Narrative Generation in Entertainment: Using Artificial Intelligence Planning

Richard A George

Email: ricgeorge21@gmail.com

Abstract

From the field of artificial intelligence (AI) there is a growing stream of technology capable of being embedded in software that will reshape the way we interact with our environment in our everyday lives. This 'AI software' is often used to tackle more mundane tasks that are otherwise dangerous or meticulous for a human to accomplish. One particular area, explored in this paper, is for AI software to assist in supporting the enjoyable aspects of the lives of humans. Entertainment is one of these aspects, and often includes storytelling in some form no matter what the type of media, including television, films, video games, etc. This paper aims to explore the ability of AI software to automate the story-creation and story-telling process. This is part of the field of Automatic Narrative Generator (ANG), which aims to produce intuitive interfaces to support people (without any previous programming experience) to use tools to generate stories, based on their ideas of the kind of characters, intentions, events and spaces they want to be in the story. The paper includes details of such AI software created by the author that can be downloaded and used by the reader for this purpose. Applications of this kind of technology include the automatic generation of story lines for 'soap operas'.

Keywords: Artificial Intelligence, Planning, Entertainment, Narrative, Generation, Automatic, Story, Storytelling, ANG

http://dx.doi.org/10.5920/fields.2015.118

Introduction

Artificial Intelligence (AI) refers to the intelligence demonstrated either by machines, or in this case, software, and incorporates functions such as reasoning, planning, learning, natural language communication, perception and robotics. Through advancements in artificial intelligence we have been able to accomplish many things, from performing tasks that would otherwise be dangerous for humans to take on, to automating mundane and tedious types of work. Through AI we will be able to tackle a vast assortment of more challenging and interesting problems in the future, such as more accurately predicting the weather, providing effective driverless transportation and eventually creating robotics so advanced that they will be able to rival the thought process and complexity of the human brain (Turner, n.d.).

A particular area of interest with room to grow is in finding new and interesting ways to use these concepts to provide a more automated process and intriguing experience in entertainment. With this in mind, this paper concerns the automatic narrative generation for storytelling.

The work behind this paper has an eventual goal of enabling software to autonomously generate dialog amongst characters using only a set of predefined attributes. For example, a user could create characters each with their own set of parameters including gender, age, experiences, personalities, ambitions, goals and different relationship variables between each other. The AI software would generate dialog between them intelligent enough to flow based on the subject of the conversation, with each character involved producing a meaningful response to what was said previously for a more natural simulation.

In order to bring this type of functionality to a software application, the idea must be brought back to the most basic aspects of storytelling. This is known as the fabula of a story, and consists of the most basic elements. These elements generally include actions, or a sequence of events that are carried out between characters and other objects in the story to create the story's skeleton. Therefore, this project aims to use various techniques in AI to allow a user with no previous programming or technical experience to shape a story based on their own ideas and story elements.

The Problem

Storytelling is traditionally produced manually by talented writers with both a natural ability, as well as skills developed during their education. These stories are for the most part developed using a writer's experiences, personality, morals, ambitions and perhaps most important, their creativity. Creative writing is therefore a process that takes quite some time to complete, and varies heavily based on the writer.

Many of these prerequisites required for creative story-telling can be artificially reproduced using predefined variables, allowing a piece of software to intelligently create narration between characters and tell a story.

Targeted Audience

The audience for this product may include companies in the entertainment industry, such as television programs or children's networks with cartoons. For example, this software could be used to take characters from television series and create unique scenarios between them to be used for episodes. Similarly, a film script could also

be generated using predefined characters and situations between them. This technique would most likely be more successful with drama genres, as they focus heavily on character development and interaction rather than action sequences.

Stories in video games could also be uniquely generated using this concept. Game developers could use the same character techniques to develop stories and interactions between different characters in the game. This could provide a unique story and user experience for each user that plays the game, offering new levels of re-playability, an important aspect to consider when purchasing a game.

This idea could also be used to generate storytelling for short books or comics. Simple stories used in children's books or interactive comics and graphic novels used on smart phones and tablets could also be uniquely generated with this method.

State of the Art

Research Groups

The work of three particular research groups are worth highlighting. They are currently working within a similar field of artificial intelligence in entertainment while incorporating human psychology and interaction. These research groups are all currently active and have undertaken many projects in various aspects of content generation.

Teeside University

Dr Julie Porteous leads a research group at Teeside University that looks into intelligent virtual environments (IVE). The group specifically deal with Human-Centred Multimedia and Human-Centred Interfaces, and are mostly involved in entertainment computing, with some work in health informatics.

This group is most noted for their work on IRIS, or Integrating Research in Interactive Storytelling. This project was aimed at achieving breakthroughs in the understanding of interactive storytelling and the development of the necessary technologies involved. It was designed to make interactive storytelling technologies in regards to performance and scalability in order to support the production of real interactive narratives. It also aimed to make the new generation of these technologies more accessible to authors and other types of content creators of various disciplines.

The group is currently working on a project called MUSE, or Machine Understanding for Interactive Storytelling. This project will introduce a new method of navigating through and understanding information through three-dimensional interactive storytelling. This system can take in text-based input of natural language and process it into knowledge representing characters, their actions, plots and the world around the characters. These are then rendered as 3D worlds that the user may navigate through with interaction, re-enactments and gameplay. (IVE Lab, 2011).

Liquid Narrative Group

The Liquid Narrative Group in North Carolina State University works with procedural content generation allowing them to create content for games and other virtual environments. Similar to this project, they use models of narrative to build stories

and tell them automatically. At the core of their work and research is *Narrative Structure and Comprehension*, which looks into computational models of narrative, its structure and learning how we build mental models of stories while we produce and comprehend them.

The group is also responsible for a project entitled *Interaction in Automatically Generated Narratives*, involving the creation of interactive experiences within a progressing story. It required the development of stories allowing user input to dynamically alter the actions that take place during the story line. This project was ended in 2005 (Liquid Narrative Group, 2014).

