DESIGN AND ANALYSIS OF ReGEN
A NARRATIVE GENERATION TOOL

Ben Kybartas

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

August 2013

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

ACKNOWLEDGEMENTS

I would like to express my thanks to Professor Clark Verbrugge for his support and guidance throughout the research and creation of this work. I would additionally like to acknowledge the continued encouragement of my family, my partner and my friends in the past two years. I would like to express specific thanks to Bentley James Oakes for coming up with the name for our system.

ABSTRACT

Using procedural narrative generation in video games provides a flexible way to extend gameplay and provide more depth to the game world at low cost to the developers. Current examples of narrative generation in commercial games, however, tend to be simplistic, resulting in repetitive and uninteresting stories. We approach these challenges and develop a system for narrative generation that uses a context-aware graph rewriting framework. We use a graph representation of the game world to create narratives which reflect and modify the current world state. These narratives are all validated to ensure that they are completable and we design a novel set of metrics which provide an approximation of narrative quality by analyzing the structure of the narratives generated. We apply these metrics throughout the generation process to ensure a certain level of quality. These metrics are then used to examine the relations between the scale of the game world with the quality of narratives generation. Additionally, we validate our graph-rewriting approach by comparing our generated narratives to other procedurally generated stories, as well as to authored narratives from commercially successful and critically praised games. The results show that our narratives compare favourably to the authored narratives. Our metrics provide a new approach to narrative analysis, and our system provides a unique and practical approach to story generation.

ABRÉGÉ

La génération procédurale d'histoire dans les jeux vidéo commerciaux permettre d'étendre le gameplay et d'approfondir l'univers du jeu à bas coup. Les jeux commerciaux qui utilisent ce genre de technologie le font de manière simpliste dont le résultat est souvent répétitif et ennuyeux. Nous avons développé un système qui crée des histoires procéduralement, ce système utilise une représentation de l'univers graphique (graphe) et réécrit cette structure afin de la faire évoluer. Cette représentation permet à notre système de comprendre l'univers, d'y modifier ce dernier et du fait créer des histoires qui respectent l'univers. Les histoires générées par le système sont validées afin d'assurer qu'elles sont logiques. De plus, nous avons défini des procédures innovatrices afin de mesurer la qualité des histoires générées. Ceci est accompli par l'entremise d'analyse de graphe. Ces procédures sont utilisées afin d'assurer une qualité minimale lors de la génération d'histoires. Par la suite, nous examinons les relations entre les histoires générées et la grandeur de l'univers ludique. Afin de valider notre approche, nous avons comparé nos histoires procédurales à d'autres systèmes procéduraux et d'histoires de jeux vidéo générées par la plume d'auteur. Nos résultats montrent que notre système se compare favorablement avec les histoires venant d'auteur humain. En sommes, notre approche permet d'analyser et mesurer des histoires en plus d'offrir une approche pratique et fonctionnelle à la génération procédurale d'histoire.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Modern trends in popular commercial video games have led to an increase in narrative-oriented games, where the person playing the game, the player, takes an active role as the lead protagonist. In such games, the gameplay is focused around completing a series of goals, where each goal drives the player forward within the story and makes new goals available. One popular genre of video games, called *Role-Playing Games* (RPGs), commonly use this format, having their players complete a number of goals which help drive the main goal of completing the game. Due to the fantasy oriented settings of these games, each of these goals are commonly referred to as *quests*, with the main goal being referred to as the *main quest*. A comparison in literature can be found with J.R.R. Tolkein's *The Lord of the Rings* trilogy, in which the main quest would be destroying the one ring, with the series of quests involving the specific tasks of the characters, such as the flight from Hobbiton, navigating the Mines of Moria, climbing Mount Doom, etc [26]. It is only by completing all these quests that the characters are eventually able to complete the main quest, and this is the structure that is reflected in the role-playing game genre.

Side quests, a specific type of quest unique to video games, are minor quests that have little or no relation to the main quest, but can still be completed by the player. These quests aim to give depth and richness to the world around them, allowing the player to explore the game's setting outside of the main quests while still following

a quest structure. An abundance of side quests can add much to the lifespan and richness of a game but are costly to develop and can detract resources away from the main quests. Due to these limitations, developers often devote less time to side quests, resulting in either only a few of them being present within a game or by having many similar, simple side quests with little variation or purpose.

The issues surrounding the cost of developing side quests has led various developers to experiment with the concept of *procedural narrative generation* as a cost-free alternative to hand-authored side quests. Beyond the complexities of generating appropriate accompanying text, however, naively generated story/quest structure tends to be easily recognized as repetitive by players [7, 9], with a consequent reduction in player interest and motivation. This problem is exacerbated by the need in real games for automatically generated quests to always be completable, and to avoid perturbing the more important main quests that drive the storyline. Providing a system which respects the constraints of the main quest, while still providing unique and complex quest structure is the goal of this research.

In this thesis we present a novel procedural narrative generation tool that was designed to overcome the short-comings of automatically generated quests in current commercial games, while still respecting the constraints and requirements of these games. Our focus was on using the system for the generation of quests, and more specifically side-quests for RPG games, while still maintaining a flexible system that could be adapted to work with a variety of narrative structures. Our system uses a *graph rewriting* approach that generates narratives based on patterns within a formalized representation of the game state or context. Use of graph rewriting allows

us to grow a story to arbitrary proportions and with arbitrary complexity through the repeated application of relatively simple narrative structure rules. Building the system with awareness of the underlying context ensures the quests match the game state, and can have observable and interesting impact on the game world, without overly interfering with the rest of the game design. This is in contrast to more traditional generation techniques that have focused on using quest templates, resulting in every quest having the same structure, to ensure that a quest is always valid and completable. The use of graph-rewriting techniques allow these conditions to be met while allowing instead for non-trivial, complex narratives that hold relevance within the game state.

Our approach incorporates the design of several simple metrics that measure the narrative quality in terms of the underlying structure. These metrics further allow for a formal analysis of the quality of the narratives generated within the system, as well as a means of comparing narratives against one another, ensuring that each quest is of an appropriate "scale" for a side quest. The various metrics evaluate basic features of the story structure, such as the size and repetition of player actions, while also examining more abstract properties such as the richness of the narrative and its effect on the game state. A validation for the choice of these metrics is provided by taking sample narratives from two large-scale, modern, commercial RPGs, *Skyrim* and *The Witcher*, and comparing the results to our narrative generation techniques. We also compare these same results to the narrative output of two other narrative generation tools.

A fundamental feature of our design is its reliance on a non-trivial representation of the game state. We provide an exploration into the effect of using our system with a variety of game states, showing the relation between the narratives generated and their context.

Quantifying story quality with respect to player experience is of course a difficult problem that depends to a large extent on the elegance of story description and other artistic properties. Our approach, however, shows that many aspects of basic structure are also important parts of the narrative design, and moreover are properties that can be captured and used to guide narrative generation.

Specific contributions of our work include:

- We describe a context-sensitive graph-rewriting approach to automatic story generation. This technique has been much less explored than search-based, goal-oriented approaches and combines awareness of game context with a flexible strategy for incrementally growing narratives to arbitrary complexity. Our approach utilizes validation to ensure that each narrative is completable.
- To quantifiably validate our narratives we define a number of game narrative metrics. These (mainly) graph-based measurements allow for numerical comparison of narrative quality, measuring both basic properties as well as abstract properties such as narrative "richness." These metrics can likewise be used during the process of generation to guide the generation process to create narratives with a user-defined level of quality.

- Using our metric design, we show our narrative generation framework can generate narrative structures of better quality than other narrative generation systems, comparable to the quality of manually designed side-quests in modern games.

In the next section we discuss related work in the field of narrative generation as well as narrative analysis. Section 3 describes our overall design for a narrative generation system using graph rewriting. In Section 4 we present the metrics used for our narrative analysis, followed in Section 5 by experimental analysis computing and comparing metrics for our system with another, successful narrative generation approach as well as with modern commercial games. Lastly, in Section 6 we conclude and discuss directions for future work.

CHAPTER 2
Related Work

In this chapter we present previous work in the field of procedural content generation, with a specific focus on the works relating to narrative generation as well as string and graph-grammar approaches to content generation. In Section 2.1 we present the studies which are most similar to our method of content generation, i.e. the grammar-based approaches to content generation. Section 2.2 discusses the planning-based narrative generation systems while Section 2.3 is focused on emergent narrative generation systems. Lastly, we discuss existing work on the analysis of certain features in narratives in Section 2.4.

2.1 Grammar Approaches to Content Generation

Our system uses graph grammars to generate stories using a set of story patterns. In doing so we will need to formalize the structure of a story such that we are able to represent stories graphically. Early work on providing such a formalism was presented by Propp in 1928 [20]. In this work, Propp presented thirty-one "functions" or *narratemes* to describe the journey of the hero in a Russian folk-tale as they relate to seven key story roles, such as the *hero*, the *villain*, and so forth. Using a thorough analysis of Russian folk-tales, Propp stated that all Russian folk tales followed these *narratemes* in sequential order, with the occasional narrateme missing. Thus to create a new folk-tale using Propp's formalism, a writer would need to create characters, setting, unique dialogue, etc. but the core events of the narrative

are unchanging. Propp called these base events the *fabula*, and the remainding pieces of the story the *syuzhet*. For our research, we are working on generating narratives at the *fabula* level, i.e. generating the base story structure while leaving the remainder of the design to be created by hand. Propp's work, however, is restrictive in that he provided only one *fabula* structure for his stories. Since we are dealing specifically with quests focused on the realm of Role-Playing games, we can expect that there may be many different base structures relating to the unique variety of quests undertaken by the player.

Following the structures provided in Propp's work enabled Lakoff to develop a grammar capable of generating fairy tales [8]. Lakoff's grammar generated fairy tale structures with the basic plot points defined by Propp, but additionally allowed for features such as embedded fairy tales. This involved the hero taking on minor challenges on their quest to resolve the main story conflict. These small challenges could take the form of a full fairy tale structure themselves, with provocation, conflict and resolution. These "embedded" stories which challenge the hero but do not detract from the main quest are similar to the *side quests* found in video games, the generation of which is one of the goals of this research.

In his analysis of Inuit folk-tales, Colby presented a slightly more dynamic system that used a *string-grammar* to generate unique story structures [4]. To accomplish this, Colby presented three abstract story categories: *motivation*, *engagement* and *resolution*. The main story was then represented first as a string generated by applying several rules relating to these story categories. An example of such a rule would be that an *engagement* event must always lead to a *resolution*. Each of these

three categories contain many possible narrative events known as *eidons*. An *engagement* could be a fight or a competition, for example. Replacing each category with an *eidon* results in a string of *eidons*, which, when read from left to right, gives the sequential order of events comprising the story. By allowing strings of arbitrary length with many possible events for each character, Colby's grammar was able to provide a much greater variety of story structures when compared to Propp. Rumelhart created a more basic grammar, but aimed his at being able to generate any kind of narrative [23]. To do this he represented narratives as events, where events, characters and settings could *allow* or *cause* certain additional events to occur. In our system, we follow closely these two approaches, by modelling our events as the actions the player will need to take to progress within the narrative, and having the events *cause* effects within the game world while *allowing* for other events to occur as a result of the previous event.

Some researchers argue that the formal story structures being developed were fundamentally inadequate. A critique of the method of presenting story structures in a formal way was proposed by Levi-Strauss [10]. He argues that stories cannot simply be presented in terms of structure, since this is ignorant of elements such as symbolism and cultural significance. This is a valid critique and remains applicable to most work in the field of narrative generation. Our research, however, is focused on narrative within games, and only in generating narrative structure while leaving the remaining pieces of the game to be designed by hand. While elements such as symbolism may be prevalent in a game narrative, they are less frequently applied to the narrative structure. In a critical and rigorous evaluation of story grammars,

Black and Wilensky argued against the expressive power of story grammars [2]. They argued that there were many valid story structures which could not be represented with any existing story grammar. This includes non-linear stories, in which parts of the story are told through flashback or are given out of order, and embedded narratives, in which a unique story occurs in the midst of a larger story. They further argued that certain non-story structures could be generated, such as instruction manuals. Garnham again stressed the importance of meaning and symbolism in stories, as well as stating that the sheer volume of possible story structures makes it impossible for a computer to be able to generate every possible story automatically [6]. One element which allows our system to avoid some of these pitfalls is that we have placed our system within a relatively narrow subset of all possible narratives, that of quests in role-playing games. This allows us to concentrate our rules on a certain type of narrative rather than aiming for a general narrative structure.

SQUEGE (Side QUEst GEnereator) was a quest generation tool developed by Onuczko et. al. [14]. Similar in purpose to our system, SQUEGE presented the quest as a graph and used graph rewriting rules in order to rewrite the original quest into increasingly complex structures. Focusing on quests involving gathering items or murdering game characters, SQUEGE was capable of generating quest structures which could contain embedded quests and branching quests. The system is also able of taking the generated quests and converting them into scripts for CD Projekt RED's *The Witcher* [15]. These quests would simply be quest structures where an external author would have to manually enter certain features such as character dialogues. SQUEGE's method of generation is most similar to our system, however SQUEGE

was ignorant of the social environment within the game, such as the relations between characters.

Graph-grammars have also been successfully applied to meta-narrative components of games, such as for the generation of role-playing game *dungeons* [5]. To do this, the dungeons are represented as directed graphs where the nodes represent each of the dungeons rooms. Rooms may contain specific events, or collectible items. By starting with a basic dungeon skeleton, transformation rules are used to create common RPG dungeon elements, such as boss-fights, locked-door puzzles, and creature encounters. This is accomplished by adding or modifying existing rooms in the dungeon. Lastly, a *shape-grammar* is used to create the shape of each of the rooms and in turn the shape of the dungeon. While not strictly related to narratives, this process is still similar in that it involves taking a player goal (to complete the dungeon) and using rewrites to expand this goal and create challenge for the player without relying on an external designer.

2.2 Planning-Based Narrative Generation Systems

Our system creates formal story structures given a game world, which is similar to the planning-based narrative generation systems. We define a planning-based system as being a system where the narrative created is essentially a formalized set of actions which aim to achieve a specific goal. Two informal sub-categories sometimes used to specify these systems further are *character goal* based systems and *authorial goal* based systems. Character goals refer to the goals of the non-player characters within the game whereas authorial goals refer to the goals set by the author. TALE-SPIN is an early narrative generation system which used a character goal-oriented

planning approach to generating narratives [13]. TALE-SPIN used a *map* to describe the game world. This map contained information about the locations, items and characters in the game world. To generate a story, TALE-SPIN would first look at the goals of one of the characters in the map. The system then simulates the actions of this character in the virtual world, biasing the world so that the main character is able to achieve their goal. Textually presenting the resulting actions taken by the main character to achieve their goal gives the final story.

The narrative generation system by Porteus et. al. aimed instead at achieving the goals of the author rather than the goals of the character [19]. The goal of their system was to allow for user interaction within a story, but keeping certain outcomes of the story predetermined by the author. To do this, the author specifies *constraints*, which are conditions that must be met at certain points throughout the story. Any changes to the game world by the user forces the system to re-plan the story events in such a way that they still achieve the author's constraints. The system was implemented in an interactive version of Shakespeare's *The Merchant of Venice* where the constraints were the key moments from the original play.

The military training tool by Zook *et. al.* also aims to achieve authorial goals [30]. Their tool generates specific training scenarios that aim to continuously improve and train the user using the simulator. The concept of small, self-contained scenarios are likened to that of quests in role-playing games, drawing similarities to our system. Each generated scenario is evaluated against the user's skill level to ensure that the scenario's always provide an increasing challenge for the user. The concept of

evaluating the results of generation against a given criteria is echoed in our research, where we instead evaluate our narratives against a set of metrics to measure quality.

For our system, the authorial goals are met by allowing the author to define all desired narrative structures manually. We do, however, generate quests based instead upon the possible goals of the characters within the game world. In this sense our system is similar to the IPOCL system by Riedl and Young. Riedl and Young's IPOCL system accounts for both the character and the authorial goals [21]. This system generates stories which aim to meet the goals given by the author, but in achieving this they ensure that each of the actions taken by the characters matches the goals of those characters. This again uses a planning approach to create a sequence of events that also forces the character goals to match the goals of the author.

2.3 Emergent Narrative Generation Systems

Our ReGEN system explicitly relies on a detailed description of the game world when generating narratives. In this sense, it shares several features with the *emergent* generation systems. In an emergent system, there is often no formally defined narrative, but the world is created in such a way that a narrative is expected to "emerge" from the interactions of characters and objects within the world. The *Virtual Storyteller* is such a system, in which the generated narrative was simply a retelling of events which occur within the simulated world [25]. To achieve interesting interactions, each character in the world is given a unique personality and goals, as well as relations to other characters within the world. In simulating the world, the actions of each of these characters comprises the story. A later version of this

system provided a modification wherein the system would only pick events relating to one particular protagonist, as it was found that the system would often output a disconnected set of events that did not seem to formulate any sort of story, making the story uninteresting for the users [24].

Chang and Soo also created an emergent system which focused on creating socially reactive AIs, this time set in the world of Shakespeare's *Othello* [3]. The AI in this system aimed to mimic each of the characters from the original play, and gives each character beliefs and motivations, allowing for deception and misjudgement as potential story events. The intention of this system was that the content of the original *Othello* play would emerge out of the interactions between AIs, or that perhaps unique but equally as dramatic situations would occur instead.

Our system, while still creating formal narratives, takes some features of these systems by modelling relationships and tracking how each event in the story will modify the interpersonal relations of the characters in the game. MEXICA, another emergent system, takes a similar approach by dynamically picking story events based off of a history of all past events in the game world and how these events changed the relations between characters [18]. The emphasis on doing this is to make narratives that feel *believable* given the world in which they occur. Our system differs in that we are still focused on creating formal narrative structures, rather than just dynamically choosing a sequence of events with the expectations that these events will constitute a narrative.

There are many emergent systems which are focused on user interaction rather than the AI simulation approach used in the above three systems. In Cavazza *et.*

al.'s recreation of *Madame Bovary*, the user physically reads the dialogue lines into a microphone. The game detects the user's emotions from this reading and updates the characters based on the perceived emotional state of the user. The game character's reaction to the user and how this modified the game world resulted in narratives "emerging" from the user's actions.

Façade is another such system where the user plays the role of an old acquaintance to the two main characters [11]. The user is able to talk to both characters in the game, and what they say changes the relations between the characters, as well as the character's relation to the user. Although largely emergent, *Façade* did include formal authorial constraints, which means regardless of user actions the story will always reach particular dramatic points. This was done to ensure a satisfactory story structure, with a build-up, climax, and denouement. The dramatic points, however, change slightly depending on the behaviour of the user, and as such still allows for a unique story on each play-through.

Prom Week, as with the above two games, focuses on creating story out of the user's power to manipulate relations in the game world [12]. These three games are sometimes referred to as *social games*, due to the fact that the games center around social relationships. In *Prom Week* the player can pick one of many possible characters, modelled after various types of high-school characters. They start the game several days before the prom and have to accomplish certain social goals before the end of the prom, such as making friends, joining groups, or even becoming the prom king/queen. To do this they have to interact with the other characters in the game and manipulate the relationships of those characters in such a way that it

achieves the goals of the player. Since our system relates to role-playing games, we do not focus on the manipulation of relationships as game-play. However, we do model relationships and the user's actions can modify any number of these relationships. Doing this means that there can be unintended consequences to the user's previous actions appearing in later game narratives. As such relations still play an important part in our system.

2.4 Metrics for Analysing Narratives

One important feature of our system was to provide a series of metrics to analyze the quality of the metrics created by our system. In previous experiments, analysis was often done by means of a human survey. Peinado and Gervás' experiment had participants rank a generated story based on *linguistic quality*, *coherence*, *interest* and *originality*[17]. We examine similar metrics to *coherence* and *originality* in a formal manner in our work while *interest* and *linguistic quality* are too dependent on personal taste to be analysed formally. *Novelty*, as with *originality*, was likewise analysed with a set of metrics by Pérez *et. al.* [18]. The aim was to ensure that each new narrative generated by their system seemed original when compared to previous generated narratives. The development of such a metric for our system remains future work, but we understand the importance of using such metrics to examine narrative quality over multiple narratives.

A set of formal metrics to specifically analyse narrative conflict were presented by Ware *et. al.* [28]. The metrics defined focused on *balance*, *directness*, *intensity* and *resolution*. Several sample narratives were evaluated using a human study where the participants were asked to rank the stories according to each of the four metrics.

These were then compared to the rankings as determined by the formulas created for each of the metrics. For our tests, we were more interested in the quality of narratives generated and use these tests also as an argument for the quality of the narratives. We instead use examples of hand-authored and other generated narratives in order evaluate narrative quality as well as supporting the argument that our metrics correspond to accurate measures of narrative quality. As well, measurements of conflict instead focus on more simple structural features, such as the number of fights present.

GADIN was a narrative generation tool that aimed to create narratives in the style of a soap-opera. These were analysed using a Turing-test where participants were given output from the system as well as summaries of actual episodes of soap-operas and were asked to pick which they felt was computer-generated [1]. While the Turing-test was designed to show the similarities between authored soap-operas and the generated soap-operas, their results gave no indication of quality. These results do not state if the soap-opera generated was a good soap-opera, or even a superior soap-opera to the hand-authored alternative. This measure of quality was the intent of our metrics, and removes the necessity for a Turing-test since we can make formal comparisons of story structure using these metrics while at the same time making formal statements regarding narrative quality.

CHAPTER 3
Narrative Generation

Our narrative generation system uses graph rewriting techniques as the main generation process. This involves representing the *game narrative* formally as a graph and then applying rewrite rules to this graph in order to generate a more structurally complex narrative. This led to the name of our system, ReGEN, which stands for **RE**writing **G**raphs for **E**nhanced **N**arratives. As was previously mentioned, one of the features of the system is to have the narratives make changes to the game world, and make all subsequent narratives respectful of these changes. This gives the player the sense that their actions have meaningful consequences in the game world. Likewise, these effects can lead to specific narratives where certain events occur because of the actions the players have taken in previous narratives (we quantify this formally in Section 4 under *Narrative Richness*). The *game world* is also modelled as a graph, using nodes as objects and labels to define relations between objects. While our thesis focuses on creating *quests* for *role-playing games*, it is designed generally enough to be applied to multiple narrative forms. Keeping in mind these intentions, we defined a set of criteria that would be suitable for such a system:

- Being powerful in its ability to generative narratives but also easy to implement and simple for potential users to understand.
- Generating narratives which are reactive to the game world environment in which they occur, and can in turn influence said game world.

- Providing uniqueness between narratives, and variety to the individual actions occurring within the narrative itself.
- Being able to generate complete and valid narratives, that do not violate any properties of the game world.

Our generation process involves two main steps. The first step involves generating the starting narrative with a few general actions, enough to constitute a basic narrative. It is important that this narrative have a defined beginning and end and be completable. The first step of generation is done using a set of rules which we call the *Initial Rewrite Rules*. Then, using a second set of rules called the *Secondary Rewrite Rules*, we rewrite the starting narrative to expand the story. This can add complex features such as player choice. This process of rewriting never modifies the beginning or end state of the narrative, thus preserving the completeness of the narrative.

The creation of both these sets of rules, as well as the design of the game world is left entirely to the user of the ReGEN system, allowing for complete authorial control over the potential narrative space. Using graph representations of both the game narrative and game world makes it easy for a user to visualize each. Additionally, we find that the process of graph rewriting is intuitive enough that even a user unfamiliar with graph rewriting should be able to quickly understand the basic logic behind the system. This is to help fulfil part of the first criteria, which states that the system must be easy to implement and understand.

Figure 3–1 displays the architecture of our system and its five central components. In this section, we will discuss the functionality of each of the five components,

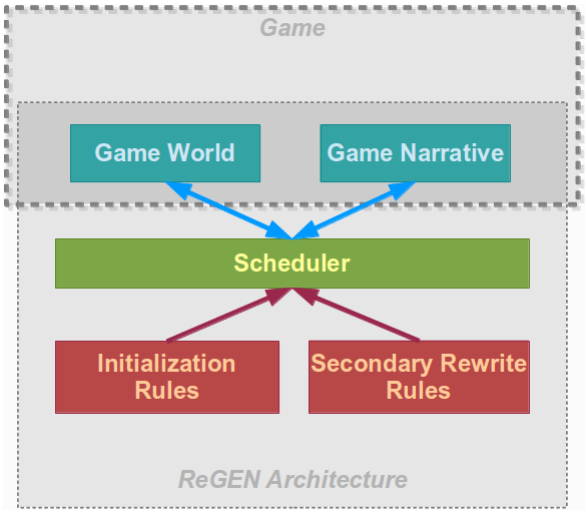

Figure 3–1: The architecture of the ReGEN system, broken down into its five central components

starting by describing the graph representations of the game world and game narrative. Following this, we will discuss the structure of both the initial rewrite rules, and the secondary rewrite rules. Lastly, we will discuss the *scheduler*, which is responsible for the generating narratives using the above four components. Note that while the scheduler is not intended to be editable by the user of the system, it is still customizable in several respects which can greatly influence the actual process of generation. This customization once again allows the user to be largely responsible for the behaviour of the system.

3.1 Game Components of ReGEN

There are five main components to ReGEN. The first two, the *game world* and the *game narrative* both must be interfaced directly to the game. The game world must be an up to date representation of all objects and relations within the game, and the changes made by the ReGEN system to this graph must then be applied

to the game itself. Likewise the game must have an interpreter that can use the narrative output from the system and convert it into a format that can be used within the game. To simplify this, both the game world and game narrative were designed to be as similar to the structures of modern role-playing computer games as possible. Further validation of this is shown in Section 5 where we are able to both convert Bethesda Softwork's *Skyrim* world into a format usable within our system, and also convert the quests from *Skyrim* and CD Projekt RED's *The Witcher* into our narrative format successfully.

The remaining components of the system, the *Initial Rewrite Rules* (IRRs), *Secondary Rewrite Rules* (SRRs) and *Scheduler* are all independent of the game itself, and are only used by the ReGEN system. This makes integration of the ReGEN system into existing games conceptually simple, requiring only that there be an existing structure to handle changes in the game world and one to interpret narratives into a playable format, which are common features in most commercial role-playing games.

