UNIVERSITY OF THE PHILIPPINES, LOS BAÑOS

Master of Science in Computer Science

DJYRON F. SARROZA

ISLA: AN ALGORITHMIC APPROACH TO ASSISTED NARRATIVE PLANNING AND ASSEMBLY

JAIME M. SAMANIEGO
Adviser

Date: **APRIL 29, 2021**

This thesis can be made available to the general public	YES
This thesis can be accessed only after consultation with the author and thesis adviser	–
This thesis can be accessed only by those bound by confidentiality agreement	–

[illegible]

DJYRON F. SARROZA

[illegible]

JAIME M. SAMANIEGO

ISLA: AN ALGORITHMIC APPROACH TO ASSISTED NARRATIVE PLANNING AND ASSEMBLY

DJYRON F. SARROZA

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL UNIVERSITY OF THE PHILIPPINES LOS BAÑOS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

(Computer Science)

APRIL, 2021

The thesis attached hereto, entitled **"ISLA: AN ALGORITHMIC APPROACH TO ASSISTED NARRATIVE PLANNING AND ASSEMBLY"** prepared and submitted by **DJYRON F. SARROZA**, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE (COMPUTER SCIENCE)** is hereby accepted.

[illegible]

JAIME M. SAMANIEGO
Chair, Guidance Committee

29 April 2021
Date Signed

JADERICK P. PABICO
Member, Guidance Committee

Date Signed

Concepcion Khan

CONCEPCION L. KHAN
Member, Guidance Committee

29 April 2021
Date Signed

Accepted in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE (COMPUTER SCIENCE)

JADERICK P. PABICO
Director, Institute of Computer Science

Date Signed

JOSE V. CAMACHO, JR.
Dean, Graduate School

Date Signed

Djyron Sarroza is currently completing his Master of Science in Computer Science in the University of the Philippines Los Baños, Philippines. His research interests include artificial intelligence and procedural content generation. This paper is primarily motivated by his interest in high fantasy novels with hard science fiction approaches – and the hopes of being a published novelist one day.

Acknowledgements

I would like to thank **Jimmy Samaniego**, for his patient and inspiring tutelage. Great appreciation to **Richard Perez** and his family for the hospitality and support during the final stretches. Gratitude to **Juan Karlo Licudine**, **Beatrice M.V. Lapa**, and **Antonio Gabriel Abad IV** for their insights and contributions in evaluating the potential of ISLA. Many thanks to **Baby Joan Aure** and **Anjii Noarin** and their respective English classes for participating in my data gathering experiments. Endless gratitude to my dear family and friends, who continue to put up with my eccentricities.

To **Kristinne Anne Cas**, who made the journey of life worth it.

Finally to my parents, **Hanah and Joey Sarroza**, for giving me life and love, and the chance to redeem myself.

Contents

List of Figures

List of Appendices

Abbreviations

CPOCL	**C**onflict **P**artial **O**rder **C**ausal **L**ink
IPOCL	**I**ntent-based **P**artial **O**rder **C**ausal **L**ink
ISLA	**I**ntelligent **S**tory **L**ayout **A**ssistant
PDDL	**P**lanning **D**omain **D**efinition **L**anguage
POCL	**P**artial **O**rder **C**ausal **L**ink
STRIPS	**St**andford **R**esearch **I**nstitute **P**roblem **S**olver

For Kristinne, Light of my life

ABSTRACT

DJYRON F. SARROZA University of the Philippines, Los Baños Jan 2021.

ISLA: An Algorithmic Approach to Assisted Narrative Planning and Assembly
Major Professor: JAIME SAMANIEGO

Intelligent Story Layout Assistant (ISLA) is a forward-chaining narrative planner based on Stephen Ware's GLAIVE. It constructs story layouts that achieve the author's goals while making sure that *most* steps in the plan have clear motivations. These layouts, or solution plans, are based on a handcrafted knowledge-base of story universe elements. The current output is capable of displaying the underlying motivations of each action, and state transition information. The output is also presented in two ways: 1) a relatively human-readable series of paragraphs and 2) a detailed state transition sequence. The objective qualities of the narrative outlines being generated are measured, quantified, and analyzed. Some limited machine learning techniques are also employed in an effort to refine the elements of the knowledge-base. As a whole, ISLA has the underlying data structures needed to potentially further assist the author in fleshing out the produced story layout.

CHAPTER 1

INTRODUCTION

There has always been abundant interest in automatic story generation in the field of artificial intelligence. The ability to generate stories on demand has great possibilities in other fields, such as entertainment and education. This thesis is motivated by the need of novice writers to have the tools to assist them in creating complex and convincing story-lines. Managing multiple complex characters, all capable of an arbitrary number of actions, and all interacting in a non-static story telling universe can be a daunting task. Human error is not just a possibility in this scenario, it is an inevitability. Having a software tool which utilizes planning paradigms and working under additional creative constraints[1] aims to alleviate such workload from a human writer.

It is necessary to note early on that this rich narrative planner does not intend to replace human writers, but is designed as a tool that will help authors focus on other creative aspects of creating stories. For example, in lieu of having to keep track of dozens of characters, their actions, and their respective effects – an author using a narrative planner can focus on "fleshing out" the scaffold provided by the planner. The rich narrative planner is merely a guide. Dialogue, personality quirks, vibrant world portrayals, these are only some story elements that require an undeniably human touch.

[1]Not merely "finding a goal state", as is the case with classical planning algorithms

This paper presents ISLA, Intelligent Story Layout Assistance, a variant of forward-chaining planner which aims to produce a cohesive narrative plan. ISLA models intentionality, conflict, and representation of alternate world-states very similar to GLAIVE [WY14]. This paper details ISLA's algorithmic approach, data structures used, knowledge-base encoding, performance measurements, and even some practical applications.

1.1. Terminology

In this section, the terminology used throughout the paper is defined. This paper used the proposed definition for a narrative, and the relevant components as described in Kybartas' and Bidarra' work [KB17], with some modifications, among other relevant terms.

1. *ISLA* stands for Intelligent Story Layout Assistant; the name of the narrative planner system described in this paper.
2. *Narrative* is defined as a *story*, which is the main content, and a *discourse*, which describes the particular telling of a story.
3. *Story* is the main content of a narrative, which contains the sequence of events happening within the current story-telling universe.
4. *Plot* is a set of events with an overall structure. This structure is necessary to describe both temporal ordering and the causal relations between events.
5. *Universe* or the story-telling universe (STU) or *domain*, includes characters/actors, setting/environment, props, or anything that is either physically or abstractly present in a narrative – called objects or *existents*. In order to represent the non-static nature of existents, the STU must encode this information as a *state/world-state*.
6. *Discourse* is the particular telling of a story. This may include the style of the STU, the ordering and duration of the events in the plot, etc.
7. *McGuffin* is a plot device in the form of some goal, desired object, or other motivator that the protagonist pursues, often with little or no narrative explanation.
8. *Existents* are defined objects that exists in a story-telling universe. These may represent physical objects or abstract ideas.
9. *Planners* are implementations of algorithms that solve the planning problem: given a domain, an initial state *I*, and a goal situation *G* consisting of a set of propositions, find a sound sequence of actions that maps the initial state into a state where *G* is true. [RY09]
10. *Intentions* are character goals that are attached to the story actors. This represents what the individuals in the narrative intend to achieve.

11. *Fabula* refers to the chronological sequence of events in a narrative; the raw materials of the story; e.g. the story.
12. *Sjuzet* is the representation of the events in a fabula (through narration, metaphor, camera angles, the re-ordering of the temporal sequence, and so on); e.g. the discourse.

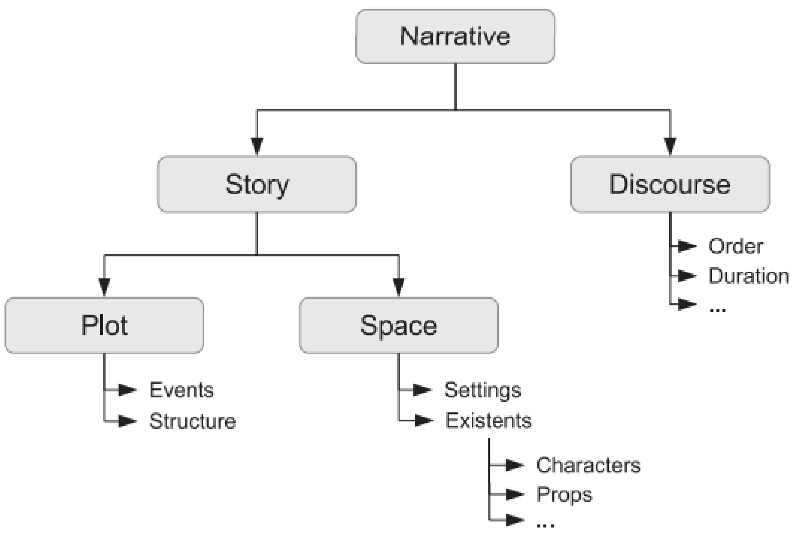


FIGURE 1.1: Proposed Structure of Narrative as described by Ben Kybartas and Rafael Bidarra

1.1.1 Objectives

1.1.1.1 General Objectives

To create a general purpose narrative planner which uses sparse initial input from the user in order to generate rich, detailed, and convincing narrative plans or outlines. The planner's output should provide enough information to the user such that the user deems the plan/outline as either complete or that it can be a solid basis for further enrichment. In other words, we aim for ISLA's output to have structural and causal sense.

1.1.1.2 Specific Objectives

- Encode a knowledge-base of narrative resources that are accessible to the narrative planning system.

- Compare different algorithms and datasets to measure performance and output quality
- Create a narrative planning system that produces plans which are sensible, convincing, and generally interesting to end users.

1.1.2 Scope and Limitations

The primary scope of this thesis encompasses the creation of the narrative planner, encoding of the knowledge-base, and analysis of the principles surrounding major design choices. Natural language processing, and machine learning are among ISLA's principal limitations. Some rudimentary attempts were made to implement these features for ISLA, but they are not the focus of this endeavor. The output outlines that ISLA produces are story skeletons which are open to numerous interpretations and mere bases for a more complete literary output.

CHAPTER 2

REVIEW OF LITERATURE

The following sections discuss the numerous inspirations for the inner workings of ISLA. With both literary theory and narrative planning having an extensive tradition in the field of artificial intelligence, there is no shortage in raw materials. Computational narratology, as described by Mani [Man13], is a relatively new field of study directly influenced by multiple aspects of technology, linguistics, and even mathematics, among others. This presents an opportunity to discover relationships and connections that will contribute to the construction and improvement of ISLA.

2.1. Transcription

In order to make the computational generation of an expression or artistic endeavor such as a *story* – which is uniquely human – possible, the space must first be transcribed into a form that can be understood by a machine and that allows computational operations and manipulation to be performed on it (i.e. represent it with what is basically recognized as a data structure.)

Second, a schema must be established to support efficient manipulation so that the story elements / existents can be instanced into a new literary piece. Ample work has been done on this matter in both literary theory and artificial intelligence that when combined form a solid foundation for a launchpad in building an artificial intelligence capable of computational generation of a narrative.

Parallels between the causal and temporal data structures of some planning algorithms and certain representation of stories used by narratologists were identified by Young [You99]. The concepts discussed by Young will resonate through many of his succeeding works, most notable and relevant for ISLA is the notion of searching through a search space of all possible plans. This is opposed to the incremental construction of a single plan.

2.1.1 Computing Narrative

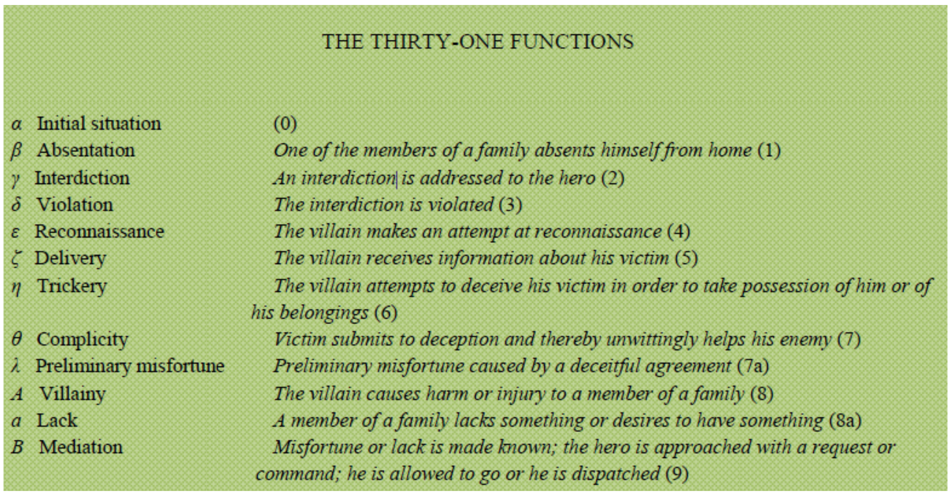
The paper by Miller explores how subjective cognitive frameworks such as narratives can and are represented as procedures and data in creative contexts [MP20]. Narratives are treated as decomposable data, where readers act as evaluators who operationalize aspects of stories, whereby metrics such as "surprisingness" can be applied to the results.

The aforementioned study further delves into concepts of narrative empiricism, narrative inference, and narrative emergence, which although impressively succinct and curated, offered limited relevance to what will emerge a very specific approach for ISLA. However, insights on how the complex and difficult task of transducing narrative features into software architecture remain valuable assets to further the work on this field.

2.2. Conceptualization

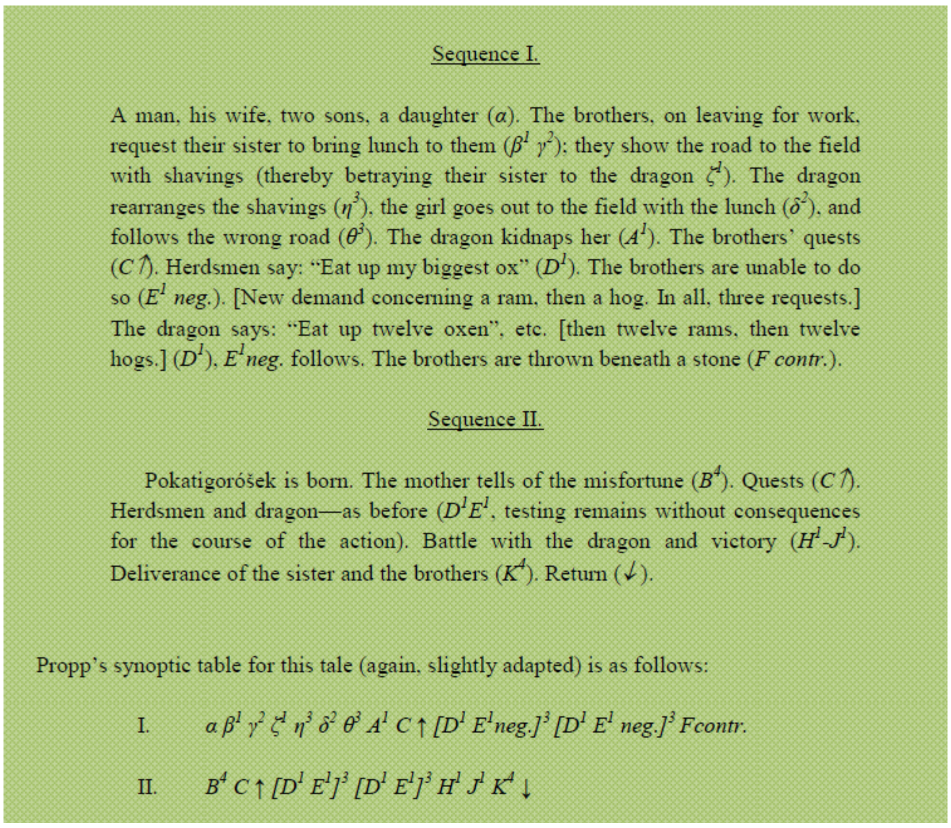
2.2.1 Literary Theory

As evidenced by works such as Propp's formalization of Russian folk tales, there are certain redundancies and patterns present in stories. Although each and every story are not exactly the same, many patterns such as the presence of heroes and villains are quite universal. Many of the efforts in developing narratology as a discipline are inspired by Vladimir Propp's *Morphology of the Folktale*, and from this seminal work sprung many publications. Too many, in fact, carry inaccurate vulgarizations of Propp's system [Agu11]. Aguirre aimed to place a


THE THIRTY-ONE FUNCTIONS

α Initial situation	(0)
β Absentation	*One of the members of a family absents himself from home* (1)
γ Interdiction	*An interdiction is addressed to the hero* (2)
δ Violation	*The interdiction is violated* (3)
ε Reconnaissance	*The villain makes an attempt at reconnaissance* (4)
ζ Delivery	*The villain receives information about his victim* (5)
η Trickery	*The villain attempts to deceive his victim in order to take possession of him or of his belongings* (6)
θ Complicity	*Victim submits to deception and thereby unwittingly helps his enemy* (7)
λ Preliminary misfortune	*Preliminary misfortune caused by a deceitful agreement* (7a)
A Villainy	*The villain causes harm or injury to a member of a family* (8)
a Lack	*A member of a family lacks something or desires to have something* (8a)
B Mediation	*Misfortune or lack is made known; the hero is approached with a request or command; he is allowed to go or he is dispatched* (9)

FIGURE 2.1: Partial list of Propp's 31 Functions


Sequence I.

A man, his wife, two sons, a daughter (α). The brothers, on leaving for work, request their sister to bring lunch to them (β^1 γ^2); they show the road to the field with shavings (thereby betraying their sister to the dragon ζ^1). The dragon rearranges the shavings (η^3), the girl goes out to the field with the lunch (δ^2), and follows the wrong road (θ^3). The dragon kidnaps her (A^1). The brothers' quests ($C\uparrow$). Herdsmen say: "Eat up my biggest ox" (D^1). The brothers are unable to do so (E^1 *neg.*). [New demand concerning a ram, then a hog. In all, three requests.] The dragon says: "Eat up twelve oxen", etc. [then twelve rams, then twelve hogs.] (D^1), E^1*neg.* follows. The brothers are thrown beneath a stone (*F contr.*).

Sequence II.

Pokatigorόšek is born. The mother tells of the misfortune (B^4). Quests ($C\uparrow$). Herdsmen and dragon—as before (D^1E^1, testing remains without consequences for the course of the action). Battle with the dragon and victory (H^1-J^1). Deliverance of the sister and the brothers (K^4). Return ($\downarrow$).

Propp's synoptic table for this tale (again, slightly adapted) is as follows:

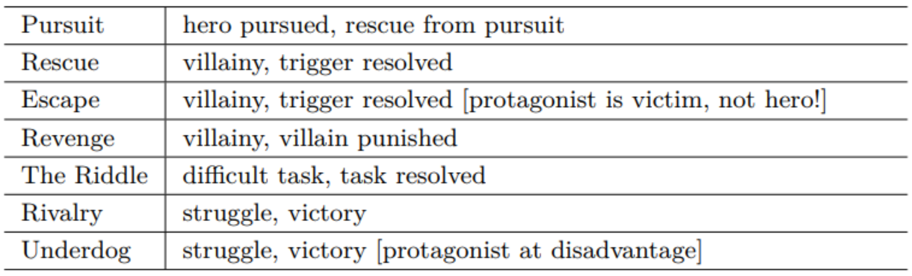
I. $\alpha\, \beta^1\, \gamma^2\, \zeta^1\, \eta^3\, \delta^2\, \theta^3\, A^1\, C\uparrow [D^1\, E^1 neg.]^3\, [D^1\, E^1\, neg.]^3\, Fcontr.$

II. $B^4\, C\uparrow [D^1\, E^1]^3\, [D^1\, E^1]^3\, H^1\, J^1\, K^4 \downarrow$

FIGURE 2.2: Example breakdown of short stories into sequences of Propp's functions

tool at the disposal of researchers in the fields of folklore and popular culture, hence is a logical place to start to get introduced to Propp's way of breaking down stories into formulaic forms.

Alexander Afanasyev's classic collection Russian Folktales (1855-64) was Propp's basis for his study. From the over six hundred folktales present in the collection, Propp derived his



Pursuit	hero pursued, rescue from pursuit
Rescue	villainy, trigger resolved
Escape	villainy, trigger resolved [protagonist is victim, not hero!]
Revenge	villainy, villain punished
The Riddle	difficult task, task resolved
Rivalry	struggle, victory
Underdog	struggle, victory [protagonist at disadvantage]

FIGURE 2.3: Paraphrases of the Elementary Plots of Tobias' in terms of Proppian character functions

structural model based on the following criteria:

1. All fairytales are constructed on the basis of one single string of actions or events called "functions".
2. Function is significant action or event defined according to its place in the plot.
3. Function, and not theme, motif, character, plot or motivation, is the fundamental unit of analysis.
4. Functions are independent of how and by whom they are fulfilled; from the standpoint of structural analysis, not doers, their method, their motivations or their psychology but the deed itself alone matters.
5. The number of functions available to fairytale-tellers is thirty-one.
6. With (codifiable) exceptions, functions always follow a strict order.
7. Tales are organized into sequences; each sequence is composed of a selection of functions in the appropriate temporal order, and constitutes a narrative episode.
8. Each function is susceptible of realization by different means ("forms of function"): Propp offers lists of the "function forms" that appear in his corpus (but warns that others are possible).
9. Only seven characters are available to fairytale-tellers: hero, false hero, villain, donor, helper, dispatcher, princess (sought-for person) and/or her father.
10. All fairytales are composed of the same functions, though not every function appears in every tale.
11. All fairytales share the same fundamental structure.

Tobias' *Master Plots* [Tob12] is another basis for story structure generalization and can be converted into Propp's formulaic way (fig.2.1 - 2.2) of describing story sequences. However, ISLA benefited more from Propp's conceptualization of narrative structures. *Master Plots* seems more of a guideline for writers, and is less useful in terms of distilling raw stories into rigorous structures such as schemata. At the very least, Propp verified the hypothesis that stories have recurring patterns that can be manipulated mathematically or algorithmically.

2.2.2 Narratology

Mieke Bal's introduction to narratology [Bal99] provides key insights on the field of study. Specifically with the definition of *fabula* – product of imagination – and *story* – product of an ordering. This insight gives enough distinction to the internal components of a narrative in order to decompose them into elements that can be manipulated by mathematical or algorithmic processes. The text also mentions that *words* are products of the use of a medium, much like programming languages are products of the use technology – both are representations of more fundamental concepts of logic and meaning.

2.2.3 Narratives and Mathematics

One product of the exploration of transcribing stories into data structures is the intuition that the connection between narratives and mathematics runs far deeper. Investigation into theorem provers yielded that narratology does indeed play crucial roles in mathematics – ranging from approaches such as algebraic semiotics [Gog99], which is being developed as a way to improve user interface functionality and quality of mathematical proof assistance tools, to studies describing the narrative structure of mathematical text and lectures [Die13] [KMRW07] [WWFC15]. These studies are can be described as "narratology for math", where mathematical concepts and elements are mapped to narratological concepts and elements in order to leverage the advantages of story telling that may be obvious to narratology, but not as much to mathematics. A great example of this are the conveyance of mathematical proofs. Proofs can be described as a linear progression from an initial state, to a final state, with annotations to other resources such as lemmas, corollaries, or other theorems. However, finding non-trivial proof requires navigating through misconceptions and errors, and these conflicts can arguably positively contribute to the understandability of the proof – much like the journey of a hero, where important conflicts within that journey is explored in order to properly understand the hero's motivations, and add value to the odyssey's resolution.

Narrative analysis using proof assistant tools [BCFC11] describes the use of narrative's formal properties in order to leverage upon theorem provers as a way to verify plot structures. This "math for narratology" approach has given credence to the intuition that stories and mathematics are more deeply connected.

2.2.4 Computational Approach - Narrative Generation Schemata

Previous work by Gervás, León, and Méndez on plot schemata [GLM15] has proved to be a valuable resource for encoding existing plot structures into ISLA's knowledge-base. This study recognized that computationally generated narrative artifacts have a tendency to rely on schema and templates, which can be instantiated into more complex structures. ISLA's vocabulary on describing narrative structures is also solidified by applying concepts detailed in the aforementioned paper. A detailed investigation on the properties of narrative schemata, Simonson's paper [Sim18] on the matter is an excellent resource. The dissertation may prove invaluable in the future specifically on the planned improvements on domain manipulation, problem extrapolation, and machine learning.

Numerous existing attempts at providing an elementary set of patterns for plot are the focus for Gervas' paper. To a large extent, oversimplification of the plot to a very abstract outline is a factor why none of these attempts were accepted as generally valid. Focus on specific aspects of a story, may cause other aspects to suffer when distilling full stories into schemata. However, Gervas was successful in analyzing these schemata as a whole, and a basic abstract vocabulary to describe different plots was defined. This may then provide the basic scaffold of a desired story, regardless of other enrichment which may later be added to it.

Taking advantage of such generalizations, ISLA is intended to utilize the same basic idea and deliver a narrative scaffold, then augmented further with supporting concepts such as character tropes, interactions, goals, conflict, and character evolution. Applying these schemas also has another advantage – they provide additional junctions for ISLA to be improved upon in the future. The current state of ISLA's capabilities does not yet handle explicit representation of individual character personalities. Because of the robust definitions and quantization of story elements by Gervás, León, and Méndez, it was deemed possible that in future works that chapter pattern selection takes into account the type of personalities that current actors have[1].

For example, it may be argued that not all personality types can particularly fit the hero role in all narrative plots (fig. 2.3). Like a hero that has a lawful-good personality type will not make much sense in a Revenge plot. Properly matching plots with personalities may produce more desirable traditional output; conversely, intentionally diverging from this mechanic may produce results that do not conform to traditional patterns. Either way, this may imbue ISLA more direct control to her story generation processes.

2.2.5 Computational Approach - Encoding the Domain

As a way to have a unified syntax for representing planning problems for the annual International Planning Competitions (IPL), the Planning Domain Definition Language (PDDL) was conceived. The main inspiration for the PDDL language was the Stanford Research Institute (SRI) Problem Solver (STRIPS) language from the early 1970s, developed at SRI International.

PDDL was a new language specifically crafted to encode two things: the planning domain, and the planning problem. In its basic form, the following are the components of a PDDL

[1]Actor and existent definition is planned to be indirectly affected by the user's desired parameters, or is directly explicitly chosen.

planning task:

Domain
    Objects: Things in the world that interest us.
    Actions/Operators: Ways of changing the state of the world.
    Predicates: Properties of objects that we are interested in; can be true or false. (2.1)

Problem
    Initial state: The state of the world that we start in.
    Goal specification: Things that we want to be true.

2.2.6 Computational Approach - Planners

Forward-chaining planning is perhaps one of the more intuitive planning approaches: beginning with the initial state, traversing the state-space by applying actions to the encountered states until a state is reached that satisfies the target condition. The strategy used to enter each state can be calculated by following the course of actions performed to travel from the initial state to the state in question: in sequence, these actions form the plan that leads to that state.

Theoretically, forward-chaining state-space traversal involves a directed representation of the graph: nodes represent states; the edges represent the applications of action between then. Nonetheless, such a structure would cause cycles to occur (corresponding to redundant parts of the plans) if a path were found back to itself from a given state. Practical implementation often demand that the search is constrained to ensure that a tree representation is sufficient: this is done by memoising the states encountered as the search progresses, and not allowing changes to those states encountered previously. Exhaustive and unguided forward-chaining search on all but the smallest of planning issues is quite infeasible. Two methods are widely used in practice to increase search efficiency: using a heuristic to direct search; and pruning the search tree.

There are numerous heuristics discussed in Cole's paper [Col07] such as memoization, loop-checking, and pruning – all of which were incorporated into ISLA. A version of the forwards reachability analysis from the initial state to reach the goal facts was later included (see sec. 3.2.4.5). This heuristic calculates which candidate action can lead to more predicates being true in the future. The assumption is: the more goal predicates which are TRUE in any given state, the closer it is to a goal state.

2.2.7 Computational Approach - Narrative Planners

There are many aspects which determine if the audience considers a story as good. Many of these aspects, such as the degree to which the audience empathizes with the protagonist, are subjective in nature. Other aspects over a wide variety of genres seem to be more universal.

The comprehension of stories requires the audience to perceive the causal connection of story events and to infer character intentionality. Accordingly, the two narrative qualities that we concentrate on in this narrative generation research are rational causal progression and believability of characters.