USC Institute for Creative Technologies

The final research group is Jonathan Gratch and his group from USC Institute for Creative Technologies. As a computer scientist and psychologist, his research consists of developing human-like software agents for virtual training environments. These methods are used to create psychological theories of human behaviour. He specifically investigates how algorithms can be used to control human behaviour in virtual environments.

One particular area of interest is emotion modelling, where Gratch looks into the role emotions play in the believability and immersion of a simulated story's world. In the Emotion Project, the group develops models that allow artificial characters to display an emotional response to events that occur within the story's world and respond with actions and behaviours that are consistent with that of humans in an emotional state.

In the Virtual Human Project, research in intelligent teaching, natural language generation and recognition, interactive narrative, emotional modelling and graphics and audio are all combined to provide a realistic and compelling training environment. The virtual humans can interact with the trainee and provide emotional responses to their actions (Gratch, 2014).

Existing Systems

Generator of Adaptive Dilemma-based Interactive Narratives (GADIN)

The GADIN (Generator of Adaptive Dilemma-based Interactive Narratives) system, developed at the University of York, is one method currently used in interactive story generation, and is currently being used in games and television. Its purpose is to satisfy essential criteria for interactive narrative used by other systems that would otherwise be incapable of being addressed.

With this system implemented, the story creator is only required to provide basic information regarding the domain background, such as information on characters and their relationships, actions and problems. Instances are then created of these problems and story actions and a planner generates a sequence of actions that lead up to a problem for the characters involved (which in this case, can also be the user). This goes a step further by allowing the user to provide input on choosing their own actions, allowing the system to adapt future storylines according to past behaviour. The effectiveness of the stories generated in this way are evaluated using criteria such as interestingness, immersion and scalability (Barber, 2008).

Suspenser

Suspense is a very important aspect of storytelling by readers and listeners. While there has been extensive research into ways of automating narrative, the subject of adding depth to these stories, such as suspense to evoke cognitive and affective responses by readers is severely lacking.

Suspenser is a specific framework in development by the Liquid Narrative Group at North Carolina State University designed to help research and develop a system that can produce a narrative designed specifically to evoke suspense from the reader. Similar to GADIN, this system can take in a data structure containing a plan comprised of goals of the story's characters and actions that they can perform in pursuit of these goals. The system uses a plan-based model of comprehension to determine the best way to output the final content in the story to best manipulate the reader's level of suspense. This is done through adopting theories developed by cognitive psychologists.

Suspenser takes three elements as input before creating a story, the first being the fabula, or the raw material of a story. The second input is a point *t* in the story's plan corresponding to a particular point where the reader's suspense will be measured. The final is the desired length of the story, so that story elements, actions and suspense can be adjusted accordingly. With this information the system can determine the *sjuzhet*, which refers to the way the story is organised. More specifically in this case, it includes the content of the story up to the point *t* allowing the reader to infer a minimum number of complete plans for the character's goal. This is done in accordance with psychological research on suspense. (Cheong & Young).

Actor Conference (ACONF)

Narrative plays an important role in understanding the events of our lives on a daily basis. The ability to generate this narrative automatically can have a huge impact on virtual reality systems designed for entertainment, training or education. This concept is complicated by two main problems: *plot coherence* and *character believability.* The coherence of the plot refers to the appearance that events in the story all lead towards some outcome or goal. Character believability refers to the appearance that the events in the story are driven by the attributes of the characters within the story. There are currently many systems capable of achieving only one of these goals, but the Actor Conference system (ACONF) presents a different approach to automatic narrative generation with the ability to generate stories with both problems addressed.

The ACONF system is specifically designed to exploit the advantages of both character-centric and author-centric techniques and ideas to achieve both problems of plot coherence and character believability. It uses a *decompositional*, *partial-order* planner to develop and assemble a sequence of actions that make up the story. These actions represent the way the characters will behave as part of this story. Using a planner for this allows for the identification of casual relationships between different actions, as well as providing an ordered sequence of operations as output that can be directly executed by agents (or characters) in this virtual world. (Riedl & Young, 2003).

AI Techniques underlying ANG

The main AI technique we will focus on is automated planning, as this process is what builds the story from its components. Automated planning or AI Planning is a part of the field of artificial intelligence that involves the automated solving of problems through the generation of action sequences or strategies. Plan generation is followed by plan execution by intelligent agents such as autonomous robots and vehicles. Planning solutions are often complex, and are constructed or discovered using various types of algorithms. A basic planning problem has a given start state, goal conditions and set of actions that may be carried out by the intelligent agent. A sequence of actions leading from the start state to the goal is then discovered. The aspects of the problem that are uncertain are the effects of the action, knowledge of the system state and a sequence of actions that may guarantee the achievement of the goal.

Similar to automatic narrative generation, interactive storytelling researchers have been using planning systems and has become the most common approach. Planning provides a well-rounded approach to this problem for many reasons. Narratives, which are also the basis for interactive storytelling, may be broken down into three levels. The lowest level is called the fabula, and is defined as ‘a series of logically and chronologically related events’ (Boutilier). Because planning consists of a series of actions that work towards a goal, it represents a good model for the fabula.

Planning Domain Definition Language (PDDL) is the language used to define and execute AI Planning, and is generally used as standard. It uses a combination of two files, including the ‘domain’, which holds all of the declarations needed. This includes variables of different types called *predicates*, as well as *actions* that can be carried out. The second file is called a ‘problem file’, which uses the variables and actions declared in the domain file along with a ‘start’ and ‘end’ state. When used with a planner, the problem file will declare the initial state of the problem, and the planner will use the domain declarations to make legal decisions governed by that file to achieve the end state (McDermott, et al., 1998) (see appendices 12-1 and 12-2 for this project’s PDDL domain and problem files). The use of PDDL therefore dictates a host of other requirements to ensure the project can be successful.