One of our main criteria for this system is having a strong correlation between the actions the user takes within each narrative and their corresponding changes in the game world. To do this we generate our narratives by looking for relational patterns in the game world based on which interesting narratives may be built. Once the narrative is created, the path the user takes through said narrative in turn causes changes to the game world. In doing so, when a new narrative is generated, it is based around the state of the game world after having completed all previous narratives.

In this section we will look at the user's flexibility and control over each element of the system by examining each component formally. Following this we will provide a brief discussion as to how each game component may interface within an actual game.

3.1.1 Game World

The *game world* is a directed labelled multigraph where we refer to nodes in the graph as being *objects* and the edges as being *relations*. An object is defined as being a node with a unique identifier, a *type*, and a set of attributes. For most games, and in particular Role-Playing Games (RPGs), there may be multiple types of objects. While the types may be customized depending on which objects are present in the game world, for our purposes the main object types were *Non-Player Characters (NPC)*, *Items*, *Locations*, the *Player* and *Enemies*. Note that a *non-player character* refers to any character within a game who is not controlled by the player. This generally refers to characters the player can interact with, such as inhabitants within a village.

Any object may be connect to any other object by a *relation*, which is represented as a directed edge in the graph. The relation edge is directed from the object who holds a relation to the specific object with which they hold said relation. We define a relation as having an *identifier* and an optional *reason*. Figure 3–2 summarizes a formal definition of the game world used.

Figure 3–3 shows an example of a very basic game world created with ReGEN consisting of only two objects and one relation. The two objects are both *NPC* types, indicating they are non-player characters. The identifiers of each of the objects are

$GameWorld = (O, R, ID, Type, Attr, Attr_{key}, Attr_{value}, R_{ID}, R_{reason}, R_{from}, R_{to})$
$n : O \times \emptyset$
$O : ID \times Type \times Attr$
$ID : O \rightarrow String$
$Type : O \rightarrow \{NPC, Player, Location, Enemy, Item\}$
$Attr : O \rightarrow Attr_{key} \times Attr_{value}$
$Attr_{key} : Attr \rightarrow String$
$Attr_{value} : Attr \rightarrow \{String, Boolean, Integer\}$
$R : R_{ID} \times R_{reason} \times R_{from} \times R_{to}$
$R_{ID} : R \rightarrow String$
$R_{reason} : R \rightarrow String$
$R_{from} : R \rightarrow O$
$R_{to} : R \rightarrow O$

Figure 3–2: Formal Definition of a Game World

shown at the top, and refer to the name of each NPC. There are two additional attributes provided here, *gender* and *alive*. These provide further details on each character: whether they are male or female, and whether the character is alive or not. These details are important when defining rules. For example, in many narratives it is required that each character in the narrative be alive, and this can be done by checking the *alive* attribute. Since the player is able to kill characters within the game world, we expect that the alive attribute will change for many NPCs, making this an important attribute to include. The attributes are all user-defined and the user may add as much or as little detail to the objects as they feel is necessary for narrative generation.

Figure 3–3 additionally shows an example of a relation between two objects. In this case we see that the King Arthur NPC *hates* the Morgan Le Fey NPC. In this case the *hates* is our relation's *identifier*. The second part of the relation gives the

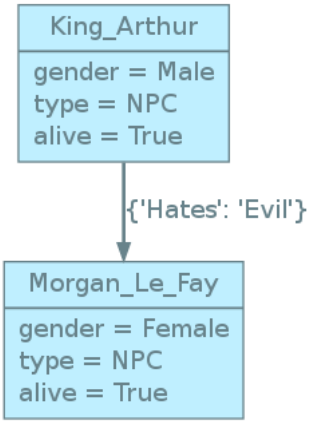

Figure 3–3: A basic example of a relationship between two NPC objects

reason, in this case *evil*. To sum up, we can say that this game world states that King Arthur *hates* Morgan Le Fay, because she is *evil*.

It is important to note here that all relations are directed and therefore not bi-directional. In this case, we have a relation between King Arthur and Morgan Le Fey, but we do not have a relation from Morgan Le Fey to King Arthur. Thus, it is entirely possible for Morgan Le Fey to have a different relation towards King Arthur. Likewise, there can be numerous relations between characters, each for different reasons. King Arthur may hate Morgan Le Fey for one reason, but love her for another reason. He may even hate her for different reasons. This openness allows for very complex interrelations between objects that can more closely represent the interrelations desired to create a realistic game world.

Of additional note, while the number of possible relations is left in control of the user, it is important to use a minimal number of relation identifiers. This is due to the generation process and will be explained in more detail in the next section.

Essentially, the system will be looking for specific relation patterns, and by using a minimal number of identifiers, it increases the likelihood of these patterns existing. For the purposes of creating a rich social environment, we defined the set of possible inter-NPC relation identifiers as *loves, hates, friends, allies, enemies, trusts, distrusts* and the slightly more specific *blackmailing*.

Figure 6–19 found on page 125 shows a larger and more detailed game world, which was later used for testing. It demonstrates the different types of objects and how they are related together. For example, NPCs generally *own* Items and *live* in Locations, etc. We believe that this system is a simple and intuitive way to represent the game world that reflects the way data is stored in most commercial RPG games.

3.1.2 Game Narrative

The *game narrative* is represented as a directed acyclic graph with labelled nodes. For a narrative, each node represents an *event*. An event is an abstraction of a specific goal to be achieved by the player. As with the game world, each event is defined using a unique identifier and a type. Additionally, each event is given a target. The unique identifier remains the same as described above in that it is simply an identifier for each event.

The type is used during the generation process and for metric analysis, as explained in Section 4, to help determine narrative *uniqueness*. In short, looking at the types of each narrative event allows us to examine the number of unique actions taken by the player in each narrative. The *target* refers to the object in the game world which is the focus of the specific narrative event. As an example, if in a narrative the player is expected to travel to a specific location, then the event type would

be a "Go To" event and the target would be the destination of the player. Reaching the destination would result in the "completion" of that event. We call the set of all target objects for a narrative the *cast* of that particular narrative.

The edges in a narrative indicate the ordering of the events. Each event is expected to have at least one incoming link and one outgoing link, with the only two exceptions being the starting and ending events. When the player completes an event then the next event in the narrative is whichever event has an incoming link originating from the current event. If the current event is linked to one or more events, then we have a *branching narrative*. Given that our research is focused around interactive narratives, it is common to have points where the character is given a choice about which action they wish to perform next, resulting in two or more possible narrative events. In this sense, the narrative experienced by the player is the path the player takes from the start event to the end event within our game narrative. Since we do not restrict the player's path through the narrative, if there are any occurrences where an event links to two or more events, then we say we have a branch in the narrative. Allowing for a branching narrative is one of the advantages of representing the narrative using a graph as opposed to alternative methods such as the string representations used by Colby, Rumelhart and Lakoff [4, 23, 8]. The formal definition of a game narrative is provided in Figure 3–4.

Note also that we have the restriction that a narrative be acyclic. This is to prevent infinite actions, where the user is able to get trapped performing the same action numerous times without progressing through the narrative. With this restriction it is assured that the user will never visit the same event more than once.

$GameNarrative = (E, L, ID, Target, Type, L_{from}, L_{to})$
$n : E \times \emptyset$
$E : ID \times Type \times Target$
$ID : E \rightarrow String$
$Type : E \rightarrow \{fixed\}$
$Target : E \rightarrow O$
$L : L_{from} \times L_{to}$
$L_{from} : L \rightarrow E$
$L_{to} : L \rightarrow E$

Figure 3–4: Formal Definition of a Game Narrative

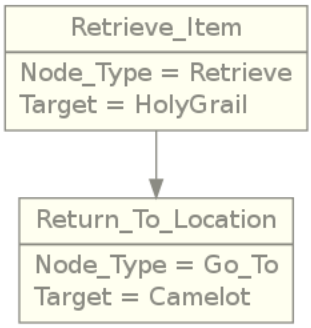

Figure 3–5: A basic sample of a narrative relating two narrative events

Figure 3–5 shows a basic narrative consisting of two events and a linking between these events. The two events are identified as "Retrieve Item" and "Return to Location". The *Node Type* refers to the type of each event. The first event is a retrieve event with the target being the Holy Grail. This indicates that to complete this event the player would have to retrieve the holy grail. Likewise to complete the second event the player would have to go to Camelot. We have a single link which states that the retrieve item event must occur before the return to location event. If the player returns to Camelot before retrieving the Holy Grail, then they will not complete the "return to Camelot" event. Each event must be performed

in the correct order. This is intended to mimic the structure of most commercial RPGs. In these RPGs the player is free to explore and interact with the world at will, however in order to progress through a quest they must complete an ordered sequence of actions.

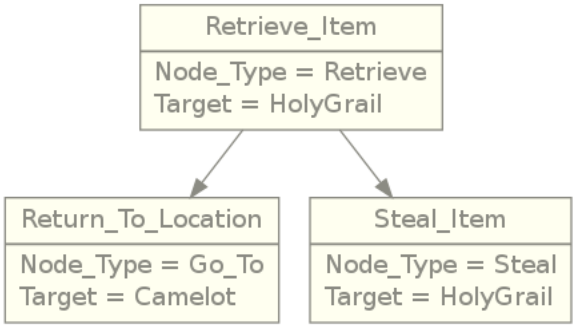

Figure 3–6: A basic sample of a branching narrative with three events and two possible player paths

Figure 3–6 demonstrates how a branching path would appear in a narrative. In this example there is a new added event, "Steal Item" which indicates that to complete the event the player must steal the Holy Grail. The two links show that from the retrieve item event, there are two possible next events. Since the player is responsible for choosing their path through the narrative, they are therefore able to select whether to return with the Holy Grail to Camelot, or to steal the Grail to keep it for themselves. Since neither of these two events has an outgoing link, they are therefore two possible end states for the narrative.

As with the game world, we find that this representation of narrative is intuitive and easy to understand when described or seen visually. It likewise mimics the structure of modern RPG quests representing a quest narrative as an ordered sequence

of actions which need to be completed, except where a player choice is provided. Validation for this structure is found in Section 5 when we are able to convert the *Skyrim* and *Witcher* narratives into this format successfully.

3.1.3 Interfacing Components within a Game Structure

Each game component was designed in such a way that they can be easily interfaced within the common structure of RPGs. To do this, we mimicked the structure of popular RPG franchises such as *Skyrim* and *The Witcher*. Within these games, *Skyrim* more noticeably, different NPCs hold different relations to each other. In current games, this is often used to define AI behaviours, for example in *Skyrim* if two NPCs "meet" (are within a certain radius) the game will base their next behaviours on the relations between the two; If they are friends, the NPCs may start a dialogue, or if they are enemies they may start fighting. Although the current use of relations is largely oriented around AI behaviour, we feel that we can use these relations for the purpose of narrative generation. Being already used in many games, it would therefore be straightforward to interface our game world graph within such a system. All that is needed is to ensure that any changes in the game world are reflected within the game itself, for example if a narrative makes two friends into enemies, the AI behaviour should be different at their next encounter.

The game narrative graph was also designed to mirror the current conventions of modern RPGs. Within most RPGs, players can perform certain actions, clicking on an item can pick it up and add it to the set of objects owned by the player (often called the *inventory*). Likewise attacking a character with a weapon may cause damage to them and eventually "kill" them, effectively removing the NPC

from the game permanently. Other actions, such as moving the player to certain game coordinates, clicking on NPCs to interact with them, are all a form of *event*, which refers to a change within the structure of the game. Linking many of these actions together is used to define the *quests* within a game, where the player is given a list of actions they must complete in a certain order. Completion of all of these actions "completes" the narrative. Thus, in our system we define our narratives in the same way, as a series of actions. Within a real game, the possible events which could occur within a narrative would be all the possible events that may occur within the game world. By making our narratives an ordered sequence of actions, we again keep the structure of modern RPGs.

Understandably, there are other features to modern games that must be taken into consideration during integration. The narrative, for example, must be communicated to the player in some fashion. Often this is by simply visually showing the player which action (or choice of actions) they need to perform next within a narrative. This is often done by displaying text such as "Talk to X" and "Go to Y". We designed the *target* to facilitate this process, since in order to convert each action to a string, the designer would simply need to take the type of the event and append the name of the target. Other common RPG features, such as dialogue, are less trivial and currently remain as future work for the system.

3.2 Rule Components of ReGEN

Graph rewriting, the approached used within our system, is a generalization of the string rewriting strategy commonly associated with language grammars for computers. In graph rewriting one defines rules that search for patterns in graphs

(as opposed to strings), rewriting the resulting matched areas to produce a new graph. This allows us to dynamically change any aspect of the graph arbitrarily. A graph rewriting system thus consists of an initial graph along with rewrite *rules* which, much like a grammar rules, consist of both a left-hand side and a right-hand side. The left-hand side shows the pattern being searched for in the main graph, and the right-hand side shows the way that a match of the pattern will be rewritten if found in the input graph. Note that within our system we define a rule as consisting of two rewrite rules, the left hand side contains both a game world and game narrative condition, whereas the right hand side contains the rewrites for both the game world and the game narrative. This structure is shown in Figure 3–7.

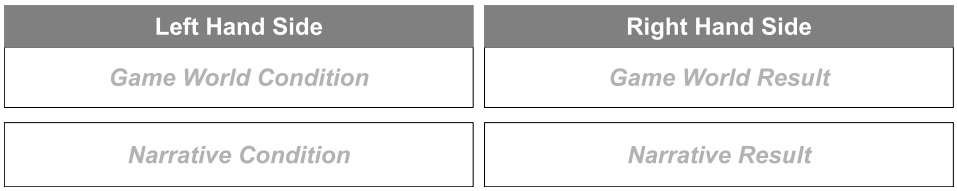

Figure 3–7: The rule definition used for our system, consisting of two conditions on the left-hand side and two results on the right-hand side.

The process of searching for a pattern within a graph is NP-complete and is a graph isomorphism problem [27]. We find, however, that the cost of this search is minimal on the size of graphs used within our system. Having labelled edges and nodes further reduces this search time. For example if we are looking for two NPCs who love each other, we search only through NPC objects and only through "love" relationships.

3.2.1 Initial Rewrite Rules

To understand the construction of the initial and secondary rewrite rules, some basic knowledge of the generation process is necessary. We will provide this in short here and expand upon it more formally in the subsequent section. The first step in generating a narrative for ReGEN is to create an initial narrative by finding a "potential story" in the game world. This is accomplished by creating a game world *condition* that may result in a specific narrative outcome. For example, if one NPC hates another NPC in our game world, they may ask the player to kill said NPC. This gives us a condition that one NPC must hate another NPC, with the outcome being a narrative in which the player must find and kill the second NPC at the request of the first. An *initial rewrite rule* is needed to accomplish this. An initial rewrite rule is composed with only a *Game World Condition* as the left-hand side, and a *Narrative Result* and *Game World Result* as the right-hand side. Note that the "rewriting" portion of this rule is essentially rewriting the empty narrative graph into one of the possible starting narratives. To define a rule, the user must define each of these three sections.

As an example of a more complex game world condition, imagine for the murder rule that if instead of desiring simply one character to hate another, we want the invoker to be a woman, who wishes her husband to be dead in order to be with her lover. Figure 3–8 shows an example of such a complex game world condition representing an adulterous love triangle. For this condition to be satisfied, there must be one female NPC who is married to a male NPC whom she hates, but at the same time loves a different male character. The *s used in the example are used to

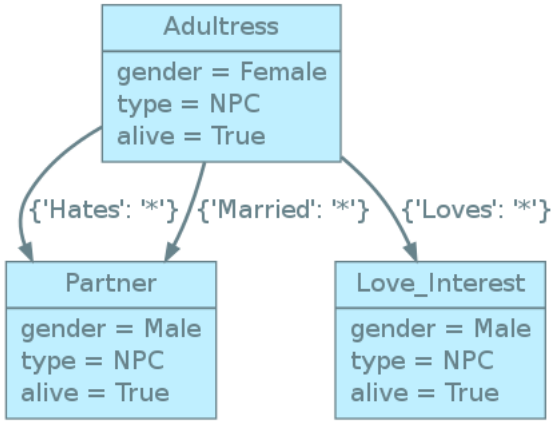

Figure 3–8: A sample game world condition showing an "adulterous" character

indicate *any reason* meaning that it does not matter why the adulterous character is married to and hates her husband or why she loves the other character. The author may choose to use reasons if they wish for a more specific condition. The ability to use or not to use reasons is left up to the user and is dependent on the amount of authorial control they want to have over the system.

In this example we indicated gender in our condition meaning that the game will ensure that this condition only exists where it is a female character as the adulterous character with male characters filling the partner and love interest roles. If the user is not interested in having this gender mix, then they simply would not include gender in their game world condition. This results in the ReGEN system looking for the same relations, but between any three NPCs of any gender mix. Once again, the amount of specificity is explicitly defined by the user using the system. Also of note is that the unique identifiers are not searched for in the game condition. The

identifiers will, however, be used as the *cast labels* used to define the narrative's cast. The narrative cast becomes important at a later point in the generation process.

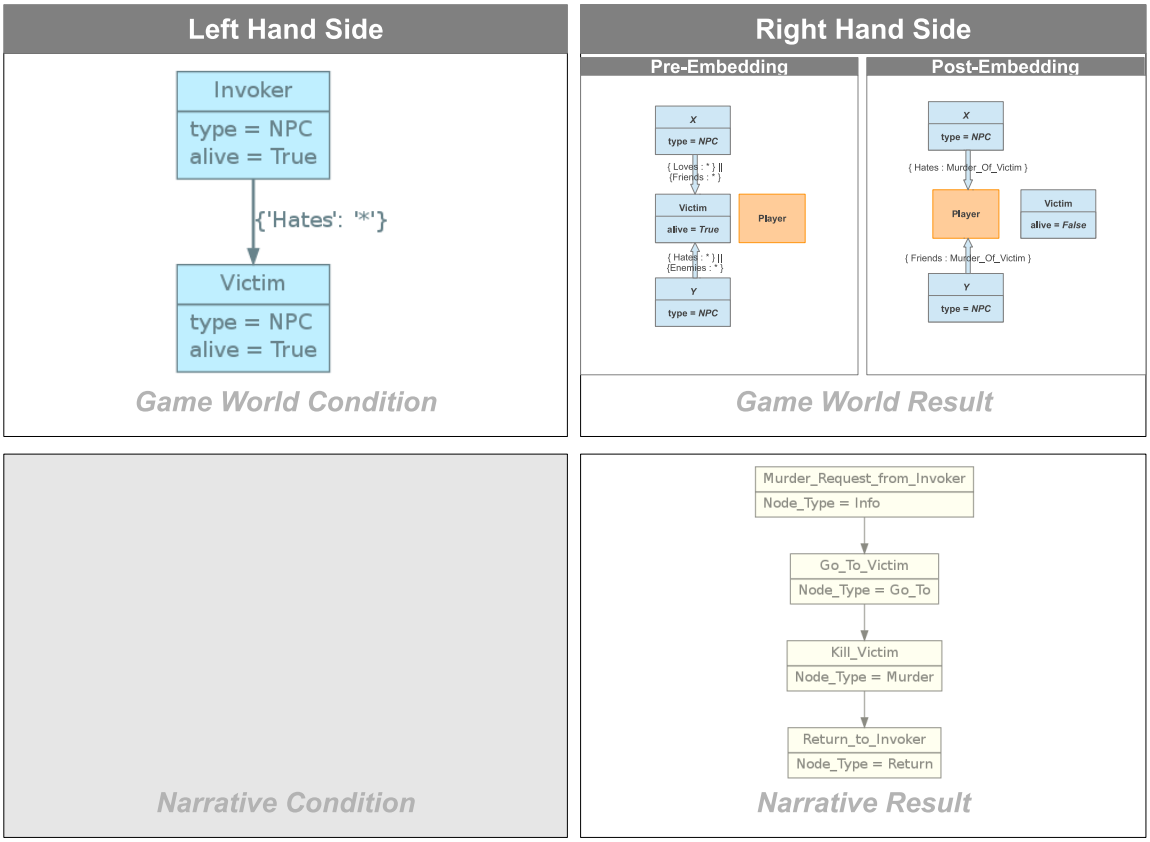

Figure 3–9: The Murder initial rewrite Rule

Figure 3–9 gives a graphical representation of the murder rule we have defined, keeping the original murder condition as opposed to the adulterous one. The left hand side refers to the *conditions* of the rule and the right hand side refers to the *results*. Note that for an initial rewrite rule we do not have a narrative condition, and that is therefore left greyed out. This condition will become important when describing our

next set of rules. The *Game World Condition* shows our hate relationship, where we ensure that the hates relationship occurs between two living NPCs.

The *Narrative Result* of an initial rewrite rule refers to the narrative that will be created provided the game world condition is met. With the murder example, we said that if there is any instance of one NPC hating another NPC, then a potential story involves having the player go and kill the second NPC. This is the narrative result, and is defined using the definition of the game narrative given above. The user may create whichever sequence of events they feel will best represent this narrative result.

The narrative result in our example shows a narrative consisting of four events where the player gets the murder request from the invoker NPC, goes to the victim, murders them, and then returns to the invoker. Note that *targets* are not assigned at this stage in the narrative, as they will be filled with whichever two NPCs fulfil the game world condition. Allowing the user full control over the narrative definition once again returns to our criteria of giving as much control over the narratives generated to the user as possible. Although our system does provide narrative evaluation tools, it is still the user who ultimately determines what is a good story. It is also the responsibility of the user at this stage to ensure that the narrative is valid and completable.

The *Game World Result* represents the effect on the game world that will occur as a result of the player completing the quest. The game world results are different to a narrative result in that they are assigned to any event in the narrative which causes a change in the game world. For example, in the murder quest, we expect

the act of murdering will result in the death of the victim NPC. This change will not occur until the player completes the murder event. The *Game World Result* is therefore a series of rules which are applied to specific events and the results will only occur if the player completes that particular event. This is important since, with a branching story, there may be certain events which the user does not complete.

The game world result in Figure 3–9 shows a summary of the results from the player completing this quest. Since it is a linear quest, we only have one possible result, however, the game world result is in general conditional, where certain relations will change only if those relations existed in the game world at the time the user completed the event. The conditions for this we call the *pre-embedding*, and the results are called the *post-embedding*. In this instance the pre-embedding checks for any NPC(s) who have a *Loves* or *Friends* relation to the Victim. These NPC(s) will then in turn hate the Player character for murdering the Victim, as shown in the *post-embedding*. Conversely, any NPC(s) who had a *Hates* or *Enemies* relation to the Victim will in turn gain a friendship relation to the player for the murder of the victim. Lastly, the victim loses all their outgoing relations upon death, since being dead they no longer have feelings towards any of the other NPCs (although NPCs maintain their relations to the dead character) and their alive value is set to False, which is again shown in the *post-embedding*. For simplicity, further examples of rules

use pseudo-code as opposed to graphical representation of these results. A complete set of all the initial rewrite rules we used for testing are found in Appendix 6.1 [1] .

3.2.2 Secondary Rewrite Rules

Following the creation of an initial story, the next step of the narrative generation process is to begin rewriting this narrative to make it more "interesting". This involves creating a new set of rules which are applied *after* creating the initial story. We call these the *Secondary Rewrite Rules* since they occur during the second phase of generation, which involves the rewriting process. This set of rules is defined similarly to the initial rewrite rules, but in this case we must also define a narrative condition as well as a game world condition. In order for the rule to be applied, both the narrative condition, and the game world condition must be met.

The *Narrative Condition* is represented as the particular event pattern we are searching for within the narrative. This event pattern is represented using the definition of a game narrative we provided above. The second stage of generation does not involve creating new narratives, rather we rewrite specific events in the narrative into new events. The narrative condition specifies which event pattern we are searching for. If it exists, then this event pattern is replaced in the narrative with the narrative result. While our rules generally keep the event pattern as being a single event, this is not necessary and it is possible for the pattern to be any length or complexity provided it is a formal game narrative.

[1] Please Note that within the appendix, * is written as 'N/A'. This is simply due to the output function and does not indicate any change in functionality.