2.2.7.1 POCL - Partial Order Causal Link

Partial Order Causal Link (POCL) plans have proved to be powerful data structures for representing stories as they form the events of a story directly along with the casual and temporal connections between them. Such program data structures can also act as metaphors for mental models formed by people who read or watch a narrative.

POCL preparation can be interpreted as a refinement search, in which a partial plan is slowly repaired or improved until either the plan is complete (and executable) or inconsistent (and unrepairable). A partial plan is annotated with flaws in this process; each flaw indicates a specific problem with the partial plan which needs to be repaired.

2.2.7.2 IPOCL - Intent-based Partial Order Causal Link

Reidl and Young [RY09] defines a method of narrative generation that models the creation process of fictional narratives as a search-based planning process. The resulting item – the plan –

is a summary of the series of acts temporarily ordered to be performed by story world characters. This design tells a story when it's implemented or made into natural language. Plans have been found to be good computational representations for narratives, since plans encode core narrative attributes: behavior, temporality, and causality [You99].

Unfortunately, solving the planning problem also does not solve the problem of narrative creation, since (non-narrative) planners do not understand many of the narrative's logical and artistic properties. In particular, planners do not consider believability of characters (or character actions). Reidl and Young defined a novel refinement narrative planner – the Intent-based Partial Order Causal Link (IPOCL) planner – that, in addition to creating causally sound plot progression, provides explanations for intentionality of character by (a) defining possible character objectives that explain their behavior and (b) creating plan frameworks that explain why those characters commit to their objectives.

The IPOCL's primary concern is the generation of a fabula. It is assumed that the sjuzet can be created from this in another process. The two attributes of narrative that IPOCL focuses on the narrative generation are 1.) logical causal progression and 2.) character believability, hence the emphasis on the goal-oriented nature of the actors present in the planning domain.

Based on a class of planning algorithms called Partial Order Causal Link (POCL) planners, the IPOCL creates narratives that, from a reader's point of view, resemble more closely the results of a simulation narrative generation system in terms of character intentionality. In particular, IPOCL is a modification of the current search-based planning algorithms to promote the intentionality of characters regardless of the intent of the author. The goal is to create narratives through a deliberative process so that characters appear to the audience in order to shape expectations and behave as if they were replicated to achieve such intentions. In this way, IPOCL may generate narratives that both have rational causal progression, meaning that they achieve states of outcomes suggested by the author, and have credible characters.

A good heuristic that ranks believability positively impacts the probability that a plan can be found. However, it is still possible for a planner to commit to a shorter, albeit less believable plan, as opposed to a longer/more expensive, but believable plan. Their conclusion is that the fabula generation problem is sufficiently different from a classical planning problem that it [narative planning] can benefit from new definitions and metrics for plan completeness.

Since actor believability is in part due to intentionality, a story is more likely to be considered believable if actors appear to be motivated by individual goals. Reidl et. al. mentions that multiple actors with different (conflicting) goals can be a complication, not to mention the difference in goals between the story characters and the human author. Fortunately, the Conflict Partial Order Causal Link directly addresses this.

2.2.7.3 CPOCL - Conflict Partial Order Causal Link

Conflict Partial Order Causal Link (CPOCL) [WY11] is an IPOCL* extension that specifically illustrates how characters can contradict each other in order to achieve their goals, which are the core of narrative conflict. CPOCL enables the failed (or partially failed) subplans, addressing a key IPOCL limitation that Riedl and Young [RY09] identified.

A good way to understand the difference between these algorithms is by comparing their solutions to the same problem. Consider this scenario: two guys, Abe (A) and Bob (B), are interested in marrying the same girl – Cat (C). While Bob had previously purchased an engagement ring (R), Abe and Cat had agreed to be married, as per the author's intention.

POCL's shortest solution would start with Bob giving Abe the ring. Though this scheme is causal, Bob's supporting his competitor doesn't make sense. This problem would not be solved by IPOCL*, because Abe and Bob can't both meet their goals. CPOCL is able to find a solution that guarantees causal well-being and good motivation for character.

2.2.7.4 GLAIVE

GLAIVE is a state-space heuristic search planning algorithm that explicitly explains intentionality of character and describes alternative worlds to promote thinking regarding phenomena such as conflict. It is based on Hoffman and Nebel's Fast-Forward planner [HN01], but similar to both IPOCL and CPOCL algorithms, GLAIVE also keeps track of the causal history of each proposition.

GLAIVE solves the problem of intentional planning described by Riedl and Young [RY09]: A valid plan is one that achieves the goals of the author but is only composed of steps that are explained in terms of the individual goals of the characters who take them. Ware and Young [WY14] have expanded this issue to include approaches in which some characters' plans fail. It does this considering that there are multiple agents which sometimes collaborate and sometimes clash as they are directed towards the objective of the author by an unseen puppet master.

GLAIVE is a state-space planner, that is, it begins at the initial state of the problem and takes steps that alter the state until it finds a state in which the expectations of the author are valid. The search space can be described as a directional tree. A node in the tree represents a state; an edge $n_1 \rightarrow n_2$ represents applying the effects of step s to the node $n1$ state to create the new node $n2$ with a different state. In practice, a node also represents a plan composed of steps taken from the root to that node on the path. The tree's root is the initial problem state, and contains an empty plan. GLAIVE explicitly tracks how the preconditions of later steps are met by earlier steps, a set of goals or intentions for the actors, and a set of unexplained steps.

2.2.8 Other Related Systems

One of the earliest software that generates stories is Meehan's TALE-SPIN [Mee77]. His approach is very similar to GLAIVE – story characters are given goal, and searches are made,

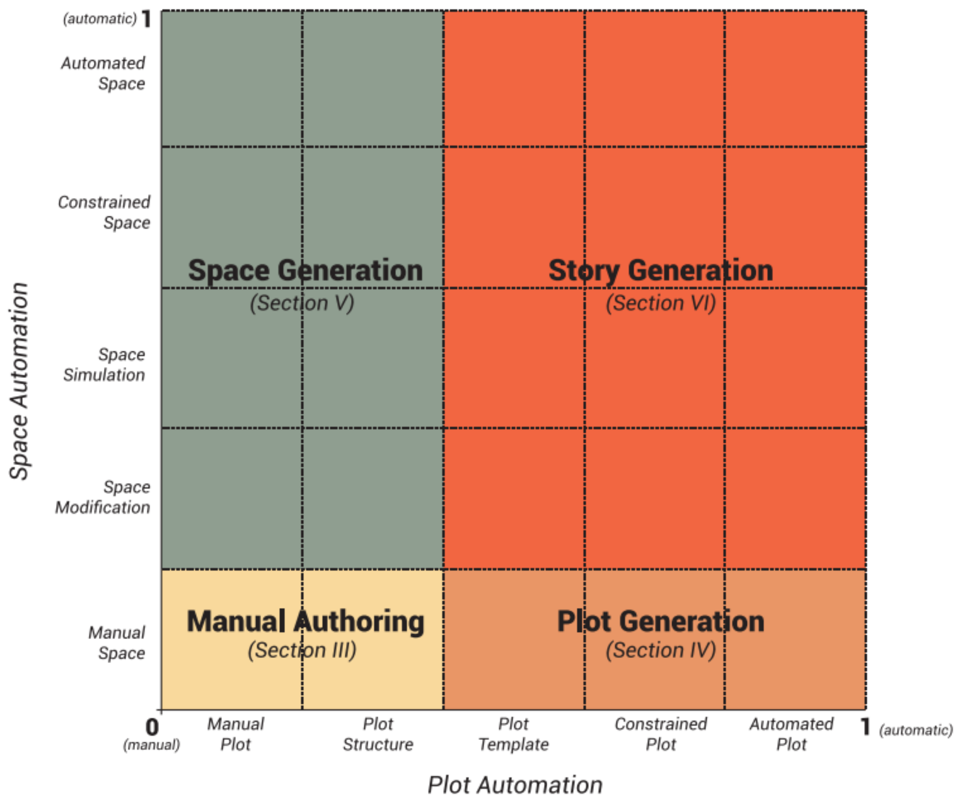


FIGURE 2.4: Story automation range, expressed in terms of the degrees of automation for *plot* and *space/universe/STU* generation

with the help of planning rules, in order to fulfill these goals. The traces of these searches are used as the basis for the output narrative.

A goal-based model of character personality is explored by Bahamón et. al. [BBY15]; character belief as modeled by Shrivani et. al. [SWF17] also plays a crucial role in describing how characters act based on their personal knowledge of the world – which may or may not be true. Both papers influenced how ISLA's planning mechanisms are designed, although goal-based character models have a more direct impact. The tree-like data structures required to describe alternate world-states are already present, and only the addition of *epistemic edges* are required to fully represent character/actor beliefs.

Assistive tools such as LISA: Lexically Intelligent Story Assistant [SHCK17] also exists in a different branch of the narrative software taxonomy. LISA focuses on real-time feedback specifically for lexical inconsistencies in the story.

Narrative planners are also used as tools for natural language story scripts [SWS$^+$18]. Sanghrajka et. al. describes a tool that assists writers with the aspects of "story world book-keeping", which is especially critical when creating new stories within already-established story-telling universes.

2.3. Plot Evaluation

2.3.1 Challenges with Objective Evaluation

Narrative, like most objects studied by humanists, is a subjective, culturally-dependent phenomenon. As such, attempts to model it fall victim to the problem of validation in the humanities; each reading of a narrative serves as one of many possible readings by one of many possible readers [MP20].

Although the question of computational creativity evaluation is non-trivial, Jordanous presented an impressive attempt to create a standardized procedure to evaluate creative systems [Jor12]. Researchers who tackle relatively ill-defined and definitely highly subjective topic of creativity research, tools such as the Standardised Procedure for Evaluating Creative Systems makes creativity more tangible to work with.

2.3.2 Tellability

Capturing the quality measure of plot, independent of discourse is a necessity for systems such as ISLA. Tellability, a concept introduced by narratologists to describe the potential suitability of a configuration of events to be rendered into discourse, presents itself as candidate for such a measure of plot quality [Ber17]. If a narrative planning system is to implement measuring tellability, it must possess 1) a character architecture that represents beliefs and intentions as a propositional knowledge base, 2) representation of well-formed plots and 3) implicit or explicit representation of character conflict.

The presence of complex plot units are offered as the key component of the measure of tellability. In Lehnert's work [Leh81], she introduced several intra-character and cross-character complex plot units. Those units are by no means complete and comprehensive, since there is no theoretic limit to the size of these units and more can be derived.

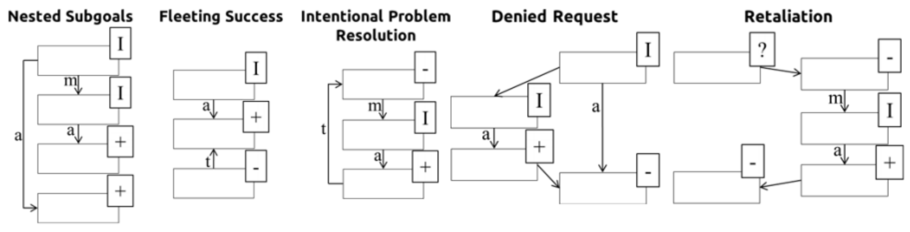


FIGURE 2.5: Examples of complex plot units; '?' is a wildcard for arbitrary vertex type.

In fig. 2.5, the vertices are denoted as:

- **+**: an internal or external event that is appraised with a positive emotion by the character,
- **–**: an internal or external event that is appraised with a negative emotion by the character,
- **I**: a neutral internal event that denotes the setting of an intention by the character

And these vertices can be connected by five types of edges:

- **m**: a motivation edge can lead from any type of vertex to an I vertex and denotes that the former event causes the intention,
- **a**: an actualization edge can lead from an I vertex to a + or - vertex and denotes that the intention causes the latter event,
- **t**: a termination edge can connect either two non-neutral, or two I, vertices and leads from the event that terminates an affective/intensional state to the one that is terminated (respectively supplanted in the I case)
- **e**: an equivalence edge can connect two non-neutral, or two I vertices, and denotes that an event has multiple affective evaluations, or two intentions are congruent; the edge direction is anti-temporal,
- **a cross-character edge** is temporally directed and can connect all types of vertices, so long as they belong to different characters' subgraphs; it denotes that an event is affecting several characters, prominently also including speech acts.

2.4. Theoretical Framework

The relationship of narratives with mathematics and logic establishes the groundwork for the ideas that will eventually become ISLA. The recurring theme that narratives can be expressed by an ordering of smaller elements, is a central tenet to ISLA's design philosophy. This ordering of elements is an indication that methods and tools from other domains of knowledge can then be applied to creating narrative artifacts. Studies in condensing corpora to schemata [McI11] provided key insight that pre-existing stories does have structure and affirms the assertions of earlier works such as Propp's fairy tale morphology and Tobias' Master Plots.

With the abundance of techniques and algorithms relevant to narrative generation [KB17], it might be best to first try determine where in the general taxonomy of narratology tools ISLA may belong to. With the components of a narrative succinctly decomposed as essentially *plot*[2] and *space*[3], taxonomic classification can then be based on how *plot* or *space* is automated. Plot generation (manual space, plot template / constrained plot / automated plot – fig. 2.4) fit the original vision for ISLA, which is aimed to create story outlines based on rules set by the user. This design choice has narrowed down the potential algorithm candidates which primarily deal with narrative planning specifically POCL, IPOCL, and CPOCL, which will be discussed further in later sections. These are different from other narratological algorithms which try to directly influence the *space* of the narratives – which may include discovery and optimization of actions, or perhaps modification of actor attributes and personalities. The deeper investigation of these kinds of algorithms is not included in the scope of this paper.

Perspectives and insights from existing works by Riedl, Young, and Ware's work on narrative planner algorithms [You99] [RY09] [WY11] [WY14] provided the backbone for ISLA's

[2]A set of events with an overall structure which represents both the temporal ordering, and the causal relations between the events

[3]The characters, settings, props, and anything which is present either physically or abstractly in the space of the narrative (e.g. existents)

own implementation of representing logical causality, with intentionality and conflict. Shrivani et. al. [SWF17] highlighted specific concepts such as belief modeling, emphasizing that there is still ways to grow systems like ISLA. Berov and Lehnert's [Ber17] *tellability* concept has provided a concrete way to measure the quality of the narrative outline output.

CHAPTER 3

ISLA: INTELLIGENT STORY LAYOUT ASSISTANT

3.1. Knowledge-base Encoding

3.1.1 Domain-Problem

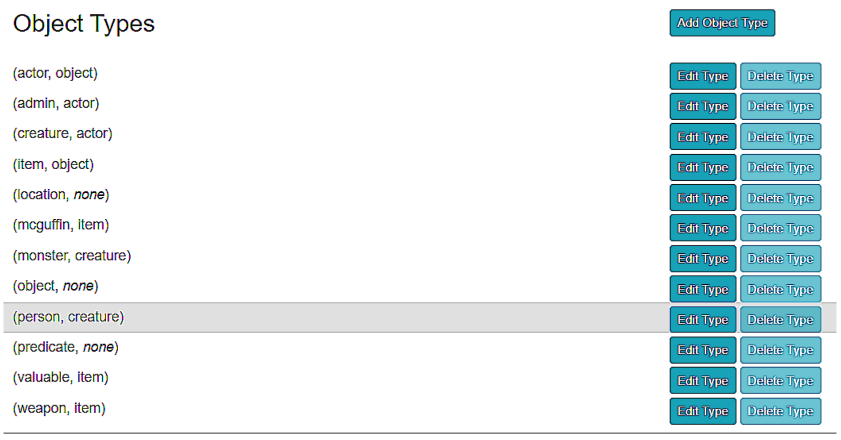
In order to properly represent the narrative universe, the Planning Domain Definition Language (PDDL) was chosen to encode information about world-states, allowable actions, actor intentionality operators, object/existent descriptors, author and character goals, and other metadata. ISLA utilizes a knowledge-base encoding language based on PDDL, but with a number of customizations, such as explicit intentionality operators, conditional statements, and author-related statements.

Allowable actions are controlled by action parameters and precondition input. As illustrated in figure 3.4, all *precondition* predicates must be satisfied before the action can be applied to a world-state. The *effect* section denotes which state predicates are applied (or removed in case of a *not* operator). *Parameters* and *agents* facilitate ease of parsing and action validity evaluation.

As for intentionality restrictions, only the *actor* supertype is allowed to have intentions. If, for example, we wish to force certain event where there are no valid actors that can be involved (e.g. forces of nature, accidents, etc.), intentions are attached to the mandatory *author* type.

$$
\begin{array}{lll}
\mathit{Domain} & \langle T, P, A \rangle & \\
\mathit{where} & T & \text{is a set of object/existent type definitions} \\
 & P & \text{is a set of state predicate definitions} \\
 & A & \text{is a set of allowed actions}
\end{array}
\tag{3.1}
$$

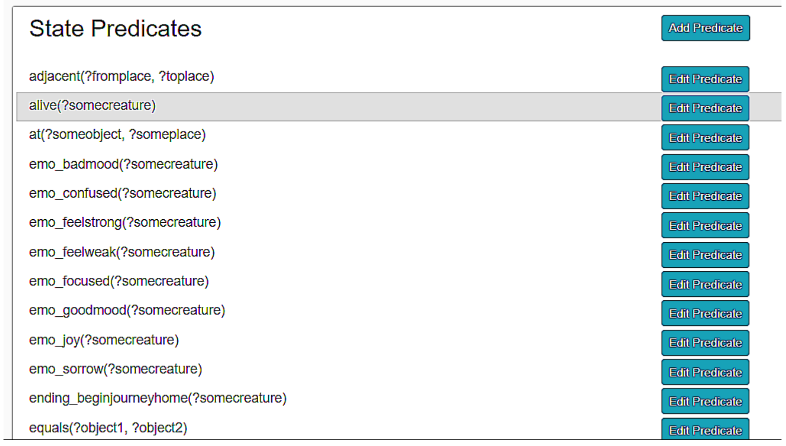
$$
\begin{array}{lll}
\mathit{Problem} & \langle E, S_0, S_f \rangle & \\
\mathit{where} & E & \text{is a set of object/existent instances} \\
 & S_0 & \text{is the initial state} \\
 & S_f & \text{is the goal state}
\end{array}
\tag{3.2}
$$



Object Types

(actor, object)
(admin, actor)
(creature, actor)
(item, object)
(location, *none*)
(mcguffin, item)
(monster, creature)
(object, *none*)
(person, creature)
(predicate, *none*)
(valuable, item)
(weapon, item)

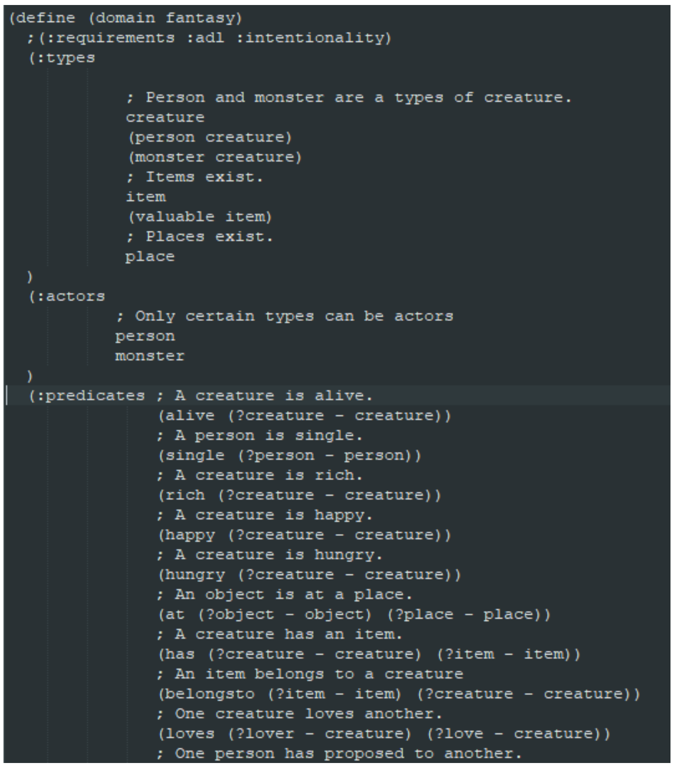
FIGURE 3.1: Domain - Object Types



State Predicates

adjacent(?fromplace, ?toplace)
alive(?somecreature)
at(?someobject, ?someplace)
emo_badmood(?somecreature)
emo_confused(?somecreature)
emo_feelstrong(?somecreature)
emo_feelweak(?somecreature)
emo_focused(?somecreature)
emo_goodmood(?somecreature)
emo_joy(?somecreature)
emo_sorrow(?somecreature)
ending_beginjourneyhome(?somecreature)
equals(?object1, ?object2)

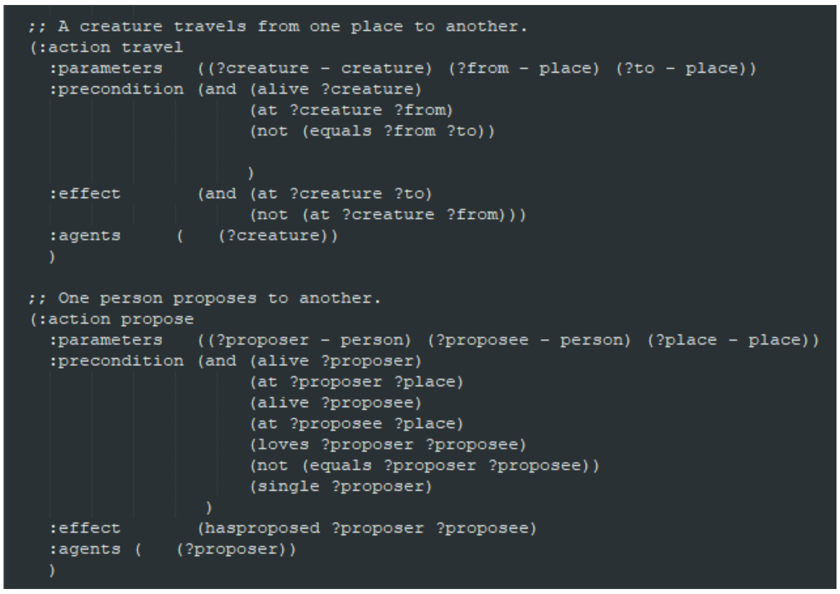
FIGURE 3.2: Domain - Object Types



```
(define (domain fantasy)
  ;(:requirements :adl :intentionality)
  (:types

            ; Person and monster are a types of creature.
            creature
            (person creature)
            (monster creature)
            ; Items exist.
            item
            (valuable item)
            ; Places exist.
            place
  )
  (:actors
           ; Only certain types can be actors
           person
           monster
  )
  (:predicates ; A creature is alive.
               (alive (?creature - creature))
               ; A person is single.
               (single (?person - person))
               ; A creature is rich.
               (rich (?creature - creature))
               ; A creature is happy.
               (happy (?creature - creature))
               ; A creature is hungry.
               (hungry (?creature - creature))
               ; An object is at a place.
               (at (?object - object) (?place - place))
               ; A creature has an item.
               (has (?creature - creature) (?item - item))
               ; An item belongs to a creature
               (belongsto (?item - item) (?creature - creature))
               ; One creature loves another.
               (loves (?lover - creature) (?love - creature))
               ; One person has proposed to another.
```

FIGURE 3.3: Domain - Types and Predicates, PDDL format



```
;; A creature travels from one place to another.
(:action travel
  :parameters   ((?creature - creature) (?from - place) (?to - place))
  :precondition (and (alive ?creature)
                     (at ?creature ?from)
                     (not (equals ?from ?to))

                     )
  :effect       (and (at ?creature ?to)
                     (not (at ?creature ?from)))
  :agents     (   (?creature))
  )

;; One person proposes to another.
(:action propose
  :parameters   ((?proposer - person) (?proposee - person) (?place - place))
  :precondition (and (alive ?proposer)
                     (at ?proposer ?place)
                     (alive ?proposee)
                     (at ?proposee ?place)
                     (loves ?proposer ?proposee)
                     (not (equals ?proposer ?proposee))
                     (single ?proposer)
                 )
  :effect       (hasproposed ?proposer ?proposee)
  :agents (   (?proposer))
  )
```

FIGURE 3.4: Domain - Actions definition, PDDL format

3.1.2 Narrative Structures

Plot structures used by ISLA are based on work by Gervás et. al. [GLM15]. This provided a convenient source of standardized plot structures distilled from multiple plot descriptions.

In order to properly encode these plot structures, custom narrative structure components were crafted which facilitate key plot elements and domain state predicate behavior. This encoding approach enables ISLA to have a non-deterministic output when tasked to produce a chapter pattern instance.

In the current state of ISLA, she is using a simple hierarchy to describe the current storytelling universe:

```
Narrative Chapter Structure Hierarchy
Domain
  +– Chapter Patterns
  |   +– Sequence Terms
  +– Predicate Descriptor Definitions
  +– Object Name Lookup
```
(3.3)

The domain defines the base rules of the storytelling universe: what kind of objects can exist, what their relationships with each other are, and how they interact. Story patterns are frameworks that define how the story outline should progress. This is achieved by using sequence terms (which is functionally a "story chapter") and chapter patterns.

3.1.2.1 Sequence Terms and Chapter Patterns

Sequence terms are fundamental knowledge-base component that defines specific intentions, or sub-goals, that are required to be fulfilled per story chapter. In other words, a sequence term represents a story chapter which the desired result or goals are known. The specific steps on how the story outline instance will take, however, is determined during the *planning phase*. This means that ISLA is not guaranteed to produce the exact same sequence of actions when creating plans from the same sequence term.

To have further control over the output narratives, ISLA also employs chapter patterns. The chapter pattern knowledge-base component represents the general structure of the whole narrative, which is composed of an ordered sequence of sequence terms. This allows the definition of common and special component sequence terms which can be arranged in any arbitrary way to conform to any story or narrative pattern paradigm.

3.1.2.2 Predicate Descriptor Definitions

Another supporting knowledge-base component is the predicate descriptor definitions. This augments the domain state predicate definitions by adding more metadata such as likelihood to spawn at the initial state and cardinality. For example, the predicate *at* two entries in the predicate descriptor definitions: one for each of its parameters, the *object* and the *place*. The *object* parameter has a cardinality of one-to-many relative to *place*, which means that an object can only be at one place at a time, but places can have multiple (or no) object located on it.

$$\begin{gathered}\textbf{State Predicate - "At":}\\ \text{(at (?object - object) (?place - place))}\end{gathered} \tag{3.4}$$

3.1.2.3 Object Name Lookup

ISLA employs a simple object naming lookup. Database tables store arbitrary object names, and ISLA chooses object instance names randomly. Internal mechanisms in ISLA prevent names from being chosen multiple times, as this can cause confusion during the planning phase.

3.1.3 Others

As of time of writing, individual character personalities have not yet been implemented. Numerous code junctions have been identified for future work on this aspect. For example, individual action pools can be assigned to each actor, which can be filtered/augmented based on that actor's personality. Run-time initial state refinement and state consistency mechanics may also benefit from the additional information maintained in the predicate descriptors.

3.2. Core Planner

Perhaps the single most important piece of code in ISLA is the core planner module (CPM). The CPM takes a fully defined domain-problem object[1]instance, and produces these key data structures as output:

- Goal Graph
- Plan Graph

The solution paths, which represent the actual relevant per-chapter solutions, are built while growing the plan graph. The succeeding sections provide more details on how solutions are discovered using the two aforementioned data structures.

3.2.1 Based on GLAIVE

Heavily influenced by the work of Ware et. al. [WY14], the core planner module is very similar to GLAIVE. ISLA's CPM is a forward-chaining planner and functions like so: beginning with the initial state, the search progresses from the initial state towards a goal state, applying appropriate actions along the way. Formally, it is a search through a landscape where each node is defined by a tuple $< S,\Pi >$. S is a world state comprised of predicate facts and Π is the plan (a series of ordered actions) used to reach S from the initial state. The initial state is denoted as: $< S_0, \{\} >$

Directed edges between pairs of nodes in the search landscape correspond to applying actions to lead from one state to another. When an action A_n is applied to a search space node

[1]A program object, not an *existent* as defined on section [1.1]

$< S, \Pi >$ the node $< S', \Pi' >$ is reached, where S' is the result of applying the action A_n in the state S, and Π' is determined by appending the action A_n to Π. As such, $\Pi = A_0 ... A_n$. Forward-chaining search through this landscape is restricted to only considering moves in a forward direction: transitions are only ever made from a node with plan Π to nodes with a plan Π' where Π' can be determined by adding (or 'chaining') actions to the end of Π.