Requirements Specification

User Requirements

This project is not governed by any specific user (individual or company), and therefore has user requirements specified by myself, as well as the recommendations of this project’s supervisor. These requirements for the software prototype portion of this project are as follows:

Tools

Using Prolog logic programming language is essential to creating a prototype for this product. Prolog is a **logic programing** language that will allow me to declare

relations with specific facts and rules, as well as create domains and problems that can be tested using search algorithms.
As discussed earlier, **PDDL** will be the main method of testing problem files and programming characters with many different attributes.

For a possible user interface, **Java** will be required to ensure it can be as effective as possible. It will require looking into ways of having Java communicate with other types of necessary languages to allow the creation of characters and attributes in this way. Doing so will allow any user, even without a programming background, to be able to create characters for storytelling.

It will also be required to learn methods of connecting the planning engine to a shell to translate the output into **natural language** for demonstration purposes.

Software
Eclipse IDE will be used to program in Java for a user interface allowing the creation of characters. If time permits, an interface will be created that users can use to create characters, attributes and relationships that can be translated into PDDL using Java. Java with Swing should allow for a user-friendly end product that would be relatively straight-forward to understand for the average user. Java is also cross platform and can communicate with many other programming languages, keeping options open to unexpected requirements as progress is made through the project.

Planners will also be utilized to plan out steps from storytelling problem files and provide a possible output solution for a story. This will need to be translated into a script using Java.

Product Specification
Product Specification
Interface
The prototype will include an interactive interface to allow a user to create their own characters and other story elements with specific attributes and relationships to generate their own stories with the help of planners. This would be ideal, as it would prove that any number of characters and scenarios could be generated using this idea, and show that anyone, with or without a development background, can be capable of producing content in this way.

Data Definition
The software prototype will be capable of receiving user input in the form of story element objects and characters and their relationships between one another. These variables will be predefined to ensure that they have the appropriate effect on the characters and the story's content. This will output a planning problem in PDDL with various specific states and parameters used to define the particular story. Actions may also be defined to provide specific effects of particular events that may occur.

These PDDL files will be able to be parsed by various planners available across the web. An attempt to integrate one or more planners into this application will be made to try and keep everything in one area and application. Once a plan is created, the software will be able to parse the steps and adjust it into the form of a narrative with

a clearly define sequence of events, with variables and solutions based on the user's initial input.

Once content is generated, the application will attempt to be hooked up to a shell capable of processing text into natural language. This shell will then be capable of reading out the dialog with different characters for demonstration purposes.

Functional Specification

Services

The final product of this project will be a software prototype designed to prove the concept of automatic narrative generation based on a set of predefined attributes for characters, goals and stories. This application will be programmed using characters with variables including gender, age, experiences, personalities, ambitions, goals and different relationship variables between each other. Ultimately, stories generated in this way will be able to be read out loud by natural language processing software.

Development Method

This project will be developed using a ***software prototyping methodology***. A prototype in this case refers to a piece of software that is in its early stages designed to test a concept or process that can be learned from. This is incomplete from a final piece of software as it only focuses on achieving specific goals within a future larger project. The process of software prototyping therefore involves creating prototypes built to simulate core aspects of the final application that is in development.

This type of software methodology is useful when assessing the requirements that will make the final product successful. It also allows developers to test the feasibility of the product without having to construct the entire system, thus saving time and money.

This process can be broken down into four phases:

1. **Identify Initial Requirements.**
   Also referred to as a *prototype plan*, this phase is used to determine the basic objectives of the prototype including the necessary input and output. From here the prototype's functionality may also be defined.
2. **Development.**
   During this phase the first prototype is developed as an executable piece of software containing the user interfaces and possibly very limited functionality.
3. **Evaluation.**
   In the evaluation phase, everyone involved with determining the final product, including customers and end-users, review the prototype and provide feedback for changes and additional functionalities to ensure the development is going in the right direction.
4. **Enhancement.**
   Based on the feedback received the software specifications and prototype can be improved. This means that steps 2 through 4 may be repeated as necessary.

Prototyping may also be broken down into several different types, including *throwaway, evolutionary, incremental, operational* and *extreme prototyping.* This project will focus on using **throwaway prototyping**.

Throwaway prototyping involves creating a model of the core functionality of the system at an early stage in development. Once all necessary preliminary requirements are gathered and understood, a simplified working version of the application is created to visually demonstrate the capabilities of the system.

This type of prototyping can be completed quickly and is useful if some aspects of the requirements specification is not fully understood. This means that these requirements can be explored and identified in depth and tested very quickly before any heavy development takes place.

Generally, once a working model is demonstrated and agreed upon, it is essentially discarded to begin formal development of the system. As the final goal is to create a prototype, the development process will terminate after the prototyping stage.

Prototyping does have its disadvantages however. Its insufficient analysis can sometimes cause developers to lose focus on the final solution by overlooking superior solutions that may be better to maintain. Developers may also spend too much time and money developing a prototype, or become attached to it making it difficult to throw away. In this case sometimes developers would try to alter prototypes for use as a final product when it does not provide an acceptable underlying architecture. If moving onto a final product, these problems would have to be taken into consideration. However, in this case, throwaway prototyping is perfect for ensuring that the core goals of the product are understood, and can prove the concept visually and audibly to users of the system (Beaudouin-Lafon).

Design

Use-Case Diagram

Figure 1 refers to a Use-Case diagram, which outlines all of the various actions that can be carried out by the user, and how they are achieved using both the software itself and an additional planner. A more in depth analysis of how every aspect of the system works together is described in the Implementation section of this document.

