in every instance of this domain, although there is no guarantee they will be reachable. The gatehouse also exists in every "door" domain. Six predicates with typed arguments describe qualities of characters, rooms, and items. For example, the *has* predicate has a character and item argument, and denotes possession of an item in this domain. Some predicates exist in every domain, like the = predicate to test for equality, and the *intends* predicate which denotes an agent's goals.

Actions are parameterized with preconditions, effects and agents. For example, the *give* action has as its preconditions that the giver and receiver are not the same character, that the giver and receiver are in the same place, that the giver has an item to give and that the receiver has not had the item before. The effect is that the giver no longer has the item, and that the receiver has the item. The agents which need to consent to the action for it to succeed are both giver and receiver. The *take* action is similar, but only needs consent from one character, while a hypothetical *lose* action would move an item to the ground without consent from any agent.

Finally we have axioms. In this case: adjacency is symmetric, and if someone has an item he's had it. Axioms are simply implications with a triggering condition and an effect implied by that condition. These axioms are simple, but universal and existential quantors, conjunctions, and disjunctions can be used to form complex implications. For example, we may have an axiom which states that if no other character is in the same room as a character, he or she is alone.

The problem defines tokens for the types. In this case, the knight, evil_knight and princess are characters. The bedroom, armory, hallway and cell are rooms. No items other than the constants defined in the domain exist in this specific problem. The initial state is a set of propositions. Most of the propositions speak for themselves, some special attention should be given to the *intends* predicates which denote goals for the three actors. A character goal is a tuple of a character and a proposition which represents the character intends to make the proposition true in the current state. Finally, we have an author goal, which represents the goal state the planner works towards.

Both the evil knight's character goal and the author goal are fairly convoluted in order to get a story with conflicts. The author goal would seem to ensure conflict, but would be satisfied if the knight opened the door, and then politely handed his axe to the evil knight guarding it. We avoid that by giving the evil knight no intention to have key or axe after the door is already open, so he will not consent to take those items after the door has been opened. It is the experience of the author that regardless of the domain, a *good story* in Glaive is made by very specific tailoring of character and author goals.

Planning and Plan Graphs

A valid intentional plan in Glaive is a sequence of actions such that each actions preconditions are true when the step is taken, the author goal is true after the last step, and every character who consents to a step has a reason to agree to that step. This agreement follows from the intentions of the characters and if there is a possible world in which this step would be on a path to fulfill the agent's intentions. For example, in the "door" scenario, consider the solution in Figure 3. In step 4 the knight moves to the gatehouse, with the intention to open the door using the key. This series of actions would lead to him reaching his goal. Although the plan fails, the knight had a good reason to take all the steps leading up to that point.

Glaive manages to do this very fast, using fast-forward planning with a tailored heuristic and an early selection on the possible actions that characters may consent to. This selection is made using *goal graphs*. This is a graph which contains those actions that may make a character goal true at the lowest level, and then those actions that satisfy a precondition of an action on the level below it at each higher level. An action can only be in the graph once, and is not added again if it already exists at a lower level. Since characters will only take actions on a path leading to the completion of one of their goals, and all steps leading to goals are on the goal graph, the algorithm can ignore any step that's not on a goal graph.

In addition to that, Glaive uses plan graphs which are an extension of those used by Fast-Forward. A plan graph has layers containing propositions and actions. The first layer has those steps which are true in the current state. The second layer is made by selecting those actions whose preconditions can be satisfied by the propositions in the first layer, and adding those propositions which may now be true. Only actions to which an agent may consent (and actions which require no consent) are used for this graph. This process is repeated until a layer is reached where all goal propositions may be true. Once this is so, a relaxed plan can be extracted, which ignores delete lists but gives a fair estimate of the length of a solution to the problem. Also important for Bardiche is that if no such relaxed plan can be found, a solution is not possible.

Creating Good Stories

Changes in Domain and Problem

Bardiche's input is slightly different from the input Glaive takes. In addition to precondition, effect, and agents the actions in the domain now also take an initiator. When we query the user for an action, we do not want to give him the option to take actions in which he is an agent, but not the initiator. For example, it would be strange if the protagonist could take an action to have the princess give the key to him, but in the domain used by Glaive, the agents are not in a determined order. The only change to the domain of 'door' to convert it to a Bardiche domain is to add an initiator to each action, which follow reasonably easily from the semantics.