3.2.2 Goal Graph

To facilitate the search for solutions given a set of known intentions, a structure known as a *goal graph* is required. The goal graph will take part in constructing the actual plan graph by providing a heuristic when choosing the next viable steps from the current *plan graph* node state.

The goal graph is a layered tree-like directed graph with state predicates as nodes and action instances as edges. Construction begins at the "goal" layer, which is comprised of state predicates from all known *intentions*. Intentions are special state predicates denoting that an actor *A* intends state predicate *P* to be true at some point in the future. As such, it represents one of *A*'s possibly many goals in the narrative.

$$\begin{aligned} &intends(A, P) \\ &intends(Author, has(A1, Item1)) \end{aligned} \tag{3.5}$$

From the goal layer, the goal graph is actually grown backwards – with the parent's edges denoting an action that will resolve into one of (in this case, an intention) the state predicates as its child. In ISLA's implementation, progression from one goal graph node to another is somewhat loose.

Let P_{effect} be the set of effects for some action A_n, $P_{precondition}$ be the set of preconditions for some action A_{n+1}. The positive goal graph chaining rule is as follows:

Goal Graph Chaining Rule

$$\begin{aligned} if \quad & |(P_{\text{effect}} \cap P_{\text{precondition}})| >= |P_{\text{precondition}} - \text{t}| \\ then \quad & A_{\text{n}} \rightarrow A_{\text{n+1}} \end{aligned} \tag{3.6}$$

The *goal graph* chaining rule only checks how compatible A_n's effects are with A_{n+1}'s precondition by counting their intersecting terms. The rule does not take into account any configuration of the *plan graph*'s current state. Therefore, any path[2]in the goal graph leading to some goal node G_{goal_a} is a *potential* series of plan actions that can fulfill the goal predicate represented by G_{goal_a}

ISLA maintains one massive goal graph, as opposed to GLAIVE, which maintains several smaller goal graphs – one for each actor. The hypothesis is that having one goal graph for all active actors will inherently encode possible interactions with multiple actors and goals.

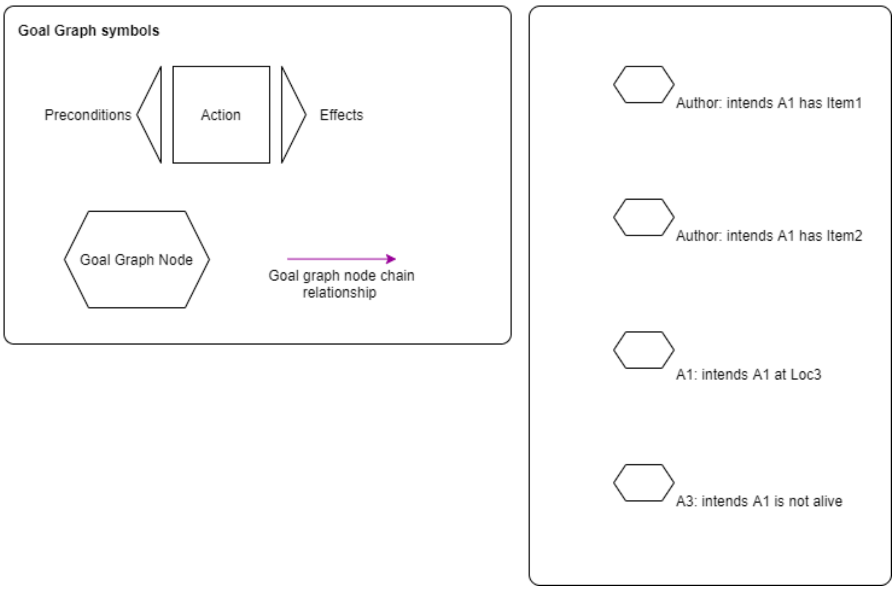

Goal Graph symbols
Preconditions
Action
Effects
Goal Graph Node
Goal graph node chain relationship
Author: intends A1 has Item1
Author: intends A1 has Item2
A1: intends A1 at Loc3
A3: intends A1 is not alive

FIGURE 3.5: Goal Graph initiation – Setting of all known goals/intentions as nodes in goal graph goal layer.

[2]As illustrated in figures 3.15-3.21

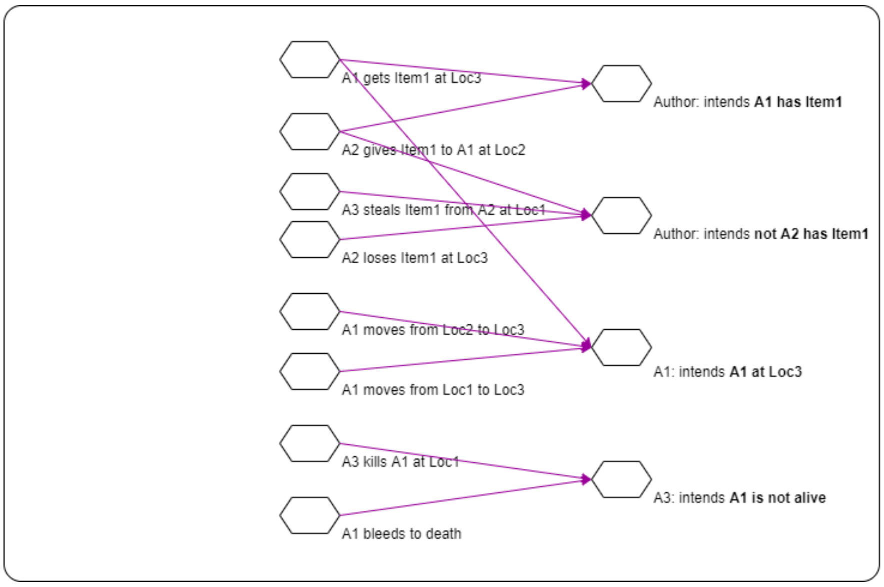

A1 gets Item1 at Loc3
A2 gives Item1 to A1 at Loc2
A3 steals Item1 from A2 at Loc1
A2 loses Item1 at Loc3
A1 moves from Loc2 to Loc3
A1 moves from Loc1 to Loc3
A3 kills A1 at Loc1
A1 bleeds to death
Author: intends **A1 has Item1**
Author: intends **not A2 has Item1**
A1: intends **A1 at Loc3**
A3: intends **A1 is not alive**

FIGURE 3.6: Goal Graph progression – Growing the goal graph one layer at a time. The nodes in preceding layers are possible actions that can be taken in order to potentially fulfill the intentions in the goal layer. These nodes only take into account how compatible the current action is (e.g. A1 gets Item1 at Loc3) with another goal node (Author intends A1 has Item1) based on the goal graph chaining rule (eq. 3.6)

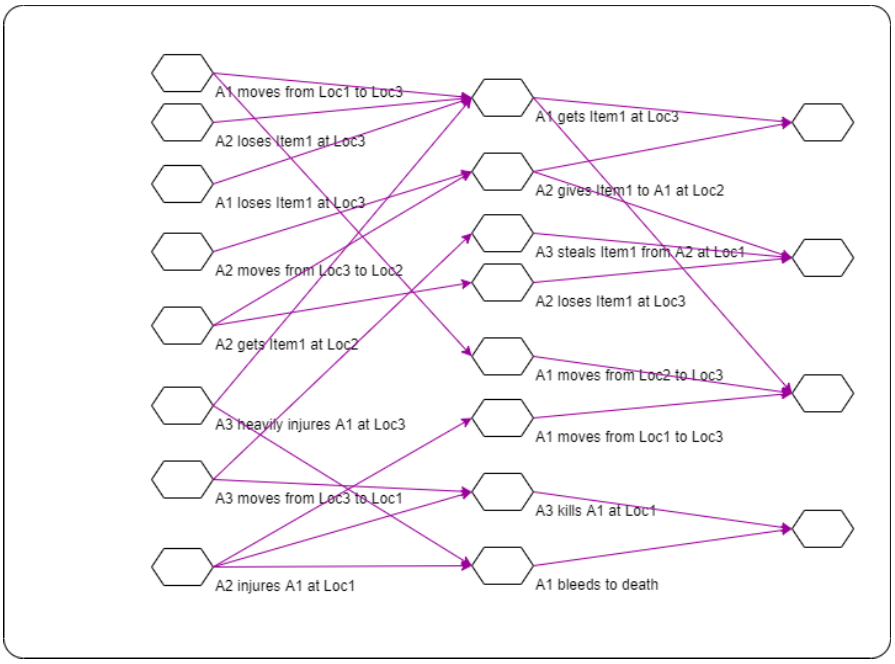

A1 moves from Loc1 to Loc3
A2 loses Item1 at Loc3
A1 loses Item1 at Loc3
A2 moves from Loc3 to Loc2
A2 gets Item1 at Loc2
A3 heavily injures A1 at Loc3
A3 moves from Loc3 to Loc1
A2 injures A1 at Loc1
A1 gets Item1 at Loc3
A2 gives Item1 to A1 at Loc2
A3 steals Item1 from A2 at Loc1
A2 loses Item1 at Loc3
A1 moves from Loc2 to Loc3
A1 moves from Loc1 to Loc3
A3 kills A1 at Loc1
A1 bleeds to death

FIGURE 3.7: Goal Graph progression – The goal graph is grown until it reached an arbitrary depth.

3.2.3 Plan Graph

The actual data structure that contain possible solutions is the plan graph. The plan graph is also a tree-like directed graph with full states – as opposed to mere predicates – as nodes,

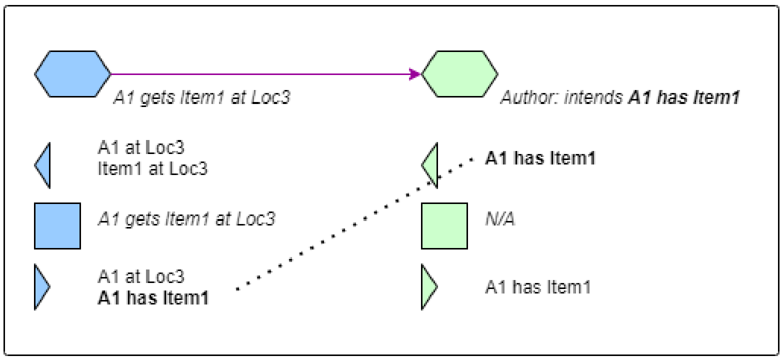


FIGURE 3.8: Details of a normal goal graph node connected to a goal layer node (or simply a goal node). The goal node's *precondition partial state* contains the state predicate instance that a particular actor (the Author, in this case) want to achieve. Since the *action* is not applicable in this layer, the *effect* wil be equivalent to the *precondition*.

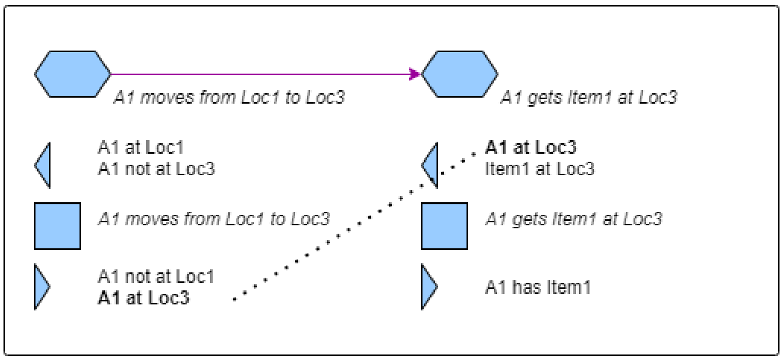


FIGURE 3.9: The normal goal graph nodes have all three components properly populated. In this example, the action *move()*'s preconditions reside in the goal graph node's *precondition partial state*, and its effects reside in the corresponding *effect partial state*

and actions as directed edges. It has a root node denoting the current state of the narrative S_0. From this starting point, candidate actions are chosen using the goal graph as a heuristic guide. The plan graph grows in a breadth-first manner. If certain conditions are met (e.g. solutions have been found and unexplained steps/actions are below threshold), plan graph construction is halted. Valid plans are then extracted as paths from the current state to solution state/s.

It is noteworthy that a breadth-first-search algorithm is used to explore the plan graph since it is often beneficial to find the shortest solution path, specifically when solving for chapter goals quickly

3.2.3.1 Backward Reasoning

The goal graph assists in growing the plan graph by way of an estimated sequence of actions. This sequence is derived by creating a backward reasoning path, with an arbitrary length *l*, from some goal graph node G_n towards some goal graph node in the goal layer G_{goal_a}. This marks a key difference between GLAIVE and ISLA; while GLAIVE considers which character's goal is closest to being achieved – as it maintains a separate goal graph for each active actor, ISLA chooses at random. This junction in ISLA's algorithm can improved in future works to fine-tune the range of the final solutions. For example, certain characters may get special priority because the current chapter is focused on them, or all characters are alternated equally so all their goals progress at roughly the same rate.

3.2.3.2 Intentional Paths

In order to represent character intentionality, ISLA employs relaxed intentional paths for step explanation based on GLAIVE's more rigid implementation [WY14].

Definition 1. A *causal link* $A_s \xrightarrow{p} A_t$ exists from step *s* to step *t* for some predicate p, if and only if step *s* has effect *p*, and step *t* has precondition *p*.

Definition 2. An *intentional path* for some character *c* and some goal *g* is a sequence of *n* causally linked actions $\langle A_1, A_2, ...A_n\rangle$ such that:

1. Character c consents to all actions
2. *c* intends *g* is true before action A_1 and true until action A_n
3. Action A_n has effect *g*
4. For *i* from 1 to *n* − *1*, there exists a causal link $A_i \rightarrow A_{\text{i+1}}$

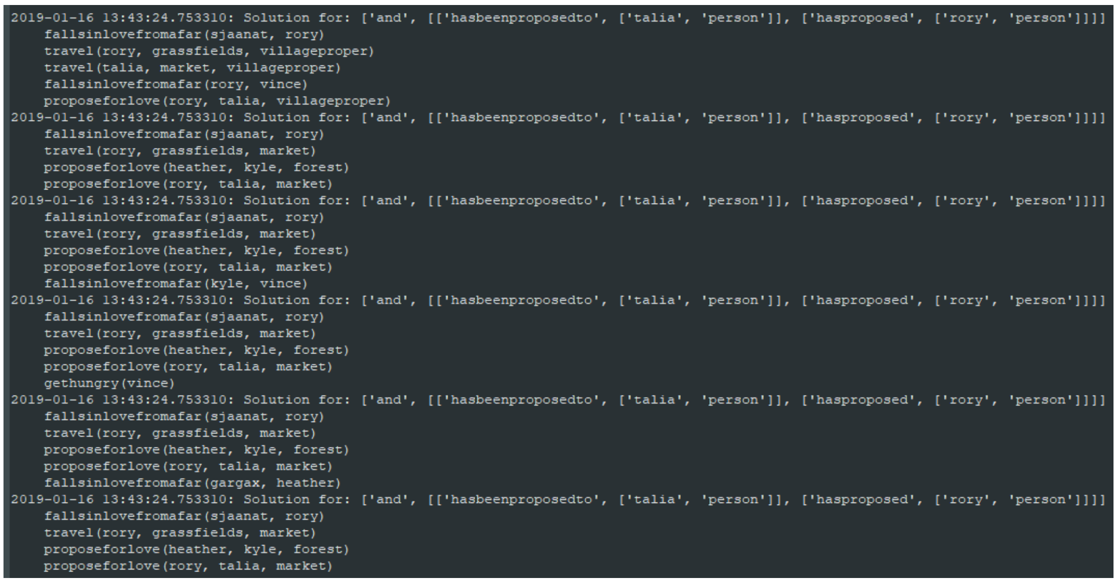
Definition 3. A step *s* is *explained* if and only if:

1. $\forall$ consenting character *c*, *s* in on an intentional path for *c*.
2. All other steps on that intentional path are explained.

Definition 4. A valid intentional plan is a sequence of *n* actions such that:

1. Each action A_i is causally link to action $A_{\text{i+1}}$
2. After action A_n is applied, the author's goals are satisfied
3. The number of unexplained steps does not reach some unexplained step threshold

Any candidate solution plan found during a plan graph search would still need to be a valid *intentional plan*. This means that the algorithm will not terminate upon locating a node containing a state that satisfy the author's goal – it will continue to locate partial solution nodes that will explain a sufficient number of steps already present in the candidate solution plan. For the sake of performance and flexibility, ISLA is equipped to control the unexplained step threshold. If the author desires that each and every step be explained (i.e. all character actions are properly *motivated*), then the threshold can be set to 0%; otherwise, the author can set this threshold closer to 100%, which may be described as complete chaos – with characters apparently in the mercy of ISLA choosing at random what they are to do next.



```
2019-01-16 13:43:24.753310: Solution for: ['and', [['hasbeenproposedto', ['talia', 'person']], ['hasproposed', ['rory', 'person']]]]
    fallsinlovefromafar(sjaanat, rory)
    travel(rory, grassfields, villageproper)
    travel(talia, market, villageproper)
    fallsinlovefromafar(rory, vince)
    proposeforlove(rory, talia, villageproper)
2019-01-16 13:43:24.753310: Solution for: ['and', [['hasbeenproposedto', ['talia', 'person']], ['hasproposed', ['rory', 'person']]]]
    fallsinlovefromafar(sjaanat, rory)
    travel(rory, grassfields, market)
    proposeforlove(heather, kyle, forest)
    proposeforlove(rory, talia, market)
2019-01-16 13:43:24.753310: Solution for: ['and', [['hasbeenproposedto', ['talia', 'person']], ['hasproposed', ['rory', 'person']]]]
    fallsinlovefromafar(sjaanat, rory)
    travel(rory, grassfields, market)
    proposeforlove(heather, kyle, forest)
    proposeforlove(rory, talia, market)
    fallsinlovefromafar(kyle, vince)
2019-01-16 13:43:24.753310: Solution for: ['and', [['hasbeenproposedto', ['talia', 'person']], ['hasproposed', ['rory', 'person']]]]
    fallsinlovefromafar(sjaanat, rory)
    travel(rory, grassfields, market)
    proposeforlove(heather, kyle, forest)
    proposeforlove(rory, talia, market)
    gethungry(vince)
2019-01-16 13:43:24.753310: Solution for: ['and', [['hasbeenproposedto', ['talia', 'person']], ['hasproposed', ['rory', 'person']]]]
    fallsinlovefromafar(sjaanat, rory)
    travel(rory, grassfields, market)
    proposeforlove(heather, kyle, forest)
    proposeforlove(rory, talia, market)
    fallsinlovefromafar(gargax, heather)
2019-01-16 13:43:24.753310: Solution for: ['and', [['hasbeenproposedto', ['talia', 'person']], ['hasproposed', ['rory', 'person']]]]
    fallsinlovefromafar(sjaanat, rory)
    travel(rory, grassfields, market)
    proposeforlove(heather, kyle, forest)
    proposeforlove(rory, talia, market)
```

FIGURE 3.10: Result Logs - Samples of actual found solutions

To elaborate, consider the following example. Given the follow author goals [eq.3.7] and actor intentions [eq.3.8], ISLA produced a solution plan [eq.3.9].

Author's goal:

$$\begin{array}{ll} and(& (I_{\text{a1}}) \\ \quad hasbeenproposedto(talia) & \\ \quad hasproposed(rory) & \\ ) & \end{array} \tag{3.7}$$

Actor intentions:

Talia - person

(intends talia (not (single talia))) (I_{t1})

(intends talia (not (at villagesquare talia))) (I_{t2})

Rory - person

(intends rory (happy rory)) (I_{r1})

(intends rory (not (single rory))) (I_{r2}) (3.8)

(intends rory (hasproposed rory)) (I_{r3})

Sja-anat - monster

(intends sjaanat (not (single sjaanat))) (I_{s1})

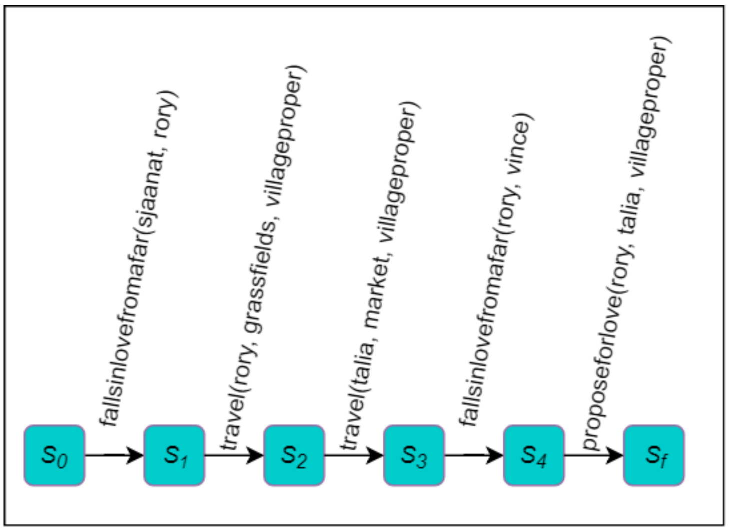
(intends sjaanat (marriedto heather kyle)) (I_{s2})

There are 3 active actors in this scenario, with distinct intentions of their own. Each action is attempted to be explained by exploring the plan graph space for states which satisfy any of the actor's intentions. In equation3.10, step/action π_1 can be explained if ISLA encounters an action *marry(sjaanat, rory, <anyplace>)* which results in some state $S_{f'}$. In practice, however, steps are not always easy to explain – the succeeding section will discuss how the chapter chainer module in conjunction with knowledge-base elements work together to solve this subproblem.

Found solution:

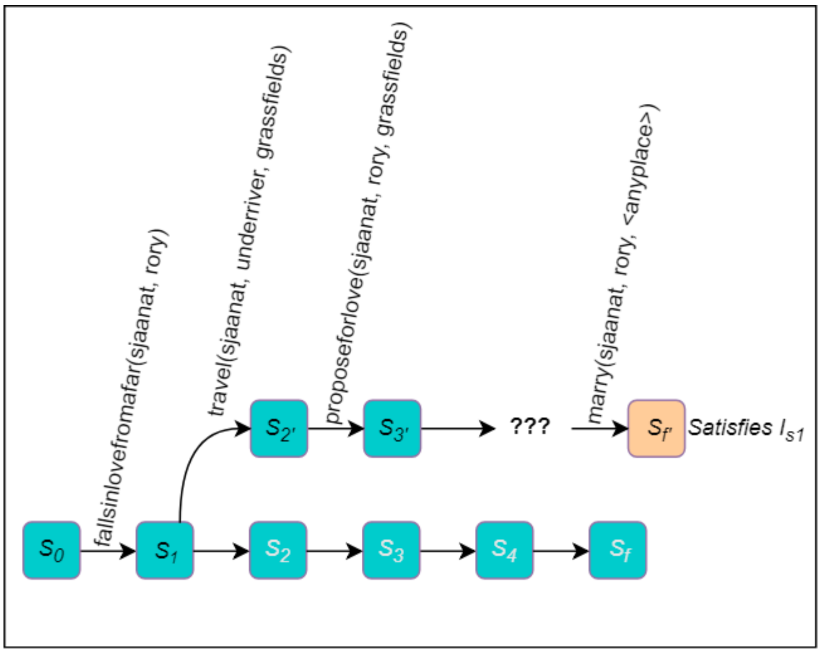
$$\begin{array}{ll} fallsinlovefromafar(sjaanat, rory) & (\pi_1) \\ travel(rory, grassfields, villageproper) & (\pi_2) \\ travel(talia, market, villageproper) & (\pi_3) \\ fallsinlovefromafar(rory, vince) & (\pi_4) \\ proposeforlove(rory, talia, villageproper) & (\pi_5) \end{array} \tag{3.9}$$

Steps π_2, π_3, and π_4 are easily explained. Fig.3.13 shows how step π_2 was explained when the state $S_{f''}$ is encountered. Take note that only preceding actions performed by *consenting active actors* whose intention/s are fulfilled by state $S_{f''}$ are explained. Hence for the aforementioned example, only actions **proposeforlove(rory, talia, villageproper)** and **travel(rory, grassfields, villageproper)** are explained, *not* fallsinlovefromafar(sjaanat, rory), even though it is part of the causal link that spawned state $S_{f''}$ on this particular "alternate reality".



fallsinlovefromafar(sjaanat, rory)
travel(rory, grassfields, villageproper)
travel(talia, market, villageproper)
fallsinlovefromafar(rory, vince)
proposeforlove(rory, talia, villageproper)
S0
S1
S2
S3
S4
Sf

FIGURE 3.11: Plan Graph - Solution found as detailed in [eq.3.9]



fallsinlovefromafar(sjaanat, rory)
travel(sjaanat, underriver, grassfields)
proposeforlove(sjaanat, rory, grassfields)
marry(sjaanat, rory, <anyplace>)
S2'
S3'
???
Sf'
Satisfies Is1
S0
S1
S2
S3
S4
Sf

FIGURE 3.12: Plan Graph - Attempting to explain step π_1

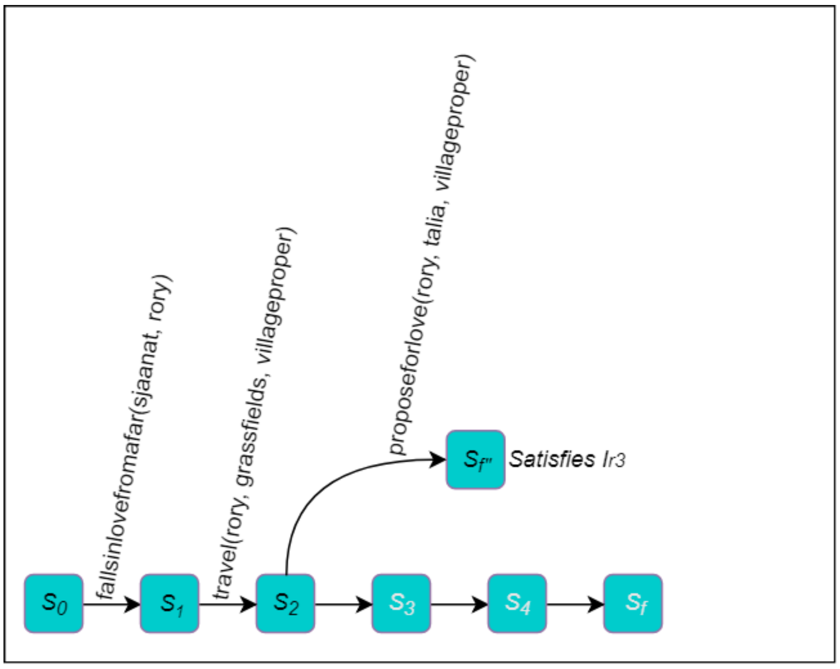

fallsinlovefromafar(sjaanat, rory)
travel(rory, grassfields, villageproper)
proposeforlove(rory, talia, villageproper)
S0
S1
S2
S3
S4
Sf
Sf''
Satisfies Ir3

FIGURE 3.13: Plan Graph - Attempting to explain step π_2

Step explanation:

(π_1)

fallsinlovefromafar(sjaanat, rory)
travel(sjaanat, underriver, grassfields)
proposeforlove(sjaanat, rory, grassfields)
...
marry(sjaanat, rory, <anyplace>) Satisfies I_{s1}

(π_2)

travel(rory, grassfields, villageproper)
proposeforlove(rory, talia, villageproper) Satisfies I_{r3}

(3.10)

(π_3)

travel(talia, market, villageproper) Satisfies I_{t2}

(π_4)

fallsinlovefromafar(rory, vince)
...
marry(vince, rory, <anyplace>) Satisfies $I_{\mathrm{r1}}, I_{\mathrm{r2}}$

(π_5)

proposeforlove(rory, talia, villageproper) Satisfies $I_{\mathrm{r3}}, I_{\mathrm{a1}}$

3.2.4 Plan Graph Sub-algorithm Comparison

During the course of ISLA's development, different approaches are implemented and tested in an attempt to improve performance. In this section, we describe the difference sub-algorithms used in the coreplanner; specifically on how the coreplanner chooses the next action to take from any given plan node.

3.2.4.1 Full Random

On full random setting, ISLA just chooses randomly from *all* legal actions. Only the action's ability to be executed on the current state is tested by ISLA. Results using the full random setting is included as the control set in all sub-algorithm comparison experiments.

3.2.4.2 Goal Graph Paths

Based on an interpretation of the GLAIVE algorithm, this is the initial sub-algorithm that was equipped to ISLA's coreplanner. ISLA utilizes the paths present in the goal graph[3]– a pre-generated data structure that represents the potential paths towards story goals. ISLA does this by searching the goal graph for a node with highest potential match to the current plan node, then tracing a path to a known goal node.