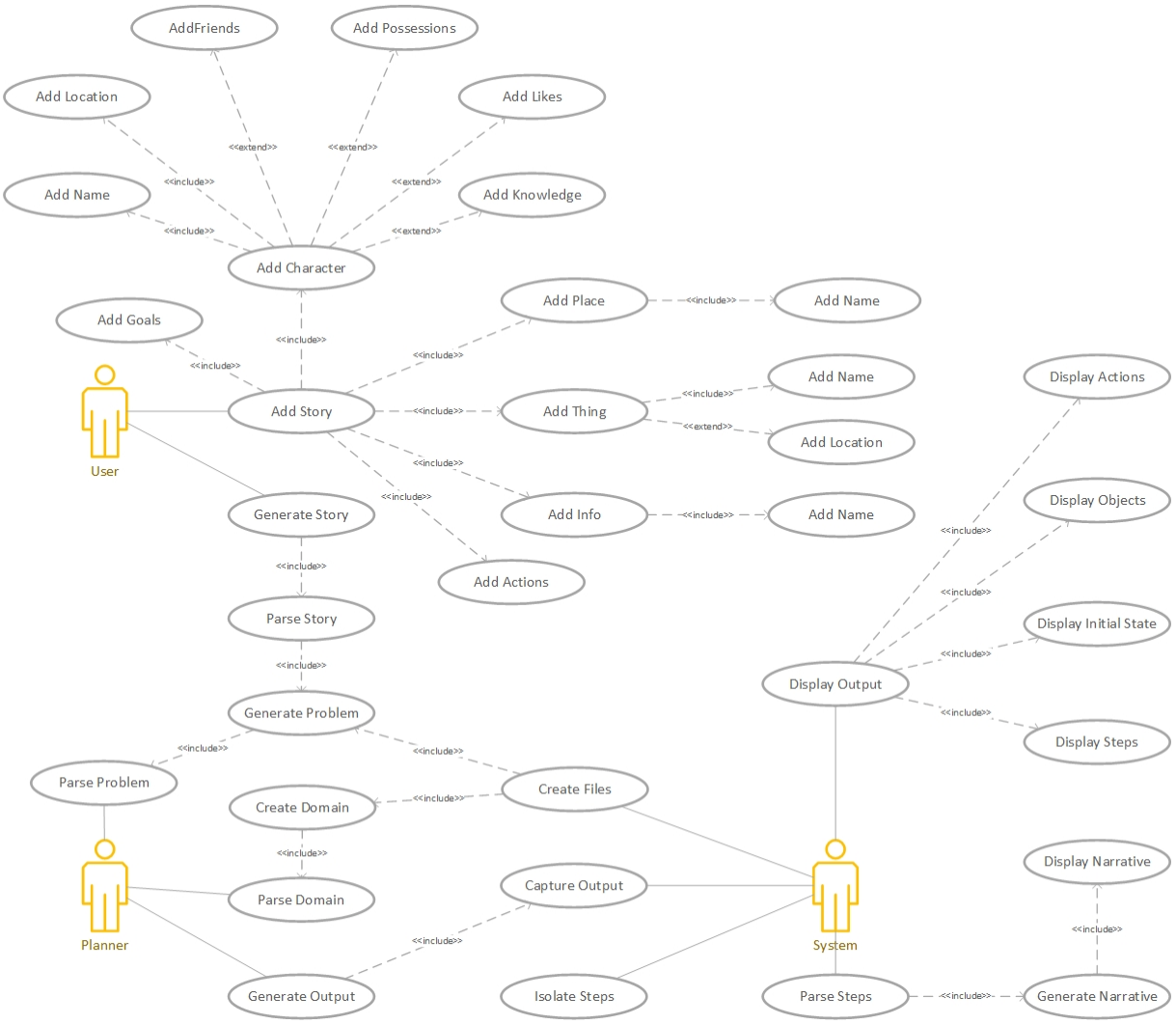

Figure 1. Use Case diagram outlining the potential actions that may be carried out by the user and the software.

Implementation

After research into the feasibility of producing a product on this scale, even as a prototype, some compromises had to be made. Rather than creating an application with the ability to create characters with different personality traits, research lead me in a different direction with a heavy focus on what defines a story. This implementation therefore revolves around the ability for anyone without programming to build a short story using the programmed interface. These stories are created using a user interface to add story elements, from which a PDDL problem file is created. Together with a predefined domain, a planner is used to find a solution that can be converted into a narrative-like structure.

Structure

`Stories` are comprised of 8 main elements, including characters, places, things (objects), information (knowledge), goals, actions, a protagonist and an antagonist. The Story class has therefore been designed to hold this type of information, with array lists used to hold groups of characters, places, things and information.

The `Character` class has been designed to hold extra information, including a name, a current location (place), a list of friends (characters), a list of things in their current possession, a list of objects they like, a list of objects they would prefer not to get rid of, and a list of information that they may have. All of these attributes can be added or removed by the user when adding characters to a story.

The `Thing` class has a similar design, allowing each individual item to have a name and an initial location. The `Place` and `Info` classes simply contain a field for an appropriate name. The `Goal` class contains three Strings to be used for parsing, which include the first party involved (a Character or Place), the second party (Character, Thing, Info or Place depending on the first party), and a conjunction String such as ‘has’, ‘at’, ‘knows’, etc.