The problem (see Figure 4) has a new field identifying the protagonist, and the author goal is replaced by a set of good endings and a set of bad endings, which are represented by an expression. In the case of the 'door' problem, there are two good endings: the door is open and the knight has the key, or the door is open and the knight has the axe. There is one bad ending: the evil knight has both the key and the axe.

Endings

An ending is not simply a final state. Let's assume we add another princess to our example. We'll call the princess that was already in the domain Princess Anna, and the new one is called Princess Belle. Both have a key to the gate, both want to help the knight, and the knight has no attachment to either one. Whether our story goes `(princess Anna gives her key to the knight [...] the knight escapes from the castle)` or `(princess Belle gives her key to the knight [...] the knight escapes from the castle)` does not change the ending, even though in the first version princess Belle still has a key, and in the second version Anna does.

An ending is also not 'the final action'. If we change our story in such a way that the knight wants to escape with the princess, we could conceive of a version where the knight escorts the princess out of the castle but is intercepted by the evil knight at the gate. Two possible endings are: `(the princess runs out of the gate while the knight fights his opponent, and in the end he defeats him.)` and `(the evil knight kills the princess as she attempts to flee, but in the end the knight defeats him.)`. In both cases the final action is the same, but the ending is radically different.

In our opinion Bardiche would need to have a semantic understanding of the domain to determine which different states constitute different endings, and this is not in the scope of the thesis. Therefor, we explicitly define good and bad endings as logical expressions in the problem file. Bardiche then converts these endings to a suspense goal.

Suspense Goal

To create Bardiche suspense goal expressions, we need to define two logical operators:

Definition 3. *select* is an n-ary boolean operator which is true if and only if exactly one of its arguments is true.

Definition 4. *possible* is a modal operator which is true when its argument can be satisfied in a plan graph from the current state, but is not true in the current state.

Since it's based on a heuristic, *possible* may return true when it is not, in fact, possible to generate a plan which will make its argument true. This is, however, relatively rare. The only way to guarantee that an expression can be satisfied in the domain is to generate a full plan which satisfies the expression. That is not acceptable because it will slow the program down too much, so we accept the occasional failure to generate a story in order to keep the program fast.

Given a set of good and bad endings given as logical expressions in a problem file, Bardiche generates a goal expression. Let $g_0 ... g_n$ denote the set of good endings, and $b_0 ... b_n$ the set of bad endings. $\Diamond$ will denote the possibility-operator, Sel_1 denotes 'select'.

Definition 5. *suspense goal* $= Sel_1(\Diamond g_0, ..., \Diamond g_n) \wedge (\Diamond b_0 \vee ... \vee \Diamond b_n) \wedge \neg(g_0 \vee ... \vee g_n) \wedge \neg(b_0 \vee ... \vee b_n)$

In words: The *suspense goal* is a state in which exactly one good ending is possible, at least one bad ending is possible, and no good or bad ending is true.

```
objects: knight evil_knight princess - character
         bedroom armory hallway cell - room

protagonist: knight
init: (at knight cell)                  (adjacent hallway cell)
      (at evil_knight gatehouse)        (adjacent hallway gatehouse)
      (at princess bedroom)             (adjacent hallway bedroom)
      (has princess key)                (adjacent gatehouse armory)
      (intends knight (doorOpen))       (in axe armory)
      (intends evil_knight (not (doorOpen)
      (intends evil_knight (has evil_knight key))
      (intends evil_knight (has evil_knight axe))
      (intends princess (has knight key))

bardicheGoal:
    (good (and (has knight key) (doorOpen))
          (and (has knight axe) (doorOpen)))
    (bad (and (has evil_knight key) (has evil_knight axe)))
```