3.2.4.3 Directed Random

It has been observed that when using goal graph paths (section 3.2.4.2) alone, ISLA often gets lost by "travelling too often". Since travelling from one location to another is one of the easiest[4]actions to enact, ISLA's choices tend to get flooded by different combination of travel actions. This results in a too-large plan graph with little state changes that progresses (or potentially progresses) the story. To alleviate this issue, the directed random approach is formulated.

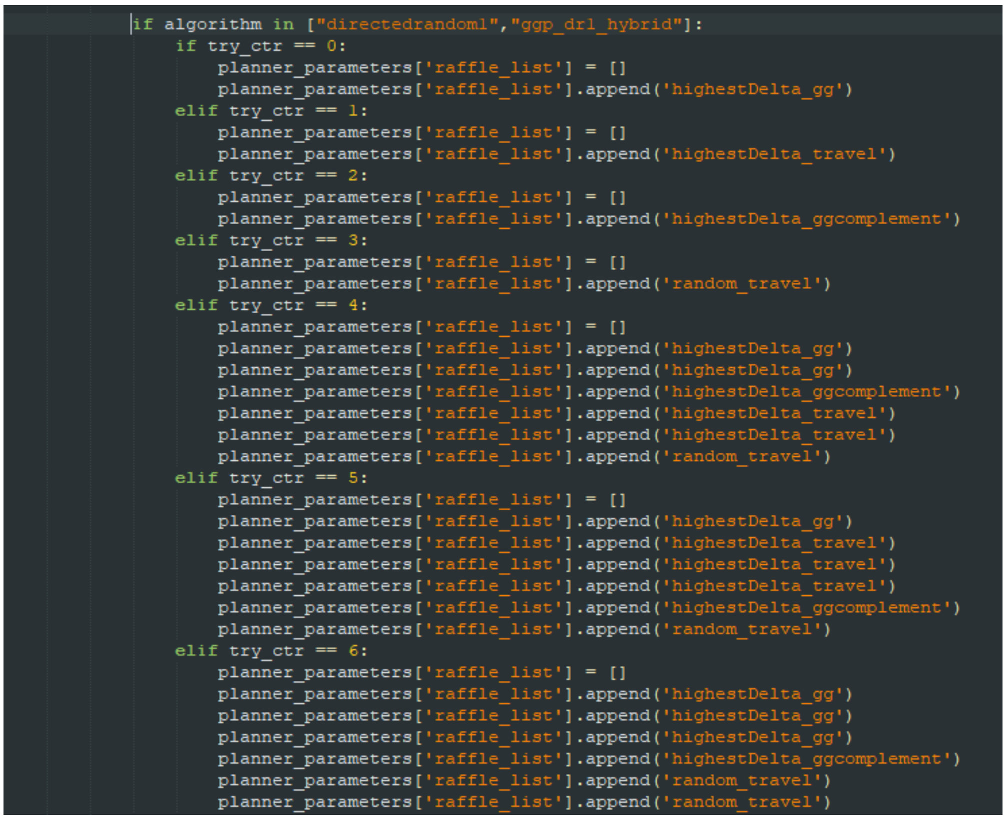
This 'directed random' approach is an attempt to improve the performance by guiding ISLA through manually defined buckets and using two different selection criteria.

- Action pool buckets
  - Actions present in the goal graph (GG-Actions)
  - Actions not present in the goal graph (GG-Complement-Actions)
  - Travel actions (Travel-Actions)
- Selection criteria
  - Highest Δ(Delta)
  - Random

The highest-delta selection criteria evaluates the score of each action in the bucket according to how much change the action will have an effect on the current state, and then selecting

[4]Just needs the creature to be alive, and that the source is not the same as the destination

the action with the highest score. This approach effectively reduces the chance of ISLA stalling during the cultivation of the plan graph.



```
if algorithm in ["directedrandom1","ggp_drl_hybrid"]:
    if try_ctr == 0:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('highestDelta_gg')
    elif try_ctr == 1:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('highestDelta_travel')
    elif try_ctr == 2:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('highestDelta_ggcomplement')
    elif try_ctr == 3:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('random_travel')
    elif try_ctr == 4:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('highestDelta_gg')
        planner_parameters['raffle_list'].append('highestDelta_gg')
        planner_parameters['raffle_list'].append('highestDelta_ggcomplement')
        planner_parameters['raffle_list'].append('highestDelta_travel')
        planner_parameters['raffle_list'].append('highestDelta_travel')
        planner_parameters['raffle_list'].append('random_travel')
    elif try_ctr == 5:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('highestDelta_gg')
        planner_parameters['raffle_list'].append('highestDelta_travel')
        planner_parameters['raffle_list'].append('highestDelta_travel')
        planner_parameters['raffle_list'].append('highestDelta_travel')
        planner_parameters['raffle_list'].append('highestDelta_ggcomplement')
        planner_parameters['raffle_list'].append('random_travel')
    elif try_ctr == 6:
        planner_parameters['raffle_list'] = []
        planner_parameters['raffle_list'].append('highestDelta_gg')
        planner_parameters['raffle_list'].append('highestDelta_gg')
        planner_parameters['raffle_list'].append('highestDelta_gg')
        planner_parameters['raffle_list'].append('highestDelta_ggcomplement')
        planner_parameters['raffle_list'].append('random_travel')
        planner_parameters['raffle_list'].append('random_travel')
```

FIGURE 3.14: Directed Random: Bucket-Selection combinations

To further increase ISLA's chances of finding the most effective action, ISLA also has several passes or attempts at this selection process. Each pass has a different action pool, and is basically a way to encode the "trying different things when one thing did not work" heuristic into ISLA.

1. Highest delta among GG-Actions
2. Highest delta among Travel-Actions
3. Highest delta among GG-Complement-Actions
4. Random Travel-Action
5. Others. Randomly chosen from an arbitrary mix of the four previous (see figure 3.14)

3.2.4.4 DR-GGP Hybrid

The directed random - goal graph paths (DR-GGP) hybrid seeks to improve the base goal graph path sub-algorithm by adding the directed random approach as the catching mechanism when GGP fails to find a suitable action.

3.2.4.5 Goal Graph Multi-Paths

The goal graph multi-path algorithm (GGMP) is an improved version of GGP. Instead of selecting a single path from the goal graph, GGMP looks for multiple paths in order to have more suggested actions for the plan graph to try to apply to the current state. Figures 3.15 to 3.22 illustrates how this works.

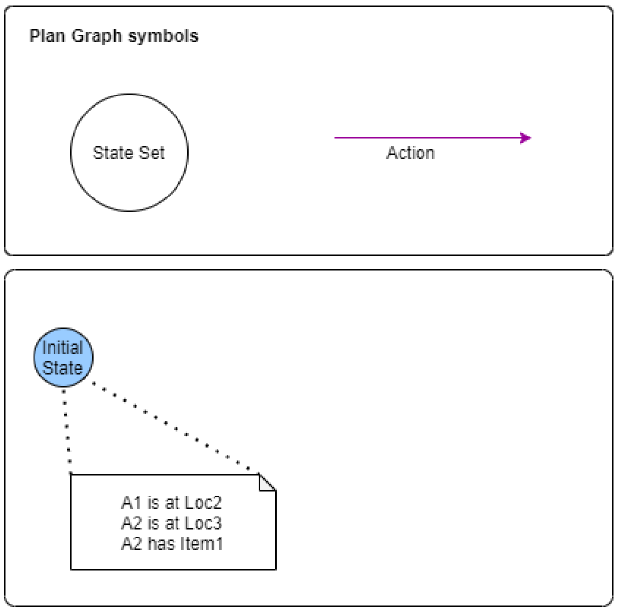


FIGURE 3.15: Plan Graph initiation – After the goal graph is generated, the plan graph is started up as a single node which contains the current state, which in the case of our example is the set *{A1 is at Loc2, A2 is at Loc3, A2 has Item1}*

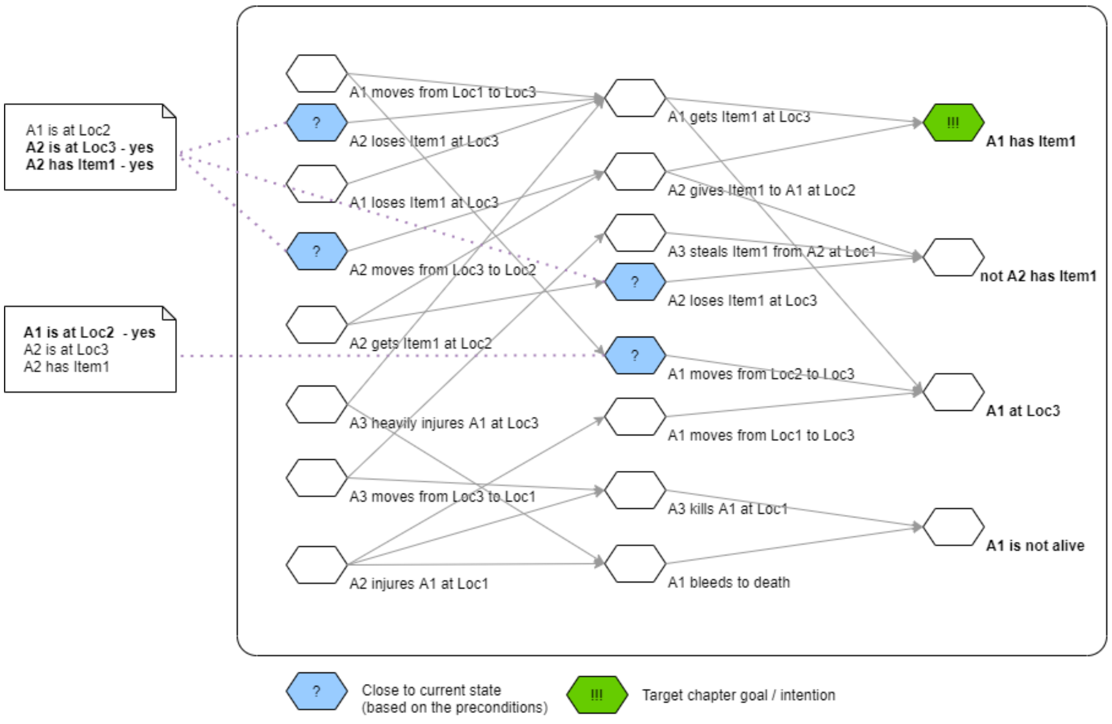


FIGURE 3.16: Using the goal graph from figure 3.7, ISLA identifies which goal graph nodes have an acceptable intersection with the current plan graph node's *state*. These candidate starting nodes (marked as '?') are potential starting points for paths to the relevant goal node (marked as '!!!') which represents the specific goal which ISLA trying to solve for.

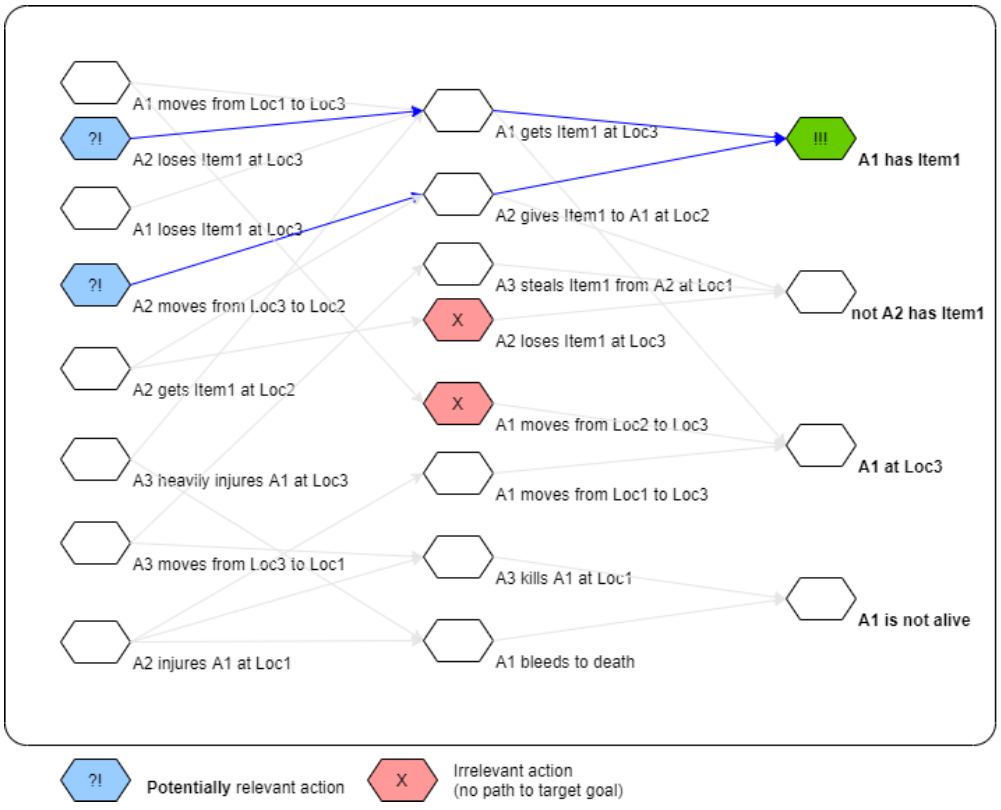


FIGURE 3.17: Goal graph nodes with paths to the relevant goal nodes are chosen as the actions to be applied to the current plan graph node (see fig. 3.18)

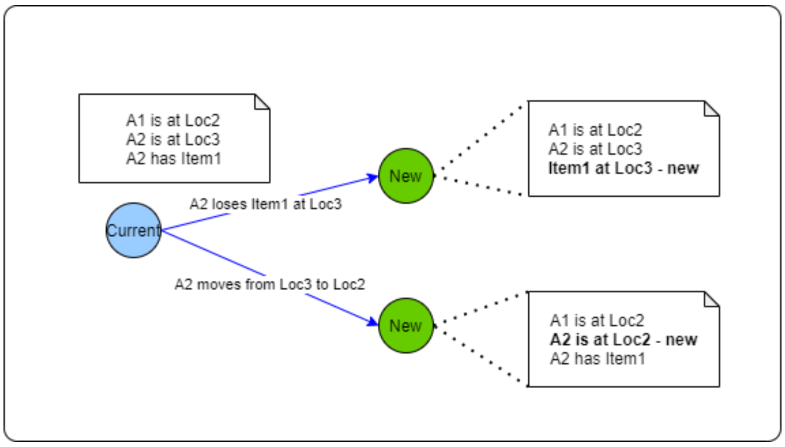


FIGURE 3.18: The plan graph grows by applying the actions identified by the goal graph as likely to move the current state closer to one of the goals. In this case, both actions *can* be applied to the current state – their complete preconditions are satisfied – however, this is not always the case (see fig. 3.20)

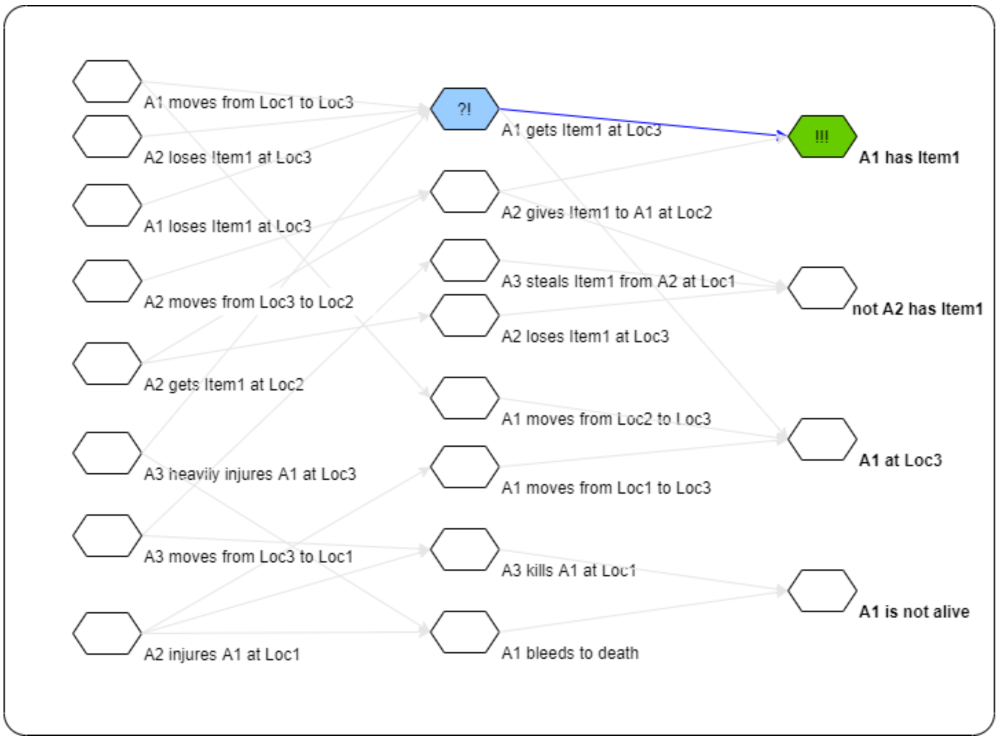


FIGURE 3.19: The *goal graph* continues to identify the next action to be tried by the *plan graph*.

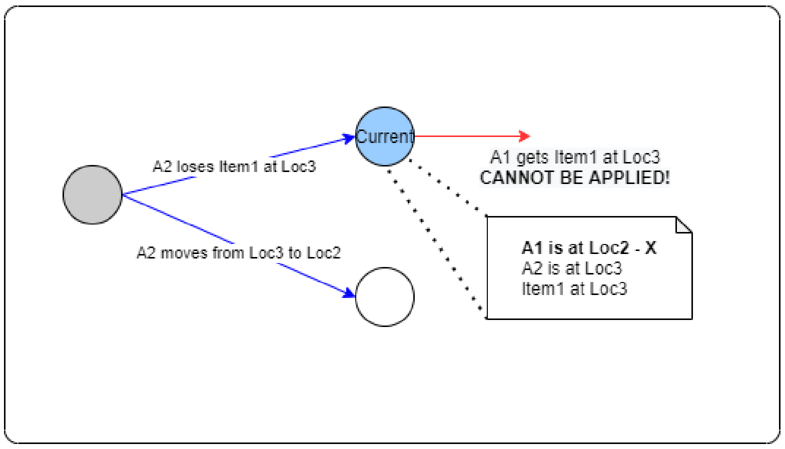


FIGURE 3.20: The process of *goal graph*-assisted *plan graph* growth continues, but the action identified step (fig. 3.19) is not compatible with the current state, and the now-current plan graph node is a dead end for the goal intention ***Author** intends A1 has Item1*

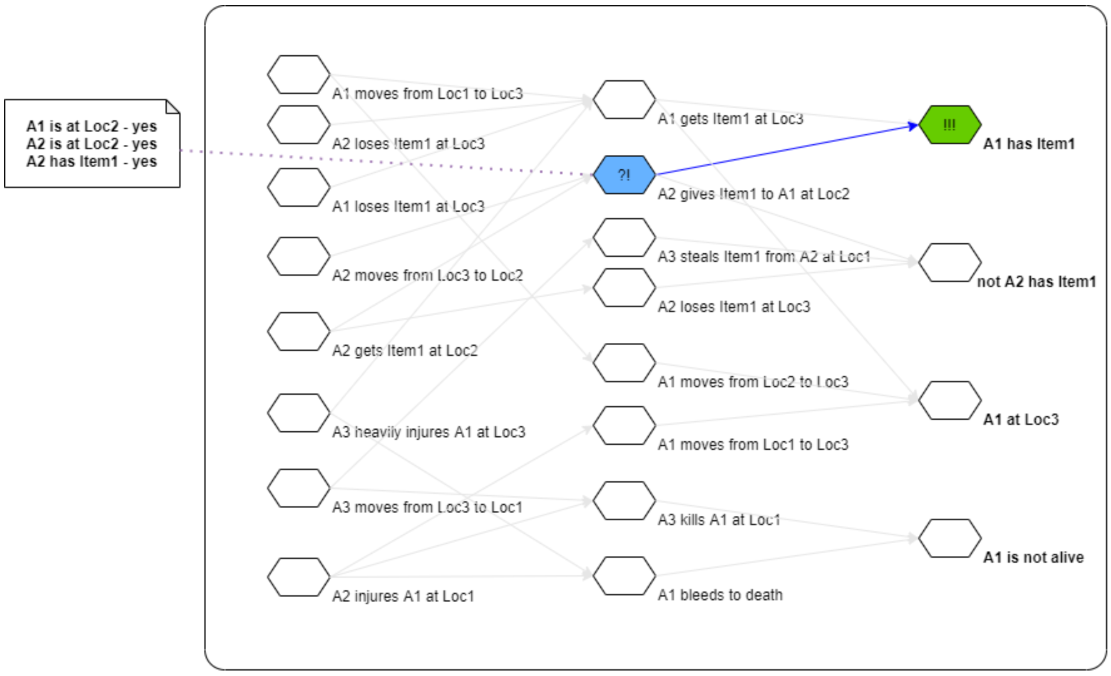


FIGURE 3.21: The *goal graph* continues to identify the next action to be tried by the *plan graph*.

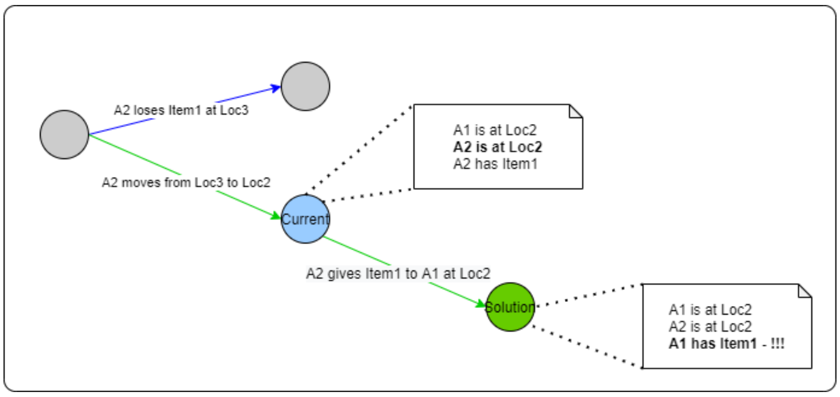


FIGURE 3.22: The latest action applied to the now-current plan graph node fulfills the intention *Author A1 has Item1* because the action *A2 gives Item1 to A1 at Loc2* has the relevant effect predicate: *A1 has Item1*.

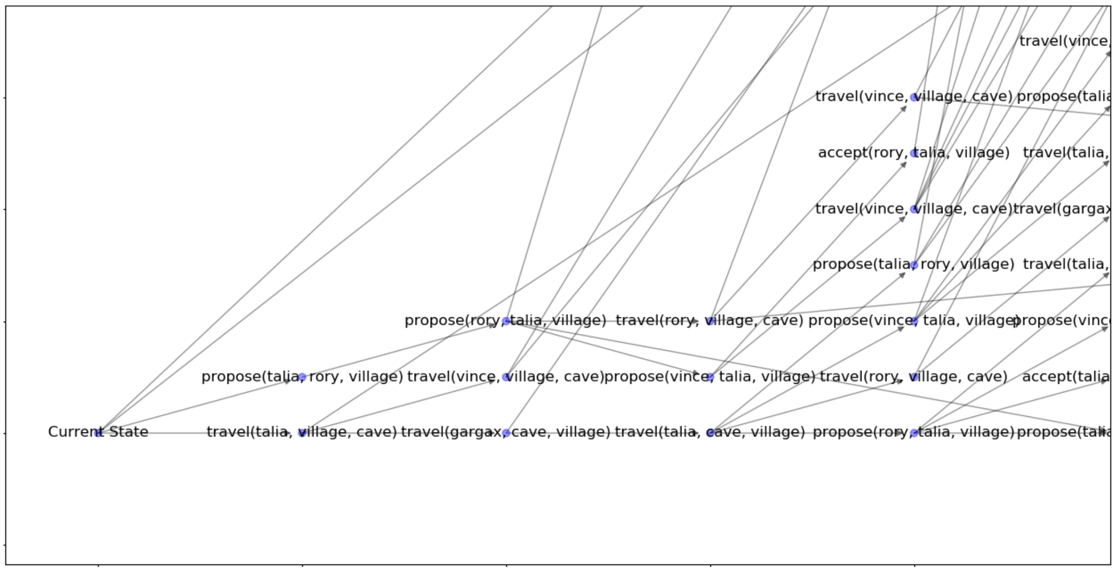

travel(vince,
travel(vince, village, cave) propose(talia
accept(rory, talia, village) travel(talia,
travel(vince, village, cave)travel(gargax
propose(talia, rory, village) travel(talia,
propose(rory, talia, village) travel(rory, village, cave) propose(vince, talia, village)propose(vinc
propose(talia, rory, village) travel(vince, village, cave)propose(vince, talia, village) travel(rory, village, cave) accept(talia
Current State travel(talia, village, cave) travel(gargax, cave, village) travel(talia, cave, village) propose(rory, talia, village) propose(talia

FIGURE 3.23: Plan Graph - From the initial state (taken from an earlier version of ISLA)

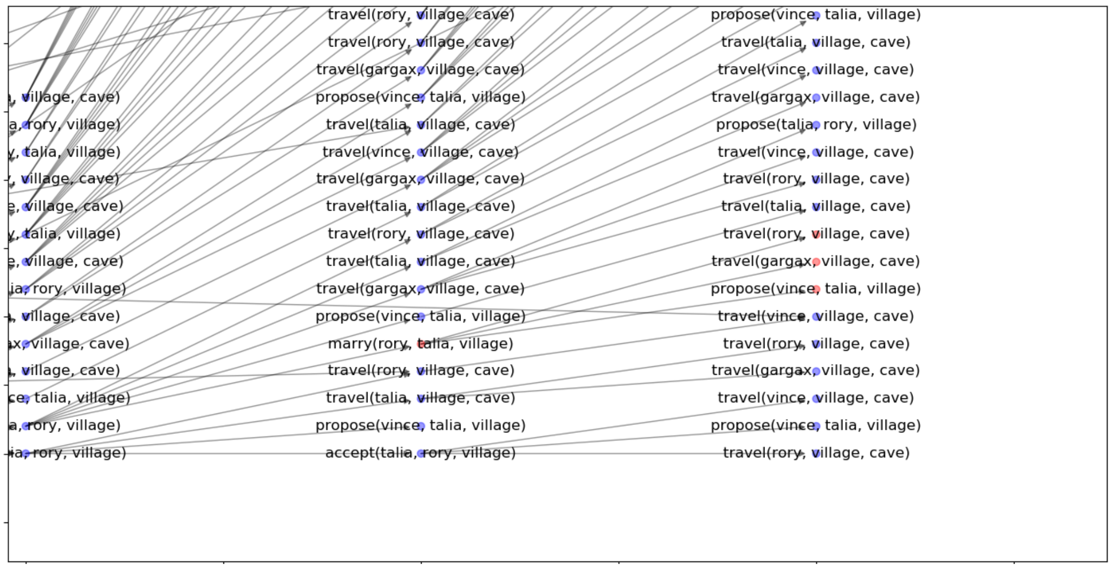

travel(rory, village, cave)
travel(rory, village, cave)
travel(gargax, village, cave)
propose(vince, talia, village)
travel(talia, village, cave)
travel(vince, village, cave)
travel(gargax, village, cave)
travel(talia, village, cave)
travel(rory, village, cave)
travel(talia, village, cave)
travel(gargax, village, cave)
propose(vince, talia, village)
marry(rory, talia, village)
travel(rory, village, cave)
travel(talia, village, cave)
propose(vince, talia, village)
accept(talia, rory, village)
propose(vince, talia, village)
travel(talia, village, cave)
travel(vince, village, cave)
travel(gargax, village, cave)
propose(talia, rory, village)
travel(vince, village, cave)
travel(rory, village, cave)
travel(talia, village, cave)
travel(rory, village, cave)
travel(gargax, village, cave)
propose(vince, talia, village)
travel(vince, village, cave)
travel(rory, village, cave)
travel(gargax, village, cave)
travel(vince, village, cave)
propose(vince, talia, village)
travel(rory, village, cave)

FIGURE 3.24: Plan Graph - Red nodes are solutions (taken from an earlier version of ISLA)

3.3. Chapter Chainer

Finding an intentional path for an entire narrative proved to be an intractable problem for ISLA. In order to find the long term author goals, ISLA needs a very large plan graph (with a depth of at least 8 nodes). Related to this issue was the difficulty in explaining all the steps in a candidate solution sequence. Steps such as π_1 and π_4 often have long causality link chains associated with them, which cause plan graphs to become very large. These two issues had a dramatic negative effect on ISLA's ability to find solutions for even for the most trivial problems during development.