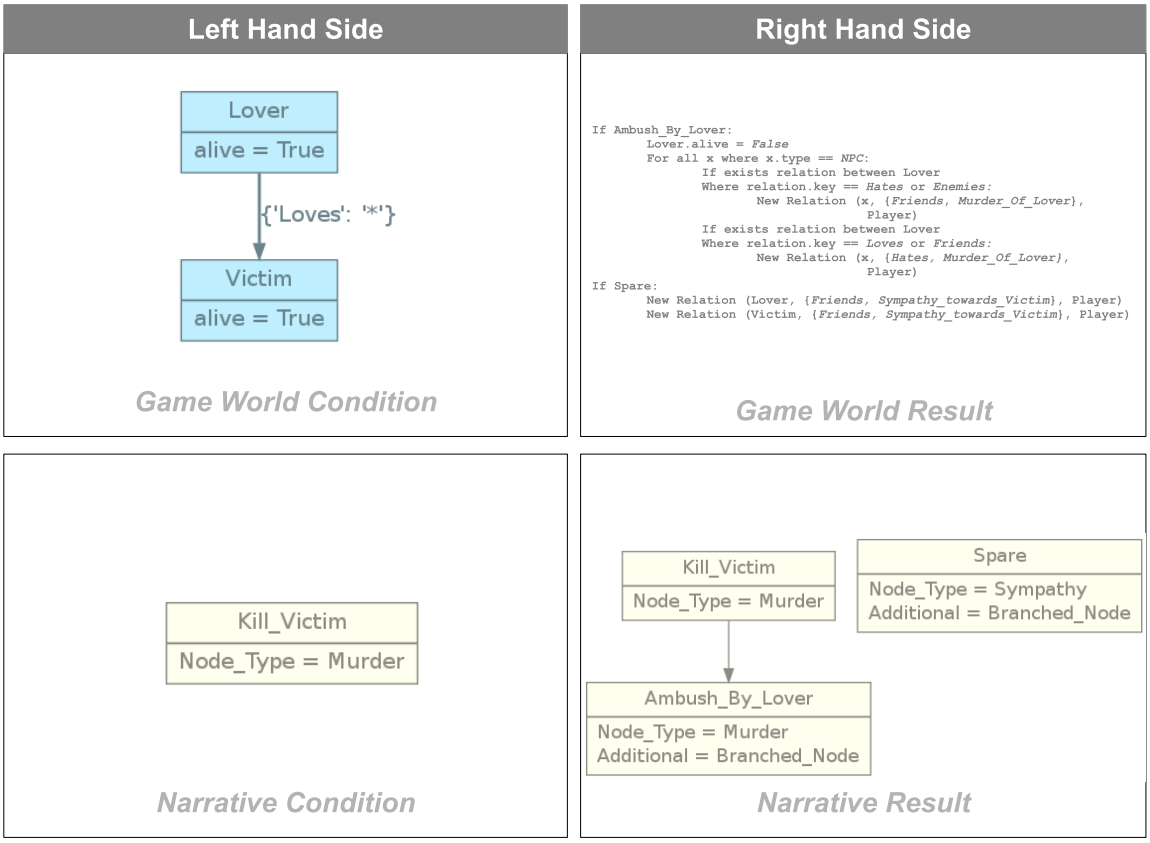

Figure 3–10: An example of a Secondary Rewrite Rule

Figure 3–10 gives a visual example of a secondary rewrite rule used within the ReGEN system. This rule relates back to the murder rule discussed in the initial rewrite rules section. The purpose of this rule is to allow the murder victim's loved ones to ambush the player after the murder has occurred. The left hand side shows both the game world condition and the narrative condition. The game world condition provided shows that in order for the condition to be met, there must be an NPC with a loves relation to the victim. Likewise, there must be a murder event occurring somewhere within the existing narrative. If both these conditions are met, then the original murder event is rewritten into three events, as shown in the narrative result. The first event is the murder event, the second is an ambush by the lover of the victim, where the player is forced to kill an NPC who has a love relation to the victim being murdered. The last event allows the player to spare the victim's life. Note that during the rewrite, all previous incoming edges to the narrative condition become incoming edges to any events in the narrative results which do not have incoming edges. Likewise all the outgoing edges to the narrative condition become outgoing edges to any events in the narrative results that do not have outgoing edges. In this case, any previous incoming edges to the murder event will become incoming edges into the murder and spare event, and any outgoing edges will become outgoing edges to the ambush by lover and spare event. This means that we have effectively created a branching narrative, since there are now two paths available to the player. The first path available involves the player killing the victim, and then being ambushed by the victim's lover, thus having to kill them too. Alternatively, the player may spare the victim's life and thus not have to fight the victim's lover.

The game world results for this rewrite are conditional in this case, since we have two alternate paths for the player. Note that we keep the kill victim event, which has the modification we described above where all NPCs who liked the victim now hate the player and vice-versa. If the player takes the kill victim path, they will now also kill the lover, meaning that now all NPCs who had loves or friends relations to the lover now also hate the player and any NPCs who had hates or enemies relations to the lover are now friends with the player. This modification is shown under the *If Ambush By Lover* portion of the pseudocode, indicating that this modification occurs only if the player completes the ambush by lover event. If the player spares the victim, then both the victim and the lover will now have a friends relation to the player due to their sympathy towards the victim. This modification is shown under the *If Spare* section.

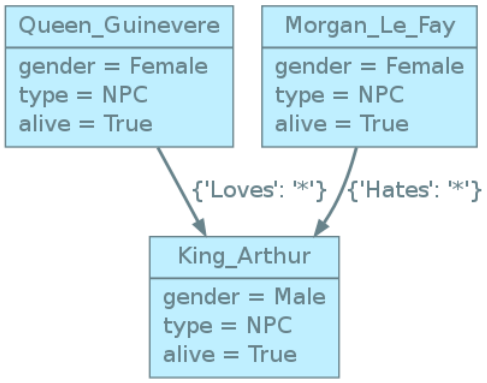

Figure 3–11: A sample game world containing three NPCs with two relations

As a complete example, imagine we have a game world with three NPCs, *Morgan le Fey*, *King Arthur* and *Queen Guinevere*. In this world, *Morgan le Fey* hates *King Arthur*, and *King Arthur* and *Queen Guinevere* both love each other. This game

world can be seen in 3–11. Imagine we have defined the IRR given in Figure 3–9 which states that if one NPC hates another, then a potential quest structure would have the player murder the enemy of that NPC. With this game world, a plausible narrative would be one in which the player aims to kill *King Arthur*, as requested by *Morgan le Fey*.

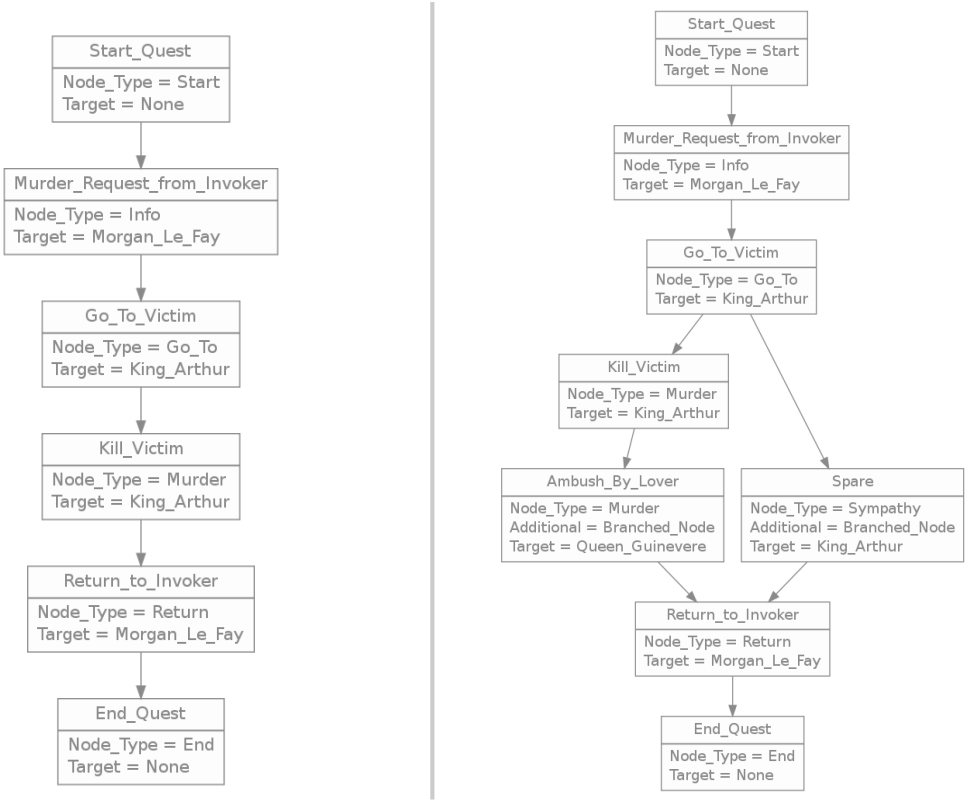

Figure 3–12: The initial murder story generated *(left)* and a rewritten version using the secondary rewrite rule provided in Figure 3–10 *(right)*

The resulting narrative structure could be represented as four events, where the player receives the murder request, goes to the victim, murders the victim, and then

returns for a reward. This structure is shown on the left side of Figure 3–12. In the next phase, the system will see if there is a valid way to rewrite this narrative using the SRR in Figure 3–10. It will note that there is an event with *Murder* as the *type*, meaning that the narrative condition has been met. At the next step, the system will replace the *Victim* object in the game world condition with the target *Victim* in the murder event. The system will then check if this updated game world condition is valid. In this example, *King Arthur* is the victim, so the system will check if there are any alive NPCs that have a *loves* relation towards *King Arthur*. This condition is satisfied since *Queen Guinevere* loves *King Arthur* in our game world. Since both conditions have been met, we may rewrite this event using our *narrative rewrite*. We start by replacing the *Lover* and *Victim* targets with the actual targets, *Queen Guinevere* and *King Arthur* respectively. Next we replace the original murder event with the new graph. We attach the events following the logic explained above, where any events without incoming edges, will be linked by an incoming edge to any events which were previously linked to the murder event by an outgoing edge. In this example this corresponds to the *Go to Victim* event being linked to the *Murder* and *Spare* events by outgoing edges. Likewise any events without an outgoing edge are linked with an outgoing edge to any events which were previously linked to the murder event by an incoming edge. In this example this corresponds to the *Return* event being linked to the *Spare* and *Ambush* events by incoming edges. The resulting narrative is shown on the right side of Figure 3–12.

The above example shows a relatively small game world with only one of each rule. From this we were still able to generate a story of some complexity which

relates closely to the relations defined with in our game world. By defining a large game world with many rules, the user may create a variety of unique and complex narratives.

3.3 Generation Process

In this section we present in detail the generation process as shown in Figure 3–13. We will begin by examining the first, initial steps in the generation process which involves generating a starting narrative. Following this, we present the rewrite process, where our secondary rewrite rules are systematically applied in order to generate complex, interesting variations of the starting narrative. We then discuss the process of performing a metric-enhanced rewrite and describe the validation process. Lastly, we discuss the process of updating the game world, which involves applying the game world rewrites used in each rule.

3.3.1 Initial Narrative Generation

Figure 3–13 gives a flowchart summarizing the generation process performed by the scheduler. The first step in our narrative generation process is to examine our game world environment for a *potential story*. To do this we define what could be known as a set of *initial rewrite rules* (IRR). It is using these rules that we create our initial narrative. The IRR has a left-hand side which searches for a condition within our game world, which we will henceforth call the *game world condition* and the right-hand side generates a narrative and in turn modifies the game world according to the actions performed within the narrative. The game world condition is also a directed labelled multigraph and the system checks if this graph exists as a subgraph

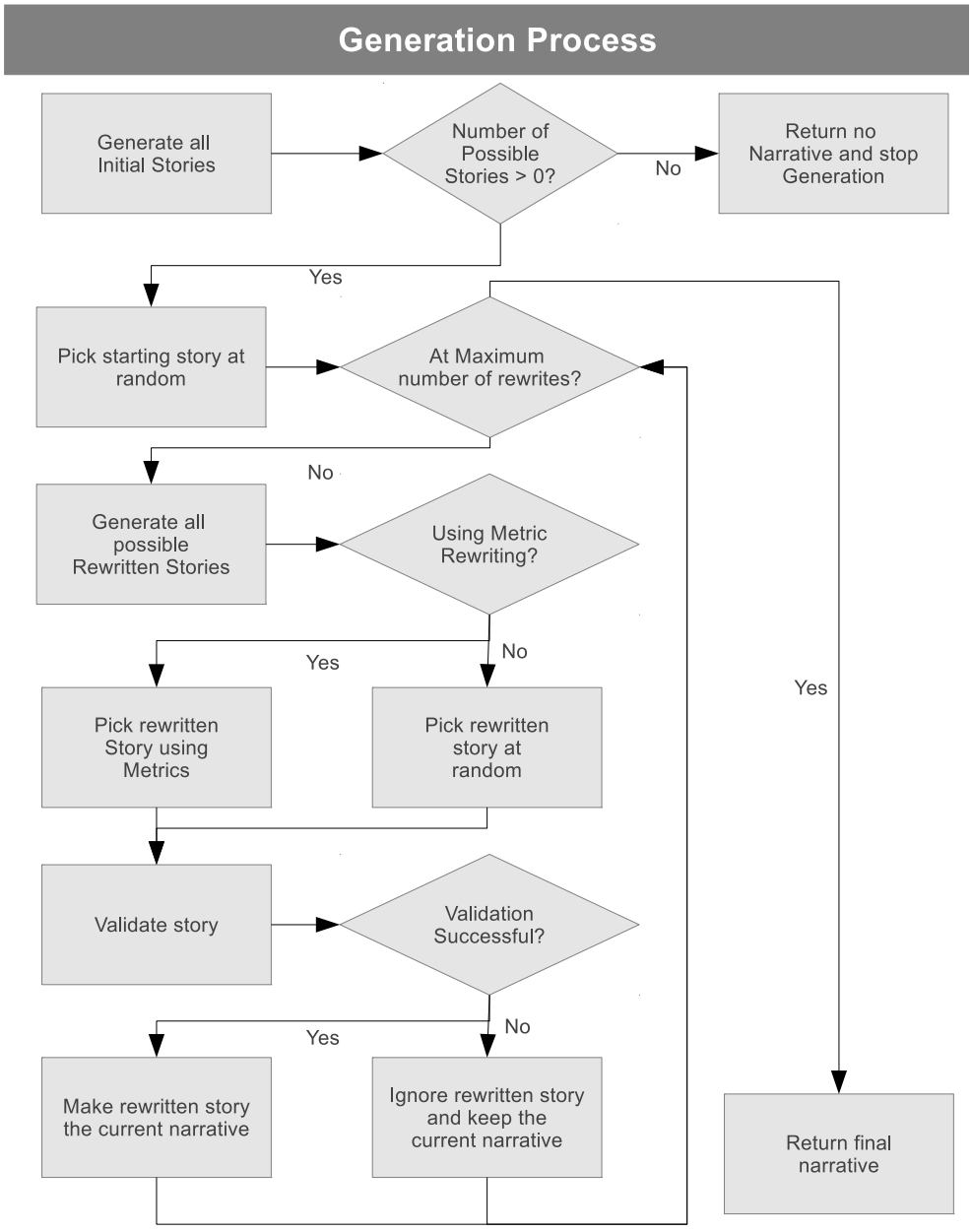

Figure 3–13: A flowchart displaying the generation process

within our main game world graph. If this condition exists, then the resulting right-hand narrative is generated. The right-hand narrative is user-defined and consists of a complete narrative as defined above as a directed acyclic graph. We impose the condition that the narrative defined must have a beginning and end and consists of at least a single event. The length and detail of this narrative is completely customizable. We find that this helps alleviate the disconnect created between author and narrative found, for example, in emergent narrative systems [24]. The cast of the generated narrative is the resulting objects which met the condition we searched for in the game world graph.

At this point our generation process resembles the *Radiant Quest* used from *Skyrim* in that we have essentially created a skeleton narrative which is filled in by targets in the game world [9]. The main difference is that the *Radiant Quest* system looks primarily at types, for instance a role can be filled by a "bartender' and this could be any "bartender" within the *Skyrim* universe. Our system uses a relation graph which can represent any number of roles and interrelations, and we use sub-graph isomorphism to determine whether this graph exists in our game world. This means we can search for much more complex conditions, such as love-triangles or adulterous characters. We use the object attributes and relation identifiers/reasons to tune our search.

3.3.2 Rewrite Process

The next step of the generation process involves rewriting the starting narrative to make it more "interesting". To do this we define a new set of rules, which we call *secondary rewrite rules* (SRR). The SRR differ from the IRR in that they can

contain both a game world condition *and* a narrative condition with the results being a narrative rewrite and a game world modification. For a narrative to be a potential candidate for a given SRR, it must first satisfy the narrative condition. The narrative condition is represented as a directed acyclic graph and must be present as a sub-graph within the actual narrative in order for the condition to be met. If the narrative condition is met, then it is verified that the game world condition is also met. The game world condition is defined in the same manner as the game world condition in the IRR. The game world condition for these rules can also reference objects in the cast of the narrative. This is done by having labels for the cast. For example, the murder victim may fall under the "Victim" label. Thus, they can be referred to as victim in the game world condition and the system will automatically fill this in with the corresponding cast object which fills the role of victim in the narrative. Note that the game world rewrites only occur during the simulation stage, after we have generated the narrative. This will be explained below in the *Updating Game World* section.

The number of times this rewrite process occurs is left up to the user. This entire process is handled by the *Scheduler*, which is meant to be largely invisible to the end user. Certain features, however, can be tweaked which radically change how the rewrite system works. Having a larger number of rewrites, for example, leads to much more complex and lengthy narratives. In the next section we present a unique feature of the system which aims to generate narratives of a specific quality, related to a unique set of metrics we defined to represent narrative quality. This is entirely

optional, and if left off then the system instead chooses a valid rewrite at random during each rewriting stage.

3.3.3 Metric Analysis

Metric enhanced rewrites are an optional feature of the ReGEN. Enabling metric enhanced rewrites allows the system to determine which rewrite rule to apply based on a set of metrics and weights. This set of metrics and weights are again customizable by the user. When rewriting, the system picks the best resulting narrative based on these metrics, with higher weighted metrics being given higher priority. Using this option provides the advantage that we are able to customize the rewrite process based on the priorities of the user.

The complete list of metrics are defined in Section 4. The user may select any combination of these metrics, save for *weight of choices*, for reasons which will be explained in the corresponding section. The set of metrics and weights to use when rewriting, as well as the option of whether or not to use metric rewriting are configurable properties of the ReGEN scheduler.

In addition to specifying each metric desired, the user must also specify a *weight* for each metric. The weights are simply an integer value which is used to indicate the importance of each metric. A positive integer indicates that a higher value in that metric correlates to improved narrative quality. Also, a larger positive integer relates to a higher importance for that particular metric. Contrarily, a negative number means that having a higher score in a given metric is a detriment to story quality. A lower negative number symbolizes an increased detriment to narrative quality.

The actual process of a metric enhanced rewrite is as follows, the system first gathers all the possible narratives based on every valid rule which can be applied. These narratives represent what our main narrative would look like, if that particular rule is applied. Each of these possible narratives are then evaluated according to the metrics requested by the user. To determine the best possible narrative we assign each narrative a *score* based on how well the narrative scores against the metrics. To do this, for each narrative we gather scores for all metrics, and then combine these scores to get a final score.

The system runs through all metrics, and evaluates the metric results for each narrative. It then finds the narrative which scored the highest for this particular metric. The score of this narrative for this particular metric is equal to the weight of the given metric. To score the remaining narratives we take the ratio of the metric results of each narrative to the results of the best narrative and then multiply this by the weight. This gives each of the remaining narratives a score based on how well each narrative scored proportionally to the highest scoring narrative. This step is shown in Equation 3.1.

$$score_{narrative,metric} = \frac{result_{narrative,metric}}{result_{narrative_{best},metric}} \times weight_{metric} \tag{3.1}$$

This step is repeated for each of the metrics in question. The final score for each narrative is therefore the summation of all the individual scores for each metric. This is shown in Equation 3.2. By using this method, we allow higher scores to be associated to metrics with higher weights. This fulfils the concept that these metrics are more important, since scoring well in these metrics gives a higher final score.

Likewise, if we had a negative weight, scoring well would give a negative value that would reduce the final score. Again, a higher negative weight would result in an even greater decrease in the final score representing the heightened importance of that weight in terms of narrative quality.

$$finalscore_{narrative} = \sum_{x \in metrics} score_{narrative,x} \tag{3.2}$$

The last step is simply to take the narrative with the highest final score as being the "best" narrative. This will be the narrative picked for this iteration of the rewrite process. The process repeats for each iteration of the rewrite process, where the narrative picked will always be greedily selected as the highest scoring narrative for that particular iteration. While this solution does not guarantee the ideal narrative, it a fast and simple method that still allows for a certain amount of optimization during the rewrite process. Importantly, though, is that it gives the user some control over how the generation process will occur without requiring the user to understand how the scheduler works.

3.3.4 Validation

After each rewrite, the system *validates* that the currently generated story is *complete* and *valid*. We define a complete narrative as a narrative where the player may take any possible path through the narrative and always end up at the end node. We define a valid narrative as one which does not violate any constraints of the game world, for example being asked to kill a character who is already dead. While the story is only rewritten when the game world conditions are met, there may still be occurrences where the performing a specific action will result in an invalid change to

game world, or that a specific action earlier in the narrative may make a later action impossible. By performing this step, we can formally state that each narrative we generate is complete and valid within the current state of the game world.

The first step in validation is to analyse each story event in terms of what is required for the story action to occur, and what this changes in the game world. This breaks down into three types of conditions, which we call the *preconditions*, *postconditions* and *lost conditions*. The preconditions represent which conditions need to be met in order for the story event to be valid. Given the murder example, in order to murder a character we require that the victim is alive. Therefore, the victim being alive is a precondition for performing a murder event. Lost conditions refer to which relations have been lost by performing an action. When a victim is murdered, they lose all their outgoing relations[2] . As such, all those relations are now considered lost conditions. Post conditions refer to relations or object attributes which are modified or created due to the story event. Returning again to the murder example, the *alive* attribute for the NPC would be set to *False* following the murder. This attribute change is a post condition of performing a murder.

This analysis is done based on the modifications provided by the user when they declare the *narrative result* for a particular event. Modifications are provided using a built in basic scripting language that allows for certain methods to be run in the game

[2] Other objects may still hold a relation to the victim, such as an NPC having a loves relation towards the dead victim. The victim themselves, however, no longer have any relations to any objects.

world graph. Examples of these are modifying attributes and creating/destroying relations. We have stored the changes to the game world for all these methods in terms of pre/post/lost conditions. This means that the system can automatically determine all three conditions for any modification, simply by accessing this stored data. We based these conditions directly on the changes which occur as a result of the method in order to ensure that all the conditions are correct for each method. Thus in the first step the system simply iterates through all story events and determines these conditions.

Once the system has all the necessary condition data, it can begin the process of *refining* the lost conditions. To do this, we start at the starting event of the story, and from there we "carry down" the postconditions. By carry down, we mean that we explore every event which occurs after the starting event, and then every event which occurs after these event, etc. until we reach the ending event(s). At every event, we examine if these postconditions become lost conditions due to a story event. For example, when a player moves from one location to another, this is represented by adding a *Currently In* relation leading from the player to the location where they have moved and removing the previous *Currently In* relation. Since the previous *Currently In* relation is removed, this makes the relation a lost condition of the move action. In the first step we could not determine this lost condition because we simply looked at each event individually, and we do not know if the player has moved at some point previously in the narrative. All we know is where the player was at the time of generation. So in the refinement step, we check the event's lost conditions against the postconditions for any events which preceded it. If we find that there is

a conflict between a lost condition and any previous postcondition, we remove the conflicted lost condition and replace it with the postcondition. We then no longer need to "carry down" that postcondition since we know it becomes a lost condition and is therefore no longer needed for the remainder of the narrative. As an example imagine if the player is currently in the castle and the first event has them move to the forest. Later in the narrative, there is an event which has them move to the cave. During the first step, the system will iterate through the postconditions and find that event has a postcondition that the player is in the forest and the second has a postcondition that the player is in the cave. However, when iterating through the lost conditions it will find that the first and later event both have a lost condition that the player is currently in the castle, since at the time of generation the player is currently at the castle. However, in this step, we will carry down the postcondition that the player is now in the forest. When we come to refining the "go to cave" event, the system will notice that the lost condition that the player is in the castle conflicts with the post condition that the player is in the forest. As such it will replace the "player in castle" lost condition with the "player in forest" lost condition. Now the narrative conditions properly show that the "player in castle" condition is lost at the first event, and the "player in forest" condition is lost in the second go to event. The algorithm for this step is shown in Algorithm 1.

Once the refinement is complete, the final step is to validate the narrative. What makes an invalid story is an instance where the preconditions for an event conflict with the post conditions of the events which preceded it. For example if we need an NPC alive for a particular event and that NPC has been killed in an earlier event

Algorithm 1 The algorithm for performing a refinement step during validation

```
function RefineNarrative(narrative)
    node_starting := GetStartingNode(narrative)
    Refine(node_starting, [ ])
end function
function Refine(node, postconditions)
    lostconditions := GetLostconditions(node)
    for condition_lost ∈ lostconditions do
        for condition_post ∈ postconditions do
            if Conflict(condition_lost,condition_post) then
                RemoveLostCondition(node, condition_lost)
                AddLostCondition(node, condition_post)
                postconditions := postconditions − condition_post
            end if
        end for
    end for
    postconditions := postconditions ∪ GetPostconditions(node)
    edges_outgoing := GetOutgoingEdges(node)
    for edge ∈ edges_outgoing do
        node_to := GetToNode(edge)
        Refine(node_to, postconditions)
    end for
end function
```

then we have an invalid story. To do this validation, we start at the end event and travel "upwards" through the story events. This means that we visit the events in the reverse order, visiting every event with an inbound link to the previous event until we eventually arrive at the starting event. [3] We keep a list of all the preconditions at each story event. During a visit to an event, we compare the preconditions list to the postconditions of that event. If there is a conflict, then we have found the story to be invalid. If there is no conflict, then we first check if any of the postconditions are equal to any of the preconditions in the list. If there is such an instance, we remove that precondition from the list since we now know that that precondition exists only *after* that particular event, and occurs as a result of that event. This means that the precondition does not need to be met for any previous events and as such we do not need to keep checking it. Note that with this process, the system evaluates each possible path through the story for validity. If there is a branching sequence of events in the narrative, then the each path will be validated separately. There is no validation occurring between branching paths, because only one path will be taken. We must ensure, however, that any event occurring before the branching paths does not invalidate any of the possible paths, meaning that no path becomes invalid due to previous player actions. We perform this validation until all events have been visited, or until we find a conflict. If there is any conflict then the story

[3] We use reverse ordering here because we need to compare each event to all the events to occur after it to ensure that it does not invalidate any events later in the narrative.

is invalid, but if we are able to visit all events with no conflicts then we can say the story is valid. The algorithm for this step is shown in Algorithm 2. By performing this process, we meet the criteria that all stories generated must be correct and free of conflict by ensuring that no changes made by the player make later events in the narrative impossible.

Algorithm 2 The algorithm for performing the final validation step

```
function ValidateNarrative(narrative)
    node_ending = GetEndingNode(narrative)
    Validate(node_ending, [ ])
end function
function Validate(node, preconditions)
    postconditions = GetPostconditions(node)
    for condition_post ∈ postconditions do
        for condition_pre ∈ preconditions do
            if Conflict(condition_pre,condition_post) then
                return Invalid Story
            end if
        end for
    end for
    preconditions = (preconditions ∪ GetPreconditions(node)) - (preconditions ∩ postconditions)
    edges_incoming = GetIncomingEdges(node)
    for edge ∈ edges_incoming do
        node_from = GetFromNode(edge)
        Validate(node_from, preconditions)
    end for
end function
```

3.3.5 Updating Game World

The last step in our process is to simulate each event in the narrative and determine its *effect* on the game world. Note that we *provided this simulation* for testing purposes, in reality, the updates would occur as the player completes the

actions within the narrative. These effects are declared when the author defines an event in the story, as the game world results explained previously. The system proceeds through the narrative in a linear fashion and whenever an event has an effect, the system applies it to the game world. The final result is a modified game environment. Note that in the case of branching narratives, a different final game environment is created for each branch. In our murder story example the act of murdering the victim may make all of the victim's friends hate the player. This modified environment serves as the starting point for the next narrative, again going through the potential IRR rules and picking one at random. Within a game setting, the path taken by the player would be the path used to update the game world in real-time. By performing this step, our generation tool is able to generate narratives "on the fly". This means that we are able to generate narratives one after the other, and that each new narrative respects the changes to the game world of all the previous narratives.