Figure 4: BARDICHE: A door problem

```
1.  (princess (move princess bedroom hallway))
2.  (knight (move knight cell hallway))
3.  (knight (take knight key princess hallway))
4.  (knight (move knight hallway gatehouse))
5.  (evil_knight (take evil_knight key knight gatehouse))
6.  (knight (move knight gatehouse armory))
7.  (knight (pickup knight axe armory))
8.  (knight (move knight armory gatehouse))
9.  (knight (open knight))

1.  (princess (move princess bedroom hallway))
2.  (knight (move knight cell hallway))
3.  (knight (move knight hallway gatehouse))
4.  (princess (move princess hallway gatehouse))
5.  (knight (move knight gatehouse armory))
6.  (knight (pickup knight axe armory))
7.  (knight (move knight armory gatehouse))
8.  (evil_knight (take evil_knight axe knight gatehouse))
9.  (princess (give princess key knight gatehouse))
10. (knight (open knight))
```

Figure 5: BARDICHE: two solutions to the ‘door’ problem

Assuming the domain and problem have been set up in such a way that all the good endings are possible at the start, this guarantees that all other good endings will be made impossible during the story. In this way we build suspension to a climax where one more setback will make all good endings impossible.

Algorithm 1 The Bardiche Algorithm

```
1: Let L be the list of steps in the story.
2: D ← domain from file.
3: P ← problem from file.
4: generate a suspense goal G from the good and bad endings in P.
5: set G as the goal of P.
6: buildingSuspense ← true
7: while (!complete) do
8:   GlaivePlan ← generate(P, D)
9:   if (GlaivePlan != null) then
10:    complete ← GlaivePlan is fully executed.
11:  else
12:    break ("unfortunately, Bardiche was unable to generate a story.")
13:  end if
14:  ExecutedPlan ← The executed portion of GlaivePlan
15:  if (!complete ∨ buildingSuspense) then
16:    if (!complete) then
17:      step ← let the user pick an intentional step initiated by the protagonist which has valid preconditions in the state at the end of ExecutedPlan.
18:      add step to ExecutedPlan
19:      complete ← G is true in the state at the end of ExecutedPlan
20:    end if
21:    if (complete) then
22:      buildingSuspense ← false
23:      complete ← false
24:      set goal G to the only possible good ending
25:    end if
26:    create new problem p with the initial state corresponding to the state of ExecutedPlan with goal G.
27:    P ← p
28:  end if
29:  add all steps of ExecutedPlan to L.
30: end while
31: output L
```

Goal Resolution

Once Bardiche has generated a story which satisfies the suspense goal, by definition only one good ending is still possible. This is the climax point for the suspense in the story. After this point, we want to work towards resolution.

Definition 6. The *final goal* is the only good ending which is still possible once the suspense goal has been satisfied.

At this point we set the goal for the planner to the final goal. If the endings have been picked in such a way that the character can not make all but one of the good endings impossible for him or herself, this guarantees conflict and suspense, thus guaranteeing the generation of a *good story.*

Interactivity in Bardiche

Bardiche does not just generate *good stories*, it does so interactively. A user controls the agent marked as the protagonist in the problem file.

Algorithm 2 generate(problem, domain)

```
1: let G be the goal of the problem.
2: if (goal G is new) then
3:   create a plan graph for goal G (which is used for the 'possible' operator).
4: end if
5: generate GlaivePlan using Glaive with the current problem and domain.
6: if (GlaivePlan != null) then
7:   repeat
8:     doContinue ← false
9:     output those steps of GlaivePlan that have been executed.
10:    if (!GlaivePlan is fully executed) then
11:      suggestedStep ← first non-approved step in GlaivePlan in which the protagonist is an agent or initiator.
12:      doContinue ← query if the user wants to execute suggestedStep
13:    end if
14:    if (doContinue) then
15:      approve the suggested step.
16:    end if
17:  until (!doContinue)
18: end if
19: return GlaivePlan
```

Process of Interaction

Bardiche Plans consist of Intentional Steps.

Definition 7. An *intentional step* is an action which is potentially motivated.

Definition 8. A step s which requires the consent of characters C is *potentially motivated* if and only if for each character c in C, there exists some goal g such that c intends g in the current state, and s appears somewhere in the goal graph for c *intends* g.