The solution was, instead of setting lofty character goals (as is GLAIVE's approach) and letting ISLA find solutions by herself, character goals are divided into manageable sub-goals. These smaller goals are then solved and can be better understood if viewed as individual chapters of a larger narrative. Aside from solving the long causality link chain problem, having a *chapter chainer* module also facilitates more structured narratives to be produced.

3.3.1 Definitions

The narrative chapter structures introduced in section 3.1.2 are more properly described with the chapter chainer in the context. Each specific sequence term is linked to its successor, much like a causality link. Formally, a *chapter link* is defined as:

$$C_{\text{n}}\langle S_{\text{n}}, G_{\text{n}}\rangle \rightarrow C_{\text{n+1}}\langle S_{\text{n+1}}, G_{\text{n+1}}\rangle$$

where C is a chapter
S is the initial state of C
G is a set of author's goals for state C (3.11)

S_{n+1} is the final state of chapter C_n

A *chapter chain* is a sequence of chapter links from C_0 to C_{n}; where S_0 is the initial state of chapter 1, G_0 is the author's goal for chapter 1; S_1 is the initial state of chapter 2, G_1 is the author's goal for chapter 2; and so on ... , until C_{n}.

3.3.2 Benefits of the Chapter Chainer

Using a "quest story" as an example, the following sections discusses the benefits of implementing the chapter chainer in ISLA.

The Quest
(1) Difficult Task Arises
(2) ???
... (3.12)
(n-1) ???
(n) Quest Resolution

During the early stages of ISLA's development, the coreplanner was expected to produce solution plans using only two inputs:

- a set of initial predicates (the initial state)
- a set of target final predicates (author's goals)

This resulted in ISLA trying to navigate very large plan graphs in an effort to produce a solution path. The hypothesis was that if the singular author goal needs to be broken down into a series of easier-to-achieve sub-goals, ISLA can reach the author's goals by solving these sub-goals sequentially. By using known narrative structures (sec. 2.2), we are able to represent these sub-goals as story chapters – essentially a smaller narrative in itself – which ISLA can solve using the coreplanner.

The Quest
(1) Difficult Task Arises
(2) Quest Assigned
(3) Departure 1
(4) Departure 2
(5) Confront Guardian
(6) Quest Resolution
(3.13)

Since the number of chapters is known beforehand, a depth-first-search is employed to look for a suitable sequence of chapters for the narrative. By traversing the depth of a potential chain of chapters, ISLA attempts to locate complete chapter chains first before branching out to different paths.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1. Initial Outputs and Analysis

Given a reasonably crafted domain, an earlier version of ISLA is able to find narrative solutions on most runs. Initial tests involved a domain with around 30+ actions and a comparable amount of state predicate definitions, however, this relatively average domain complexity proved to be too much for ISLA to handle. With this in mind, a much smaller domain (10 action definitions) was crafted specifically to help conduct repeatable tests to quantify ISLA's performance. This smaller domain describes a simple quest story, where characters can delegate intentions, travel, obtain and lose items, and harm other characters. Succeeding sections will discuss in more detail several factors affecting ISLA's efficacy. Findings from these experiments have led to numerous changes to ISLA, and she can now consistently produce story outlines from larger domains, at higher success rates.

4.1.1 Core Planner

4.1.1.1 Input Domain factors

Initial experimental results are based on a relatively tiny domain – *The Quest*. This domain contains a mere 10 action definitions and 7 state predicate definitions.

Chapter Pattern (The Quest):
HeroDispatched
Departure1 (journey of hero)
Departure2 (continuation of journey)
Guardian (confront the item guardian)
QuestResolution (hero obtains the mcGuffin)

Actions (The Quest):
delegate_getquest
creaturegetsitemfromplace
creaturelosesitematplace
creaturegiveitemtocreature
steal
creaturekillcreature_withweapon
creaturekillcreature_noweapon
creatureinjurescreature_withweapon
creatureinjurescreature_noweapon
creaturebleedsout

State Predicates (The Quest):
(adjacent (?fromplace - place) (?toplace - place))
(alive (?somecreature - creature))
(at (?object - object) (?place - place))
(has (?somecreature - creature) (?item - item))
(hasweapon (?somecreature - creature))
(isinjured (?somecreature - creature))
(isheavilyinjured (?somecreature - creature))

(4.1)

Even without directly modifying the domain, incrementally adding locations to the dataset has significant impact on ISLA's performance. Figures 4.1 and 4.2 show goal graph growth from 23% to up to 40%. Figures 4.4 and 4.5 show action instance growth from 26% to up to 35%. This translates to direct increase in runtimes, as shown in figures 4.7 and 4.8.

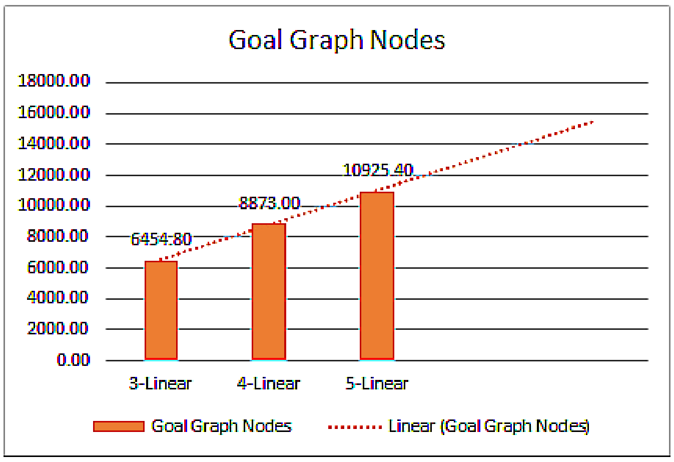


FIGURE 4.1: Goal Graph node count for linear terrain layout

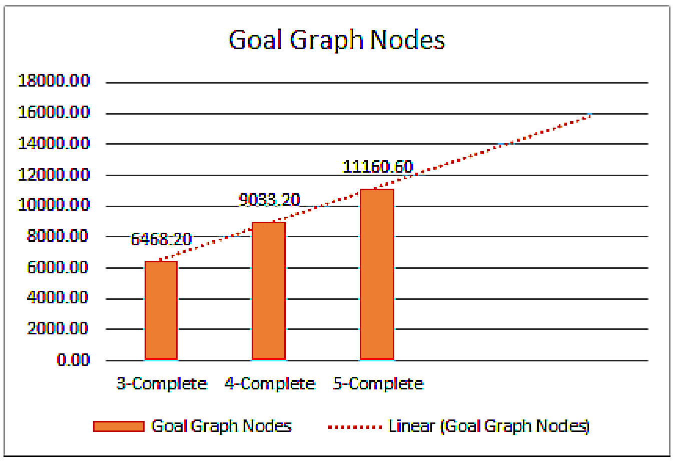


FIGURE 4.2: Goal Graph node count for complete terrain layout

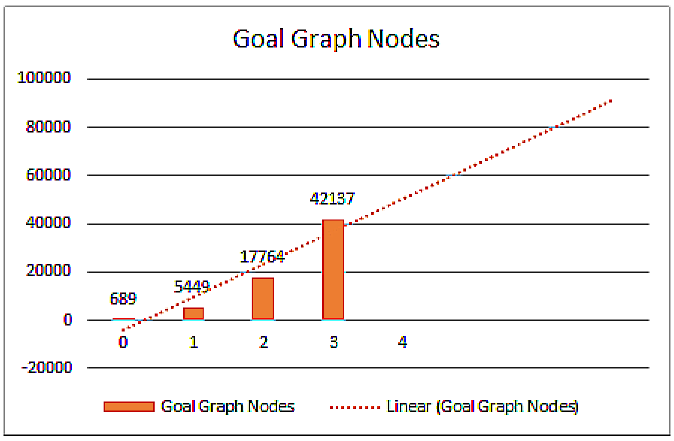


FIGURE 4.3: Goal Graph node count for varying optional characters setup

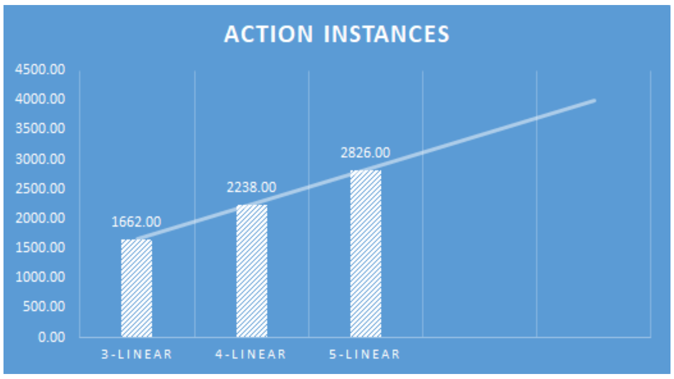


FIGURE 4.4: Action instance count for linear terrain layout. Locations are stringed together sequentially.

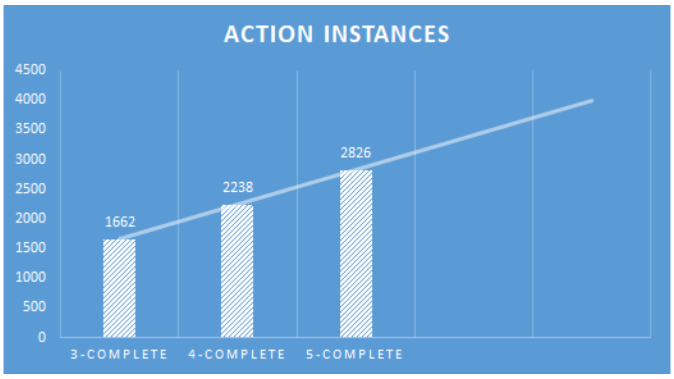


FIGURE 4.5: Action instance count for complete terrain layout. Locations are all connected to each other like a complete graph.

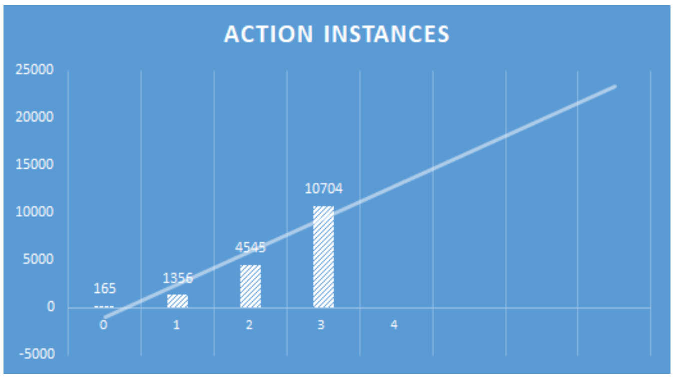


FIGURE 4.6: Action instance count for varying optional characters setup

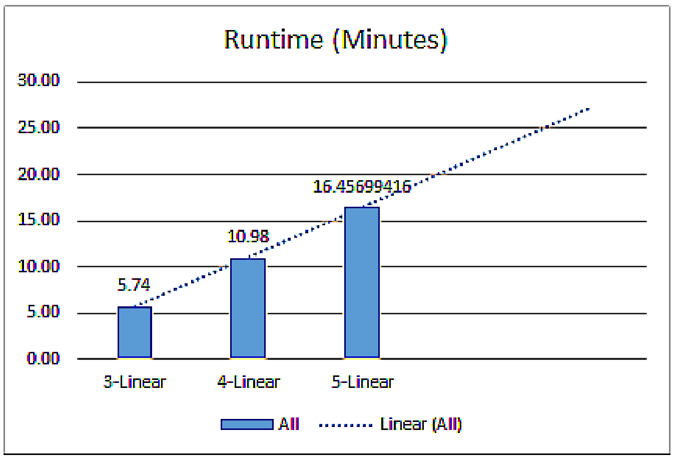


FIGURE 4.7: Runtime (minutes) for linear terrain layout

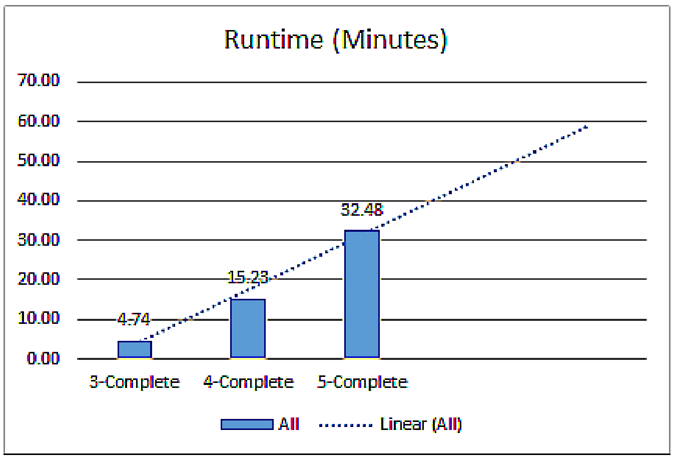


FIGURE 4.8: Runtime (minutes) for complete terrain layout

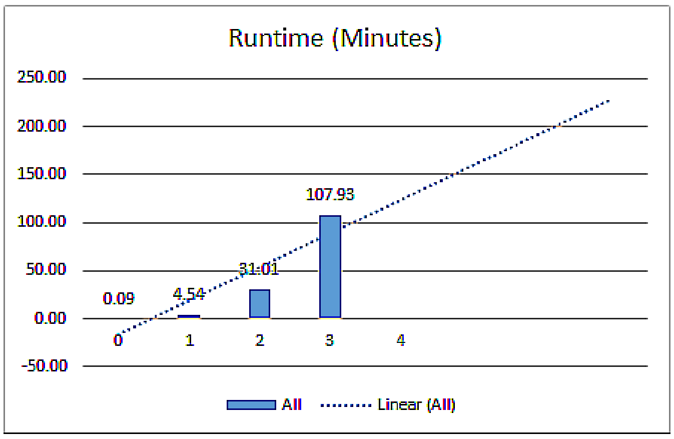


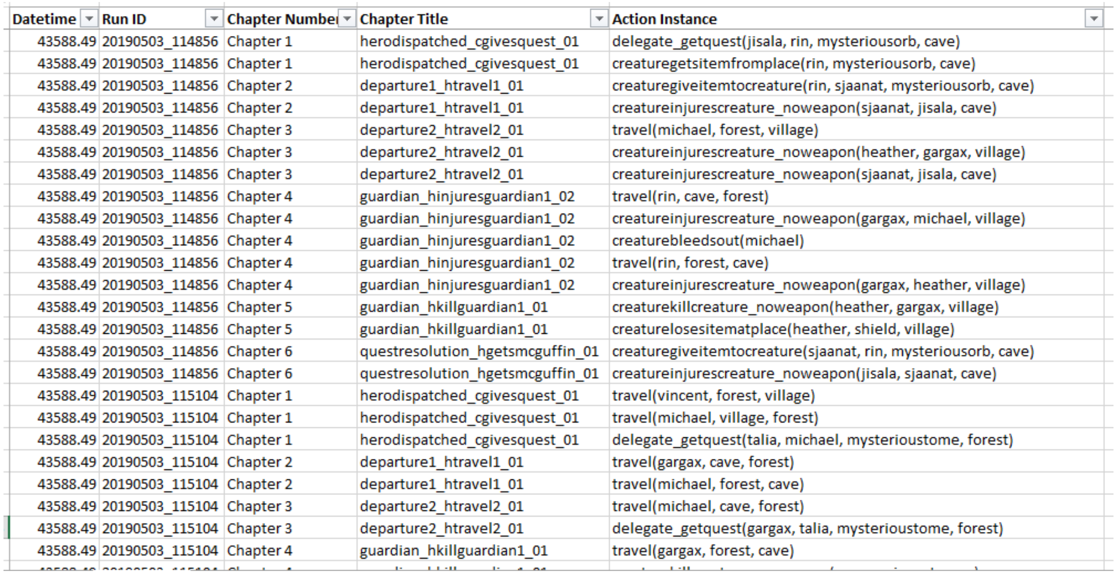
FIGURE 4.9: Runtime (minutes) for varying optional characters setup

4.1.1.2 Output Outline

Figures 4.10, 4.11, and 4.12 shows sample output outlines. The following outline is a good example of what ISLA is capable of producing:

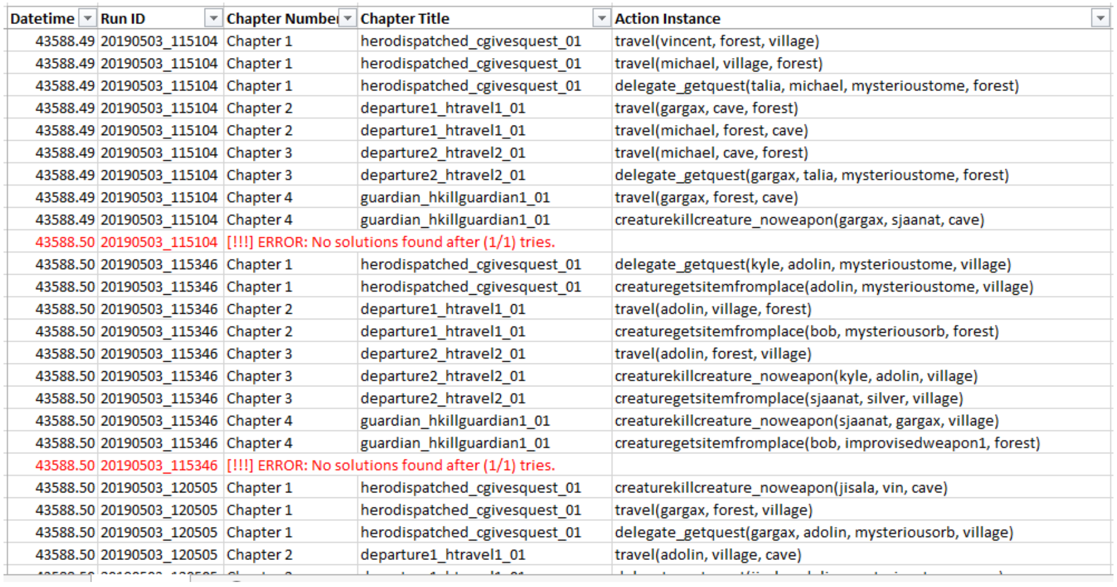
```
Run ID 20190503_114856: Outline
Chapter 1
    delegate_getquest(jisala, rin, mysteriousorb, cave)
    creaturegetsitemfromplace(rin, mysteriousorb, cave)
Chapter 2
    creaturegiveitemtocreature(rin, sjaanat, mysteriousorb, cave)
    creatureinjurescreature_noweapon(sjaanat, jisala, cave)
Chapter 3
    travel(michael, forest, village)
    creatureinjurescreature_noweapon(heather, gargax, village)
    creatureinjurescreature_noweapon(sjaanat, jisala, cave)
Chapter 4
    travel(rin, cave, forest)                                          (4.2)
    creatureinjurescreature_noweapon(gargax, michael, village)
    creaturebleedsout(michael)
    travel(rin, forest, cave)
    creatureinjurescreature_noweapon(gargax, heather, village)
Chapter 5
    creaturekillcreature_noweapon(heather, gargax, village)
    creaturelosesitematplace(heather, shield, village)
Chapter 6
    creaturegiveitemtocreature(sjaanat, rin, mysteriousorb, cave)
    creatureinjurescreature_noweapon(jisala, sjaanat, cave)
```

Recall that the goal of *The Quest* is for the hero to obtain a special item type – the McGuffin (4.1). Run ID 20190503_114856 Outline (figure 4.2) is particularly interesting because that goal was achieved right away on Chapter 1; the hero, *Rin*, immediately finds the *MysteriousOrb* after being given the quest by another character, *Jisala*. One might expect that ISLA will halt at this point since the author's goal has been met, however the requirement that the story outline match the story pattern **HeroDispatched-Departure1-Departure2-Guardian-QuestResolution** has not been satisfied.



Datetime	Run ID	Chapter Numbe	Chapter Title	Action Instance
43588.49	20190503_114856	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(jisala, rin, mysteriousorb, cave)
43588.49	20190503_114856	Chapter 1	herodispatched_cgivesquest_01	creaturegetsitemfromplace(rin, mysteriousorb, cave)
43588.49	20190503_114856	Chapter 2	departure1_htravel1_01	creaturegiveitemtocreature(rin, sjaanat, mysteriousorb, cave)
43588.49	20190503_114856	Chapter 2	departure1_htravel1_01	creatureinjurescreature_noweapon(sjaanat, jisala, cave)
43588.49	20190503_114856	Chapter 3	departure2_htravel2_01	travel(michael, forest, village)
43588.49	20190503_114856	Chapter 3	departure2_htravel2_01	creatureinjurescreature_noweapon(heather, gargax, village)
43588.49	20190503_114856	Chapter 3	departure2_htravel2_01	creatureinjurescreature_noweapon(sjaanat, jisala, cave)
43588.49	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	travel(rin, cave, forest)
43588.49	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creatureinjurescreature_noweapon(gargax, michael, village)
43588.49	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creaturebleedsout(michael)
43588.49	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	travel(rin, forest, cave)
43588.49	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creatureinjurescreature_noweapon(gargax, heather, village)
43588.49	20190503_114856	Chapter 5	guardian_hkillguardian1_01	creaturekillcreature_noweapon(heather, gargax, village)
43588.49	20190503_114856	Chapter 5	guardian_hkillguardian1_01	creaturelosesitematplace(heather, shield, village)
43588.49	20190503_114856	Chapter 6	questresolution_hgetsmcguffin_01	creaturegiveitemtocreature(sjaanat, rin, mysteriousorb, cave)
43588.49	20190503_114856	Chapter 6	questresolution_hgetsmcguffin_01	creatureinjurescreature_noweapon(jisala, sjaanat, cave)
43588.49	20190503_115104	Chapter 1	herodispatched_cgivesquest_01	travel(vincent, forest, village)
43588.49	20190503_115104	Chapter 1	herodispatched_cgivesquest_01	travel(michael, village, forest)
43588.49	20190503_115104	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(talia, michael, mysterioustome, forest)
43588.49	20190503_115104	Chapter 2	departure1_htravel1_01	travel(gargax, cave, forest)
43588.49	20190503_115104	Chapter 2	departure1_htravel1_01	travel(michael, forest, cave)
43588.49	20190503_115104	Chapter 3	departure2_htravel2_01	travel(michael, cave, forest)
43588.49	20190503_115104	Chapter 3	departure2_htravel2_01	delegate_getquest(gargax, talia, mysterioustome, forest)
43588.49	20190503_115104	Chapter 4	guardian_hkillguardian1_01	travel(gargax, forest, cave)

FIGURE 4.10: Outline output excerpt



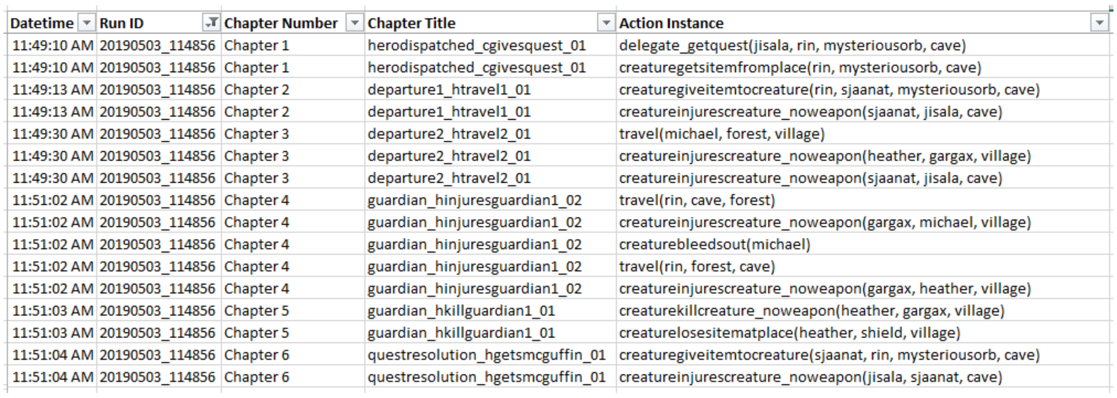
Datetime	Run ID	Chapter Numbe	Chapter Title	Action Instance
43588.49	20190503_115104	Chapter 1	herodispatched_cgivesquest_01	travel(vincent, forest, village)
43588.49	20190503_115104	Chapter 1	herodispatched_cgivesquest_01	travel(michael, village, forest)
43588.49	20190503_115104	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(talia, michael, mysterioustome, forest)
43588.49	20190503_115104	Chapter 2	departure1_htravel1_01	travel(gargax, cave, forest)
43588.49	20190503_115104	Chapter 2	departure1_htravel1_01	travel(michael, forest, cave)
43588.49	20190503_115104	Chapter 3	departure2_htravel2_01	travel(michael, cave, forest)
43588.49	20190503_115104	Chapter 3	departure2_htravel2_01	delegate_getquest(gargax, talia, mysterioustome, forest)
43588.49	20190503_115104	Chapter 4	guardian_hkillguardian1_01	travel(gargax, forest, cave)
43588.49	20190503_115104	Chapter 4	guardian_hkillguardian1_01	creaturekillcreature_noweapon(gargax, sjaanat, cave)
43588.50	20190503_115104	[!!!] ERROR: No solutions found after (1/1) tries.		
43588.50	20190503_115346	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(kyle, adolin, mysterioustome, village)
43588.50	20190503_115346	Chapter 1	herodispatched_cgivesquest_01	creaturegetsitemfromplace(adolin, mysterioustome, village)
43588.50	20190503_115346	Chapter 2	departure1_htravel1_01	travel(adolin, village, forest)
43588.50	20190503_115346	Chapter 2	departure1_htravel1_01	creaturegetsitemfromplace(bob, mysteriousorb, forest)
43588.50	20190503_115346	Chapter 3	departure2_htravel2_01	travel(adolin, forest, village)
43588.50	20190503_115346	Chapter 3	departure2_htravel2_01	creaturekillcreature_noweapon(kyle, adolin, village)
43588.50	20190503_115346	Chapter 3	departure2_htravel2_01	creaturegetsitemfromplace(sjaanat, silver, village)
43588.50	20190503_115346	Chapter 4	guardian_hkillguardian1_01	creaturekillcreature_noweapon(sjaanat, gargax, village)
43588.50	20190503_115346	Chapter 4	guardian_hkillguardian1_01	creaturegetsitemfromplace(bob, improvisedweapon1, forest)
43588.50	20190503_115346	[!!!] ERROR: No solutions found after (1/1) tries.		
43588.50	20190503_120505	Chapter 1	herodispatched_cgivesquest_01	creaturekillcreature_noweapon(jisala, vin, cave)
43588.50	20190503_120505	Chapter 1	herodispatched_cgivesquest_01	travel(gargax, forest, village)
43588.50	20190503_120505	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(gargax, adolin, mysteriousorb, village)
43588.50	20190503_120505	Chapter 2	departure1_htravel1_01	travel(adolin, village, cave)

FIGURE 4.11: Outline unsuccessful output

The primary story-line progresses on chapters 1,2,3,5, and 6 like so:

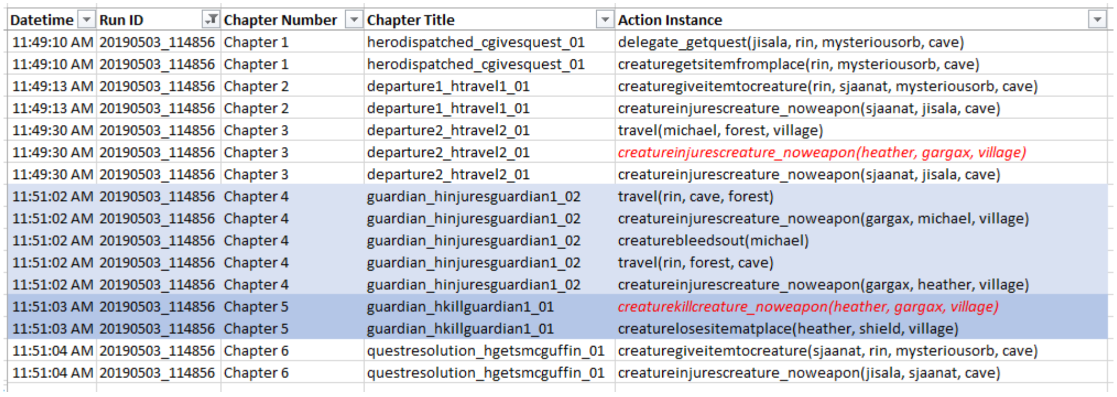
Run ID 20190503_114856: Primary Story-line
Chapter 1: *HeroDispatched*
  Rin obtains quest from Jisala
  Rin obtains MysteriousOrb
Chapter 2: *pre-QuestResolution*
  Rin loses MysteriousOrb to Sja'anat
Chapter 3: *Guardian (injure)* (4.3)
  Heather injures Gargax (the Guardian) in the Village
Chapter 5: *Guardian (kill)*
  Heather kills Gargax in the Village
Chapter 6: *QuestResolution*
  Sja'anat gives the MysteriousOrb to Rin

Since ISLA assembles the story using actions from different actors with potentially different intentions, it is observed that some narrative sub-goals *can* be satisfied by actions performed by a different actor from the actor with the goal *intent*. For example, the story sub-goals Guardian (injure)[1] and Guardian (kill)[2] were fulfilled by Heather (a supporting actor) and not the main hero actor, *Rin*. When the narrative reached outline Chapter 4, ISLA was looking to fulfill the intention *intends(rin, isinjured(gargax))*. However, that state predicate is already true, thanks to the action *creatureinjurescreature_noweapon(heather, gargax, village)* applied on outline Chapter 3.