The ReGEN system takes much of the philosophy behind the socially themed narrative generators, but provides an alternative to emergent and goal-oriented planners. The user is responsible for defining the initial story structures as well as any additional narrative rewrites, giving complete authorial control over the potential events and event structures which occur as a result of the generation process. One of the downsides of not providing an emergent narrative system is that we restrict the freedom of player choice. We can however provide branching narratives which allow for player choice and give some measure of freedom. This design trades potential emergence for narrative completeness and control, a trade-off that seems appropriate

for quests in commercial games. An advantage of the use of Initial Rewrite Rules and Secondary Rewrite Rules is that we begin from and ensure a successful narrative at all points, but can incorporate relatively arbitrary constraints. The metrics we describe in the next section, for instance, can be evaluated at each stage to ensure we only pick the "best" rewrite rule at each step in the generation process using our metric rewriting process. Likewise by using validation we can guarantee that the stories generated could viably exist within the current state of the game world, meaning no quests are generated which are impossible for a player to complete.

CHAPTER 4
Metrics

A central difficulty in narrative generation is to evaluate the quality of the narratives generated. Player opinion and enjoyment are of course paramount but require procedurally complex and necessarily noisy human evaluation, which is typically done in the form of questionnaire evaluations [17, 1]. Our approach here is to develop a novel set of metrics that intuitively relate to *narrative quality*, at least in terms of overall narrative structure. Our metrics take into account basic narrative features such as length and story branching, as well as more complex inter-story relations that are commonly associated with narrative depth/complexity. We define an *inter-narrative* metric as being any metric which serves as an evaluation between different narratives, while an *intra-narrative* metric refers to an analysis on a single story, which does not relate to any other of the stories generated. Note that most metrics used in this research are intra-narrative metrics, while defining more inter-narrative metrics remains a future goal.

Our metrics are likewise intended to allow for personal interpretation. This is specifically to allow interpretation by the user of the system. As mentioned in the previous section, the only time the metrics affect the generation process is if metric-enhanced rewriting is used. Otherwise, the story metrics are provided as is. The intention is that the user may use the metrics to understand the structure of stories

currently being generated by the system. They are then able to modify the system as necessary in order to improve performance.

Below we define nine metrics that are used to give formal and quantifiable insights into the quality of the narrative being evaluated. The metrics are *narrative content*, *longest/shortest path*, *number of branches*, *cost*, *highest/lowest cost*, *encounters*, *uniqueness*, *narrative richness* and *weight of choices*. We will explain what each metric defines and provide justification in choosing these metrics in the following section. Following this, we provide a brief discussion on the correlations between metrics.

4.1 Narrative Content

Our first metric relates simply to narrative length, and the total number of story events. While it is impossible to directly relate story length to how good a story is (such as determining if short stories are better than novels or vice versa), there are certainly extremes on both sides that could negatively impact story quality. For example, a quest involving the player going to collect one flower may be disappointingly short, while an alternative quest involving the player going to collect one-hundred flowers may feel much too long. The "One Hundred Flowers" quest could also appear lengthy due to the fact that the player is repeating the same actions for the entire quest, which we will discuss below in the "uniqueness" metric. One measure of narrative length is by simply looking at how many narrative events there are overall, regardless of whether the player will be able to experience all events in a single play-through of the quest. We will dub this basic metric *Narrative Content* as this

is not necessarily a measure of narrative length, but rather of how many potential narrative events could occur within the quest.

4.2 Maximum/Minimum/Average Path Length

We cannot directly measure the length of a narrative when games allow for branching stories as each branch could be any arbitrary length. This means that the best measure for narrative length would involve looking at both the longest, shortest and average path through a narrative, and the number of narrative events experienced by the player when taking either of the extreme paths. As with most metrics, it is controversial to state these directly relate to the narrative quality, but we can say that they are one of many metrics which are important to understanding narrative quality. For example, a large difference between the amount of narrative seen when taking the longest path as opposed to the shortest path may show that the player is missing out on much of the potential *narrative content* if they take the shorter path. Conversely, if both the longest and shortest path are very short compared to the total overall *narrative content*, we may be in a position where there are many short branching paths, which can be viewed as either a good or bad feature depending on what is expected by the user.

4.3 Maximum/Minimum/Average Number of Branches

We take the number of branches as being the maximum, minimum and average number of player-experienced events in our narrative graph that lead to two or more possible events. Note that we are not analyzing the total number of paths, rather, we are trying to analyze the number of times within the narrative that the player will be given a choice, depending on the path they take through the given

narrative. Thus we are aiming to determine, the most, fewest and average number of choices a player could be given within the quest. This is one metric which would not normally be a feature of more traditional narrative analysis, but it is specific to an interactive narrative context. We can view the number of branches in a narrative as a measurement for both player freedom and narrative complexity.

4.4 Highest/Lowest/Average Cost

Cost is a metric closely tied with narrative generation systems. In our system, actions such as murdering a person or destroying an item make irreversible changes to the game world. After a point these actions could lead to states where it is no longer possible to generate any narratives since most of the NPCs in the world had been murdered by a player in previous quests. We therefore suggest that certain actions should have a *cost* associated with them. This leads us to assign any event in the narrative that resulted in an object being effectively removed from the game world as having a cost of one.

In *Skyrim*'s *Radiant Quest* system, this potential cost is by-passed by having most of the targets be procedurally generated [9]. For example, in a murder quest, the game will procedurally generate an arbitrary NPC to be the player's target. Thus, murdering them does not make any lasting changes to the game world. While this alleviates the concerns of cost, this takes away a feeling of importance from the quest. It is interesting to have quests which make definite and consequential changes, even if at a high level these pose a threat to the size or state of the game world. The *weight of choices* described later in this section was our way of analyzing the relation between player actions and the corresponding change in the game world.

Cost thus represents an important property of narrative quality when narratives have a meaningful interaction with the game world.

For our analysis of cost, we take a similar approach to the narrative length in that we are interested in the highest, lowest and average possible cost. Again these values let us determine if we are entering any extremes, as well as exploring the variety of possible outcomes based off of player choice. We do not see any purpose in including a metric of *Cost Content*, similar to *Narrative Content* as we are not interested in the overall presence of cost events, we are simply interested in how much cost is guaranteed to occur (*lowest cost*), how much cost is possible to occur (*highest cost*) and how much cost is expected to occur (*average cost*).

4.5 Most/Fewest/Average Encounters

Encounters is a metric which can be considered an alternative to the cost metric. Essentially, an encounter relates to an exciting event such as a fight with a monster that does not make any irreversible changes to the game world. This metric has the disadvantage that in many of the examined narratives, the game world contained many random monsters placed in the world environment, but fighting them was incidental and not directly represented as part of the quest structure. We retained this metric since there were some quests where certain types of encounters were deemed important to quest progression, such as quests wherein the player must kill X number of creatures to proceed, or ambushes by groups of enemies the player must kill in order to proceed. Once again, we analyze both the most and fewest possible encounters, as well as the average number of encounters.

4.6 Highest/Lowest/Average Uniqueness

As mentioned in the preceding section, for metric analysis we assigned a key action to each event in our story. This we use to help define our *uniqueness metric*. For example if a node involves murdering someone, then the node is a *Murder* node. If the node involves travelling to a location, then this is a *Go To* node, and so on. Using this we can determine how many *unique node types* there are in a story. A story with two murder nodes, for example, would qualify as having only one unique node type. If we divide the number of unique node types by the *narrative content*, we get a measure of what percent of our total story nodes are unique, which we choose to call the story's *uniqueness*. This is shown in Equation 4.1

$$uniqueness = \frac{unique\ node\ types}{narrative\ content} \tag{4.1}$$

This means that our measure of uniqueness can be considered as an *intra-narrative* measurement of uniqueness. Referring back to the *One Hundred Flowers* quest, we can break the quest down into one hundred events, where each event is a *COLLECT* event. This would yield a narrative content of 100, but a uniqueness of only one percent. Conversely, the *One Flower* quest would have narrative content of one, but a uniqueness of 100 percent. An example of this in the commercial game *Skyrim* would be a number of quests within the *Civil War* chain of side-quests. These quests, usually prefixed with *The Battle of* followed by the location name, consist entirely of killing a certain number of enemies at different locations. This results in a repetitive experience for the user which regardless of initial interest inevitably becomes dull. We do believe that uniqueness has a direct effect on narrative quality, as

it helps to determine how much *repetitiveness* a player may encounter. Again, while there have been many examples of games that use repetitive actions, these games are often not story based. Future work of this research is to create the inter-narrative equivalent of this metric, allowing us to compare narratives to see how similar one generated narrative is to another. This would prevent a case where a narrative might in itself be unique, but many of the same narratives may be occurring in a row. Returning to the *Skyrim* example, the *Battle of* side-quests are additionally repetitive in that there are several of them and in no case is there any change in the actions which need to be taken by the player. Once again, since the uniqueness of a quest may change depending on the path the player takes, we look at the highest, lowest and average uniqueness score for each narrative.

4.7 Narrative Richness

In an intuitive sense, narrative "richness" or "depth" closely relates to the how surprising or interesting narrative events appear to the player. Plot twists and realization of sub-plots add to perceived complexity and interest, but require the player to experience and even influence narrative events in a way that does not directly relate to the current goal. We interpret this as a metric in terms of the *unintentional consequences* of a narrative, since it attempts to measure how much of a given narrative has been influenced from past narratives, without being a direct goal of adjoining steps. In our system, we keep a store of all the changes made to the game world by each narrative created. We call these the *postconditions* of our narrative. We also view the game world conditions for the IRRs and SRRs as potential *preconditions*. Preconditions can either be satisfied by conditions in the game world which

were not the result of previous narratives, and conditions which were. For example, if in a previous narrative the player made two characters hate each other, then if a narrative is generated with the precondition that those two characters hated each other, the hate precondition was only satisfied because of the actions unknowingly taken by the player in the previous narrative. This then allows us to define narrative richness in terms of the percentage of the preconditions for our narrative which were satisfied by the postconditions of any previous narratives, as shown in Equation 4.2. This metric could be viewed as an inter-narrative metric.

CD Projekt RED's *The Witcher* is a game known for using this feature, where actions taken by the user in previous quests result in them experiencing different events in later quests. An example of this occurs in an early quest entitled *Of Monsters and Men*, where the player is given the choice to defend a character accused of witchcraft, or leave her to be killed. In a much later quest, *Frozen Reflections*, the player encounters the witch. If the player saved the witch, then she is alive and provides the player with potions. If the player left the witch to die, then she is instead a vengeful spirit who attacks the player. Our narrative richness metric aims to capture the concept that the narrative experienced by the player has changed due to seemingly arbitrary choices made by the player in an earlier quest. Note that this metric cannot be evaluated based on user path, as with the above metrics, since richness depends on the impact of a choice on all possible futures—we need to look at how the whole quest itself is the result of previous actions, and not just for an individual user path.

$$narrative\ richness = \frac{|postconditions_{total} \cap preconditions|}{|preconditions|} \tag{4.2}$$

4.8 Weight of Choices

Our final metric is to examine the effect that choices have on the number of final possible game worlds. As mentioned in the previous section, during the test phase whenever our stories have a branching event we split our simulation into two parts to represent the two new possible game worlds. We then continue generating narratives from these two new game worlds. After some predetermined number of iterations, we then compare each final game world to each other final game world. This involves comparing each object's attributes and relations by dividing the number of attributes and relations that are the same between both worlds by the total number of attributes and relations in each game world. This gives the similarity between each game world, as shown in Equation 4.3. We believe this metric is important since it highlights the importance of the choices made by the player in the game in a quantifiable way. This metric is the only metric which cannot be provided as an optimization metric since it is only applied in experimentation and is dependent on comparing the results of multiple stories, making it difficult to evaluate for a single narrative.

$$similarity_{1,2} = \frac{|attributes_1 \cap attributes_2| + |relations_1 \cap relations_2|}{|attributes_1 \cup attributes_2| + |relations_1 \cup relations_2|} \tag{4.3}$$

4.9 Metric Correlation

Within these nine metrics we find that there are certain correlations between metrics. *Narrative Content* and *Path Length* are both strongly correlated, for example, since each is simply a measure of the number of events within a narrative. *Path Length*, however is dependent on the player's path through the narrative whereas *Narrative Content* is simply the sum of all possible events. We can expect, then, that a larger *Narrative Content* would lead to overall longer *Path Lengths* in general. *Path length*, however, is also largely dependent on the *Number of Branches*. Since each branch creates at least two events, many branches will yield a much higher *narrative content* but the *path length* may remain the same.

Cost and *Encounters* are also closely related. A *cost* action may be the same as an *encounter* action with the only difference being the target. If the target is *renewable*, i.e. the game may generate multiple instances of this object, then we define it as being an *encounter* action. Likewise if the action removes the object permanently from the game world then we instead have a *cost* action. Although being related in this way, each metric may be different and having a higher value in one metric will not necessarily result in a lower result for the second metric.

In terms of conflicting metrics, the main conflict was found to be between the *Narrative Content* and *Uniqueness* metric. This conflict is discussed further in Section 5, however in summary it was found that higher content would lead to a reduction in *uniqueness*. This is due to there being only a finite set of possible actions a player can have within a game. Therefore, as a narrative contains more content it therefore increases the likelihood of two of the same actions occurring within a player path.

Lastly, it was noted that *Narrative Richness* and *Weight of Choices* are closely related, and that a higher value in one should correspond to a higher value in the other. This is intuitive, as the more influence a player makes on the game world, the more likely this influence will be reflected in the generated narrative. However, this is not ensured, as it is possible to have a high narrative influence with only minor changes to the game world graph, provided those minor changes lead to more potential narratives. Likewise, if there are many changes to the game world graph, but none of those changes lead to more potential narratives, then the opposite effect will be observed.

CHAPTER 5
Experimental Analysis

For our experimentation we designed tests aimed at comparing the results of our system to two other quest-oriented narrative generation systems and to two "good" narratives. The two narrative generation systems are SQUEGE and the *Skyrim Radiant Quest* tool. The two "good" narratives are the main quests from *Skyrim* and the main quests from *The Witcher*. We define "good" narratives as being written manually by an author and used as part of a commercially successful RPG which is known for strong narrative content. For our analysis we use the metrics defined in the above section, evaluated for multiple narratives from both the narrative generation systems and the "good" narratives. We hypothesize that our system will achieve better metric values than the two other generation systems, and that our narratives should be comparable metric-wise to the hand-authored quests.

We likewise sought to experiment with using the ReGEN system with multiple types of game worlds. This includes two hand-made game worlds, a small game world with sparse relations and a dense game world with dense relations. Lastly, we examined the effects of using ReGEN on a commercial level game world, by translating the *Skyrim* game data into our format and using it as the game world for several experiments. For this experiment, we hypothesized that the system should be able to produce similar quality narratives regardless of the structure of the game

world. Additionally, we were able to examine potential cost issues with using the ReGEN system on a large-scale commercial game world.

After the narrative initialization step in our generation, we have a system which is similar to *Skyrim's Radiant Quest* generation system. By using a series of secondary rewrite rules, we make the narrative more complex and arguably better. As proof of this improvement, we evaluated the results of using ReGEN as a Radiant Quest generator and then seeing the improvements by adding in our secondary rewrite rules. Once again these improvements are quantified using our metrics and our criteria for a good narrative.

Lastly, many of the previous tests noted a conflict between the narrative content and uniqueness. In summary, a longer narrative seemed to result in an inevitable decline in uniqueness. To explore this effect, we analyzed the gradual decline of uniqueness by chaining multiple quests into one large quest and then showing the results of the uniqueness metrics. This was explored both with the hand-authored, "good" narratives, and generated narratives from our own system.

In the next section we present in-depth discussions and justifications for each set of test data. Following this, we provide the criteria we created for a good narrative. Lastly, we present and discuss the results for each of the experiments.

5.1 Test Data

In this section we will present each of the test data sets in turn as well as the justification and process of their inclusion. We start by examining the three different game worlds which were tested using the ReGEN systems. Afterwards, we introduce the two narrative generation systems, SQUEGE and the *Skyrim Radiant Quests*. The

remaining sections describe the hand-authored quests from *Skyrim* and *The Witcher* that were used for testing.

5.1.1 ReGEN Game Worlds

While the variety of narrative rules is entirely dependent on the author, in many cases they will be constrained to working within the game world they were provided. As such, it is important to evaluate the effects of using the ReGEN system with different types of game world graphs. We evaluated the results for three different types of graphs, using the same set of rules for each graph. The first two graphs were hand-authored and the third graph was based of the game world of *Skyrim*. The first hand-authored graph consisted of a small world graph with sparse relations, intending to test our system with minimal data. The second hand-authored world graph was larger with a dense set of relations, intending to test our system with a larger body of data. Lastly, the *Skyrim* game world was intended to test our system with a game world at the approximate scale of a commercial game. We present each of these graphs in the next three sections along with their justification and implementation.

Sparse World Graph

We hand-authored the sparse world graph and designed it to test running the system at a minimal level. In total there were 25 objects and 61 edges, which averages to around 2.5 edges per node. We expect that the ReGEN system will have a harder time generating stories when given a smaller game world with very few relations, but expect the metric evaluation of quality to be the same. The sparse world graph can be found on page 125. In this graph, blue nodes represent NPCs, green nodes represent

locations, yellow nodes represent enemies [1] , red nodes represent objects and the pink node represents the player. This graph is unfortunately the only graph which is small enough to be shown, but note that all other game worlds are similar in structure to this game world, but with a varying number of nodes and relations. For each graph, we defined a dictionary of allowable relations. The relations allowed between NPCs are as follows: *loves, hates, friends, allies, enemies, trusts, distrusts, likes, dislikes,* and *blackmail* . For NPC to object relations, we only allow *owns* relations and for NPC to location relations, we only allow *lives* relations. Enemies are allowed to *target* certain locations and *live* in others. The player, in this representation, uses a *currently in* relation to indicate where the player is currently located in the game world. Lastly, locations may be *connected* to each other in any of the four compass directions: *north*, *south*, *east* and *west*.

Dense World Graph

We again hand-authored the dense world graph to test the system using a much larger body of data and a much denser set of relations. This graph was designed with 63 objects and 928 relations, averaging just under fifteen edges per node. This gives a heavily dense graph in which most NPC objects were designed to have at least one relation to every other NPC object in the game world. Note that the inter-NPC relations were computationally generated to ensure that each NPC had

[1] an enemy is an NPC who will immediately attack you and cannot be interacted with. The enemy likewise does not have the set of social relations for the regular NPCs.

approximately one of every type of NPC relationship possible. The remainder of the graph was all authored by hand. By having a denser graph we expect that it will be easier for the system to generate narratives. The relations allowed in the dense graph are the same as the ones allowed in the sparse world graph.

Skyrim World Graph

The *Skyrim* world was a conversion of *Skyrim* data into our game world format. In order to provide an accurate representation of the *Skyrim* world, all data was taken from the *Skyrim Creation Kit*. The *Skyrim Creation Kit* is a free tool from the *Skyrim* developers which can be used by anyone who owns a copy of the game. Within the tool kit, the user has access to all the *Skyrim* data, including the relation graph. Since *Skyrim*'s game world is similar to our description of a game world, where NPCs have relations to one another, we can easily convert the *Skyrim* relation graph into our format. This means that we are left with a relation graph of NPCs and their relations to one another. In most cases, the *Skyrim* relations matched the relation dictionary used for our hand-authored graphs. This was intentional as we used the *Skyrim* social relations as inspiration for our relation dictionary. In a few cases, we needed to replace a relation with one from our dictionary, but there was always an appropriate alternative. For example, we replaced all *Foe* relations in the *Skyrim* relation graph with our *Enemies* relation, since they imply the same relation. The other main changes were replacing the *acquaintance* relation with our *likes* relation, and the *rival* relation with our *dislikes* relation. Using the NPC data, we were then able to locate where the NPCs lived, and which items they owned. With this, we could generate our world graph consisting of NPCs, items

and locations. This limited our game world to only these three kinds of objects and only NPCs which have at least one relation to another NPC. Likewise we did not include any of the procedurally generated enemies within the Skyrim world. Since random encounters are kept out of Skyrim quests, it seemed reasonable to exclude these kinds of enemies and keep the assumption that fighting will occur as the natural consequence of the player exploring the game world. In the end this amounted to a world graph containing 757 nodes and 1360 edges. Of interest is that this amounts to only 1.8 edges per node, meaning that the density of this graph is similar to that of the Sparse World graph we defined above.

5.1.2 SQUEGE

The SQUEGE narrative generation tool as mentioned in Section 2 was a side quest generation tool which likewise used graph rewriting as the main generation process [15, 14]. Its similarity to our system allows for good comparison to be drawn between both styles of generation. Compared to our system, SQUEGE does not work with a relation-based system, keeping only lists of all NPCs and items within the game world. Likewise, there are only two main types of rewrites, one which involves quests to murder NPCs and the other quests to retrieve items. The system instead focuses on creating story structures, for instance creating branches and then randomly assigning one of the two types of quests to each of the branches. With these similarities and differences, it is worthwhile to see where our system is superior to the SQUEGE system, and where it falls short. Note that the SQUEGE output uses the same graph representation as our system, with a directed acyclic graph with

each node being an event in the narrative. This makes it possible to analyze the narratives directly without any modification.

5.1.3 Skyrim Radiant Quests

Skyrim's Radiant Quest system is one of the most well known attempts at providing generated narratives in a large-scale and successful commercial game [9]. Its generation method involves taking a quest structure and filling in the character roles with NPCs in the game world based on what type of NPC is required for each event in the quest. For example, if the quest is given by a "bartender" then the system will fill the role with a bartender NPC. The system provides 24 basic quest structures, but these are largely similar. Quest types include assassination quests, fighting monster/NPC quests, item retrieval quests, thieving quests, rescue missions, etc. If the quest calls for a murder or a gather item (what we consider a cost event), then the NPC being murdered or the object being collected are automatically generated and placed in the game world. This allows the system to theoretically generate quests forever. However, since the list of possible items to generate is finite, generated items may be repeated eventually resulting in two of the same quests being generated. The Radiant Quest system bears some similarity to our system in its consideration of certain rules about the game world when generating quests, and ensuring complete quests by adhering to strict predefined quest structures. Overall, however, the system represents a very primitive generation system and as such we expect to be able to outperform this system when compared to our metrics. Note that the *Skyrim* Radiant Quest structures were parsed into our format using the

method explained in the next section, allowing us to analyze them alongside the SQUEGE and ReGEN quests.

5.1.4 Authored Quests

For our "good" narratives we examined the main quests from *Skyrim* as well as the main quests from *The Witcher*. Both are commercially successful games known their strong narrative content [22]. The main quests should be representative of the best-written of all the game quests as they are the primary narrative for the game.

In order to faithfully convert each quest in the *Skyrim* and *The Witcher* into a form that may be analyzed by our system, we gathered the quest data from the highly detailed and exact wikis [29, 16]. It is common in gaming to have actively maintained community wikis which log the quests in detail, either through extensive play-testing or by utilizing official guides endorsed by the creators of the games. As a further validation, many wikis themselves are maintained by the developers of the game, who ensure that all information on the wiki is valid. In the case of *The Witcher*, the extensive wiki we used is supported by the developer. From this, we find that the information provided through the wiki can be viewed as the correct and complete information for both games.

In certain cases, the game may provide a *toolkit* which is a tool which allows any user to view the actual quest formats used within the game itself. This is true for *Skyrim*, where the *Skyrim Creation Kit* was provided by the developers and allows full access to the *Skyrim* quest structures. With access to this data, we are further able to validate that the quest structures present within the wiki are correct and valid representations of the quests within the game.

In the wikis, each narrative is presented in the form of the actions which are needed by the user in order to progress in the narrative. Since this matches our definition of a narrative, it is simple to convert these narratives into our story graph format with each action being converted into an event. This event is then linked to the next action(s) that will need to be taken by the player. In the case of *The Witcher*, branches are indicated by showing both possible paths with if statements. Likewise, *The Witcher*'s quest descriptions verbally state when a narrative has been influenced by previous player actions in previous narratives, which we use to evaluate narrative richness. Both these allow us to generate the complete *Witcher* quests. Note that *Skyrim*'s quests do not branch and there are no conditional events so this step is not necessary for converting the *Skyrim* narratives. Lastly, we use keyword parsing to determine the node-type of each individual event (for example, we search for Murder to label a node a murder node) intervening only when a specific action does not have an associated keyword.

Skyrim Main Quests

Skyrim was selected as the first of the authored quests due to its critical acclaim and commercial success. Using *Skyrim*'s authored quests likewise compliments our use of the *Skyrim* Radiant Quests in our experiment. For the experiment, we focused specifically on the *Main Quests*. These are the quests which must be completed in order to complete the game. We ignored the optional quests since we make the assumption that the best quality authorship would be given to the quests which are guaranteed to be experienced by all players who complete the game. *Skyrim*'s quests are limited for evaluation purposes in several ways. First, none of *Skyrim*'s quests

involve branching. There are occasionally optional actions, but these are not formally part of a path through the narrative. Secondly, *Skyrim* does not explicitly state when there are fights during a quest. Fighting in *Skyrim* is frequent, but is instead related to random enemies the player encounters on their way to completing each goal. This prevents us from analyzing the encounters metric accurately. Encounters are, however, used in the Radiant Quests for quests involving killing a specific number of enemies. The Radiant Quests likewise have occasional branches, and as such we will be analyzing the Radiant Quests for both these features but not the Main Quests. Lastly, neither type of quest has conditional events or quests, meaning we have no means of analysing narrative richness within *Skyrim*.