Steps may be *executed* or *suggested.*

Definition 9. A *suggested step* is the first step which has the protagonist as an agent or as its initiator which is not *approved*, or any step after that step.

Definition 10. An *executed step* is any step which is not a *suggested step.*

Bardiche lets Glaive generate a plan to the current goal, which is either the *suspense goal* or the *final goal.*

If the plan consists solely of *executed steps*, it is complete. If the current goal was the *suspense goal*, we set the *final goal* to be the current goal and generate a new plan. If the *final goal* was the current goal, we are done and output the entire story.

If the plan does not consist solely of *executed steps*, we ask the user to approve the first *suggested step.* If he does, we check if the plan is now complete. If not, we repeatedly ask the user to approve additional steps.

If the user does not approve a step, we ask him to pick an *intentional step* which the protagonist can initiate in the current state. We then create a new problem with the state after the execution of this new step as its initial state, and generate a new plan.

We repeat this routine until we have a completed story.

Dynamic Goal Selection

Assuming the endings defined in the domain are dependent on the actions of the protagonist, the user can exert a strong influence on the ending of the story. For an example, look at Figure 5. The first story is generated by approving every action suggested by the planner. It leads to the ending `(and (has knight axe) (doorOpen))`.

To see how the second story is generated, look at Figure 6. The user decides not to take the key from the princess, but instead to walk to the armory to get the axe right away. It's still possible for the knight to get the key, though, and taking it away from the princess would not make it impossible for the knight to get it (since he has not "had" the key, and thus could take it away from the evil knight). Taking the axe away from the knight will make it impossible for him to get it back, though, since in this domain losing an object means it's irrecoverable. Thus, this is the fastest way for the planner to eliminate one of the good endings, which makes it switch the ending to `(and (has knight key) (doorOpen))`.

In general, the planner will find the plan which leads to the elimination of all but one goal g in the lowest amount of steps, but the user does not have to follow this plan. If the actions of the user make it easier to eliminate all but one goal which is not g, the planner will work towards this new goal. In this way the *final goal* is dynamically selected from the good endings in the *suspense goal.*

Further Work

Creating domains and problems for Bardiche requires less effort than for Glaive, because the stories generated are semi-automatically forced to be "good stories", whereas Glaive has a tendency to generate stories that beeline for resolution without conflict if goals are not very specifically set to induce it. It is still nontrivial to get them right, though.

The main problem is that it is not possible to determine if a goal that is possible in the current state will become impossible in another state, except by finding a state in which it is impossible. If all goals are always possible at the same time, Bardiche will keep searching for a plan in which only one of them is true, never terminating. If, for example, we remove the condition that a character may not have had an item before from the 'door' domain, it is always possible for the knight to get the key and the axe. Since the planner does not "know" this, it will attempt to find a plan in which it's impossible for the knight to get one or the other. Some sort of logical analysis of domain and problem should be devised to find out if it's possible for goals to be impossible. We reckon this is doable, and would warrant a look.

It would also be interesting to see if good and bad endings could be parsed from the domain and initial state, but we believe this is a very difficult problem to solve. As discussed before, an ending is not a state or a final action, but a nebulous quality which is hard to formalize. It would require the program to analyze the semantics of the constants, predicates and actions in order to say which conditions have to be met for a state to be a good or bad ending. Considering the previous issue mentioned, it also has to be able to devise sets of endings which can be planned towards.

Finally, we would like to create some sort of domain/problem maker, which would make it easier to create larger domains and problems than typing them by hand.

Conclusion

Bardiche is an interactive online narrative generator which generates *good stories* with dynamic goals based on the problem and the user's actions. Bardiche uses the Glaive system to generate intentional plans in any domain, although some domains and problems are more suited to the system than others. It is fast enough to be used online for smaller problems. It is untested on large problems because they are too hard to make without bugs.

We believe Bardiche is an improvement over similar systems in that it allows the user a large degree of freedom in the actions his character takes, while the generator adapts quickly and manages to come up with new suggested story flows rapidly. In addition to that, Bardiche works for any domain given to it, and always creates "good stories" if possible.

Software

Bardiche has been implemented in Java 8. It is available on request.