User Interface

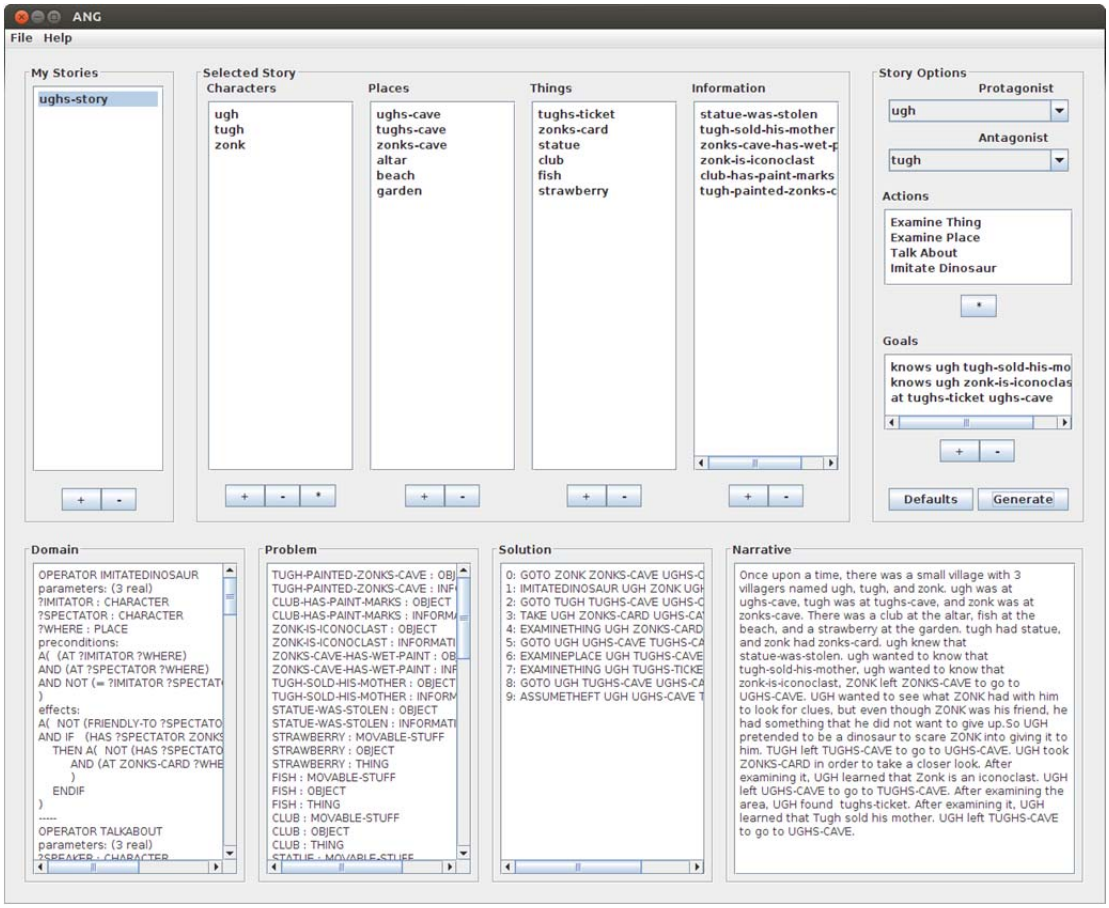

Figure 2. The main user interface after all story elements have been inserted, and a story has been generated.

The user interface begins with the main window for the `ANGgui` class, which is also the central location for creating and controlling every aspect of a user’s story. This can be seen in Figure 2 as a full story with all elements filled in and generated. To begin, a user may use the ‘My Stories’ panel to add or remove a story, while giving it a name. The panel contains a list of all individual stories created by the user. Selecting a story will update all of that story’s information in all other panels to the right of the story.

Once a story is created, the user may add any other item of their choosing. It is recommended to start with places, since both characters and things can have an initial location. Pressing the add button in the Places panel with display a simple window asking for the name of the place. Once added, the added places will appear in the list, and be added to the selected story. These places can be removed from the story and list by hitting the remove button. The Information panel works in an identical way.

Using the add function in the Things panel will display a window requiring a name and a current location. If the item created is located at a particular place that was previously created, it can be selected here. If the item is meant to be in the possession of a particular character, the location should be set to ‘none’, as this function is controlled in the Character section.