The Witcher Main Quests

In choosing *The Witcher* we provided an alternative to the *Skyrim* main quests with quests that have branching, fights and conditional events. Again, we are looking at the main quests under the assumption that these will be the best authored quests. *The Witcher* has received significant critical acclaim for its especially interesting game narrative where player actions have non-trivial and interesting consequences [22]. As mentioned above we evaluate these unintentional consequences using our narrative richness metric, meaning that we are able to analyze the Witcher for all the features missing in the Skyrim quests.

Neither the Skyrim nor the Witcher main quests are analyzed for weight of choices. In the case of Skyrim this is because without branching we cannot see how player choice changes the game world. In the case of the Witcher this is simply due to the lack of available data. We cannot gather a complete enough description

of the game world before and after each player choice to be able to evaluate the given metric. For the Skyrim Radiant Quests we can, however, safely say that the weight of choices is 0% since the quests are specifically designed to generated needed items, and as such there are no significant changes to the game world as a result of completing the quest.

5.2 Criteria for A Good Narrative

Our metric analysis is meant as a tool to be used by authors. Our metrics measure structural properties, and are meant to approximate narrative quality, but are not of course a complete replacement for aesthetic judgement. We expect an author would use metric analysis to study the types of stories produced by the *ReGEN* system. This gives insight into whether the author should add or modify specific rules in order to better control the system's output. An example of this would be an author who values player choice and therefore desires a large amount of branching in the generated narratives. If they are unhappy with the *number of branches* metric, then it may be an indication that they need to provide more rewrite rules which result in a branching narrative. For our experiment, we define the following criteria for a "good" narrative:

- A longer shortest/longest/average path is better as it increases the lifespan of the quest. Additionally, we desire shortest/longest/average paths that are roughly equal, as this indicates that regardless of player choice, they will still experience a similar number of events.
- A larger number of branches allows for more player control within a given quest, so we value a narrative with a higher number of branches. Having

roughly equal scores for all three measures, again indicates that a player is guaranteed a consistent experience which is independent of the path taken.

- A relatively low cost is preferable for all measures of cost, as it increases the lifespan of the system. Having a minimal difference between highest and lowest cost will further show that the cost effect is consistent, regardless of the choices made by the player.
- A higher number of encounters are good across all measures, as encounters provide exciting scenarios that do not reduce the size of the game world graph.
- A higher uniqueness is preferable as it indicates the presence of multiple unique actions in a narrative. Once again we search for roughly equal measures to ensure that there are no largely repetitive paths a player could take within a quest.
- A higher narrative richness shows that player actions significantly affect the narratives they experience, and is therefore viewed as a positive.
- We value a higher result for weight of choices as it indicates that the players can significantly affect the game world they are playing in.

Validity of these criteria are based on our own perceptions of what constitutes a good narrative, and further verified in our experimental results by comparing with narratives that are more well known in the gaming community. We expect that our system will be able to outperform the basic narrative generation of *Skyrim*'s *Radiant Quest* tool. We additionally wish to check for our short-comings and advantages over SQUEGE, since both our systems are aiming to produce interesting side-quests for RPGs. Lastly, we anticipate that our system will be of similar quality to that of

the main narratives from *The Witcher* and *Skyrim*. It is important to note that we are not saying that our system can compete with the main quests of either game, since we cannot compare writing, voice-acting or any of the other features provided. Rather, we are trying to show that the structure of our narrative can compare to those narratives. In the next section, we present a series of tests examining and comparing the narratives in terms of our previously defined metrics.

5.3 Experiment Results

In the following section we present the results of each of the three tests as well as the two smaller experiments. The summary of all these experiments may be found in Table 5–1. We first present the results of running our system using the three different game worlds and comparing the results of each. Following this we briefly present the performance analysis results for running the game world tests. Then, we discuss the results of analyzing the four types of narratives explained above with our set of metrics. We likewise compare these results to the results from our ReGEN tests in order to note the similarities and differences between our narrative and the other narratives. We then present the two smaller experiments; in the first experiment we adjust our system so that it mimics the *Radiant Quest* system. We then use our rewrite rules to show how the *Radiant Quest* can be improved by using our graph rewriting rules. Lastly we briefly discuss a noted issue that the narrative content and uniqueness metric are conflicting metrics, meaning that increasing one generally decreases the other.

Experiment Title	Experiment Description	Results + Significance	Page
Game World Graph Experiment	Averaged the metrics over 100 generated narratives for three types of game worlds, a sparse, dense and the *Skyrim* world.	Found that the sparse and dense worlds were similar, and both outperformed the *Skyrim* world, due in part to *Skyrim*'s relation distribution. This showed that for effective generation the rules must written with the distribution of relations in mind in order to produce the best possible narratives. It was likewise noted that the sparse world had a significant increase in *weight of choices* and *narrative richness* showing that the structure of the game world directly influences these two metrics, and that smaller game worlds allow for greater player influence and richer narratives.	82
Performance Evaluation	Monitored the average generation times for each of the game worlds	Found that generation time was up to one minute for *Skyrim* world. While still reasonable, further optimizations may greatly improve this time	89
Narrative Comparison Experiment	Averaged the metrics for various narratives, including sample narratives from two generation tools and two hand-authored narrative sets from *Skyrim* and *The Witcher*	Found that our system results were closest to the results for *Skyrim*. We generally outperformed both generation tools whereas *The Witcher* was much higher in certain metrics and much lower in others. Proved that our system can compare to commercial game narratives and that it outperforms two similar generation systems.	90
Radiant Quest Experiment	Modified ReGEN to work as the Radiant Quest generator and then tried to improve it with SRRs	Showed an improvement in narrative quality with use of SRRs. This showed the flexibility of the system and validation that the use of SRRs can improve narrative quality.	99
Analysis of *Skyrim* as One Long Quest	Created large quests consisting of many chained smaller quests for *Skyrim* and ReGEN and evaluated them for *uniqueness* and *narrative length*	Showed that *narrative content* and *uniqueness* conflict and that larger narratives inevitably lead to less unique narratives.	101

Table 5–1: A summary of each experiment, as well as its findings and significance.

5.3.1 Comparison of Results with various Game World Graphs

For comparing the results of the three different game worlds, we generated one hundred narratives for each world and then evaluated them using our metrics. The rules used for testing are all presented in Appendix 6.1. In summary, there were ten initial rewrite rules and eight secondary rewrite rules. These presented a basic set of rules that operated on the relationship dictionary we described above. Among the ten initial rewrite rules, there were rules to murder hated NPCs, give gifts to foster relations between NPCs, forge or break allegiances, blackmail NPCs or murder NPCs who are blackmailing other NPCs, steal items, fight monsters, be ambushed by the player's enemies and overthrow oppressive NPCs. The rewrite rules allow the player to be caught while stealing, perform stealth kills, be ambushed when trying to murder a character (either by an NPC who wants to save the character, or kill them themselves), have random encounters, have additional monster fights, have gifts stolen, or spare a character they are supposed to murder. Note also that for testing, we allowed five possible SRR applications and used metric rewriting. We gave a weight of one to each of the metrics we declared good our criteria, and a minus one to each of the bad metrics. This essentially meant each of the metrics had a weight of one except cost. Table 5–2 shows the average results for each of the metrics for each of the three game worlds after performing the above experiment.

Narrative Content and Path Length

In evaluating narrative content we found that the dense world had the highest average narrative content and path length. This supports our hypothesis that the dense world would be easier to generate narratives for, specifically for narrative

Metric	Sparse World	Dense World	Skyrim World
Narrative Content	5.87 ± 1.68	6.43 ± 2.07	4.64 ± 1.14
Longest Path	5.14 ± 1.38	5.36 ± 1.31	4.30 ± 0.52
Shortest Path	4.52 ± 1.53	4.68 ± 1.47	4.0 ± 0.0
Average Path	4.83 ± 1.41	5.02 ± 1.38	4.17 ± 0.25
Most Branches	0.66 ± 0.47	0.68 ± 0.71	0.29 ± 0.47
Fewest Branches	0.66 ± 0.47	0.54 ± 0.50	0.28 ± 0.45
Average Branches	0.66 ± 0.47	0.61 ± 0.59	0.28 ± 0.46
Highest Cost	0.78 ± 0.72	0.66 ± 0.87	0.30 ± 0.52
Lowest Cost	0.09 ± 0.28	0.0 ± 0.0	0.0 ± 0.0
Average Cost	0.41 ± 0.37	0.26 ± 0.33	0.15 ± 0.24
Most Encounters	0.66 ± 1.36	0.68 ± 1.36	0.0 ± 0.0
Fewest Encounters	0.48 ± 0.96	0.51 ± 0.98	0.0 ± 0.0
Average Encounters	0.57 ± 1.16	0.60 ± 1.16	0.0 ± 0.0
Highest Uniqueness	0.97 ± 0.08	0.97 ± 0.08	1.0 ± 0.0
Lowest Uniqueness	0.91 ± 0.14	0.90 ± 0.13	1.0 ± 0.03
Average Uniqueness	0.94 ± 0.10	0.95 ± 0.10	1.0 ± 0.01
Narrative Richness	0.55 ± 0.50	0.20 ± 0.40	0.19 ± 0.39
Weight of Choices	0.16	0.09	0.02

Table 5–2: The mean and standard deviation evaluated for using ReGEN with various social graphs

rewrites. Path length and narrative content are directly related to the number of rewrites performed on the initial narratives. So if there are more relations, we hypothesized that this would result in more rewrites being applied, the results of which would show up as an increase in the narrative content and path length metrics. While the values were close between the sparse world and the dense world, there was a definite decline in the performance in the *Skyrim* world. The average path length of the narratives for *Skyrim* was 4.17 compared to the sparse and dense world which scored 4.83 and 5.02 respectively. Likewise, the standard deviation for average path

length in the *Skyrim* was only 0.25, which strongly indicates that fewer secondary rewrite rules were applied on average.

Analyzing the *Skyrim* social graph gives some insight into the noticeably lower metric scores, both in narrative content and in general. First off, the density of the world was even less than the sparse world, being on average only 1.8 relations per object. Secondly, our rule set focuses heavily on working with the extreme relations, such as *hates* or *loves* relations. However in the *Skyrim* relation graph there are 41 loves relations, and only eight hates relations. There are, however, 248 allies relations and 143 friends relations. While we have initialization rules that require allies or friends relations, there are no rewrite rules that use these. These patterns could be attributed to the fact that the relations within *Skyrim*'s game world were aimed more at defining AI behaviours, as opposed to being used for narrative generation purposes. Overall, this indicates that in using ReGEN it is important to either tailor the rules to work with the predominant relations, or tailor the game world to closely match the rule set. We do, however, find it impressive that we are still able to get decent results using the *Skyrim* social graph without any changes to our existing rule set and we believe that tailoring the rules to more closely reflect the relation distribution of the *Skyrim* world would improve these results.

Using our narrative criteria we can state that a dense world provides a marginally better narrative length than a sparse world. Of important note is that, since our narratives will be generating and modifying relations, as more narratives are generated there is generally an increase in the density of the game world. This is expected behaviour and thus it is possible that a sparse world will eventually resemble a dense

world simply due to the relations created within the narrative. The downside is that the longer the system runs, the more high-cost actions become performed. This removes many relations and results in it being difficult to map the change in density in the game world over time.

Narrative Branching

The amount of branching in a narrative results from the number of secondary rewrite rules applied, since none of the initial rewrite rules create branches. This makes it very similar to the discussion for narrative content and path length and we notice the same patterns here that were noticed in the previous discussion. Again the dense world and sparse world performed very closely, 0.66 and 0.61 on average. This means that approximately every other narrative generated will have at least one branching path. The *Skyrim* world fell short scoring only 0.28. This again can be attributed to the analysis provided above; since less SRRs are applied in the *Skyrim* world, it will score much lower in any metrics which are directly affected by the secondary rewrite rules.

In this field, the dense world outperformed the sparse world using our criteria, however the standard deviations are large enough on both (0.47 and 0.59 respectively) that the scores are approximately equal. This is also noted with the narrative content and path length. Since the two worlds differ so largely in their construction, this supports the claim that having a rule set that works with the distribution of relations is more important than the density of relations. This refers specifically to metrics that are directly related to the number of SRRs applied or the quality of IRRs.

Cost and Encounters

Cost and Encounters follow the same rule as branching and narrative content, with the difference being that they also have a dependence on the initial rewrite rules. This is because whether a quest will involve either cost actions or encounters is usually based off of the initial quest generated. Thus, a world that generates more murder quests will have a higher cost whereas a world with more fight quests would result in a higher encounters score. The average cost is 0.41 for the sparse world, 0.26 for the dense world and 0.15 for the *Skyrim* world. Thus, for cost the *Skyrim* world performs the best out of all three game worlds. Returning to our more in-depth analysis of the *Skyrim* relation graph used, we again note that murders are one of the biggest causes of cost events, and that these occur due to a *hates* relation; unsurprisingly then, with only eight starting hates relations in the *Skyrim* world, there will be relatively few murders. The murders would increase if more hates relations are created, but this can be a lengthy process and it could take many quests before the number of murders match those of the dense or sparse world.

The sparse world has the worst cost metric, although with the rewrite rule that allows the player to spare their victim we do note that all three game worlds have a lowest cost of almost zero. More importantly, since the sparse world has the least number of objects, the high cost indicates that an end state in which no more narratives are generated will be reached quickly. The speed at which the end state is reached is dependent, however, on the player's choice of whether to save or kill their victims when completing the generated quests. Comparatively, the *Skyrim* world has very few cost events and many objects, meaning we expect the number

of possible narratives to be much larger before reaching the end state. While we do list cost as a negative metric, having a situation with many cost events and a small world inherently leads to a very interesting situation with the player's actions having a great impact on the world around them and many consequences for those actions. We noticed this effect reflected in the narrative richness and weight of choices metrics, which will be discussed later. This shows that again, it is up for the author to decide whether the game world should be designed to last longer, or be more dynamic.

Encounters are similar to cost, although due to an unfortunate lack of data we did not include any of the enemies in the *Skyrim* world, and thus cannot compare those results. This however, seemed fitting since *Skyrim*'s quests generally expect the player to encounter monsters randomly while travelling, and thus do not generally model these encounters in their narrative. Out of the two worlds, the score of 0.60 for the dense world was about the as the score of 0.57 for the sparse world, with high standard deviations for both.

Uniqueness

Uniqueness was generally high for all three game worlds. Interestingly, the *Skyrim* world had very high values for uniqueness. As noted during experiments, the narrative content and uniqueness seem to be conflicting metrics, and generally having a higher score in one reduces the other's score. Thus, Skyrim having every node on average being unique is the result of their much smaller narrative content. As more rewrite rules get applied, the more likely there are to be repeated actions. This is reflected with the sparse and dense world both having larger narrative content, but only scoring 0.94 and 0.95 average uniqueness respectively.

Narrative Richness and Weight of Choices

Both narrative richness and weight of choices were highly in favour of the sparse world. As mentioned before in the cost section, the small size allows for intense player interaction and consequence. Likewise, with a sparse set of relations, over time we expect that many of the relations will end up being relations created due to player actions. The metric results show the extent of this effect. The narrative richness score of 0.55 indicates that at least half the narratives generated occurred directly due to the actions taken by the player. This was a much better score than the dense and *Skyrim* worlds, which both score approximately 0.20.

For weight of choices, we see that in the sparse world, the player can generally affect around 16% of the game world with the choices they make. Note that there is no standard deviation for weight of choices since it is a single value calculated for all possible final game states. This is much more significant than the 9% score for the dense world or the even lower 2% score for *Skyrim*. In this instance, we see that the size of the game world plays a large role in the weight of choices. Also, although 16% may sound low, even this amount means that many of the final game worlds are exhausted (all NPCs dead), and no new narratives can be generated for them. This means that even small scores can result in dramatically different possible final game worlds. What is important to note is that narrative richness and weight of choices are heavily dependent on how many narratives have been generated, and tend to increase the longer the system is running. In running one hundred tests, we accounted for a fairly large number of quests for a player to complete. While similar results to the sparse world could be achieved for the other two game worlds,

this would result in the player having to complete possibly thousands of quests to see similar results. This may be desirable or undesirable depending on the author's intended use of the ReGEN system.

Summary

In this experiment we aimed to examine the relation between game world and narrative quality. We discovered that, while it was possible to use the same rule set to generate quests for *Skyrim*, the quests are not at the same quality as the game worlds that were originally used as inspirations for the rules. This means that in using ReGEN, the user must have an understanding of the structure of the game world and the distribution of relations, and use this to develop rules that work with the predominant relations. Simply doing this resulted in essentially equal results for both a sparse and dense world with regards to many metrics, such as narrative length, branching, encounters, cost and uniqueness.

Narrative richness and weight of choices were shown to be directly influenced by the size of the game world as well as the number of narratives that have been generated. While simply generating more narratives will improve both metrics over time, the best results will always be for smaller, sparse game worlds. This is because in these worlds, the player's actions have a much greater impact and it is a lot quicker to make the narrative generated relations more predominant than the initial relations.

5.3.2 Performance Evaluation

In this section, we present a very brief discussion of the performance analysis of the system during the previous experiment. To do this, we measured the average

time taken to generate a narrative as well as what percent of that time was spent initializing the narrative, and what percent was spent during the rewrite phase. For the sparse world, the average time to generate a narrative was 0.12 seconds, with 57% of the time being used for initialization and 43% used for the rewriting process. For the dense world, the average time was 3.61 seconds, with 80% of the time being used for initialization and 20% of the time for rewrites. Lastly, generating a narrative with the *Skyrim* world took on average 57.73 seconds with 77% for initialization and 23% for rewrites.

The purpose of this analysis was to examine the performance of the system with different sized game worlds. With these results we find a noticeable delay during the initialization phase. In our generation process, to initialize a narrative we start by determining all of the possible narratives. Each of these involves a subgraph isomorphism check within our game world. As more and more potential initial narratives are possible, this in turn increases the time to check and analyze each of these possible narratives. In order to reduce the amount of time spent on initialization rules, future work should aim to optimize this step, possibly by stopping as soon as at least one narrative has been found. Likewise, by optimizing the system and implementing it in a more optimized language such as C would decrease the generation time as well. We do, however, feel the generation times to be acceptable especially given the size of the larger game worlds.

5.3.3 Comparison with Other Narratives

For our tests we parsed all four of the other narratives into the format of our narratives. This was relatively straightforward as all narratives follow a similar

narrative structure that can easily be represented in graph form. As explained above, we used highly detailed wikis for to get the structure for *Skyrim* and *The Witcher* whereas the SQUEGE output is already presented in the form of a directed acyclic graph. For evaluating uniqueness, this involved giving each event in the narrative a *type*, assuming each event generally revolved around one action, such as fighting, gathering info or travelling. We could verify this once again by using the wikis, which define each action that must be taken in order to proceed in the quest. We then used our metrics to analyze each of the stories in terms of *narrative content*, *longest/shortest/average path*, *most/fewest/average branches*, *highest/lowest/average uniqueness*, *most/fewest/average encounters*, and *highest/lowest/average cost*. The results are presented in 5–3.

Metric	SQUEGE	Radiant Quests	Skyrim Main Quests	Witcher Main Quests
Narrative Content	5.08 ± 3.25	2.33 ± 0.85	5.12 ± 2.37	15.41 ± 9.52
Longest Path	4.13 ± 1.64	2.17 ± 0.80	5.12 ± 2.37	12.09 ± 6.41
Shortest Path	3.04 ± 0.79	2.17 ± 0.80	5.12 ± 2.37	11.18 ± 6.58
Average Path	3.78 ± 1.29	2.17 ± 0.80	5.12 ± 2.37	11.61 ± 6.46
Most Branches	1.63 ± 1.07	0.17 ± 0.37	0	1.26 ± 1.09
Fewest Branches	1.17 ± 0.37	0.17 ± 0.37	0	1.26 ± 1.09
Average Branches	1.48 ± 0.88	0.17 ± 0.37	0	1.26 ± 1.09
Highest Cost	1.42 ± 0.91	0	0.29 ± 0.46	0.21 ± 0.40
Lowest Cost	1.29 ± 0.93	0	0.29 ± 0.46	0.12 ± 0.32
Average Cost	1.38 ± 0.89	0	0.29 ± 0.46	0.17 ± 0.34
Most Encounters	0	0.42 ± 0.57	0	1.44 ± 1.90
Fewest Encounters	0	0.42 ± 0.57	0	1.24 ± 1.88
Average Encounters	0	0.42 ± 0.57	0	1.33 ± 1.89
Highest Uniqueness	0.68 ± 0.14	0.95 ± 0.12	0.72 ± 0.20	0.59 ± 0.25
Lowest Uniqueness	0.58 ± 0.10	0.95 ± 0.12	0.72 ± 0.20	0.53 ± 0.24
Average Uniqueness	0.62 ± 0.09	0.95 ± 0.12	0.72 ± 0.20	0.56 ± 0.24
Narrative Richness	0	0	0	0.03 ± 0.16

Table 5–3: The mean and standard deviation evaluated for various types of generated and hand-authored quests

Narrative Content

The results shown in Table 5–3 show our system's metrics being comparable to both the *Skyrim* main quests, as well as the SQUEGE output in some respects, with our system scoring 6.43 for the dense world, compared to 5.08 in SQUEGE and 5.12 in *Skyrim*. Note that we will not refer to the sparse world results except where they differ greatly to the dense world results, since in most cases both results were very close. We will also, unless otherwise stated, refer to our dense world results as our main results, since they represent an optimal configuration of the system.

Our ReGEN quests were comparable to those of SQUEGE and the *Skyrim* main quests, although scoring slightly lower. The results of the analysis for *The Witcher*'s main quest-line appear to be, in general, very different than our previously examined systems. As expected, our system, the "good" quests, and the SQUEGE output show an improvement over *Skyrim*'s *Radiant Quest* results given that we defined a larger narrative content as being indicative of a better narrative. In terms of narrative content, SQUEGE and the *Skyrim* main quests exhibit on average five distinct narrative events and our narratives have six. The *Skyrim Radiant Quests* average only two events which, by our definition above, we consider to be a poor result. *The Witcher* exhibits a much a larger set of narrative events, providing nine to ten more events on average than our system, *Skyrim*, or the SQUEGE quests.

Longest/Shortest/Average Path

Our system's longest and shortest path were 5.4 and 4.7 respectively with the average path being 5.0, meaning on average a player will see between *73-84*% of the narrative's content on a single playthrough. Comparatively, a player will see

around *59-80*% of narrative content for a given SQUEGE narrative and *72-79*% for *The Witcher* Main Quests. For the *Skyrim* main and Radiant quests we see a much higher percentage of narrative content, 100 and 96 percent respectively, which is due to the complete lack of branching paths in *Skyrim*'s main quests and the minimal use of branching paths in the *Radiant Quests*. If we follow our assumption that the "good" quests present superior quest design, and assume that a well designed quest will allow the user to see between 70-100% of narrative content independent of the path followed, then we can state that our quests fall within this boundary. Alternatively, we could state that a good branching story presents between 70-80% of narrative content, comparable to the narrative content results from *The Witcher*, and somewhat above the amount of choice (59%) present in SQUEGE output. In this case, our results still closely fall within this range. Using our criteria for a good narrative, we find that our system outperforms the *Radiant* and SQUEGE quests. We match the values of path length found within the *Skyrim* main quest, scoring on average 5.0 events comparing to the 5.1 events contained on average in a *Skyrim* main quest. Some difference between the longest and shortest path are to be expected since the lack of branching in the *Skyrim* quests mean that users will always see all possible narrative content.

Looking at the difference between the amount of narrative seen when following the longest and shortest path, based on the percentages we previously gave, we see that there is a difference of *11*%. Meaning that if the player follows the shortest path in an average narrative, they see eleven percent less of the total content than if they took the longest path. This difference is higher than the *7*% difference in

The Witcher, and less than the *21%* of the SQUEGE narratives. A lower difference implies that regardless of which path the player takes through the narrative they are still able to see much of its content, and that choices impact specific narrative results rather than representing fundamental, highly disjoint story branching. These results correspond nicely with our criterion that having a smaller difference between a longest and shortest path is indicative of a good story, and more consistent gameplay.

Most/Fewest/Average Branches

The number of branches show that on average, a little over one in every two of our generated stories will contain at least one branching path, having an average of 0.62 branches per narrative. Following our criteria, this falls well below the results of both the SQUEGE narratives and *The Witcher* narratives, where *The Witcher* has on average one branching path per narrative and the SQUEGE narratives have closer to two branching paths. The *Skyrim* main quests are always linear and therefore do not have any branches, whereas the *Skyrim Radiant Quests* very infrequently contain branching paths. Since our criteria states that a large number of branches indicate better narratives, then the SQUEGE and *Witcher* narratives are superior in this area to our generated metrics. Since we have made the assumption that *The Witcher* quests are representations of good narrative structure, we should therefore add more rewrite rules which create branching paths in the narrative in order to compete with its results. This is again one of the strong benefits of defining narrative metrics and comparing our generated narratives to others, since it gives insight into how we can restructure our narrative generation tool in such a way that it produces measurably better narratives.

Highest/Lowest/Average Cost

Comparing costs, we see that our stories contain, on average, 0.26 irreversible actions per narrative. Although observing highest and lowest cost, we see a much greater divide, with the average highest cost being 0.66 and the lowest being 0. The results for SQUEGE are even higher, with at least one cost unit per narrative regardless of path chosen. The *Radiant Quests* always have no cost because all destroyed objects are procedurally generated for each quest. While this is an interesting means of having narrative generation with no potential cost to game environment, it also takes away a sense of purpose from the side quest, since as a result the quest makes no noticeable difference in the game world. Our average cost is comparable to the *Skyrim* and *Witcher* main quests, which have costs between 0.1 and 0.3. In spite of this, our highest cost is still high and aiming to reduce the number of rules which include an irreversible action would improve not necessarily narrative quality, but rather the lifespan of the quest generation process itself.