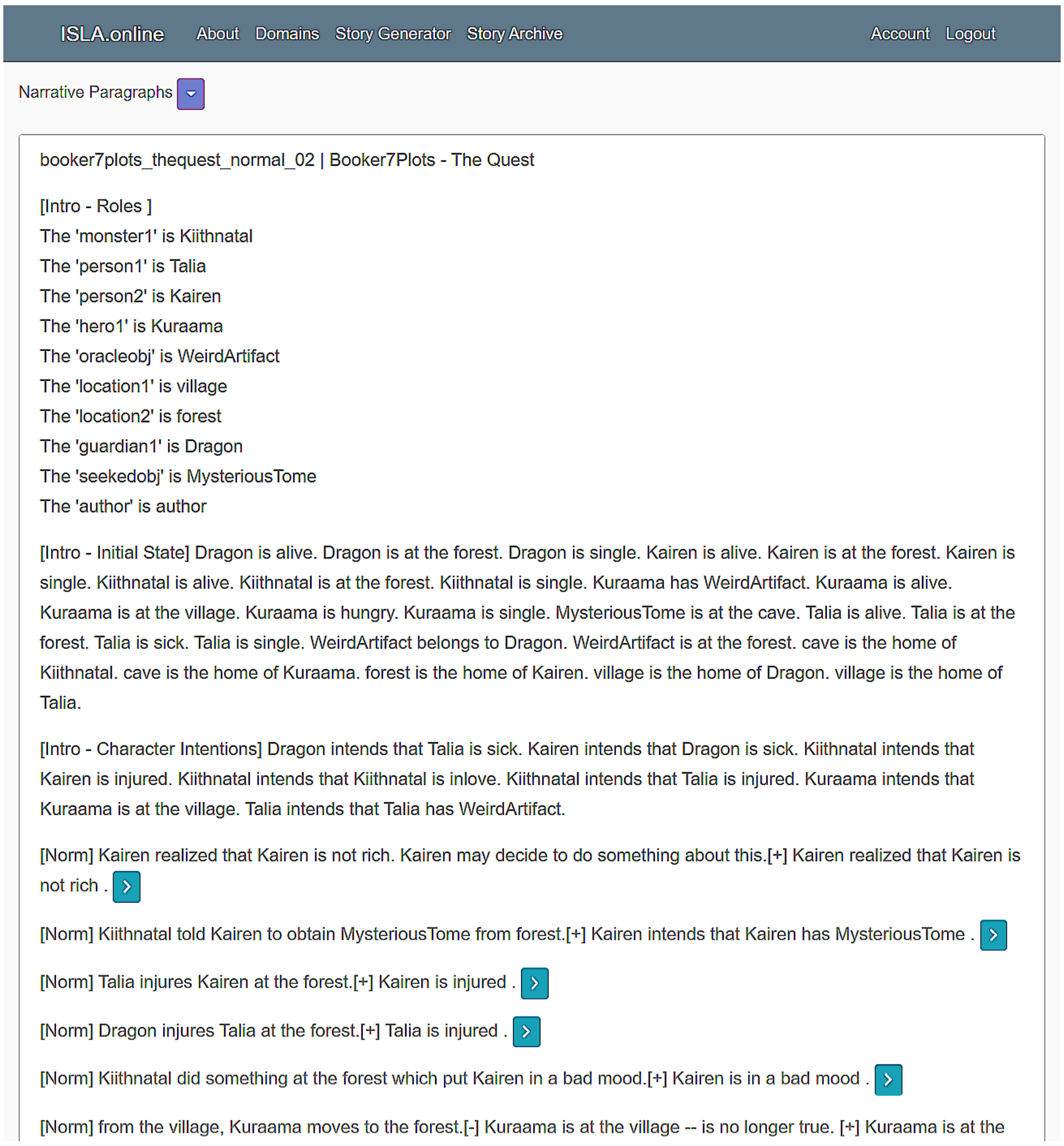
Datetime	Run ID	Chapter Number	Chapter Title	Action Instance
11:49:10 AM	20190503_114856	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(jisala, rin, mysteriousorb, cave)
11:49:10 AM	20190503_114856	Chapter 1	herodispatched_cgivesquest_01	creaturegetsitemfromplace(rin, mysteriousorb, cave)
11:49:13 AM	20190503_114856	Chapter 2	departure1_htravel1_01	creaturegiveitemtocreature(rin, sjaanat, mysteriousorb, cave)
11:49:13 AM	20190503_114856	Chapter 2	departure1_htravel1_01	creatureinjurescreature_noweapon(sjaanat, jisala, cave)
11:49:30 AM	20190503_114856	Chapter 3	departure2_htravel2_01	travel(michael, forest, village)
11:49:30 AM	20190503_114856	Chapter 3	departure2_htravel2_01	creatureinjurescreature_noweapon(heather, gargax, village)
11:49:30 AM	20190503_114856	Chapter 3	departure2_htravel2_01	creatureinjurescreature_noweapon(sjaanat, jisala, cave)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	travel(rin, cave, forest)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creatureinjurescreature_noweapon(gargax, michael, village)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creaturebleedsout(michael)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	travel(rin, forest, cave)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creatureinjurescreature_noweapon(gargax, heather, village)
11:51:03 AM	20190503_114856	Chapter 5	guardian_hkillguardian1_01	creaturekillcreature_noweapon(heather, gargax, village)
11:51:03 AM	20190503_114856	Chapter 5	guardian_hkillguardian1_01	creaturelosesitematplace(heather, shield, village)
11:51:04 AM	20190503_114856	Chapter 6	questresolution_hgetsmcguffin_01	creaturegiveitemtocreature(sjaanat, rin, mysteriousorb, cave)
11:51:04 AM	20190503_114856	Chapter 6	questresolution_hgetsmcguffin_01	creatureinjurescreature_noweapon(jisala, sjaanat, cave)

FIGURE 4.12: Outline success example



Datetime	Run ID	Chapter Number	Chapter Title	Action Instance
11:49:10 AM	20190503_114856	Chapter 1	herodispatched_cgivesquest_01	delegate_getquest(jisala, rin, mysteriousorb, cave)
11:49:10 AM	20190503_114856	Chapter 1	herodispatched_cgivesquest_01	creaturegetsitemfromplace(rin, mysteriousorb, cave)
11:49:13 AM	20190503_114856	Chapter 2	departure1_htravel1_01	creaturegiveitemtocreature(rin, sjaanat, mysteriousorb, cave)
11:49:13 AM	20190503_114856	Chapter 2	departure1_htravel1_01	creatureinjurescreature_noweapon(sjaanat, jisala, cave)
11:49:30 AM	20190503_114856	Chapter 3	departure2_htravel2_01	travel(michael, forest, village)
11:49:30 AM	20190503_114856	Chapter 3	departure2_htravel2_01	*creatureinjurescreature_noweapon(heather, gargax, village)*
11:49:30 AM	20190503_114856	Chapter 3	departure2_htravel2_01	creatureinjurescreature_noweapon(sjaanat, jisala, cave)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	travel(rin, cave, forest)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creatureinjurescreature_noweapon(gargax, michael, village)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creaturebleedsout(michael)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	travel(rin, forest, cave)
11:51:02 AM	20190503_114856	Chapter 4	guardian_hinjuresguardian1_02	creatureinjurescreature_noweapon(gargax, heather, village)
11:51:03 AM	20190503_114856	Chapter 5	guardian_hkillguardian1_01	*creaturekillcreature_noweapon(heather, gargax, village)*
11:51:03 AM	20190503_114856	Chapter 5	guardian_hkillguardian1_01	creaturelosesitematplace(heather, shield, village)
11:51:04 AM	20190503_114856	Chapter 6	questresolution_hgetsmcguffin_01	creaturegiveitemtocreature(sjaanat, rin, mysteriousorb, cave)
11:51:04 AM	20190503_114856	Chapter 6	questresolution_hgetsmcguffin_01	creatureinjurescreature_noweapon(jisala, sjaanat, cave)

FIGURE 4.13: Outline success example, with a sub-goal being achieved on a previous chapter



booker7plots_thequest_normal_02 | Booker7Plots - The Quest

[Intro - Roles]
The 'monster1' is Kiithnatal
The 'person1' is Talia
The 'person2' is Kairen
The 'hero1' is Kuraama
The 'oracleobj' is WeirdArtifact
The 'location1' is village
The 'location2' is forest
The 'guardian1' is Dragon
The 'seekedobj' is MysteriousTome
The 'author' is author

[Intro - Initial State] Dragon is alive. Dragon is at the forest. Dragon is single. Kairen is alive. Kairen is at the forest. Kairen is single. Kiithnatal is alive. Kiithnatal is at the forest. Kiithnatal is single. Kuraama has WeirdArtifact. Kuraama is alive. Kuraama is at the village. Kuraama is hungry. Kuraama is single. MysteriousTome is at the cave. Talia is alive. Talia is at the forest. Talia is sick. Talia is single. WeirdArtifact belongs to Dragon. WeirdArtifact is at the forest. cave is the home of Kiithnatal. cave is the home of Kuraama. forest is the home of Kairen. village is the home of Dragon. village is the home of Talia.

[Intro - Character Intentions] Dragon intends that Talia is sick. Kairen intends that Dragon is sick. Kiithnatal intends that Kairen is injured. Kiithnatal intends that Kiithnatal is inlove. Kiithnatal intends that Talia is injured. Kuraama intends that Kuraama is at the village. Talia intends that Talia has WeirdArtifact.

[Norm] Kairen realized that Kairen is not rich. Kairen may decide to do something about this.[+] Kairen realized that Kairen is not rich .

[Norm] Kiithnatal told Kairen to obtain MysteriousTome from forest.[+] Kairen intends that Kairen has MysteriousTome .

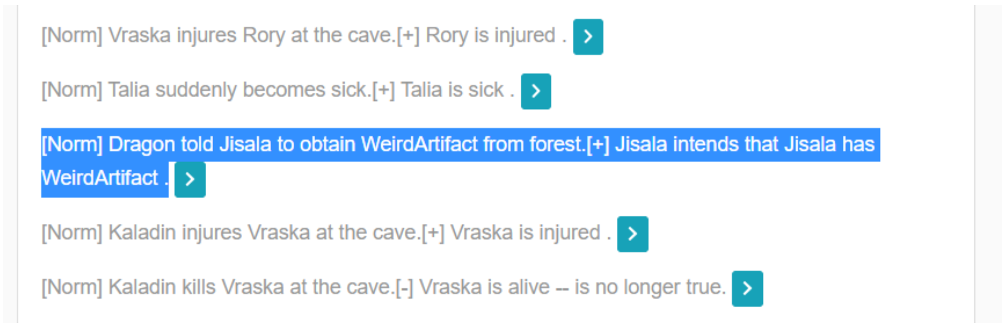
[Norm] Talia injures Kairen at the forest.[+] Kairen is injured .

[Norm] Dragon injures Talia at the forest.[+] Talia is injured .

[Norm] Kiithnatal did something at the forest which put Kairen in a bad mood.[+] Kairen is in a bad mood .

[Norm] from the village, Kuraama moves to the forest.[-] Kuraama is at the village -- is no longer true. [+] Kuraama is at the

FIGURE 4.14: The Quest - Narrative Paragraphs


[Norm] Vraska injures Rory at the cave.[+] Rory is injured .

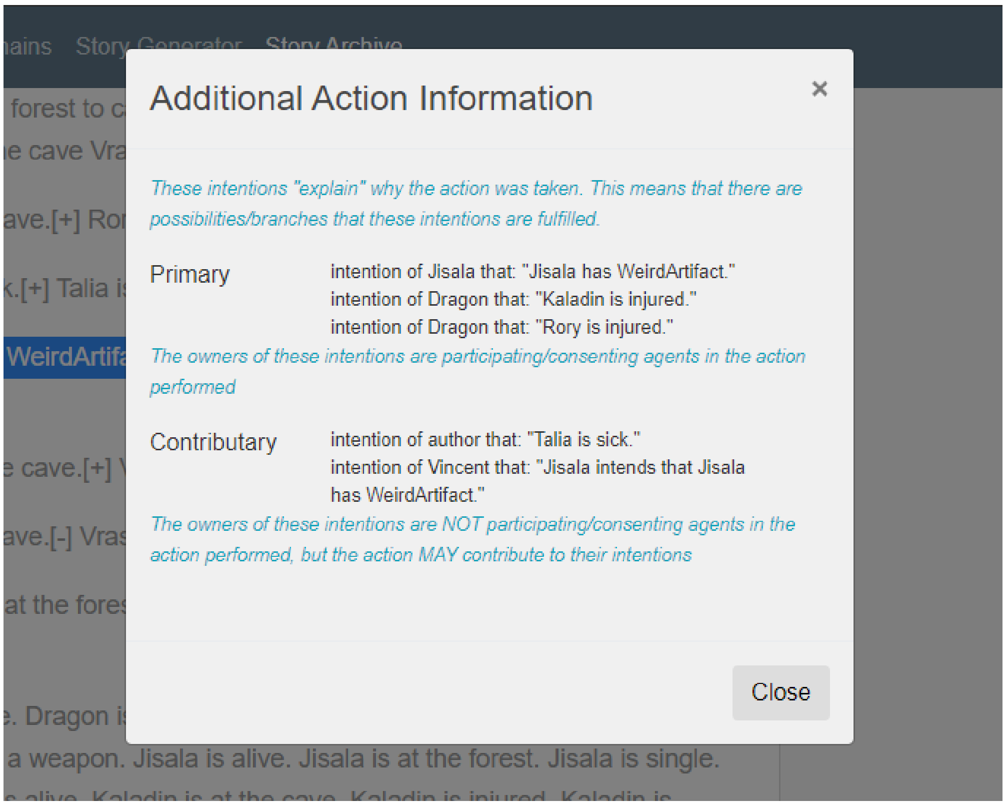
[Norm] Talia suddenly becomes sick.[+] Talia is sick .

[Norm] Dragon told Jisala to obtain WeirdArtifact from forest.[+] Jisala intends that Jisala has WeirdArtifact .

[Norm] Kaladin injures Vraska at the cave.[+] Vraska is injured .

[Norm] Kaladin kills Vraska at the cave.[-] Vraska is alive -- is no longer true.

FIGURE 4.15: The Quest - Narrative Paragraphs: Human readable action string



Additional Action Information

These intentions "explain" why the action was taken. This means that there are possibilities/branches that these intentions are fulfilled.

Primary	intention of Jisala that: "Jisala has WeirdArtifact." intention of Dragon that: "Kaladin is injured." intention of Dragon that: "Rory is injured."

The owners of these intentions are participating/consenting agents in the action performed

Contributary	intention of author that: "Talia is sick." intention of Vincent that: "Jisala intends that Jisala has WeirdArtifact."

The owners of these intentions are NOT participating/consenting agents in the action performed, but the action MAY contribute to their intentions

Close

FIGURE 4.16: The Quest - Narrative Paragraphs: Additional action information

4.1.1.3 Output Intentionality analysis

ISLA has an *unexplained threshold* of 51% during all test runs, meaning that solution outlines can have up to 51% of its steps *not* belonging to any intentional paths. Actions present in run ID 20190503_114856 outline (4.2) are *sufficiently* explained by the following intentions:

Run ID: 20190503_114856:

Narrative Intentions		
(jisala, **intends(rin, has(rin,mysteriousorb))**)	*HeroDispatched*	
(rin, at(rin, cave))	*Departure1*	
(rin, at(rin, cave))	*Departure2*[3]	
(rin, isinjured(gargax))	*Guardian* (injure)	
(rin, not(alive(gargax)))	*Guardian* (kill)	(4.4)
(rin, has(rin, mysteriousorb))	*QuestResolution*	
Random Intentions		
(sjaanat, has(sjaanat, mysteriousorb))	1	
(heather, isinjured(gargax))	2	
(heather, not(alive(gargax)))	3	

In order to help drive the story forward, ISLA assigns random intentions to some actors. Random intention 2 and 3 proved critical in both satisfying Rin's intention of injuring (and then killing) the Guardian Gargax, and adding narrative flavor and parallel sub-plots.

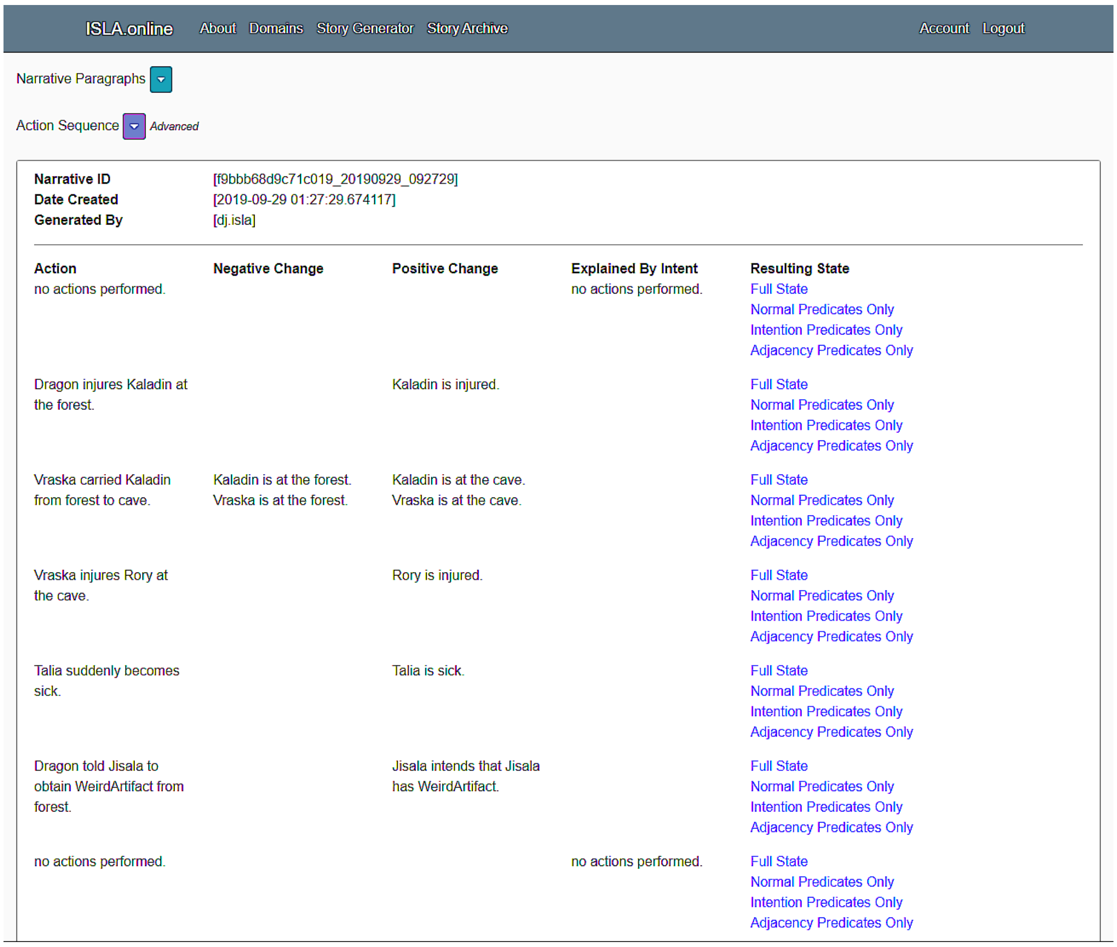
4.1.2 Chapter Chainer

Improvements were made to ISLA to properly leverage the chapter chainer module, and was outfitted more data processing layers and a basic user interface. These modifications have improved ISLA's output significantly by making data more human-readable and making key information, such as intentionality, visible to users. As such, narrative instances have two output modes: narrative paragraphs (fig. 4.14) and action sequence (fig. 4.17).

The narrative paragraphs mode is aimed to be closer to the format of the usual short story. ISLA converts the state predicates and action instances into "human readable strings". These are still defined manually during development of the knowledge-base elements. To give access to some of the underlying logic[4], the narrative paragraphs mode has a link that provides additional information in the form of a pop-up modal (figs. 4.15, 4.16).

[3]By chance, Departure1 and Departure2 have the same destination: the Cave. Also by chance, Rin is already at the Cave since outline Chapter 1

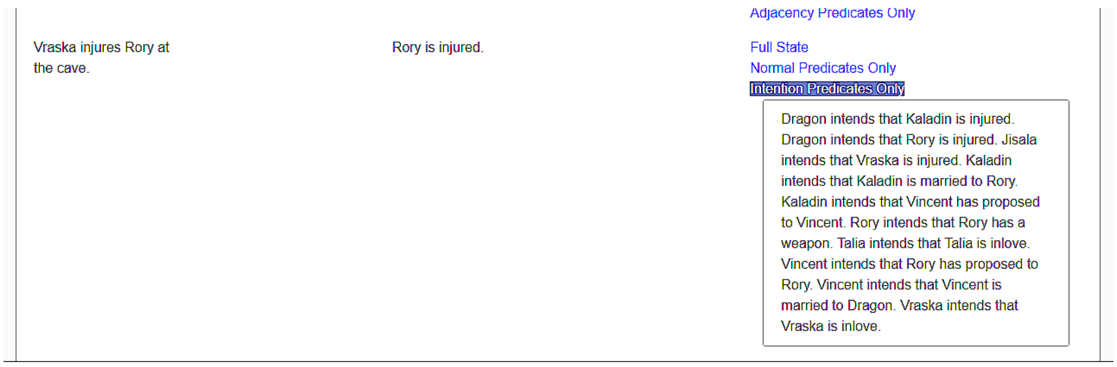
[4]As of writing time, only intentionality information is available for display.



Narrative ID	[f9bbb68d9c71c019_20190929_092729]
Date Created	[2019-09-29 01:27:29.674117]
Generated By	[dj.isla]

Action	Negative Change	Positive Change	Explained By Intent	Resulting State
no actions performed.			no actions performed.	Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only
Dragon injures Kaladin at the forest.		Kaladin is injured.		Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only
Vraska carried Kaladin from forest to cave.	Kaladin is at the forest. Vraska is at the forest.	Kaladin is at the cave. Vraska is at the cave.		Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only
Vraska injures Rory at the cave.		Rory is injured.		Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only
Talia suddenly becomes sick.		Talia is sick.		Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only
Dragon told Jisala to obtain WeirdArtifact from forest.		Jisala intends that Jisala has WeirdArtifact.		Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only
no actions performed.			no actions performed.	Full State Normal Predicates Only Intention Predicates Only Adjacency Predicates Only

FIGURE 4.17: The Quest - Action Sequence



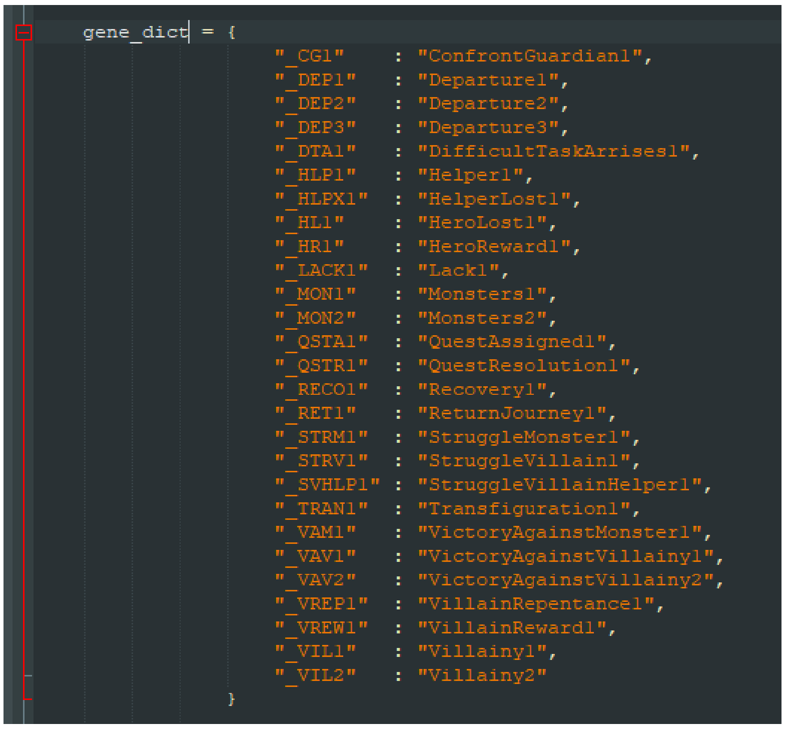
				Adjacency Predicates Only
Vraska injures Rory at the cave.		Rory is injured.		Full State Normal Predicates Only Intention Predicates Only Dragon intends that Kaladin is injured. Dragon intends that Rory is injured. Jisala intends that Vraska is injured. Kaladin intends that Kaladin is married to Rory. Kaladin intends that Vincent has proposed to Vincent. Rory intends that Rory has a weapon. Talia intends that Talia is inlove. Vincent intends that Rory has proposed to Rory. Vincent intends that Vincent is married to Dragon. Vraska intends that Vraska is inlove.

FIGURE 4.18: The Quest - Action Sequence: Additional state information

On the other hand, the action sequence mode displays mostly raw information. This mode shows the sequential steps/actions in the solution, and their effect on the narrative's state. Each action is accompanied with information on its respective negative[5] and positive change[6] to the current state.

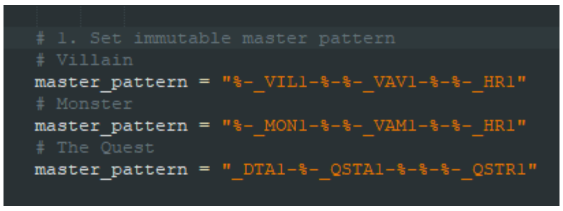
4.1.3 Story Pattern Refinement Using a Basic Genetic Algorithm

During early performance data gathering, it has been observed that many chapter patterns suffered from low success rate regardless of other factors (algorithm used and other settings). The hypothesis was that there may be more suitable story pattern sequences similar to these low-performing instances, and this subproblem may be treated as an optimization problem. Since story patterns may be represented as strings (fig. 4.19, fig. 4.21), a genetic algorithm (GA) may be used to tease out better story patterns from an arbitrary low-performing base pattern.