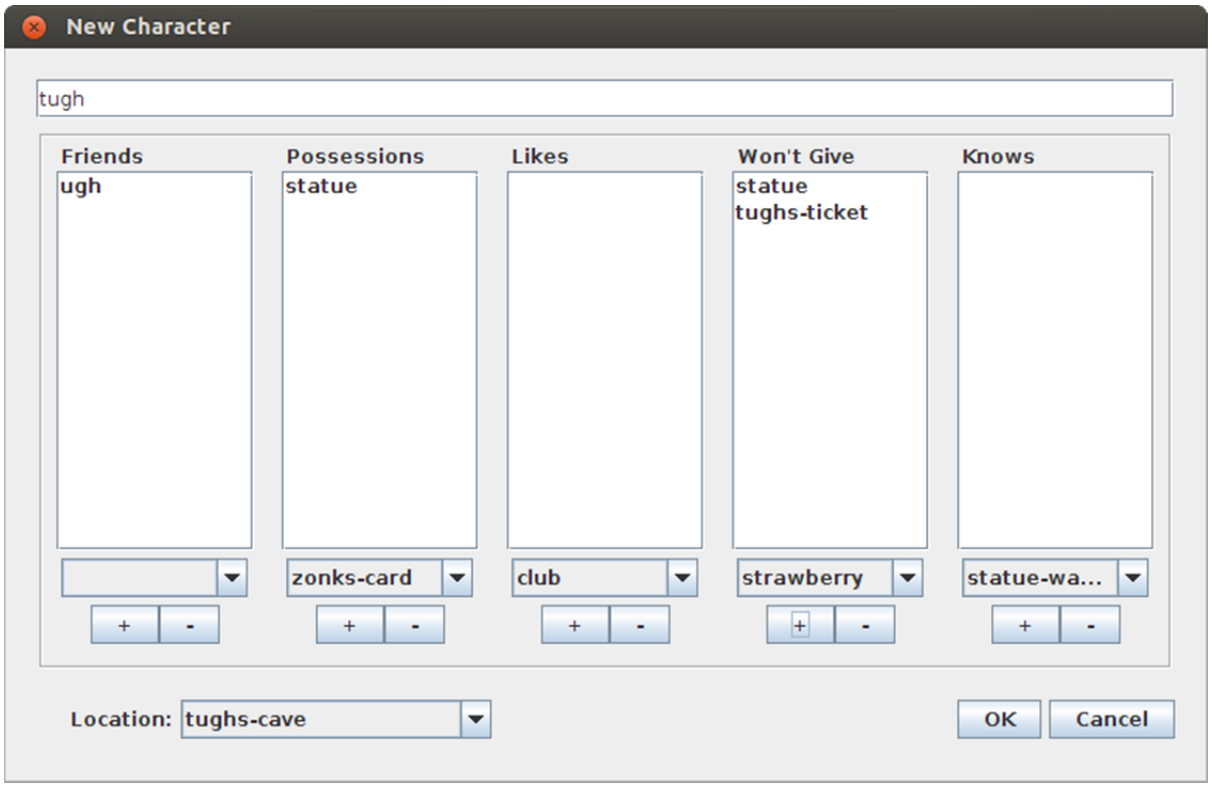

Figure 3. The ‘New Character’ interface with available objects from the story as parameters for the new character.

Using the add character function shown in Figure 3 will display a new window with all of the character options. From here the user can add the character’s name and initial location based on the places added previously. There are also options to use previously added material to include a list of that particular character’s friends, items in their current possession, items they like, items they will not give up, and pieces of information they currently know. These items can be added or removed with the interface’s buttons and combo boxes. Selected characters from the list may also be edited and deleted as necessary.

The Options panel contains an area to set the protagonist and antagonist of the current story. The protagonist is who will be used when the planner is looking for a

solution, and the protagonist has been programmed to be the current thief in this particular story domain.

The actions area is used to modify four specific available actions within a story, including examining things, examining places, talking about information and imitating dinosaurs (scaring other characters). Selecting one option in the list and pressing the edit button will bring up a special window for that action containing combo boxes with different relevant story elements that can be used in combination to add an effect to that action.

Below the actions area is an option to add goals to this particular story, which determine the steps used by the planner to achieve them. This is important, as this determines how certain characters behave and interact with objects, places and other characters. Adding a goal will bring up a window asking to choose a character or a thing. If a character is chosen, the option to know, have, be at, etc. can then be chosen. Based on that selection, the appropriate set of objects added to the story will be available in the final area. For example, if a character is chosen with the word ‘at’ selected, then the final combo box will be populated with the list of places added to the story. This would subsequently create a goal requiring that at some point, the selected character must end up at the selected location. Likewise, if a Thing is selected, a location is the only goal it may have.

Functions and Planner

Once all elements of a story have been created, the user has the option to hit the ‘Generate’ button, where a number of functions take place.

The first of these functions is done in the `CreateFile` class, where all of the story information is parsed and outputted to a PDDL problem file using the appropriate syntax. This includes object declarations and types, as well as the entire initial state, including where characters and things are, which character currently has what object, what each character knows, etc. This file also contains the goals of the story, and is located in a new directory created in the project directory to house files associated with that particular story.

Once created, a classed called `PlannerFunctions` uses the build in the Process class to run a specific command through the terminal without opening it. This command is to locate and run the ‘FF’ planner in the project directory. This command is also programmed to locate the pre-defined story domain, and the newly created problem file to be run through the planner. In order to gather further information about the domain and problem files, the planner is also run in a specific configuration to output additional information about every object created, every action with their effects, and the entire initial state. If the designed story has a solution and parses correctly, the planner will find it and output all of this information, including a list of steps to achieve the contained goals.

This planner output is captured by the function and separated into three sections, including actions, objects/initial state, and steps. These three sections are then sent back to the main window and distributed to their corresponding text areas in the output panel at the bottom of the main window.

When a list of steps are generated, they are also sent to a method that uses particular situations to parse this data into more of a narrative-like format. This story is then displayed in the final output window called ‘Narrative’. Once completed, any number of changes can be made to that story and re-generated to view different results.

Story Domain and Problem

The application has the ability to add and modify character actions from the actions section. This set of actions are used to make up the story Domain once the story is generated. The story domain used for illustration purposes in this document and during testing is based on the version created and used in interactive storytelling testing by Leandro Motta Barros and Soraia Raupp Musse from the University of the Sinos Valley (Barros & Musse, 2007). Small alterations were made to have it parse successfully with the FF Planner (Joerg) and this application.

The domain file is designed with specific objects, including all of the story elements available in the ANG interface to create a story. It also includes a set of default actions, including ‘`GoTo`’, ‘`Take`’, ‘`TalkAbout`’, ‘`ImitateDinosaur`’, ‘`ExaminePlace`’, ‘`ExamineThing`’, ‘`GivePresent`’, and ‘`AssumeTheft`’. Each action can involve different object types, and can have modified preconditions and effects based on the way the user sets up the actions in the application.

For example, when using ‘`GoTo`’, a character will move from their current location to another location. ‘`Take`’ involves a character taking an item from a place or a character. ‘`TalkAbout`’ involves two characters talking about a particular piece of information in order to learn another piece of information. ‘`ImitateDinosaur`’ is used when a character must take an object from another, but the owner of the object does not want to give it up. The other character imitates a dinosaur to scare them into giving it to them, resulting in that character no longer being friendly towards them. ‘`ExaminePlace`’ is used for a character to look for items at the particular location. ‘`ExamineThing`’ is used to gain a particular piece of information from an object. ‘`GivePresent`’ is used for one character to give a gift to another. If the other character likes the particular object given then that character becomes friendly towards the other. Finally, ‘`AssumeTheft`’ is used to put an item at a particular location.

This application may be used to create a story of any type with any objects provided by the user, since all actions and story elements may interact with each other in the way that the user chooses. Currently, ANG does not support the addition of new actions.