Most/Fewest/Average Encounters

In examining encounters, we reiterate that in most RPGs, encounters are implied but not explicitly stated in quest structure. For example, in *Skyrim*, a player encounters many monsters travelling through the game world, but these are random encounters and not stated in the quest description. SQUEGE, makes no such explicit definitions of encounter either. However, encounters are explicitly defined in our system, *The Witcher* main quests and the *Radiant Quests*. We define having more encounters as being a positive, and in this instance our system generates encounter events much less frequently than *The Witcher* and slightly outdoes the

Radiant Quest, scoring 0.60 to their 0.42, with *The Witcher* scoring the highest at 1.44. This is again, a feature which can be tweaked with the creation of and/or modification to, the sets of rewrite rules in our system.

Highest/Lowest/Average Uniqueness

The last metric, and one of the more interesting, is that of uniqueness. As mentioned before, this metric is not perfect since we take only the primary action of each narrative event to be a description of that event. This ignores, for example, the implied random encounters in the *Skyrim* world, and does not account for player preference with regards to narrative content. We still feel however, that provides strong insight into narrative quality as it is the primary events that are usually of the most interest and importance to the player. Given that we regard repetitive events as detrimental to narrative quality, this is one metric of which a higher value directly implies an increase in narrative quality.

What is noted in with these results is that the main quests of *Skyrim* and *The Witcher* have much lower uniqueness scores compared to our system and the *Skyrim* Radiant quests. The uniqueness for *Skyrim* is on average 0.72, whereas the average uniqueness for *The Witcher* is only 0.56. Conversely, ReGEN's average uniqueness for the dense world is 0.95, comparable to that of the *Radiant Quest* system. One reason for this may be that our system and the *Radiant Quest* system are both focused on creating *side quests*, which are separate from the actions in the main quest. For example, a side quest may involve stealing an item, and a player who is not interested in being a thief may not perform this quest. However, in the main quest the player must perform all of the events in order to complete the game, so

the main quest may aim to make stories which are of interest to all possible players. This is a relatively small subset of all the possible narrative events, and would result, therefore, in a lower uniqueness score. Conversely, the size of the narrative content may have an effect on the uniqueness. Since, as more content is added, the more likely it is that an action in the set of all possible actions will appear more than once. Thus, the uniqueness score of 0.5 for *The Witcher* could be a consequence of it having a much higher narrative content, that of fifteen narrative events on average per story. We found that the SQUEGE generator performed poorly in this metric, having an average uniqueness of only 0.6 despite the smaller narrative size. This poor score, though, could be more attributed to the limited number of rules as opposed to the actual functioning of the system.

Narrative Richness

We then examined both our narratives and the main quest narrative of *The Witcher* in terms of our *narrative richness* metric. The *narrative richness* metric could really only be evaluated for these two sets of quests since it is only in these two systems that actions taken by the player can have unforeseen consequences in later narratives. Although the *Skyrim* quests have some indirect effects on the *Skyrim* world, these changes do not effect the main quest, whereas in *The Witcher* indirect impact is an important part of player (and narrative) choice [22]. Our narrative richness results for all three of the game worlds previously tested were shown to be much larger than the results for *The Witcher*. The sparse world had a richness of 0.55, the dense world scored 0.20 and the Skyrim world score 0.19. *The Witcher*, however, only received 0.03. This means that our generated quests were more strongly

influenced by previous player actions. We find this to be a very positive result that indicates the potential for our system to expand upon an area of *The Witcher* that got much praise, but was still relatively undeveloped.

5.3.4 Radiant Quest Experiment

As an additional experiment, we performed a test to analyze the effect of using our system with the narratives given in *Skyrim*'s *Radiant Quest* system as the base narratives in our IRR. In this way, we could then observe the changes in metrics when our SRRs are applied to the resulting narratives. This experiment shows the flexibility of our graph rewriting system, and how it can adapted to suit the system in *Skyrim*. We also show that by applying our SRR rules, we can create measurably better narratives, which is an argument again in favour of our system as a means for narrative generation. The results for this experiment are shown in Table 5–4, with the first column showing the results for the *Radiant Quest*, the second column showing the results of simply using the base narratives from the *Skyrim Radiant Quest* system and the third showing the results after our SRR rule layer is applied.

Metric	Original Radiant Quests	Radiant Quests (ReGEN)	Radiant Quests + SRR (ReGEN)
Narrative Content	2.33 ± 0.85	2.51 ± 1.25	3.68 ± 1.95
Longest Path	2.17 ± 0.80	2.30 ± 1.12	2.74 ± 1.11
Shortest Path	2.17 ± 0.80	2.30 ± 1.12	2.36 ± 1.04
Average Path	2.17 ± 0.80	2.30 ± 1.12	2.57 ± 1.07
Most Branches	0.17 ± 0.37	0.21 ± 0.41	0.37 ± 0.48
Fewest Branches	0.17 ± 0.37	0.21 ± 0.41	0.37 ± 0.48
Average Branches	0.17 ± 0.37	0.21 ± 0.41	0.37 ± 0.48
Highest Cost	0	0.32 ± 0.47	0.66 ± 0.89
Lowest Cost	0	0.19 ± 0.40	0.0 ± 0.0
Average Cost	0	0.26 ± 0.40	0.29 ± 0.40
Most Encounters	0.42 ± 0.57	0.26 ± 0.44	0.28 ± 0.45
Fewest Encounters	0.42 ± 0.57	0.14 ± 0.34	0.0 ± 0.0
Average Encounters	0.42 ± 0.57	0.20 ± 0.36	0.12 ± 0.20
Highest Uniqueness	0.95 ± 0.12	1.0 ± 0.0	1.0 ± 0.0
Lowest Uniqueness	0.95 ± 0.12	1.0 ± 0.0	0.91 ± 0.14
Average Uniqueness	0.95 ± 0.12	1.0 ± 0.0	0.96 ± 0.07

Table 5–4: The results of running our system as a radiant quest system

The first thing to note is that, due inherently to the way we designed our system, murder actions now have cost. This is in contrast to the *Radiant Quests* in *Skyrim* where all murder victims are procedurally generated. We left in this cost simply to show how it changes once we apply our SRR rules. The first column effectively matches the metrics given in the results for the *Radiant Quests* in Table 5–3. Differences can be attributed to the fact that our system picks the next narrative at random from all potential narratives, meaning some variation is likely to occur simply due to the randomness of the selection process. After modifying the system to use our SRR graph rewrite rules, we see a noticeable improvement in the metrics based around our original definition of narrative quality, such as with the content and number of branches. There is a raise in the highest cost, which we define as detrimental to narrative quality, but interestingly the lowest cost has been reduced to zero (one of the SRR rules allows the player to spare the victim they are supposed to murder, which likely resulted in this change). The number of encounters remained the roughly the same. We noted before that our rules are weak in this area, and adding more rules would most like improve our encounter metric given in Table 5–3. The uniqueness score declined slightly, showing a loss in narrative quality in this area. This supports the argument that an increase in narrative content will invariably lead to a decrease in uniqueness since the set of possible narrative events in a game is finite.

5.3.5 Analysis of Skyrim as One Long Quest

Our final experiment was to metrically analyze *Skyrim* as one long main quest, as opposed to several shorter quests making up the main quest. Since there are

seventeen quests in the *Skyrim* main quest, we tested the result against linking seventeen of our quests together into one long quest. Here, we were trying to examine the decline of uniqueness in narratives as dependent on content. The results show that the Skyrim main quest consisted of 87 events, but its overall uniqueness was only 0.13. Likewise, when linking seventeen of our own quests together, we ended up with 100 events, but a uniqueness score of only 0.11. If we push this further, linking 100 of our side quests into one quest, we end up with 269 events but a uniqueness score of only 0.04.

These results show the decline in uniqueness for both the *Skyrim* main quests, and of our own system over time. Interestingly, since uniqueness is in terms of how many narrative events are unique, multiplying narrative content by uniqueness gives us the total number of unique narrative events for each of the narratives tested. For both our system, and *Skyrim*'s main quest, this gives a total of eleven unique narrative events possible, meaning we again have measurable comparable metrics in terms of quality to those of *Skyrim*.

Understandably, many of these metrics are the result of the ways our rules are defined, and tweaking the rules can improve/worsen many of the results. However, this is one of the important reasons why we chose to design metrics. By performing these analyses we are able to determine which story structures are considered "good" and can then tweak our generation tool to make improved narratives.

CHAPTER 6
Conclusion

In this work we presented a system for generating narratives using a graph rewriting process. The focus of this work was on creating side quests at a low cost to developers while overcoming the many shortcomings of the current level of commercial narrative generation techniques. We likewise aimed to overcome some of the issues noted in previous works in the field of narrative generation, such as long generation times or the actions taken not actually forming a cohesive story.

Since repetition and the disconnect between player actions not having an effect on the game world were two common criticisms of many generated quests, we designed a tool based on a formal graph representation of the current game state, called the *game world* when creating narratives. This allowed the narratives to be generated using the current state of the game world. We likewise allowed the narratives generated to affect the objects and relations in the game world. This meant that player actions could affect the game world and that this modified game world could in turn affect the narratives generated. Our solution to the repetitive structure in many generated quests was to use multiple iterations of the graph rewriting technique to continuously expand and modify the game narrative, adding in important features such as player choice. We used a novel set of metrics that evaluate the quality of the narratives generated by examining the narrative's structure and content. These were used to guide the generation process to create narratives of a

certain quality. We lastly used a level of validation to ensure that every narrative generated is completable.

Our nine metrics were aimed at examining basic structural properties, such as narrative length, the number of fights and uniqueness. We also examined qualities of narratives that are specific to game narratives, such as the amount of player choice and the potential irreversible actions that shrink the size of the game world. These more abstract qualities examined how the player actions lead to unforeseeable consequences in later narratives, quantifying the effect of player choice on the game world.

For our experimentation, we were interested in validating our metrics as well as examining the performance of our system. We initially sought to test how the size and structure of the game world affected the quality of the narratives. The results show that it is important to create rules that respect the distribution of relations in the game world. It was also shown that narrative richness and weight of choices are heavily dependent on the size of the game world and the amount of narratives which have been completed. We then compared our results against the main quests from *Skyrim* and *The Witcher*, several quests from the SQUEGE generation tool, and the *Skyrim Radiant Quest* templates. Our results show that our system compares favourably with the hand-authored main quests, and shows where we could improve the rule set to create even higher quality narratives. We demonstrated the flexibility of our system by modifying it to behave as the Skyrim *Radiant Quest* system, and then showed how our graph rewriting methods could create a measurable improvement in the quality of narratives generated. Lastly,

we demonstrated the conflicting relation between narrative content and uniqueness, showing how larger narratives inevitably reduce the uniqueness score. It was shown that since there are only a certain number of player actions possible, with a higher number of events we will inevitably begin to repeat actions. These results show the potential of using a formal analysis of narrative quality to examine both the narratives and the narrative generation system itself.

6.1 Future Work

The results from our experimentation into the performance of our system showed that it would take approximately a minute to generate quests using a commercial scale game world. While this may not be an ideal result, around 75% of this development time was only on the first step of generation, finding a potential story. Since this step involves finding all possible stories and picking one at random, optimizing this part of the narrative generation process could substantially reduce the generation time. This can be done by stopping the process after a single narrative has been found, for example and would make the system a viable option for commercial developers as well as research.

The metrics defined gave a good analysis of story quality, but even more metrics could be defined. There are relatively few metrics which measure the quality of a narrative against the existing narratives, what we call *inter-narrative* metrics. A specifically important metric would be an inter-narrative metric for uniqueness, which compares how similar a generated narrative is to previously generated narratives. Likewise, performing a user study would allow us to to evaluate the effectiveness

of the metrics defined. Comparing the user's opinions on narrative quality to the quality as calculated by our metrics could validate our choice of metrics.

Larger scale problems which could be the focus of later research include narrative generation for multiple stories. In most commercial games the player may be active in multiple quests at once. This would involve generating multiple narratives at a time, and ensuring that each narrative never conflicts with any of the actions in the other active narratives. Additionally, in order to be commercially viable there would need to be some form of dialogue generation. This is a research domain in itself but since most narrative games involve conversations between the player and NPCs, there would need to be some dialogue created for every quest. By providing reasons for our relations, we provided some important information that could be used in dialogue generation. The fact that most modern games contain voice-acting, however, leads to further research challenges when attempting to generate dialogue.