It requires the Glaive Narrative Planner, which is implemented in Java 7 and available at http://stephengware.com/projects/glaive/

Acknowledgements

We would like to thank Stephen G. Ware and Michael Young for authoring Glaive and confirming to us by email that we are "welcome and encouraged" to use it in this project.

References

Abbott, H. P. (2008). *The cambridge introduction to narrative.* Cambridge University Press.

Brewer, W. F., & Lichtenstein, E. H. (1982). Stories are to entertain: A structural-affect theory of stories. *Journal of Pragmatics*, *6*(5), 473–486.

Brooks, C., & Warren, R. P. (1959). *Understanding fiction* (Vol. 5). Appleton-Century-Crofts New York.

Egri, L. (1960). *The art of dramatic writing: Its basis in the creative interpretation of human motives.* Simon and Schuster.

Fendt, M. W., & Young, R. M. (2011). The case for intention revision in stories and its incorporation into iris, a story-based planning system. In *Intelligent narrative technologies.*

Fendt, M. W., & Young, R. M. (2014). Adapting iris, a non-interactive narrative generation system, to an interactive text adventure game. In *The twenty-seventh international flairs conference.*

Herman, D., Manfred, J., & Marie-Laure, R. (2010). *Routledge encyclopedia of narrative theory.* Routledge.

Hoffmann, J., & Nebel, B. (2001, May). The ff planning system: Fast plan generation through heuristic search. *J. Artif. Int. Res.*, *14*(1), 253–302.

Li, B., & Riedl, M. O. (2015). Scheherazade: Crowd-powered interactive narrative generation.

Mateas, M., & Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. In *Game developers conference* (Vol. 2).

Prince, G. (2003). *A dictionary of narratology.* University of Nebraska Press.

Riedl, M. O., & Young, R. M. (2010, September). Narrative planning: Balancing plot and character. *J. Artif. Int. Res.*, *39*(1), 217–268.

Roberts, D. L., Riedl, M. O., & Isbell, C. L. (2009, September). Beyond adversarial: The case for game ai as storytelling. In *Breaking new ground: Innovation in games, play, practice and theory.* Brunel University.

Ware, S. G. (2011). A computational model of narrative conflict. In *Proceedings of the 6th international conference on foundations of digital games* (pp. 247–249). New York, NY, USA: ACM. doi: 10.1145/2159365.2159401

Ware, S. G., & Young, R. M. (2014). Glaive: A state-space narrative planner supporting intentionality and conflict. In I. Horswill & A. Jhala (Eds.), *Proceedings of the tenth AAAI confer-*

ence on artificial intelligence and interactive digital entertainment, AIIDE 2014, october 3-7, 2014, north carolina state university, raleigh, nc, USA. AAAI.

```
  (:executed (princess (move princess bedroom hallway)))
  (:suggested (knight (move knight cell hallway)))
take action ((move knight cell hallway))? (y/n)
y

  (:suggested (knight (take knight key princess hallway)))
take action ((take knight key princess hallway))? (y/n)
n

0) (move knight hallway gatehouse)
1) (move knight hallway cell)
2) (move knight hallway bedroom)
3) (take knight key princess hallway)
0

  (:executed (princess (move princess hallway gatehouse)))
  (:suggested (princess (give princess key knight gatehouse)))
allow action ((give princess key knight gatehouse))? (y/n)
n

0) (move knight gatehouse hallway)
1) (move knight gatehouse armory)
2) (take knight key princess gatehouse)
1

  (:suggested (knight (pickup knight axe armory)))
take action ((pickup knight axe armory))? (y/n)
y

  (:suggested (knight (move knight armory gatehouse))))
take action ((move knight armory gatehouse))? (y/n)
y

  (:executed (evil_knight (take evil_knight axe knight gatehouse)))

Suspense building complete

  (:suggested (princess (give princess key knight gatehouse))))
allow action ((give princess key knight gatehouse))? (y/n)
y

  (:suggested (knight (open knight))))
take action ((open knight))? (y/n)
y

Story Complete
```

Figure 6: BARDICHE: generating the second story of Figure 5.