The story problem used for this story domain consists of the declared objects, and the initial states of those objects. At the end contains the goals for the story, which result in the planner attempting to achieve those goals to provide a sequence of events for that story. Changing the types of goals will modify the outcome of every story in an attempt to achieve all of these goals.

Maintenance

Work in Progress

In the interest of time, the ability to save created stories to a file that can later be retrieved to avoid having to re-create stories was omitted. As this is a prototype to prove a particular concept, this function was considered less important than getting core functionalities working correctly. This could be done by creating a simple file when saving a story that contains lines of Strings including all of a story's information. A heading for each set of lines, such as Character or Place, would be useful in ensuring the information is structured in an organised way for easy retrieval. These files could then be retrieved at launch and their stories re-created accordingly.

Future Enhancements

This application is designed only to create a story's fabula, or sequence of events that take place. It cannot create dialog between characters. Future enhancements could use AI learning techniques to determine how a particular character might speak to another, or go into more depth about what action they may decide to take. This would require substantially more research for a working solution.

With further enhancements, the ability to attach the outputted narrative to a shell capable of reading it out loud with natural language processing could also be developed to provide further functionality for demonstration purposes.

Evaluation

Product

This entire project has an ultimate goal of allowing the ability to generate dialogue between characters with predefined attributes. After extensive research into this particular area of artificial intelligence, and into the structure of story-telling in general, it was decided that before this type of dialogue can be created, the story's most basic level, the fabula, must be constructed. This consists of low level actions and events that happen within a story. In that regard, the developed ANG prototype successfully achieves this goal, and allows anyone with no programming experience to use an interface to develop and generate these stories on his or her own.

The product includes certain predefined actions that may exist in any story, such as characters moving from one place to another, or taking objects from certain places. Additional actions, such as speaking to one another and examining objects and places can potentially include many different effects on characters and objects. By including the ability to manipulate these effects in the more complex story actions, there is virtually no limit to the complexity of the objects being created by the user to determine a full story.

While there are many way to improve the overall rigidity of the application, such as additional error handling and a more fluid narrative output, the product is a prototype that successfully delivers on its required specifications and ability to prove a concept.

Conclusion

Artificial Intelligence is rapidly shaping the future of many important aspects of everyday life. One particular area with growing interest in future advancements is

entertainment, including television, film, gaming and general interactive and non-interactive means of story-telling.

Directors, producers, designers and developers are often looking for new techniques for engaging audiences into the world that they have created. A sense of immersion is key, and even with the best visual effects can be lost without interesting and unpredictable stories and characters within them.

The focus of this research and project is to analyse the feasibility of automating the most fundamental and lowest level component of a story; the fabula, which is comprised of a sequence of events that occur within a story. Through research it has been determined that in order to consider building an application complex enough to develop dialog between characters with predefined attributes regarding their personality, a story fabula must first be created. As these events are effect simple steps taken to achieve an ending, artificial intelligence planning has proven to be an effective way of providing results.

Automated story-telling through the use of artificial intelligence planning has proven to be an effective means of creating stories with varying results based on the intended goals. By altering these goals with new additions, the story results can change depending on those goals, and the type of search algorithms used within the planner.

The story example and its variations that can be developed using this product are very simple as a result of them being based on only the story's sequence of events. At present, this would be most useful as a teaching tool, allowing younger audiences to develop their own stories using characters. This would be useful for the development of their creativity and reading skills, and provide a more interactive and enjoyable way of learning these skills.

Future development into this idea has massive implications on what will be possible in the near-future. With these techniques, various aspects of scripts can already be pre-written, and computer-controlled characters in games can give the illusion of free will based on the way a user interacts with them. The limits of the human imagination are already being challenged with modern advancements into researching these topics, including going as far as scientists theorising that everyday life could possibly be nothing more than a simulation (Kinder, 2013). It is an exciting, if not somewhat terrifying thought when considering what may be possible within this lifetime.

References

Barber, H. (2008, October). *Generator of Adaptive Dilemma-based Interactive Narratives.* York: Department of Computer Science, The University of York.

Barros, M., & Musse, S. R. (2007). *Planning Algorithms for Interactive Storytelling.* Universidade do Vale do Rio dos Sinos, Brazil. ACM Compt. Entertaint.

Beaudouin-Lafon, M. (n.d.). *Prototyping Tools and Techniques.* Retrieved from Laboratoire De Recherche En Informatique (Laboratory for Computer Science): https://www.lri.fr/~mackay/pdffiles/Prototype.chapter.pdf

Boutilier, C. (n.d.). *Stanford.* Retrieved from Stanford University: http://www.stanford.edu/group/nasslli/courses/boutilier/Lecture1.pdf

Cheong, Y.-G., & Young, M. (n.d.). *Narrative Generation for Suspense: Modeling and Evaluation.* North Carolina.

Gratch, J. (2014). *USC ICT: Gratch*. Retrieved from USC Institue for Creative Technologies: http://people.ict.usc.edu/~gratch/

IVE Lab. (2011). *Intelligent Virtual Environments*. Retrieved from IVE Teeside University: https://ive.scm.tees.ac.uk/

Joerg. (n.d.). Fast-Forward.

Kinder, L. (2013, November 20). *Do we live in the Matrix? Scientists believe they may have answered the question*. Retrieved from The Telegraph: http://www.telegraph.co.uk/science/science-news/10451983/Do-we-live-in-the-Matrix-Scientists-believe-they-may-have-answered-the-question.html

Liquid Narrative Group. (2014). *Liquid Narrative Group: Projects*. Retrieved from Liquid Narrative Group: http://liquidnarrative.csc.ncsu.edu/index.php/research/projects/projects-by-topic

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., . . . Wilkins, D. (1998). *PDDL - The Planning Domain Definition Language.* New Haven, CT: Yale Center for Computational Vision and Control.

Riedl, M., & Young, M. (2003). *Character-Focused Narrative Generation for Execution in Virtual Worlds.* Toulouse: ICVS 2003.