Appendix A - Initialization Rules

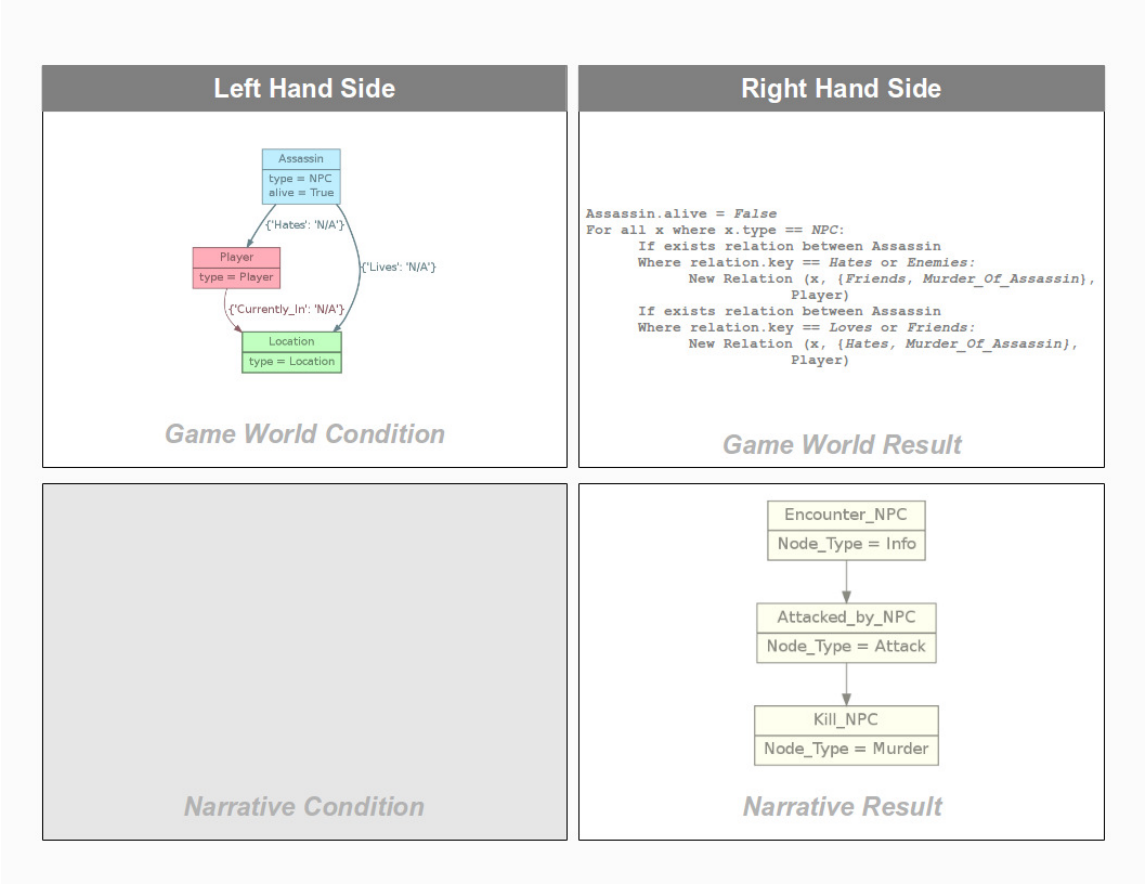

Figure 6–1: Assassination Initialization Rule

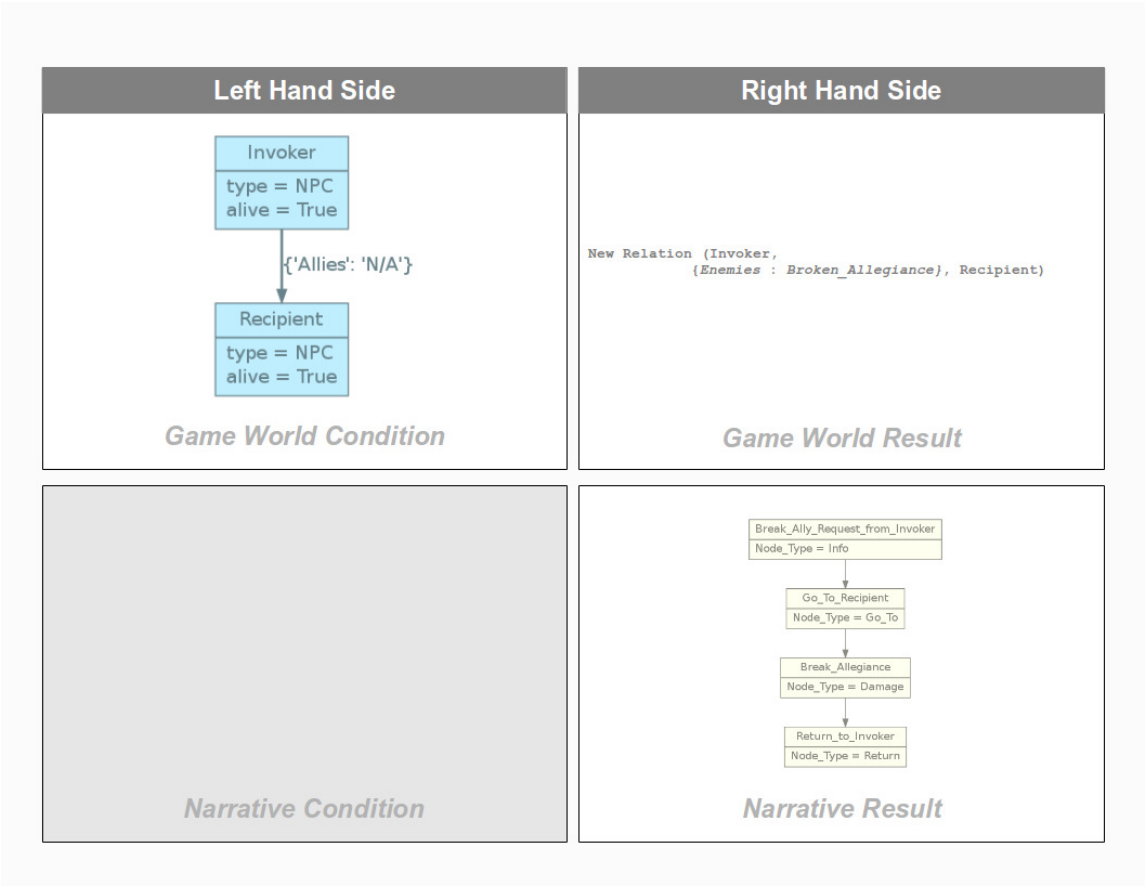

Figure 6–2: Break Ally Initialization Rule

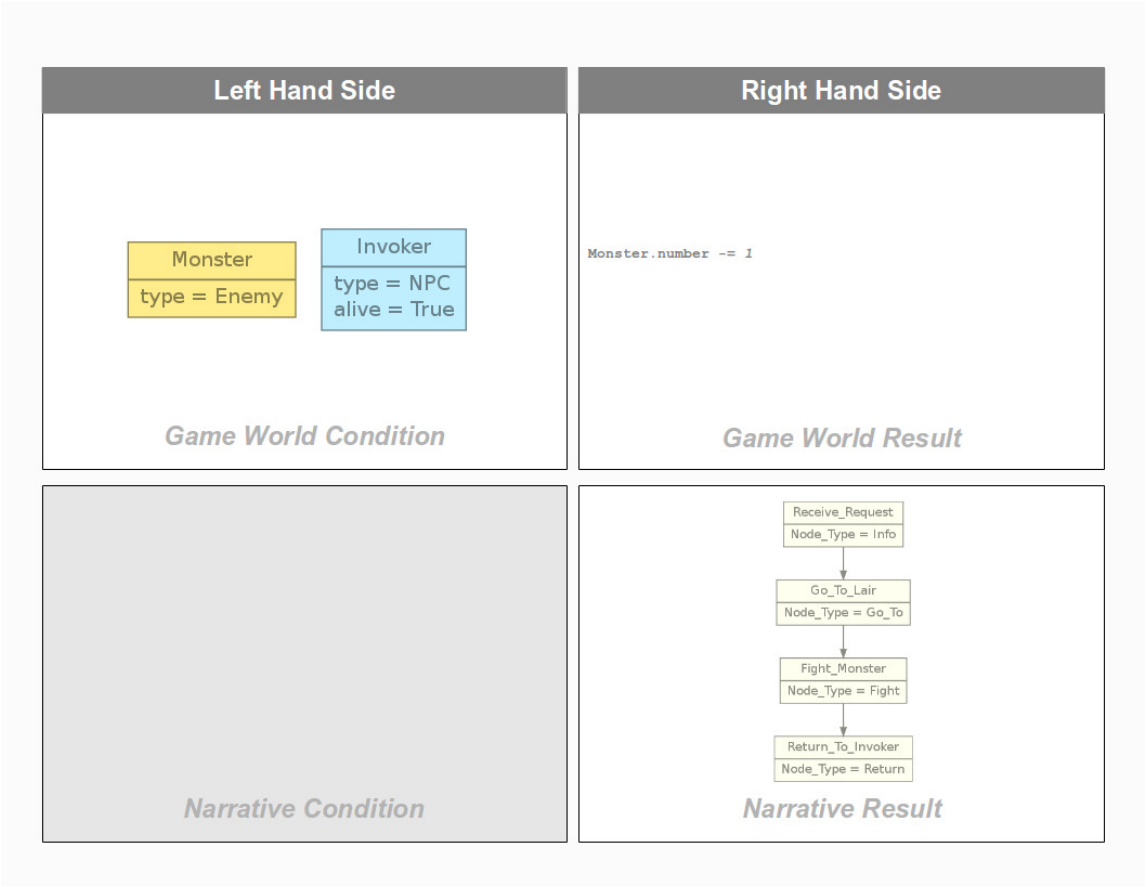

Figure 6–3: Fight Initialization Rule

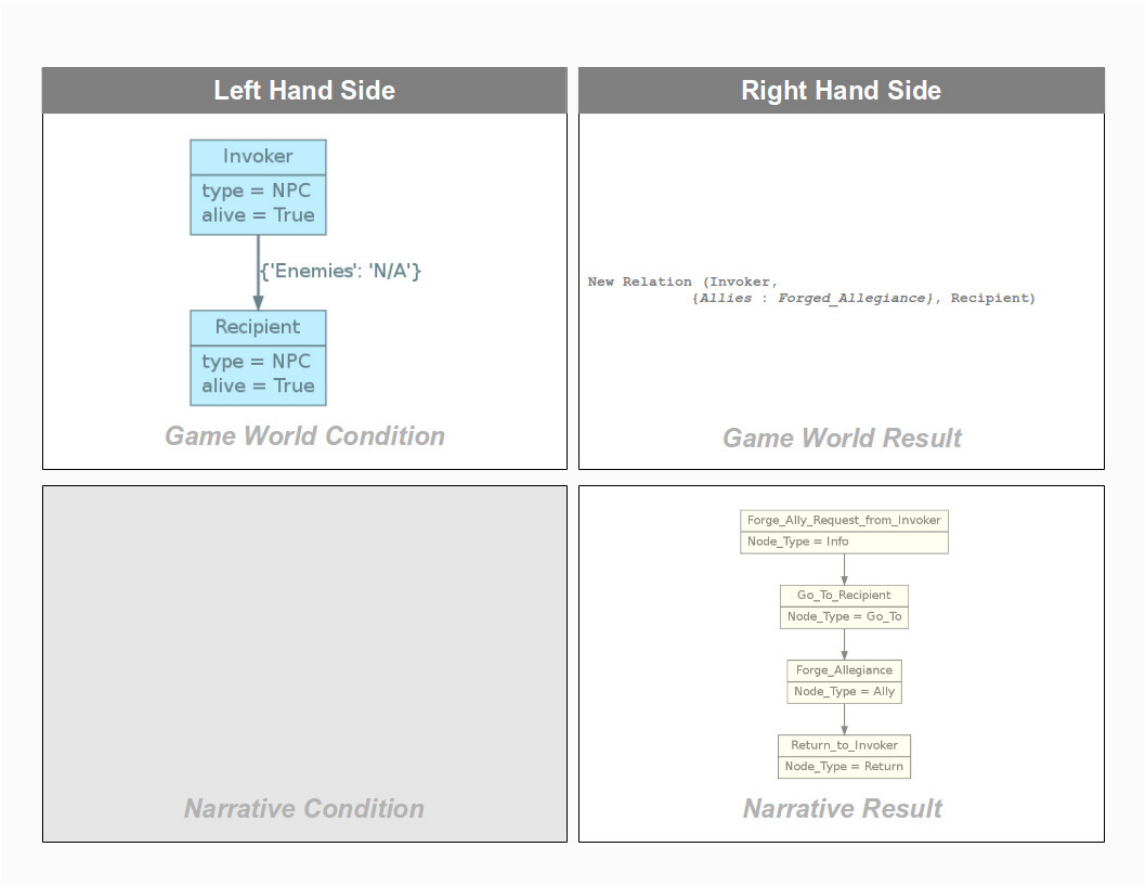

Figure 6–4: Forge Ally Initialization Rule

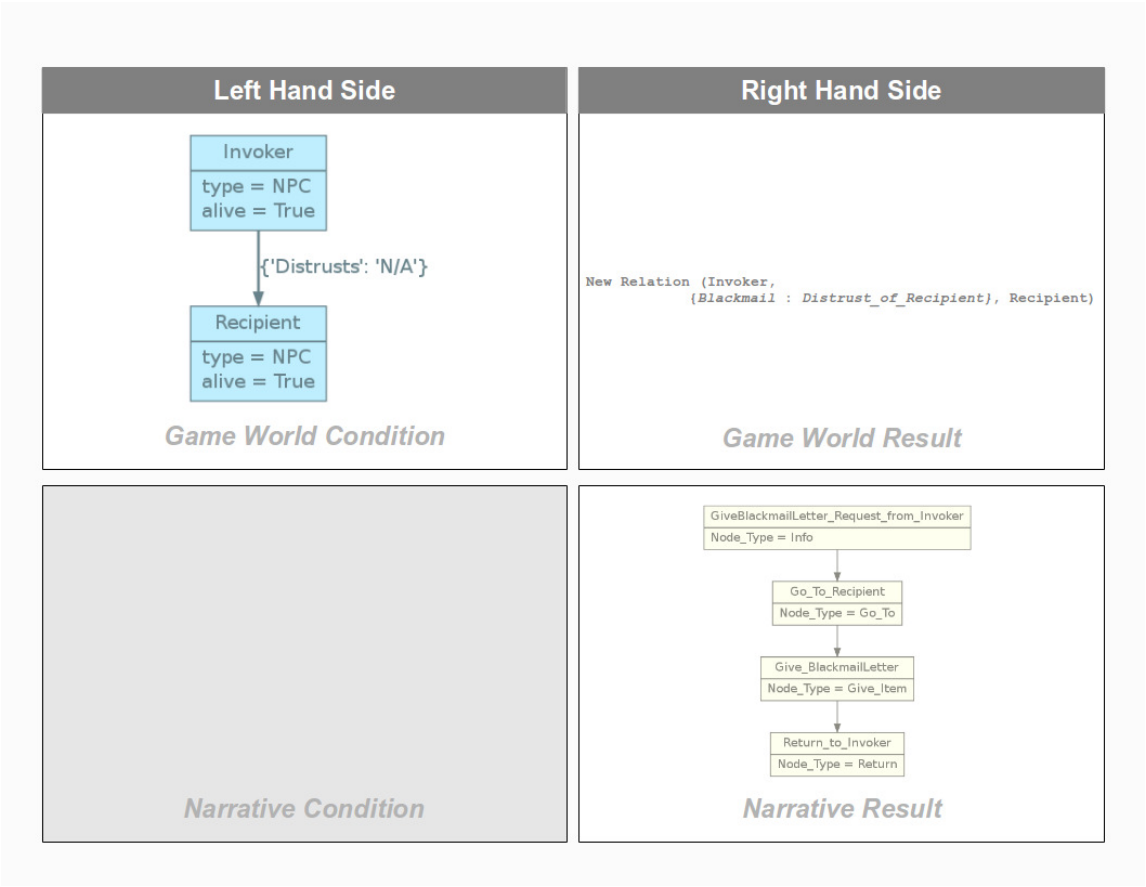

Figure 6–5: Give Blackmail Letter Initialization Rule

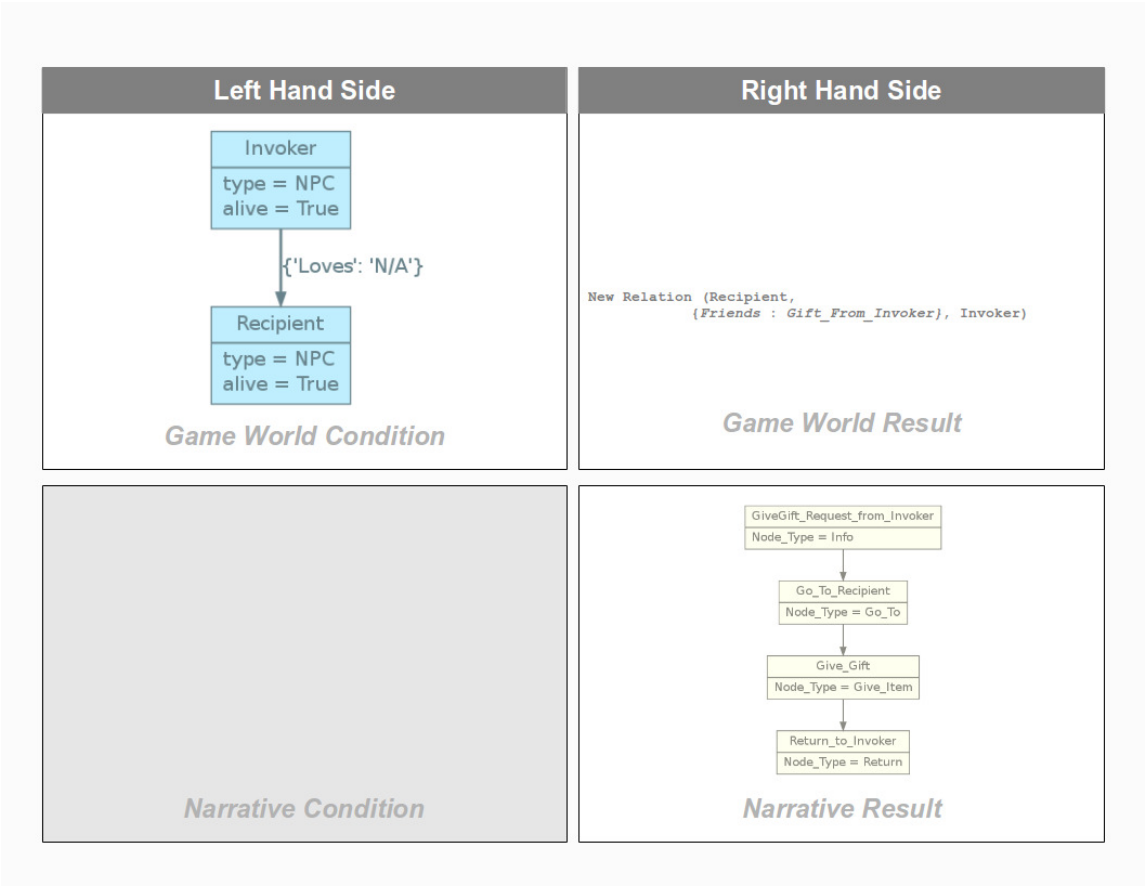

Figure 6–6: Give Gift Initialization Rule

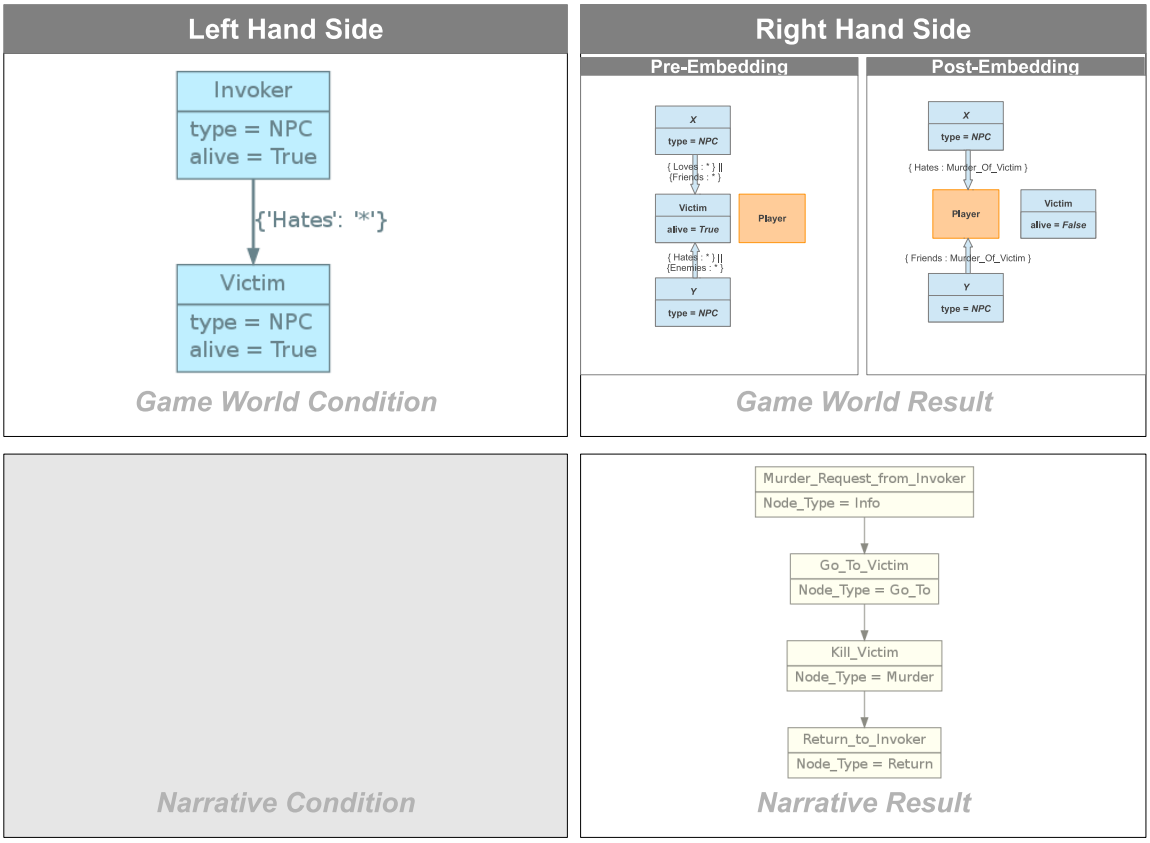

Figure 6–7: Murder Initialization Rule

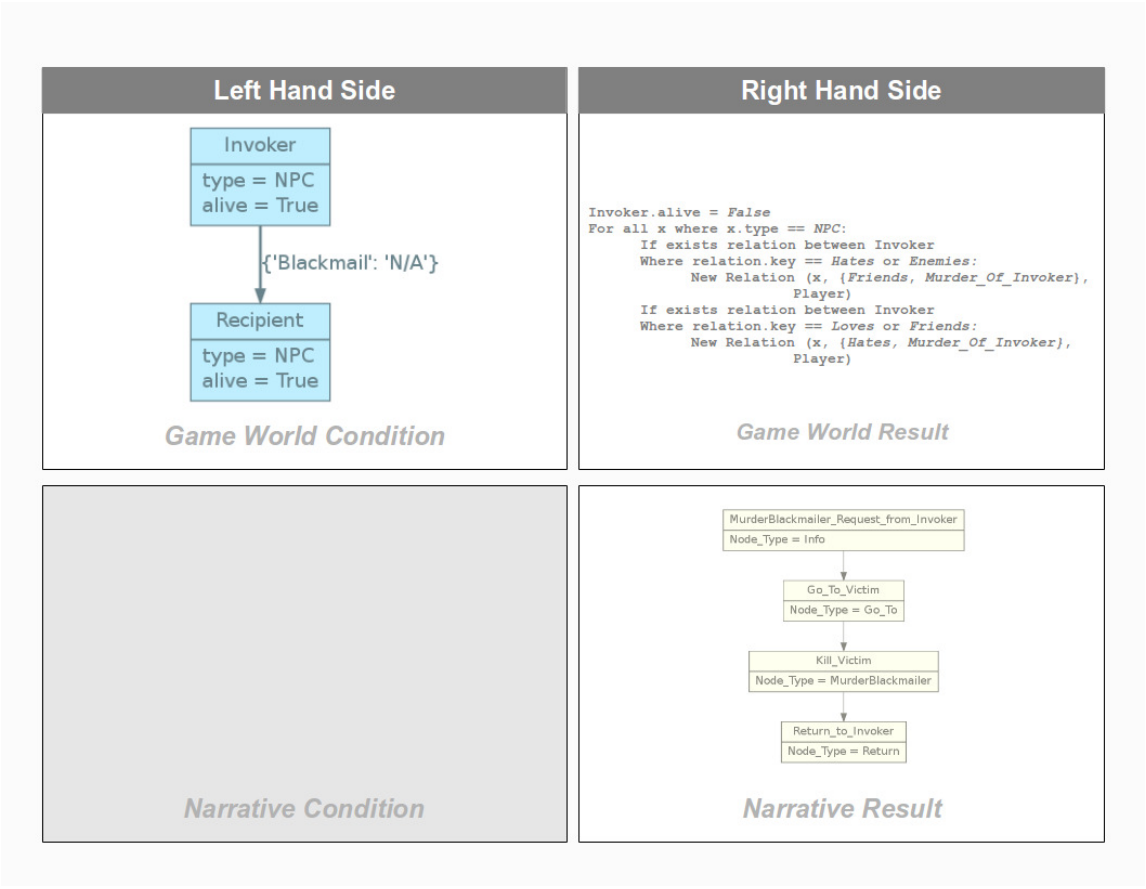

Figure 6–8: Murder Blackmailer Initialization Rule

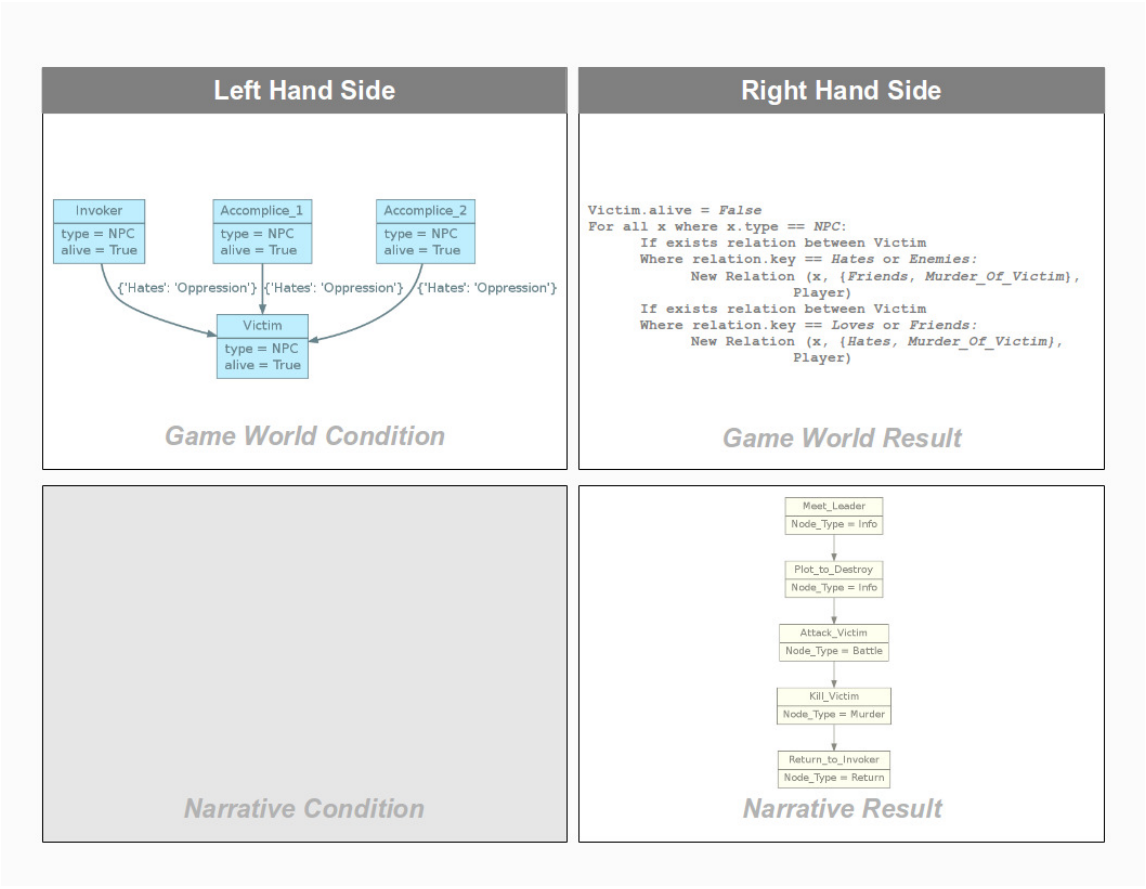

Figure 6–9: Rebel Initialization Rule

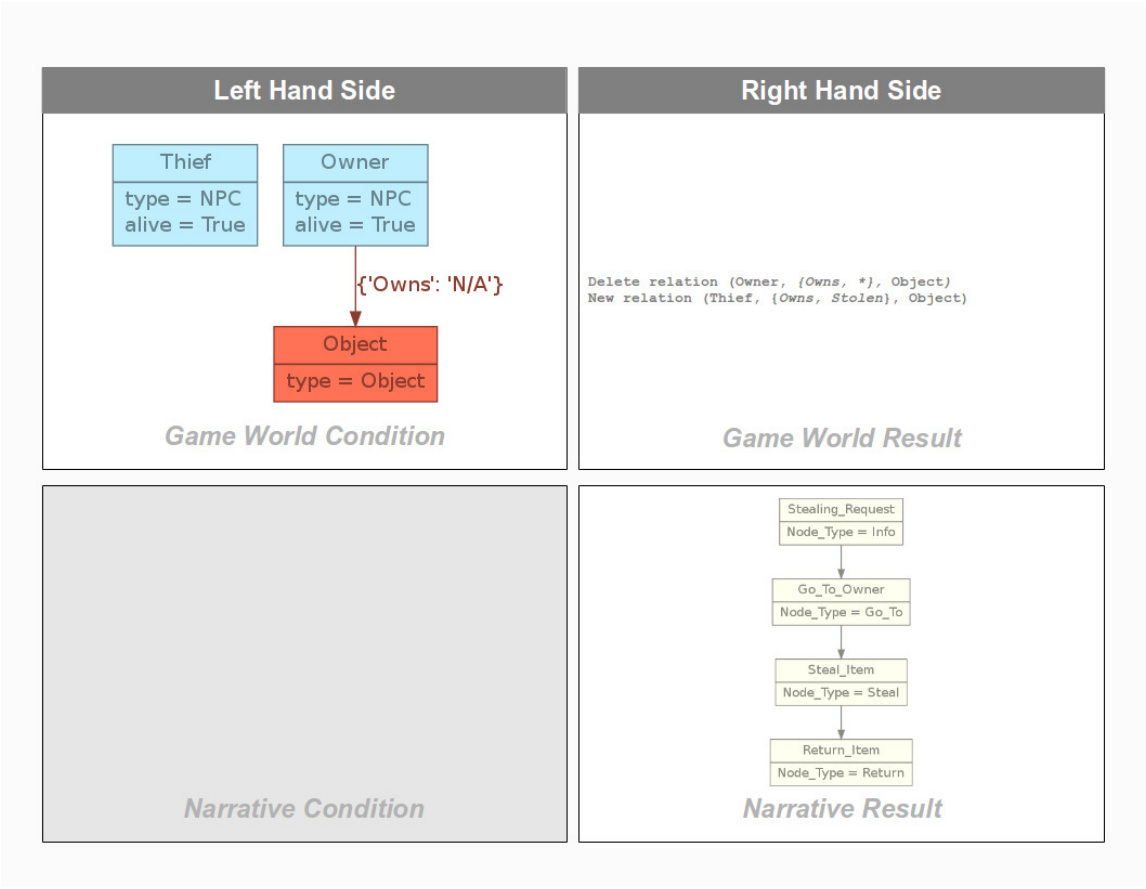

Figure 6–10: Steal Initialization Rule

Appendix B - Secondary Rewrite Rules

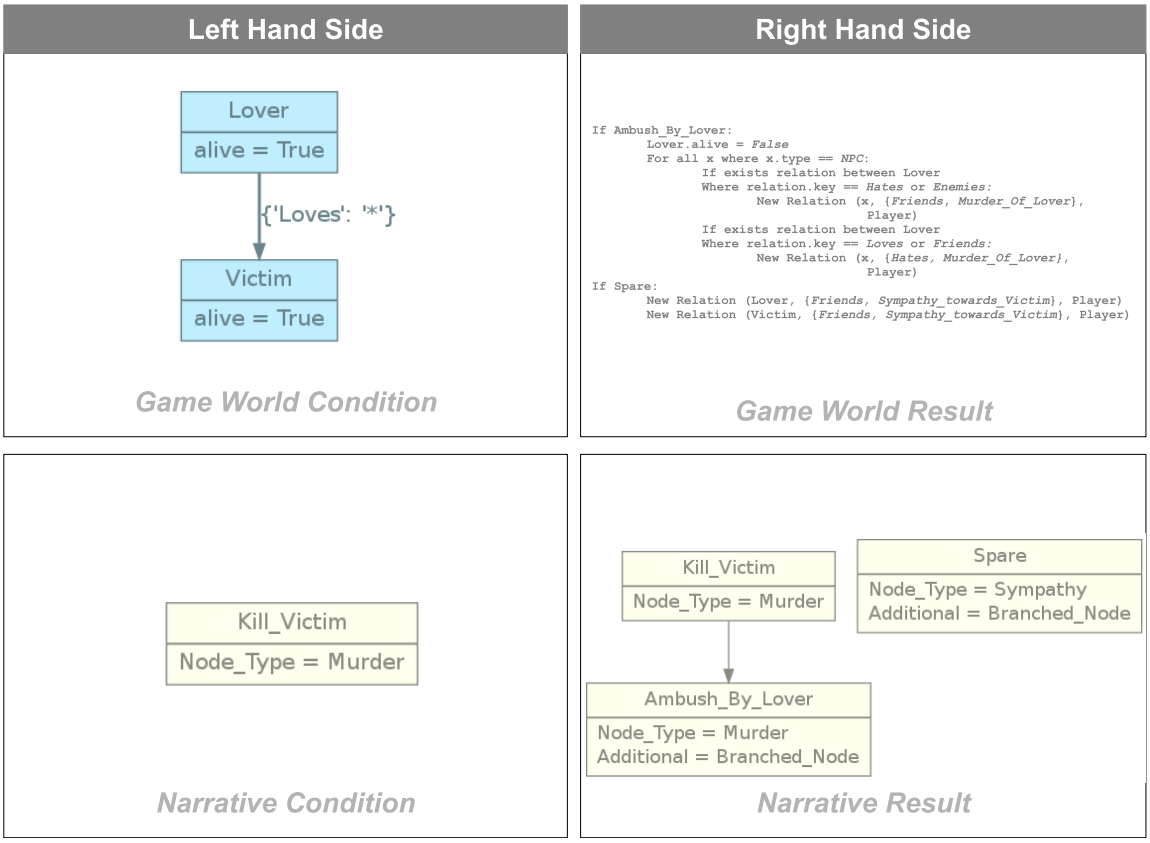

Figure 6–11: Ambush Rewrite Rule

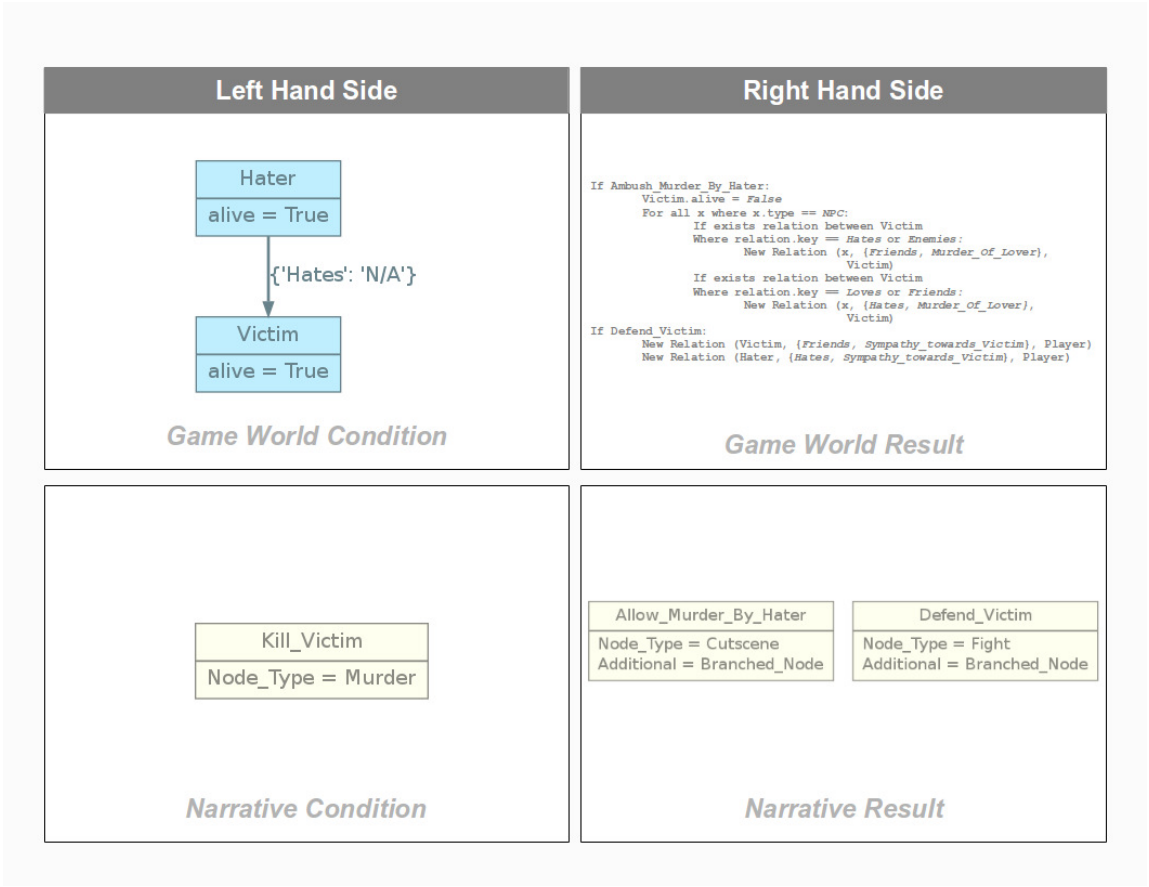

Figure 6–12: Ambush by Hater Rewrite Rule

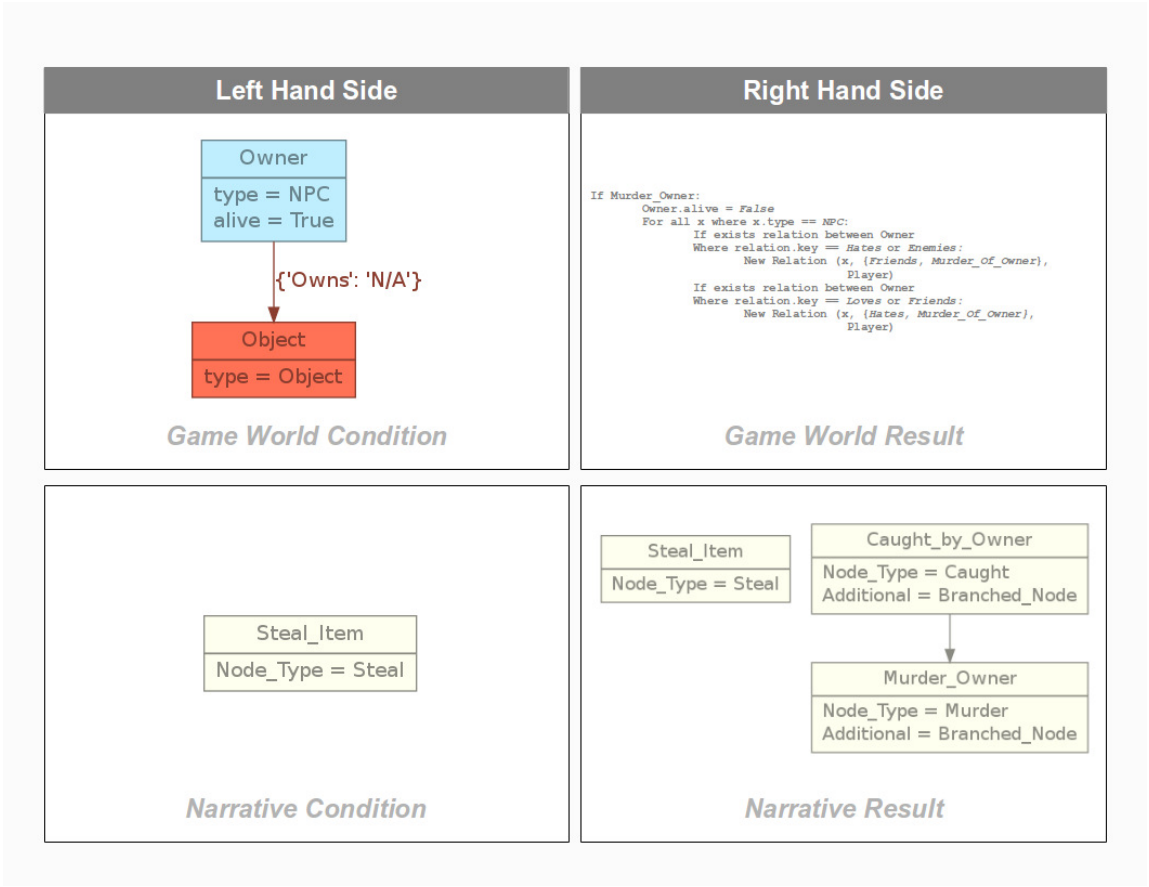

Figure 6–13: Caught Rewrite Rule

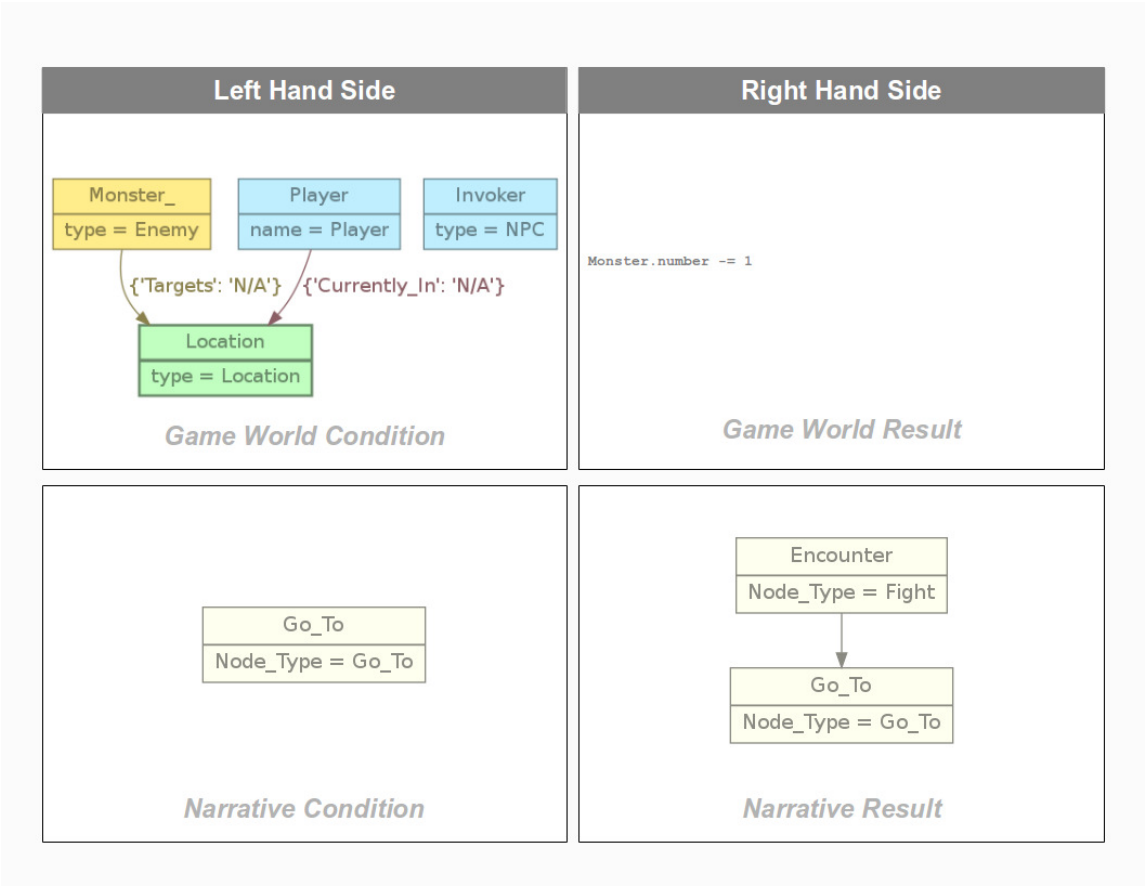

Figure 6–14: Encounter Rewrite Rule

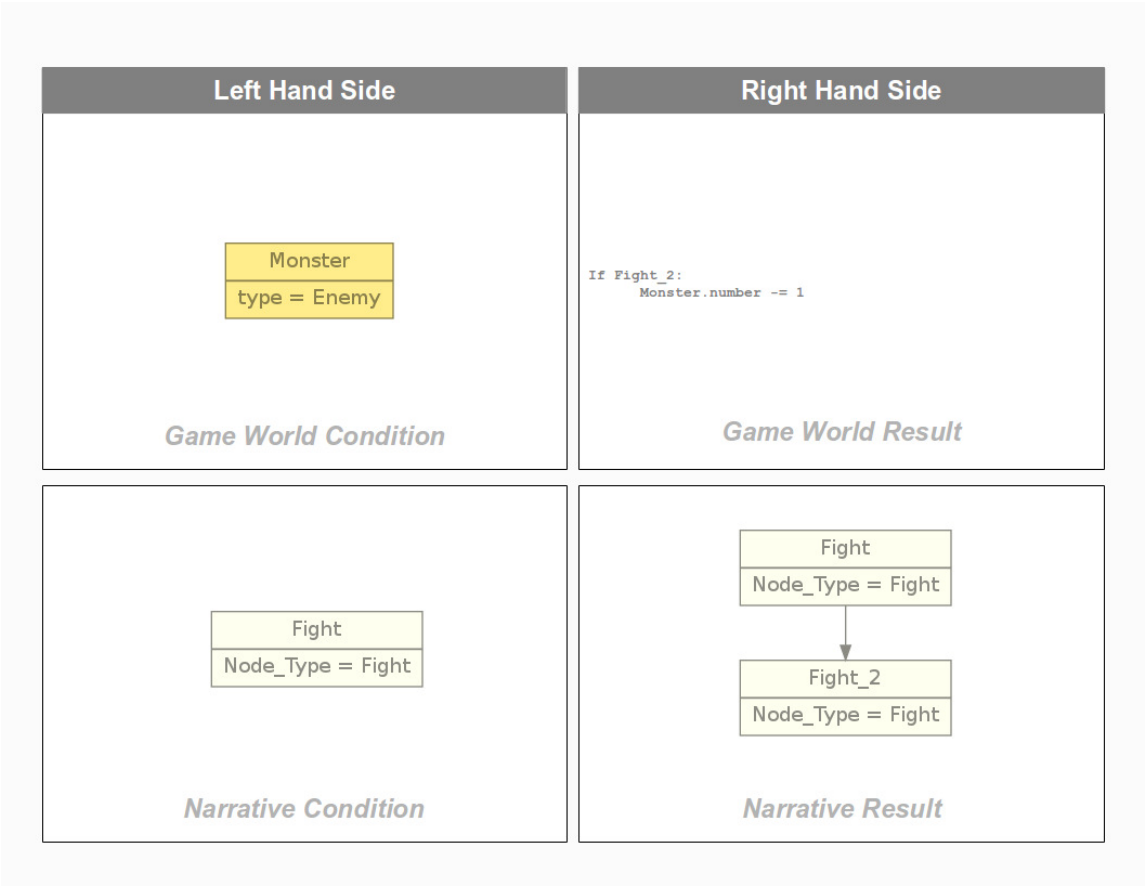

Figure 6–15: More Fight Rewrite Rule

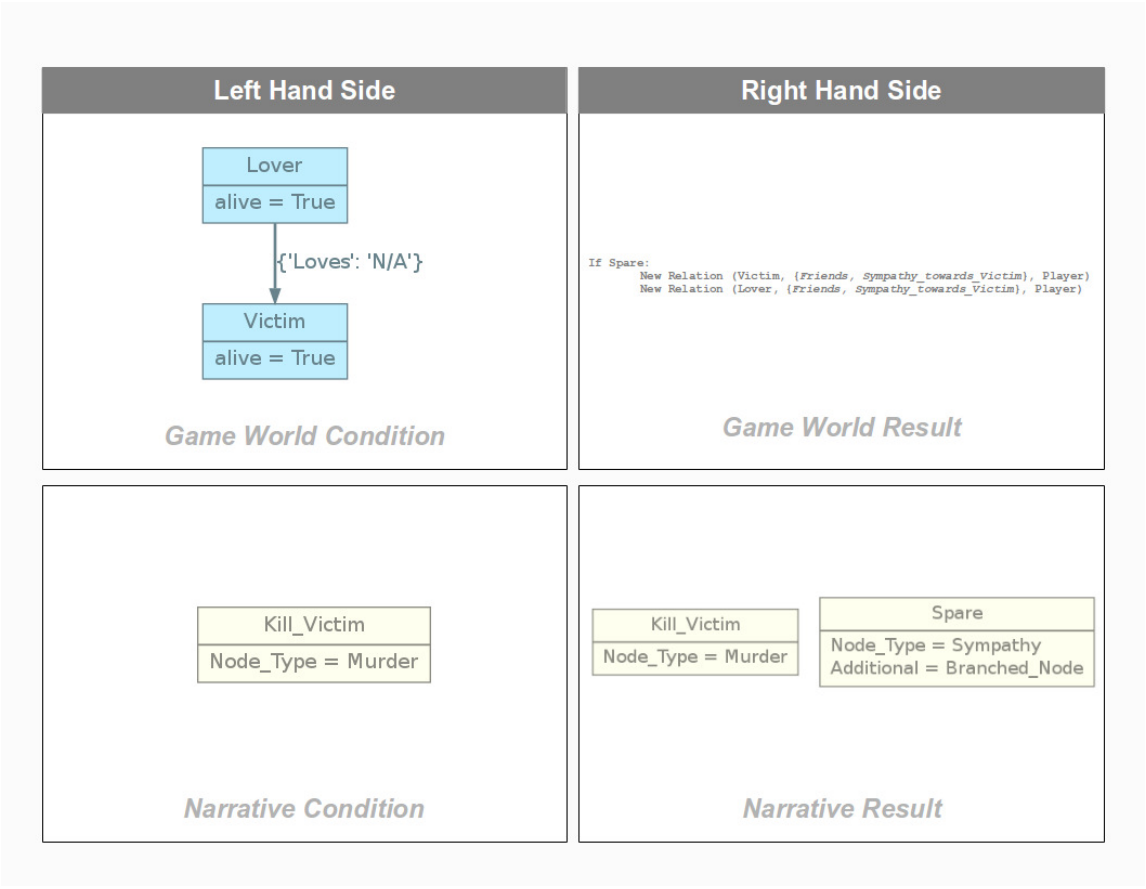

Figure 6–16: Spare Rewrite Rule

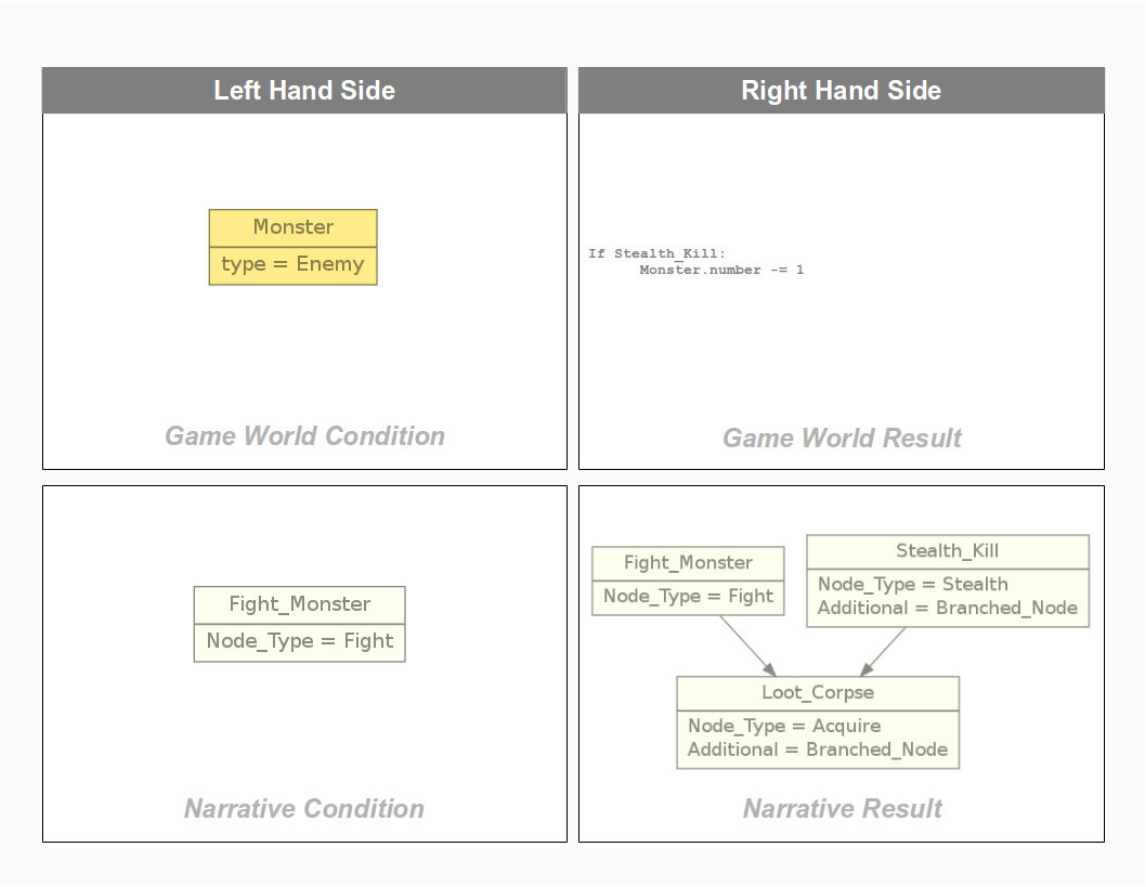

Figure 6–17: Stealth Kill Rewrite Rule

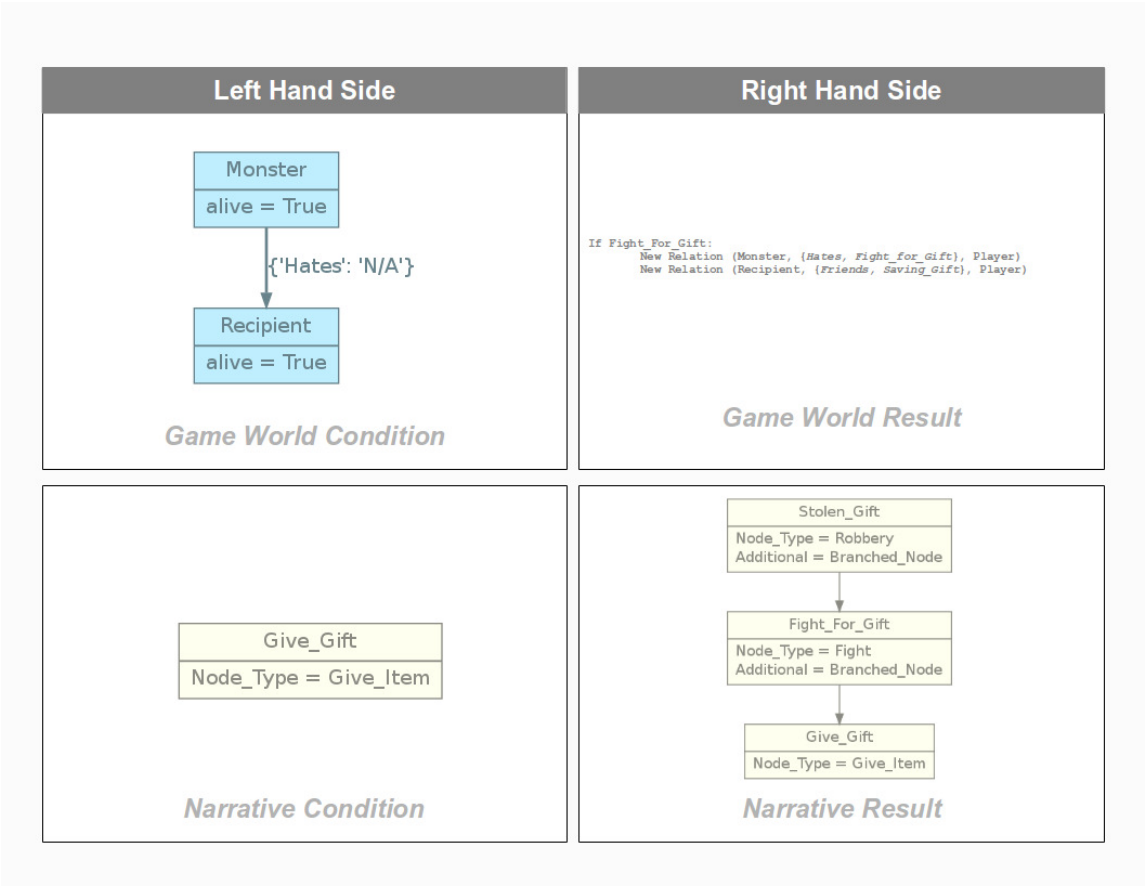

Figure 6–18: Stolen Gift Rewrite Rule

Appendix C - Game World Graph

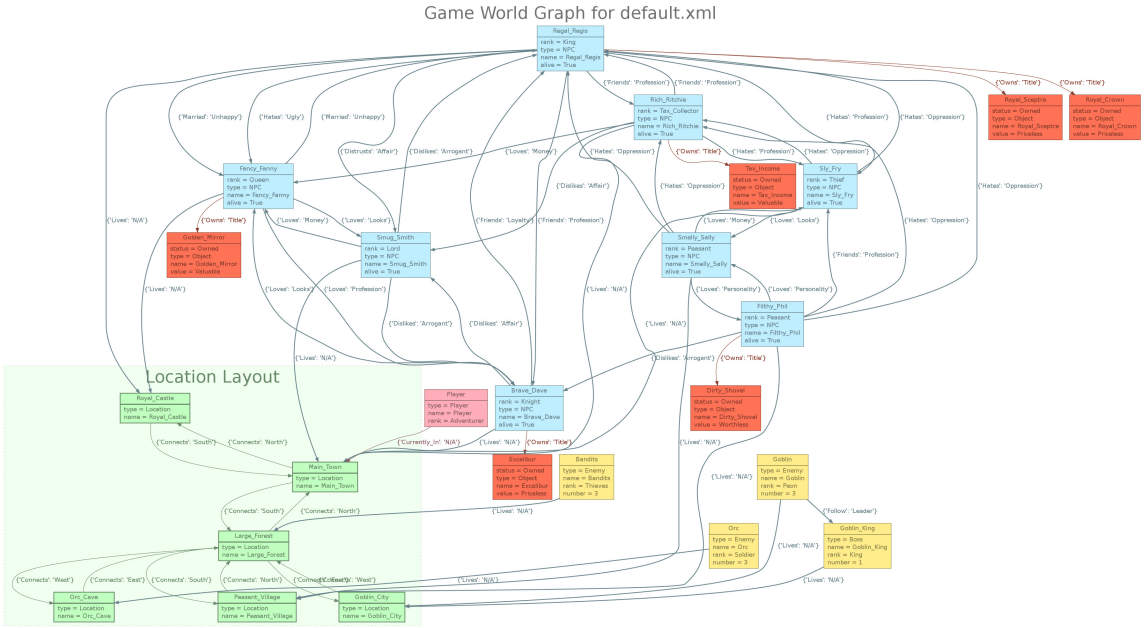

Figure 6–19: The Basic Game World Graph used for experimentation

References

[1] Heather Barber and Daniel Kudenko. Generation of dilemma-based interactive narratives with a changeable story goal. In *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment*, INTETAIN '08, pages 6:1–6:10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[2] John B. Black and Robert Wilensky. An evaluation of story grammars. *Cognitive Science*, 3(3):213–229, 1979.

[3] Hsueh-Min Chang and Von-Wun Soo. Planning-based narrative generation in simulated game universes. *IEEE Trans. Comput. Intellig. and AI in Games*, 1(3):200–213, 2009.

[4] B. N. Colby. A partial grammar of eskimo folktales. *American Anthropologist*, 75(3):645–662, 1973.

[5] Joris Dormans. Level design as model transformation: a strategy for automated content generation. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, PCGames '11, pages 2:1–2:8, New York, NY, USA, 2011. ACM.

[6] Alan Garnham. What's wrong with story grammars. *Cognition*, 15(13):145 – 154, 1983.

[7] Julian Horsey. The elder scrolls v: Skyrim radiant quest system, will customise your game as you play. `http://www.geeky-gadgets.com/the-elder-scrolls-v-skyrim-radiant-quest-system-will-cusomtise-your-game-as-you-play-10-11-2011/`, November 2011.

[8] George P. Lakoff. Structural complexity in fairy tales. *The Study of Man*, 1:128–150, 1972.

[9] Heinrich Lenhardt. Bethesdas nesmith reflects on the difficult birth of skyrims radiant story system. `http://venturebeat.com/2012/01/27/bethesdas-nesmith-reflects-on-the-difficult-birth-of-skyrims-radiant-story-system/`, January 2012.

[10] Claude Levi-Strauss. Structure and form: Reflections on a work by vladimir propp. In *Structural Anthropology, Volume 2*. University of Chicago Press, 1983.