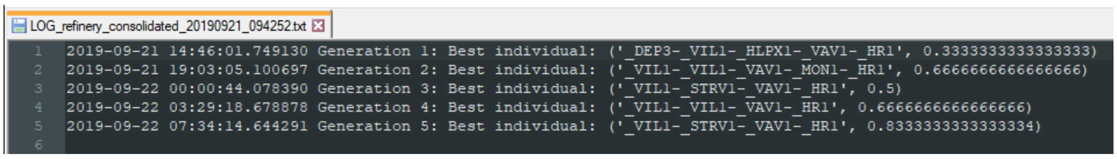
```
    gene_dict = {
                    "_CG1"    : "ConfrontGuardian1",
                    "_DEP1"   : "Departure1",
                    "_DEP2"   : "Departure2",
                    "_DEP3"   : "Departure3",
                    "_DTA1"   : "DifficultTaskArrises1",
                    "_HLP1"   : "Helper1",
                    "_HLPX1"  : "HelperLost1",
                    "_HL1"    : "HeroLost1",
                    "_HR1"    : "HeroReward1",
                    "_LACK1"  : "Lack1",
                    "_MON1"   : "Monsters1",
                    "_MON2"   : "Monsters2",
                    "_QSTA1"  : "QuestAssigned1",
                    "_QSTR1"  : "QuestResolution1",
                    "_RECO1"  : "Recovery1",
                    "_RET1"   : "ReturnJourney1",
                    "_STRM1"  : "StruggleMonster1",
                    "_STRV1"  : "StruggleVillain1",
                    "_SVHLP1" : "StruggleVillainHelper1",
                    "_TRAN1"  : "Transfiguration1",
                    "_VAM1"   : "VictoryAgainstMonster1",
                    "_VAV1"   : "VictoryAgainstVillainy1",
                    "_VAV2"   : "VictoryAgainstVillainy2",
                    "_VREP1"  : "VillainRepentance1",
                    "_VREW1"  : "VillainReward1",
                    "_VIL1"   : "Villainy1",
                    "_VIL2"   : "Villainy2"
                }
```

FIGURE 4.19: GA Refinement - Gene Dictionary



```
# 1. Set immutable master pattern
# Villain
master_pattern = "%-_VIL1-%-%-_VAV1-%-%-_HR1"
# Monster
master_pattern = "%-_MON1-%-%-_VAM1-%-%-_HR1"
# The Quest
master_pattern = "_DTA1-%-_QSTA1-%-%-%-_QSTR1"
```

FIGURE 4.20: GA Refinement - Immutable Patterns



```
2019-09-21 14:46:01.749130 Generation 1: Best individual: ('_DEP3-_VIL1-_HLPX1-_VAV1-_HR1', 0.3333333333333333)
2019-09-21 19:03:05.100697 Generation 2: Best individual: ('_VIL1-_VIL1-_VAV1-_MON1-_HR1', 0.6666666666666666)
2019-09-22 00:00:44.078390 Generation 3: Best individual: ('_VIL1-_STRV1-_VAV1-_HR1', 0.5)
2019-09-22 03:29:18.678878 Generation 4: Best individual: ('_VIL1-_VIL1-_VAV1-_HR1', 0.6666666666666666)
2019-09-22 07:34:14.644291 Generation 5: Best individual: ('_VIL1-_STRV1-_VAV1-_HR1', 0.8333333333333334)
```

FIGURE 4.21: GA Refinement - Gene Pattern Instances: OvercomeVillainy

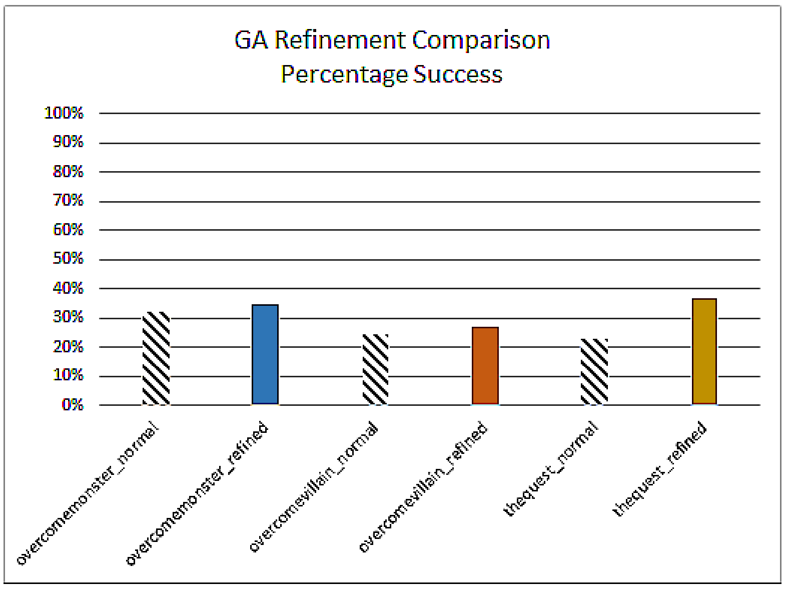


FIGURE 4.22: GA Refinement - Refined Patterns Performance Results

A simple genetic algorithm was used in the attempt to refine the three story patterns with the lowest performance. First, a gene dictionary was manually established (fig. 4.19). This represents all the current chapter patterns implemented in the active domain.

Second, the immutable patterns are also manually defined (fig. 4.20). These are the genes which cannot be changes during GA refinement. For example the immutable pattern for OvercomeVillainy is as follows:

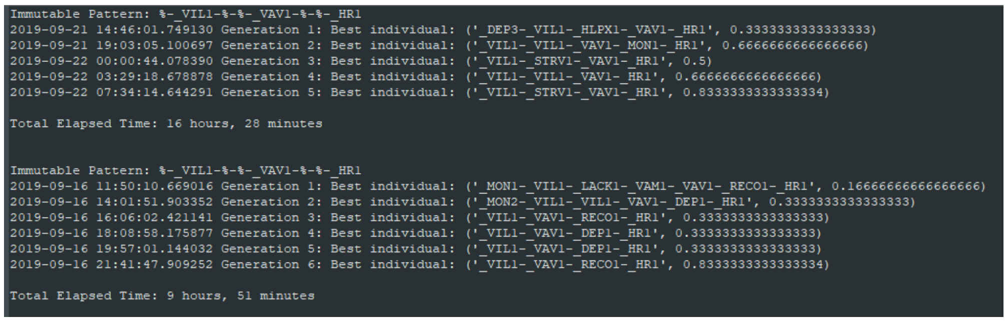
$$\%\text{-\_VIL1-}\%\text{-}\%\text{-\_VAV1-}\%\text{-}\%\text{-\_HR1} \tag{4.5}$$

The wildcard genes (percent symbols) and named genes are delimited by dashes. A wildcard means that there may exist 0-3 named terms in that space during the randomized "population individual" spawning. A named gene may not be mutated (changed into another random gene). The pivot point for splicing during crossover operation is the middle named gene. All immutable patterns have three named genes for consistency.

Third, the refinement process is initiated. An initial population is spawned using the immutable pattern as a base, with random genes from the gene dictionary is inserted in the wildcard spaces. The population is always kept at 6 individuals per generation. The individuals are subjected to 10 cycles of planning through ISLA's chapterchainer module, the success rate becomes the individual's fitness score. If an individual has passed the success threshold, the refinement process is terminated and the newly refined story pattern is saved as a file. Else, the top 3 individuals are chosen, and are put through the cross-over operation. Once the new population is derived, the mutation phase is initiated. Every individual has a small chance to mutate (5%). If an individual mutates, a random non-immune[7] gene will be changed to another random gene. The new population will then repeat the refinement process until a suitable individual emerges, or the generation threshold is reached.

$$
\begin{array}{ll}
\multicolumn{2}{c}{\textbf{GA Refinement: Cross-over Narrative Pairs}} \\
\textbf{Parents} & \\
 & I_{AA} : AaAb \\
 & I_{BA} : BaBb \\
 & I_{CC} : CaCb \\
\textbf{Cross-over pairs} & \\
 & I_{AA} \times I_{BB} = I_{AB} : AaBb \\
 & I_{BB} \times I_{CC} = I_{BC} : BaCb \\
 & I_{AA} \times I_{CC} = I_{AC} : AaCb \\
\textbf{Total Population} & \\
 & I_{AA} : AaAb \\
 & I_{BA} : BaBb \\
 & I_{CC} : CaCb \\
 & I_{AB} : AaBb \\
 & I_{BC} : BaCb \\
 & I_{AC} : AaCb
\end{array}
\tag{4.6}
$$

After refinement, the refined story patterns are re-tested in ISLA for success rate performance improvements. Unfortunately, after an extremely lengthy refinement process (fig. 4.23),



```
Immutable Pattern: %-_VIL1-%-%-_VAV1-%-%-_HR1
2019-09-21 14:46:01.749130 Generation 1: Best individual: ('_DEP3-_VIL1-_HLPX1-_VAV1-_HR1', 0.3333333333333333)
2019-09-21 19:03:05.100697 Generation 2: Best individual: ('_VIL1-_VIL1-_VAV1-_MON1-_HR1', 0.6666666666666666)
2019-09-22 00:00:44.078390 Generation 3: Best individual: ('_VIL1-_STRV1-_VAV1-_HR1', 0.5)
2019-09-22 03:29:18.678878 Generation 4: Best individual: ('_VIL1-_VIL1-_VAV1-_HR1', 0.6666666666666666)
2019-09-22 07:34:14.644291 Generation 5: Best individual: ('_VIL1-_STRV1-_VAV1-_HR1', 0.8333333333333334)

Total Elapsed Time: 16 hours, 28 minutes


Immutable Pattern: %-_VIL1-%-%-_VAV1-%-%-_HR1
2019-09-16 11:50:10.669016 Generation 1: Best individual: ('_MON1-_VIL1-_LACK1-_VAM1-_VAV1-_RECO1-_HR1', 0.16666666666666666)
2019-09-16 14:01:51.903352 Generation 2: Best individual: ('_MON2-_VIL1-_VIL1-_VAV1-_DEP1-_HR1', 0.3333333333333333)
2019-09-16 16:06:02.421141 Generation 3: Best individual: ('_VIL1-_VAV1-_RECO1-_HR1', 0.3333333333333333)
2019-09-16 18:08:58.175877 Generation 4: Best individual: ('_VIL1-_VAV1-_DEP1-_HR1', 0.3333333333333333)
2019-09-16 19:57:01.144032 Generation 5: Best individual: ('_VIL1-_VAV1-_DEP1-_HR1', 0.3333333333333333)
2019-09-16 21:41:47.909252 Generation 6: Best individual: ('_VIL1-_VAV1-_RECO1-_HR1', 0.8333333333333334)

Total Elapsed Time: 9 hours, 51 minutes
```

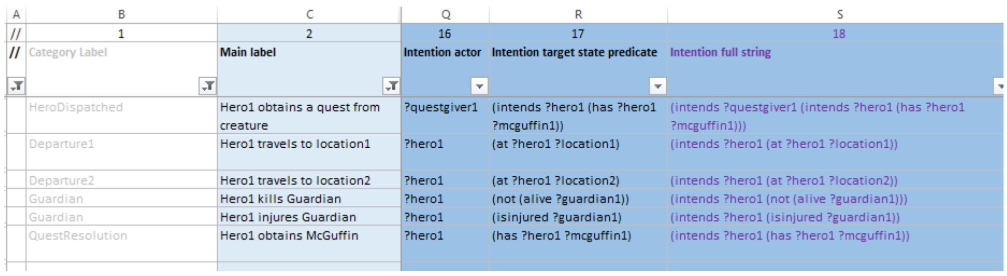
FIGURE 4.23: GA Refinement - Refinement Results

the success rate improvement is marginal at best (fig. 4.22). After this experiment, the conclusions garnered are as follows:

1. Story pattern sequence refinement process may be trumped by random favorable initial states, giving false positives during fitness evaluation.

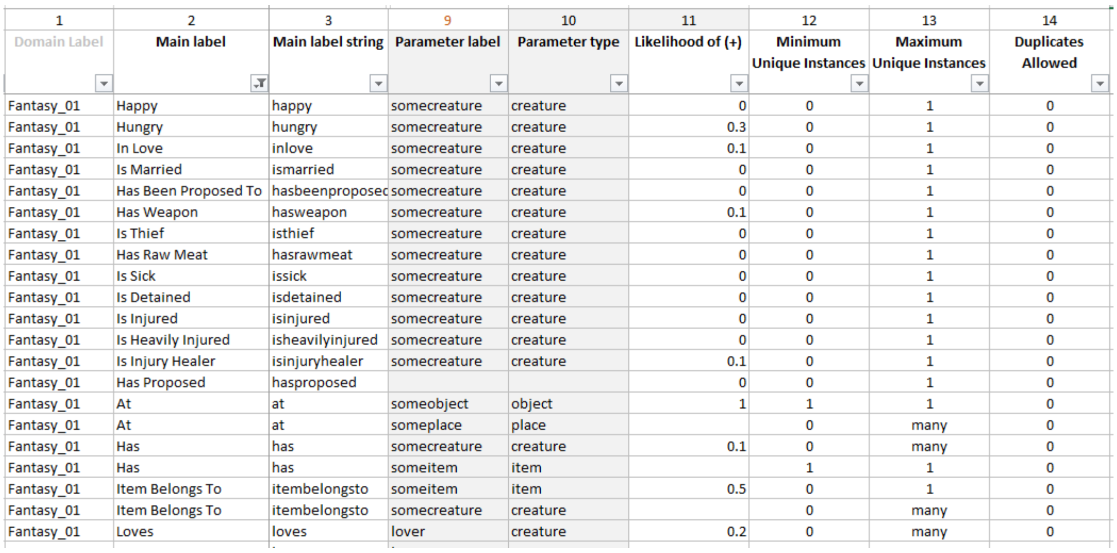
2. Story pattern sequence refinement process may be further improved by faster fitness evaluation techniques, standardized initial state generation for all population individuals, settings tweaking (success threshold, multiple final-refined individuals, population settings, etc.)

3. Initial state generation that considers the story pattern, as opposed to full random, may have a bigger positive impact to planning success rates.



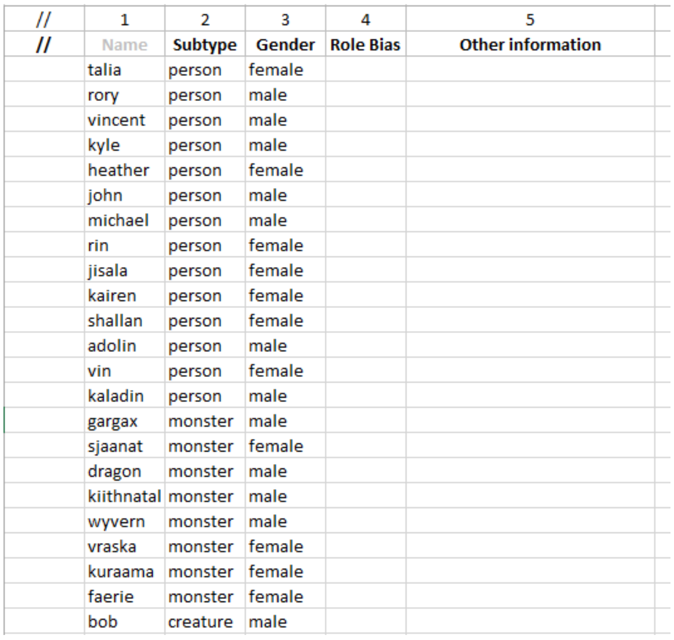
A	B	C	Q	R	S
//	1	2	16	17	18
//	Category Label	Main label	Intention actor	Intention target state predicate	Intention full string
	HeroDispatched	Hero1 obtains a quest from creature	?questgiver1	(intends ?hero1 (has ?hero1 ?mcguffin1))	(intends ?questgiver1 (intends ?hero1 (has ?hero1 ?mcguffin1)))
	Departure1	Hero1 travels to location1	?hero1	(at ?hero1 ?location1)	(intends ?hero1 (at ?hero1 ?location1))
	Departure2	Hero1 travels to location2	?hero1	(at ?hero1 ?location2)	(intends ?hero1 (at ?hero1 ?location2))
	Guardian	Hero1 kills Guardian	?hero1	(not (alive ?guardian1))	(intends ?hero1 (not (alive ?guardian1)))
	Guardian	Hero1 injures Guardian	?hero1	(isinjured ?guardian1)	(intends ?hero1 (isinjured ?guardian1))
	QuestResolution	Hero1 obtains McGuffin	?hero1	(has ?hero1 ?mcguffin1)	(intends ?hero1 (has ?hero1 ?mcguffin1))

FIGURE 4.24: Sequence Terms - defines individual intentions that will drive the story forward



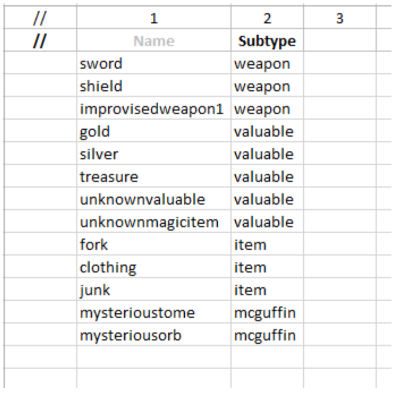
1	2	3	9	10	11	12	13	14
Domain Label	Main label	Main label string	Parameter label	Parameter type	Likelihood of (+)	Minimum Unique Instances	Maximum Unique Instances	Duplicates Allowed
Fantasy_01	Happy	happy	somecreature	creature	0	0	1	0
Fantasy_01	Hungry	hungry	somecreature	creature	0.3	0	1	0
Fantasy_01	In Love	inlove	somecreature	creature	0.1	0	1	0
Fantasy_01	Is Married	ismarried	somecreature	creature	0	0	1	0
Fantasy_01	Has Been Proposed To	hasbeenproposec	somecreature	creature	0	0	1	0
Fantasy_01	Has Weapon	hasweapon	somecreature	creature	0.1	0	1	0
Fantasy_01	Is Thief	isthief	somecreature	creature	0	0	1	0
Fantasy_01	Has Raw Meat	hasrawmeat	somecreature	creature	0	0	1	0
Fantasy_01	Is Sick	issick	somecreature	creature	0	0	1	0
Fantasy_01	Is Detained	isdetained	somecreature	creature	0	0	1	0
Fantasy_01	Is Injured	isinjured	somecreature	creature	0	0	1	0
Fantasy_01	Is Heavily Injured	isheavilyinjured	somecreature	creature	0	0	1	0
Fantasy_01	Is Injury Healer	isinjuryhealer	somecreature	creature	0.1	0	1	0
Fantasy_01	Has Proposed	hasproposed			0	0	1	0
Fantasy_01	At	at	someobject	object	1	1	1	0
Fantasy_01	At	at	someplace	place		0	many	0
Fantasy_01	Has	has	somecreature	creature	0.1	0	many	0
Fantasy_01	Has	has	someitem	item		1	1	0
Fantasy_01	Item Belongs To	itembelongsto	someitem	item	0.5	0	1	0
Fantasy_01	Item Belongs To	itembelongsto	somecreature	creature		0	many	0
Fantasy_01	Loves	loves	lover	creature	0.2	0	many	0

FIGURE 4.25: Predicate Descriptors



//	1	2	3	4	5
//	Name	Subtype	Gender	Role Bias	Other information
	talia	person	female		
	rory	person	male		
	vincent	person	male		
	kyle	person	male		
	heather	person	female		
	john	person	male		
	michael	person	male		
	rin	person	female		
	jisala	person	female		
	kairen	person	female		
	shallan	person	female		
	adolin	person	male		
	vin	person	female		
	kaladin	person	male		
	gargax	monster	male		
	sjaanat	monster	female		
	dragon	monster	male		
	kiithnatal	monster	male		
	wyvern	monster	male		
	vraska	monster	female		
	kuraama	monster	female		
	faerie	monster	female		
	bob	creature	male		

FIGURE 4.26: Creature name lookup



//	1	2	3
//	Name	Subtype	
	sword	weapon	
	shield	weapon	
	improvisedweapon1	weapon	
	gold	valuable	
	silver	valuable	
	treasure	valuable	
	unknownvaluable	valuable	
	unknownmagicitem	valuable	
	fork	item	
	clothing	item	
	junk	item	
	mysterioustome	mcguffin	
	mysteriousorb	mcguffin	

FIGURE 4.27: Item name lookup

4.2. Overhaul of the Coreplanner, Chapter Chainer, and GUI

4.2.1 Improvements to Coreplanner and Chapter Chainer

During the course of development, several algorithms were implemented for the coreplanner module: with goal graph paths (GP), directed random (DR), and goal graph paths - directed random hybrid (GPDR) during the initial phase, and goal graph multipaths (GMP) developed and implemented during the overhaul phase. The pre-overhaul algorithms seemed to work, but suffered from significant failure rates and high runtimes during executions that are bound to fail (see figure 4.29). This triggered the re-evaluation of the core codes and eventually led to the creation GMP (expounded on section 3.2.4.5). Comparative tests indicated that even though GGMP has a higher average execution time (fig. 4.28), the fact that it does not fail has proven that it is the most desirable algorithm that can (currently) be implemented by ISLA.

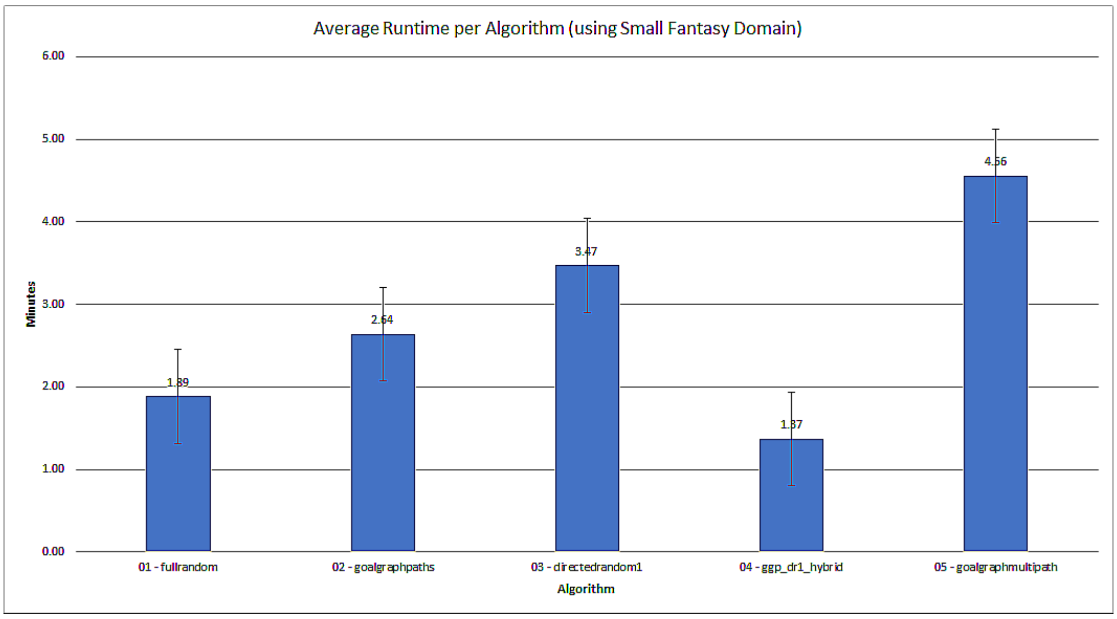


FIGURE 4.28: Average Runtime per Algorithm (using Small Fantasy Domain)

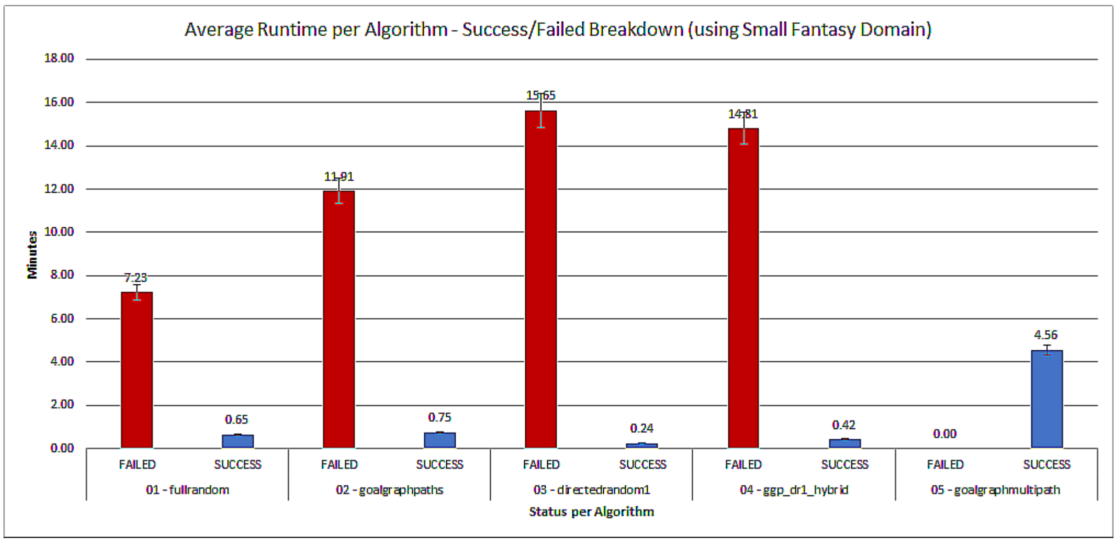


FIGURE 4.29: Average Runtime per Algorithm - Success-Failed Breakdown (using Small Fantasy Domain)

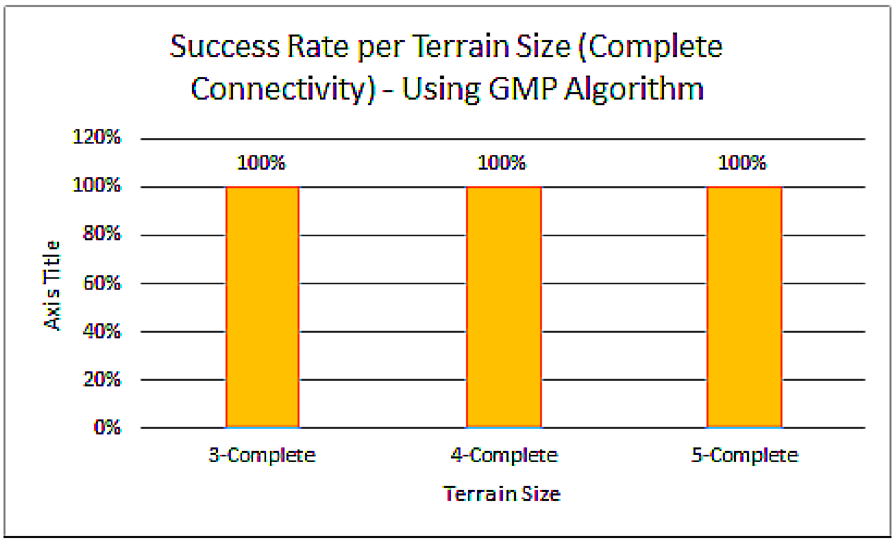


FIGURE 4.30: Success Rate (Complete-GMP)

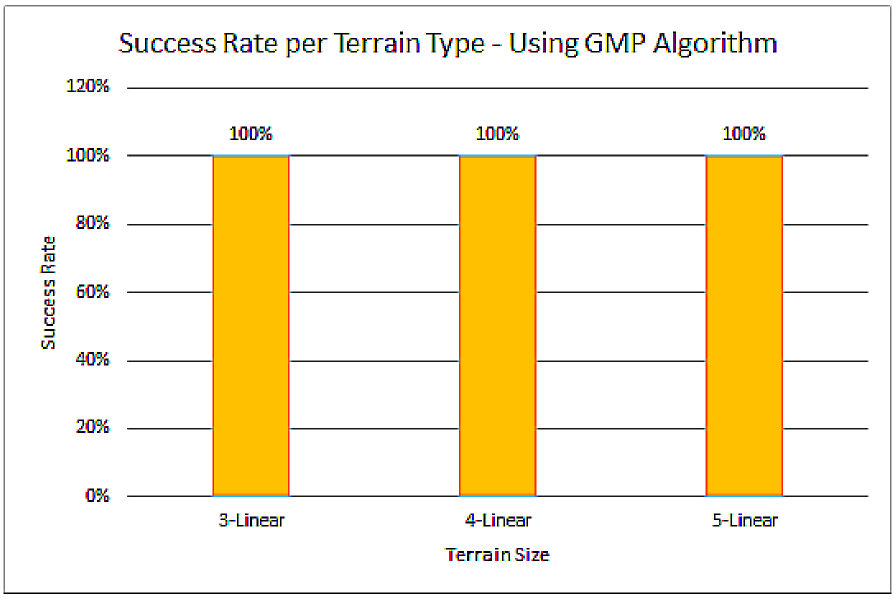


FIGURE 4.31: Success Rate (Linear-GMP)

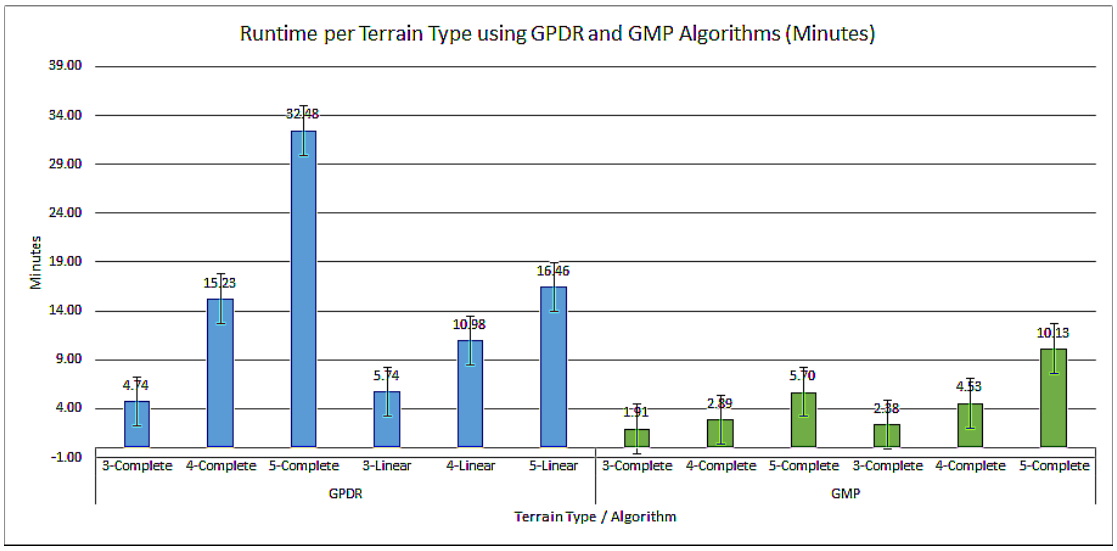


FIGURE 4.32: Runtime per Terrain Type using GPDR and GMP Algorithms (Minutes)

4.2.2 Using Tellability as a Measure

Complex plot units as explored by Berov [Ber17] and introduced by Lehnert [Leh81] closely resemble ISLA's *chapter patterns*; since chapter patterns are manually defined via the narrative structures tool, administrator users of ISLA have direct control over the tellability quality of the system's output. In doing so, administrator users *manually guarantee* a certain level of tellability to ISLA's output. This differs from GLAIVE, since GLAIVE's output does not consider the output's substructures (i.e. chapter patterns); even if the output is logically sensible based on intentionality and other factors, there is no guarantee that the output is tellable.