Turner, B. (n.d.). *10 Ways Artificial Intelligence Will Affect Our Lives*. Retrieved from Discovery: http://www.discovery.com/tv-shows/curiosity/topics/ways-artificial-intelligence-will-affect-our-lives.htm

Appendices

The files shown in this section represent stories generated using the story domain used for testing by Leandro Motta Barros and Soraia Raupp Musse from the University of the Sinos Valley (Barros & Musse, 2007). These versions are modified to work correctly with ANG and with the planner used during testing and for illustration purposes.

Sample Full Story Domain (after story generation)

Modified Version (Barros & Musse, 2007)

```
(define (domain ughs-story)
(:requirements :strips :typing :equality :conditional-effects
        :existential-preconditions :negative-preconditions)
 (:types place information movable-stuff - object thing character -
movable-stuff)
 (:predicates (at ?what - movable-stuff ?where - place)
        (has ?who - character ?what - thing)
        (is-protagonist ?who - character)
        (knows ?who - character ?what - information)
        (likes ?who - character ?what - thing)
        (friendly-to ?the-who ?the-friend - character)
        (wont-give ?who - character ?what - thing)
        (everybody-knows ?what - information)
        (is-the-thief ?who - character))
 (:action GoTo
      :parameters (?who - character ?from ?to - place)
      :precondition (and (at ?who ?from)
                (not (= ?from ?to)))
      :effect (and (at ?who ?to)
            (not (at ?who ?from))))
 (:action Take
      :parameters (?who - character ?what - thing ?where - place)
      :precondition (and (at ?who ?where)
                (at ?what ?where))
      :effect (and (not (at ?what ?where))
            (has ?who ?what)))
 (:action GivePresent
     :parameters (?giver ?receiver - character ?present - thing)
     :precondition (and (not (wont-give ?giver ?present))
               (not (= ?giver ?receiver))
               (has ?giver ?present)
               (exists (?p - place)
                   (and (at ?giver ?p)
                      (at ?receiver ?p))))
     :effect (and (not (has ?giver ?present))
           (has ?receiver ?present)
           (when (likes ?receiver ?present)
              (friendly-to ?receiver ?giver))))
 (:action AssumeTheft
     :parameters (?thief - character ?location - place ?stolen-thing -
thing)
     :precondition (and (at ?thief ?location)
          (is-the-thief ?thief)
          (has ?thief ?stolen-thing))
     :effect (at ?stolen-thing ?location))
```

```
  (:action ExamineThing
      :parameters (?examiner - character ?what - thing)
      :precondition (has ?examiner ?what)
:effect (and (when (= ?what tughs-ticket)
(knows ?examiner tugh-sold-his-mother))
(when (= ?what zonks-card)
(knows ?examiner zonk-is-iconoclast))
(when (= ?what club)
(knows ?examiner club-has-paint-marks))
))
  (:action ExaminePlace
      :parameters (?examiner - character ?where - place)
      :precondition (and (at ?examiner ?where)
(not (exists (?c - character)
(and (not (= ?c ?examiner))
(at ?c ?where)))))
:effect (and (when (= ?where tughs-cave)
(has ?examiner tughs-ticket))
(when (= ?where zonks-cave)
(knows ?examiner zonks-cave-has-wet-paint))
))
  (:action TalkAbout
      :parameters (?speaker ?listener - character ?topic - information)
      :precondition (and (friendly-to ?listener ?speaker)
(not (= ?speaker ?listener))
(knows ?speaker ?topic)
(exists (?p - place)
(and (at ?speaker ?p)
(at ?listener ?p)))):effect (and (knows ?listener ?topic)
(when (and (= ?listener zonk)
(= ?topic zonks-cave-has-wet-paint))
(knows ?speaker tugh-painted-zonks-cave))
(when (and (= ?listener tugh)
(= ?topic zonks-cave-has-wet-paint))
(knows ?speaker tugh-painted-zonks-cave))
))
  (:action ImitateDinosaur
      :parameters (?imitator ?spectator - character ?where - place)
      :precondition (and (at ?imitator ?where)
(at ?spectator ?where)
(not (= ?imitator ?spectator)))
:effect (and (not (friendly-to ?spectator ?imitator))
(when (has ?spectator zonks-card)
(and (not (has ?spectator zonks-card))

(at zonks-card ?where))))))
```

Sample Story Problem (after story generation)

Modified Version (Barros & Musse, 2007)

```
 (define (problem ughs-story-act1)
  (:domain ughs-story)
  (:objects
   ugh - character
   tugh - character
```

```
   zonk - character
   ughs-cave - place
   tughs-cave - place
   zonks-cave - place
   altar - place
   beach - place
   garden - place
   tughs-ticket - thing
   zonks-card - thing
   statue - thing
   club - thing
   fish - thing
   strawberry - thing
   statue-was-stolen - information
   tugh-sold-his-mother - information
   zonks-cave-has-wet-paint - information
   zonk-is-iconoclast - information
   club-has-paint-marks - information
   tugh-painted-zonks-cave - information
   )
  (:init
   (is-protagonist ugh)
   (at ugh ughs-cave)
   (at tugh tughs-cave)
   (at zonk zonks-cave)
   (at club altar)
   (at fish beach)
   (at strawberry garden)
   (has tugh statue)
   (has zonk zonks-card)
   (knows ugh statue-was-stolen)
   (likes zonk strawberry)
   (friendly-to tugh ugh)
   (friendly-to zonk ugh)
   (wont-give tugh tughs-ticket)
   (wont-give tugh statue)
   (wont-give zonk zonks-card)
   (is-the-thief ugh))
  (:goal (and (knows ugh tugh-sold-his-mother)
      (knows ugh zonk-is-iconoclast)
      (at tughs-ticket ughs-cave)
)))
```