[11] Michael Mateas and Andrew Stern. Procedural authorship: A case-study of the interactive drama facade. In *In Digital Arts and Culture (DAC*, 2005.

[12] Josh McCoy, Mike Treanor, Ben Samuel, Brandon Tearse, Michael Mateas, and Noah Wardrip-Fruin. Authoring game-based interactive narrative using social games and comme il faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate (ELO 2010)*, Providence, Rhode Island, USA, June 2010.

[13] James R. Meehan. Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 1*, IJCAI'77, pages 91–98, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

[14] Curtis Onuczko, Duane Szafron, and Jonathan Schaeffer. Stop getting side-tracked by side-quests. In *AI Game Programming Wisdom 4*. Course Technology, 2008.

[15] Curtis Onuczko, Duane Szafron, Jonathan Schaeffer, Maria Cutumisu, Jeff Siegel, Kevin Waugh, and Allan Schumacher. A demonstration of squege: A crpg sub-quest generator. In *AIIDE*, pages 110–111, 2007.

[16] The Unofficial Elder Scrolls Pages. Skyrim: Main quest. `http://www.uesp.net/wiki/Skyrim:Main_Quest`, 2013.

[17] Federico Peinado and Pablo Gervs. Evaluation of automatic generation of basic stories. *New Generation Computing*, 24:289–302, 2006. 10.1007/BF03037336.

[18] Rafael Pérez ý Pérez and Mike Sharples. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):119–139, 2001.

[19] Julie Porteous, Marc Cavazza, and Fred Charles. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Trans. Intell. Syst. Technol.*, 1(2):10:1–10:21, December 2010.

[20] Vladimir Propp. *Morphology of the Folktale*. University of Texas Press, 1968.

[21] Mark O. Riedl and R. Michael Young. Narrative planning: balancing plot and character. *J. Artif. Int. Res.*, 39(1):217–268, September 2010.

[22] RPS. Game logic vs choice & consequence. `http://www.rockpapershotgun.com/2011/11/21/game-logic-vs-choice-consequence/`, November 2011.

[23] D. E. Rumelhart. *Notes on a schema for stories*, pages 211–236. Academic Press, Inc, 1975.

[24] Ivo Swartjes and Mariët Theune. A fabula model for emergent narrative. In *Proceedings of the Third international conference on Technologies for Interactive Digital Storytelling and Entertainment*, TIDSE'06, pages 49–60, Berlin, Heidelberg, 2006. Springer-Verlag.

[25] M. Theune, S. Rensen, R. op den Akker, D. Heylen, and A. Nijholt. Emotional characters for automatic plot creation. In *Technologies for Interactive Digital Storytelling and Entertainment. Second Intern. Conf., TIDSE 2004*, pages 95–100. Springer-Verlag, 2004.

[26] J.R.R. Tolkein. *The Lord of the Rings*. Harper Collins Publisher, 2001.

[27] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, January 1976.

[28] Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. Four quantitative metrics describing narrative conflict. In *Proceedings of the 5th international conference on Interactive Storytelling*, ICIDS'12, pages 18–29, Berlin, Heidelberg, 2012. Springer-Verlag.

[29] The Witcher Wiki. The witcher primary quests. `http://witcher.wikia.com/wiki/The_Witcher_primary_quests`, 2013.

[30] Alexander Zook, Stephen Lee-Urban, Mark O. Riedl, Heather K. Holden, Robert A. Sottilare, and Keith W. Brawner. Automated scenario generation:

toward tailored and optimized military training in virtual environments. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, pages 164–171, New York, NY, USA, 2012. ACM.