Another interesting point about tellability is that it is relative to specific chapter patterns. In other words, the output can be measured by comparing it to a known tellable pattern. For example, outputs are more tellable as 'fleeting success stories' than as 'tragendy' stories (see figure 4.33). In order to show this, chapter patterns (both for the base patterns for comparison, and ISLA's outline output) are represented as a string of characters using the following legend:

- **I**: a neutral event denoting the setting if an intention of a character
- **p**: a positive intention event; positively contributing to a character's intention
- **n**: a negative intention event; negatively contributing to a character's intention
- **+**: a positive general event; generally desirable event for a character that may or may not be related to any intention
- **-**: a negative general event; generally undesirable event for a character that may or may not be related to any intention
- **?**: a neutral/unknown/irrelevant general event

Known tellable patterns (can be arbitrary or based on externally evaluated patterns, of which is out of scope of this paper) are then set as standards which ISLA's output should measure up to (see x-axis of figure 4.33. ISLA's output are then converted into the same type of string and compared to the 'standard patterns' using a Javascript string similarity routine based on Dice's Coefficient. The higher the string similarity, the more "tellable as <standard pattern>"

the specific output is. Figure 4.33 shows that since the chapter pattern used for all 93 outputs are based on the "fleeting success" pattern, they have a generally similar pattern to the 3 variants of the fleeting success standard pattern. To highlight this, the outputs are also compared to two different standard patterns based on ’delayed success’ (multiple ’?’ events before a ’p’ event) and ’tragedy’ (multiple ’n’ and ’-’ events).

This initial result shows that ISLA’s output tellability quality can be directly controlled, although the measurement metrics can be improved upon by using better chapter pattern conversion mechanics and better (or more tailor-made) pattern similarity algorithms.

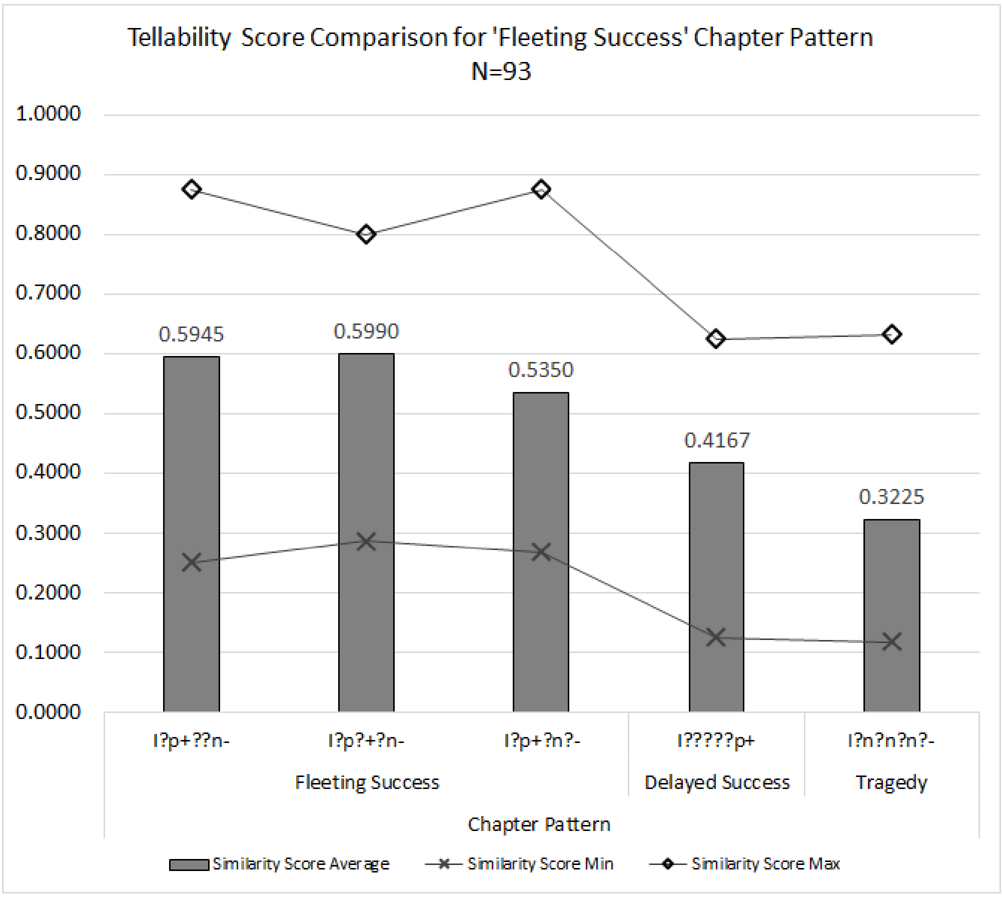


FIGURE 4.33: Tellability score comparison of 93 story outlines compared to 5 known tellable patterns.

4.2.3 GUI

Small improvements like parameter modification, improved add/edit/delete functionality, error correction are implemented to aid in the modification of existing domains and other narrative data structures. These changes vastly improve the quality of life during the experimentation phase as domain changes no longer need to go through manual processes which are clunky and prone to error.

Improvements on the GUI for the narrative instances are also implemented with the aim to expose underlying data structures to end-users.

4.3. Demonstration to Industry Professionals

In order to gauge ISLA's practical potential, demonstrations to a few industry professionals were held to showcase an earlier version of ISLA. This proceeded starting with a short presentation of ISLA's background and technical details, a live demonstration of ISLA's capability, and finally an open question and answer session in order to gather the attendees' sentiments. The following sections will highlight the results.

4.3.1 Survey Results (Industry Professionals)

Part 1: Average score, with a range between -3 and +3

1. Did you find ISLA easy to use? **0.33 - "Neutral"**
2. Did you find ISLA helpful in layouting stories? **1.33 - "Slightly Helpful"**
3. How sensible is this particular generated story? **1.33 - "Slightly Sensible"**

Part 2:Average score, with a range between 0 and 5

1. How helpful is ISLA as a tool for authors / creative writers? **3 - "Useful, but needs more improvements in many areas"**

2. How helpful is ISLA as a tool for game developers / game content creators? **2 - “ISLA has a tiny glimmer of potential”**

4.3.2 Interview Highlights

“[This story] was more interesting for me. But it might also be because I can visualize possible ’reasons’ behind the unfolding events more. Could be cause I favor the genre more.”

“Lots of potential with what you have so far. Lots of potential functions in different ways.”

Adjectives: arbitrary, dramatic, twisty, interesting, amusing, cute, structured, dull, objective, plain

Antonio Gabriel "Tobie" Abad IV
Creative Director, Head Game Designer of Taktyl Studios

—

“It seems harder to setup something than just write an outline then make a story from there. I do usually start with world building first, because the rules will always be established first before you can make stories out of it. I have a bit of difficulty using the tool. But it’s great to know that you can control scenarios and not just everything is randomly generated, so I think this tool is best used for game designers that concentrate on narratives. Just needs a bit of tweaking.”

Core suggestion: Fix UI

Dr. Beatrice Margarita V. Lapa
Professor at De La Salle-College of Saint Benilde, Senshi.Labs

—

“Great potential! Not for established authors, since they already have their own mental model for creating stories, but seems like a good tool for writers when used in writing exercises. ISLA looks useful for game development purposes.”

Inception: Inspired the idea to use ISLA as a teaching tool

Juan Karlo Licudine
Co-Founder and Lead Game Developer Mindcake Games

4.4. ISLA as a Teaching Tool Experiment

We conducted a simple exercise designed to measure ISLA in two aspects: viability of ISLA as a teaching tool, and sensibility and helpfulness of ISLA's narrative outline outputs. The participants are two teachers teaching high school English, with a total of 100 students between them, across 4 sections.

The experiment proceeded as follows.

1. The teachers (playing as the client) and myself (playing as the ISLA platform's roll-out team) conducted initial meetings to gather simple parameters in order to design the class exercise/s that will be deployed later on. Details such as "Which story genres are acceptable to use" and "Does your class require active technical support during the exercises" are agreed upon.
2. The roll-out team pre-generates narrative outlines using ISLA before the class exercises.
3. The roll-out team dispenses the exercise page URLs to the participating teacher and students, and briefs them on how to use ISLA
4. The class proceeds as normal, and the students providing the survey answers near the end of the class. These student surveys measure the sensibility and helpfulness of ISLA's narrative outline outputs
5. The client teacher then answers a separate survey form to evaluate the viability and usability of ISLA as a teaching tool

4.4.1 Survey Results (Students)

4.4.1.1 Score Highlights

The lowest scores that ISLA received are for the usability and GUI aspects. A significant portion of the student participants reported that the information displayed by ISLA seems to be overwhelming, "needs improvement as it was hard to navigate", "requires specific prerequisite knowledge to use". Total score: 0.33 - Neutral.

Majority of the participants have recognized the usefulness of ISLA's output as basis for more fleshed-out storied. Both questions 4 and 5 received decent overall marks (fig. C.1.6 and C.1.7 with 3.63 and 3.85 average respectively.

4.4.1.2 Feedback Highlights

"Although ISLA is not user friendly because it requires specific knowledge about codes, it is still helpful to students like me because it generated a story outline for me who cannot connect my own ideas when writing a story. The action/initial state generated can help the author in producing more ideas for their story yet the resulting state is confusing. Overall, the website is confusing as well as the terms used yet I see some potential."

"Personally, I think I would opt to use ISLA if I had more time to understand its features (for I only get the gist of it). I like how it gives the author a, somewhat, clear summary of the flow of their story and other possible events within the story."

"I think this is perfect for games that follow a storyline. I do not really see the potential of ISLA for professional writers."

4.4.2 Survey Results (Teachers)

ISLA is generally acceptable to the participating teachers, and have acknowledged ISLA's capacity as a teaching tool. Initial comments have described ISLA's impact as "time-saving" and "creates interest in the subject".

CHAPTER 5

FUTURE WORK

5.1. Functional Improvements

5.1.1 Problem extrapolation from sparse input

Pattern matching a user's limited input to ISLA's knowledge-base may yield better narrative results, as opposed to selecting from predefined chapter patterns. Context extraction from user input may yield a better experience when used as a stand-alone system or as a supporting component in a bigger ecosystem (e.g. a game).

5.1.2 NLP functions

Extending ISLA's capability may require concepts from natural language processing (NLP) and natural language generation (NLG) domains as detailed in McIntyre's work [McI11]. Applying such capabilities to ISLA will greatly increase the efficacy at which the output is understood, and will therefore be more useful to a wider range of audiences.

5.1.3 Machine Learning

Since it is possible to define both initial state, domain, and final state, it is interesting to employ supervised machine learning to help fine-tune internal algorithm junctions like those mentioned in [sec.3.1.2] and [sec.3.2.3.1]. The initial attempt to refine chapter sequences 4.1.3 is a step in the right direction; however, further research and experimentation is required.

5.1.4 Other Algorithmic Improvements

Implementation of character beliefs by applying and utilizing epistemic edges on the plan graph, initial state refinement, state consistency mechanics, and individual functional actor personalities may be looked into in order to improve the overall experience and performance of ISLA in the future.

5.2. Non-functional Improvements

5.2.1 Performance

A key hindrance in ISLA's further development is the extreme runtimes she experiences when performing planning operations. Several factors are being looked into to address this: parallel processing, programming language (compiled vs. interpreted), core code re-engineering and optimization.

5.2.2 Others

Output quality, extensibility, and usability are all aimed to be primary design concerns that needs to be considered also on succeeding incarnations of ISLA. Improving upon the GUI to be more user-friendly requires the assistance of knowledge from the UI/UX field, especially with issues such as conveying fundamentally technical information to a non-technical audience.

CHAPTER 6

CONCLUSIONS

A forward-chaining narrative planner, which supports intentionality, is constructed. This narrative planner, called the Intelligent Story Layout Assistant (ISLA), is able to construct story layouts that achieve the author's goals while making sure that *most* steps in the plan have clear motivations. These layouts, or solution plans, are based on a handcrafted knowledge-base of story universe elements. The current output is capable of displaying the underlying motivations of each action, and partial/complete state transition information.

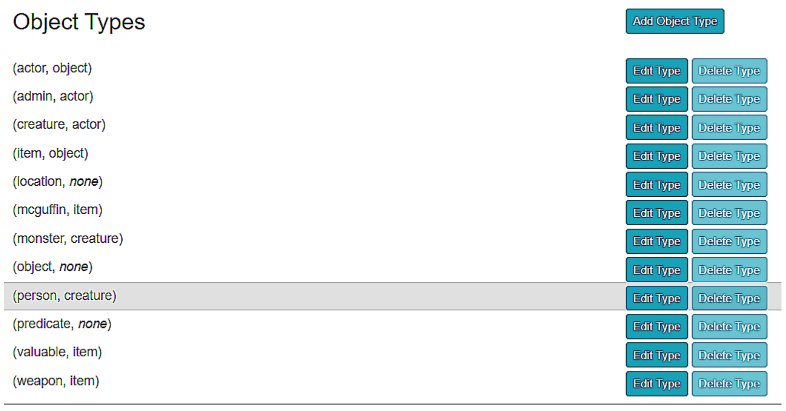
The knowledge-base can be improved by manually encoding more domains, and by refining knowledge-bases elements by employing machine learning techniques. Performance issues are obvious concerns. Compatibility with external systems will require further re-engineering and research.

In our analysis, ISLA's output proved sufficiently sensible on most cases, but has significant challenges particularly with performance, user-friendliness. ISLA's usefulness to professional writers have colloquially low score; on the other hand, ISLA is surprisingly useful in the realm of teaching as a tool to teach students how to write creative stories of their own. ISLA's narrative outline outputs performed decently on the student surveys even in the presence of the aforementioned difficulty of presenting fundamentally technical aspects of the narrative to a predominantly non-technical audience. ISLA currently has the *underlying* data structures needed to further assist the author in fleshing out the produced story layout. These have been exposed to some extent and has been recognized as helpful information for content authors.

APPENDIX A

GUI

A.1. Domain Objects, Predicates, and Actions


Object Types

(actor, object)
(admin, actor)
(creature, actor)
(item, object)
(location, *none*)
(mcguffin, item)
(monster, creature)
(object, *none*)
(person, creature)
(predicate, *none*)
(valuable, item)
(weapon, item)

FIGURE A.1.1: Domain Object Type list

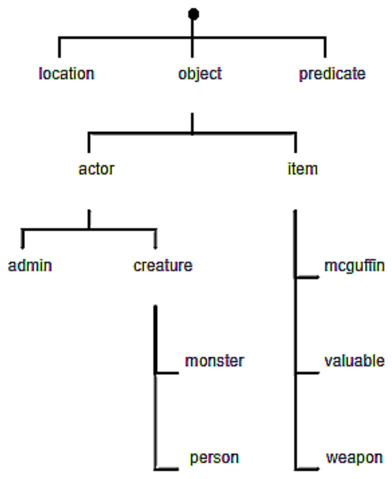
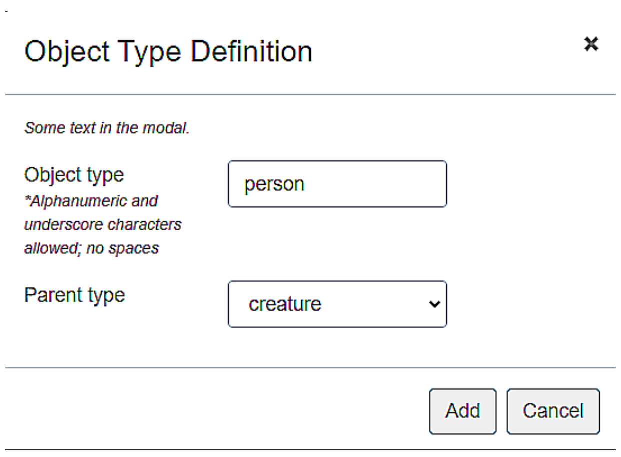


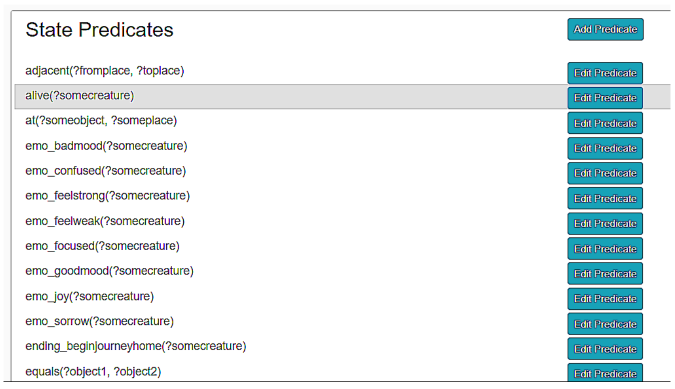
FIGURE A.1.2: Domain Object Type hierarchy display


Object Type Definition

Some text in the modal.

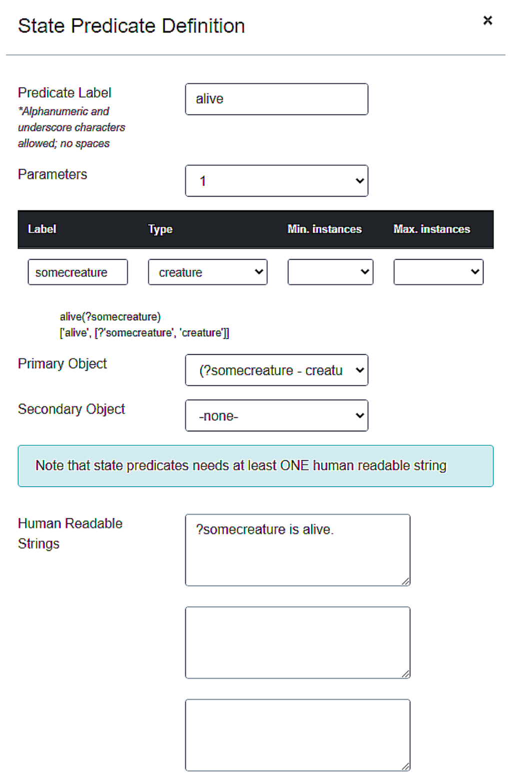
Object type **Alphanumeric and underscore characters allowed; no spaces*	person
Parent type	creature

FIGURE A.1.3: Domain Object Type definition


State Predicates

adjacent(?fromplace, ?toplace)
alive(?somecreature)
at(?someobject, ?someplace)
emo_badmood(?somecreature)
emo_confused(?somecreature)
emo_feelstrong(?somecreature)
emo_feelweak(?somecreature)
emo_focused(?somecreature)
emo_goodmood(?somecreature)
emo_joy(?somecreature)
emo_sorrow(?somecreature)
ending_beginjourneyhome(?somecreature)
equals(?object1, ?object2)

FIGURE A.1.4: Domain State Predicates list


State Predicate Definition

Predicate Label **Alphanumeric and underscore characters allowed; no spaces*	alive
Parameters	1

Label	Type	Min. instances	Max. instances
somecreature	creature		

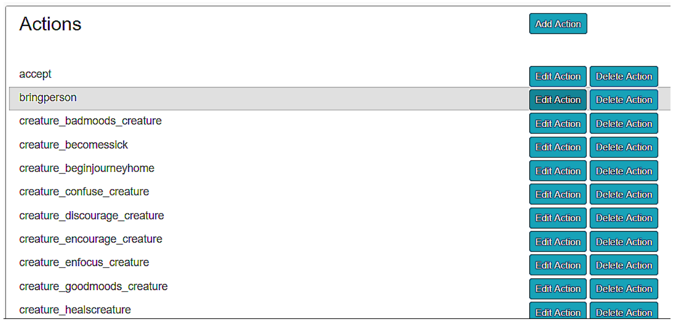
alive(?somecreature)
['alive', [?'somecreature', 'creature']]

Primary Object	(?somecreature - creatu
Secondary Object	-none-

Note that state predicates needs at least ONE human readable string

Human Readable Strings	?somecreature is alive.

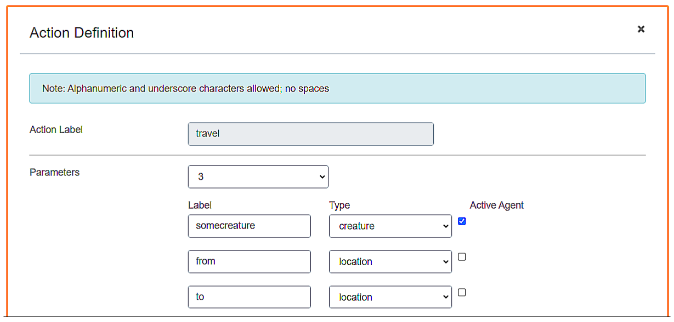
FIGURE A.1.5: Domain State Predicates definition interface



Actions

accept
bringperson
creature_badmoods_creature
creature_becomessick
creature_beginjourneyhome
creature_confuse_creature
creature_discourage_creature
creature_encourage_creature
creature_enfocus_creature
creature_goodmoods_creature
creature_healscreature

FIGURE A.1.6: Domain Actions list



Action Definition

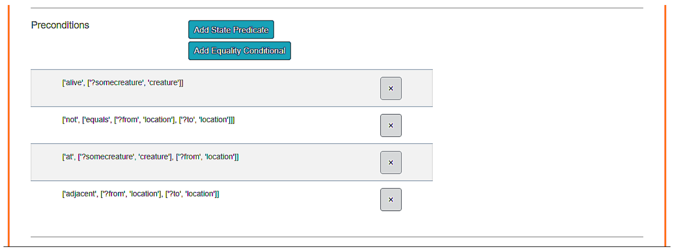
Note: Alphanumeric and underscore characters allowed; no spaces

Action Label: travel

Parameters: 3

Label	Type	Active Agent
somecreature	creature	☑
from	location	☐
to	location	☐

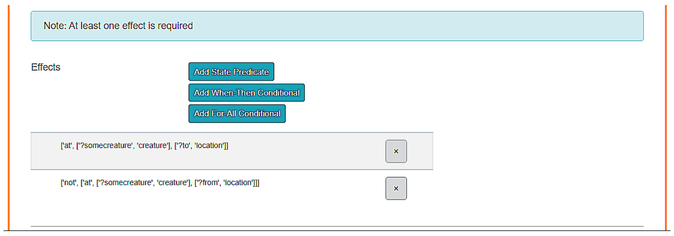
FIGURE A.1.7: Domain Action Definition - Parameters



Preconditions

['alive', ['?somecreature', 'creature']]
['not', ['equals', ['?from', 'location'], ['?to', 'location']]]
['at', ['?somecreature', 'creature'], ['?from', 'location']]
['adjacent', ['?from', 'location'], ['?to', 'location']]

FIGURE A.1.8: Domain Action Definition - Preconditions

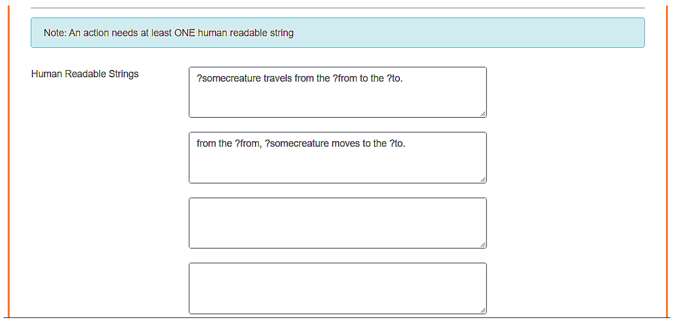


Note: At least one effect is required

Effects

['at', ['?somecreature', 'creature'], ['?to', 'location']]
['not', ['at', ['?somecreature', 'creature'], ['?from', 'location']]]

FIGURE A.1.9: Domain Action Definition - Effects



Note: An action needs at least ONE human readable string

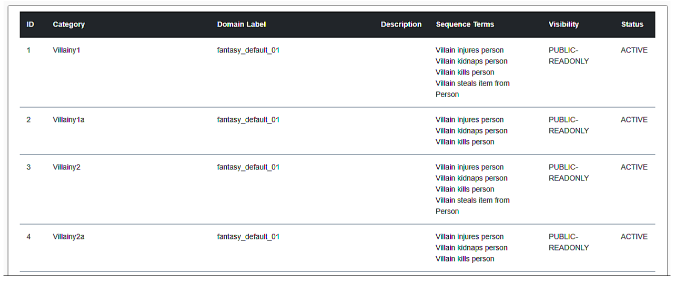
Human Readable Strings

?somecreature travels from the ?from to the ?to.

from the ?from, ?somecreature moves to the ?to.

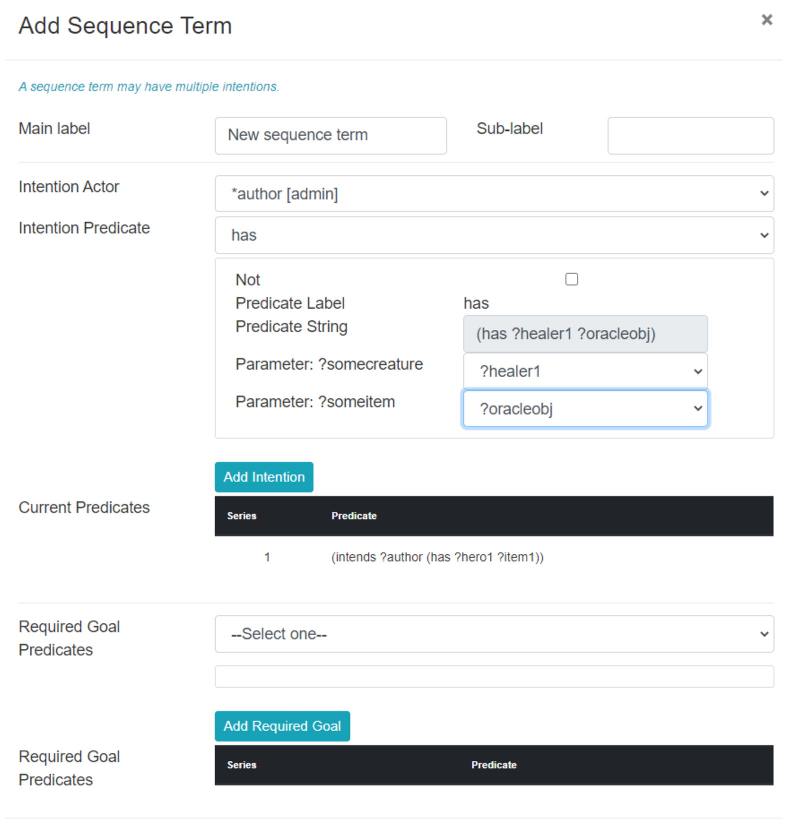
FIGURE A.1.10: Domain Action Definition - Human-readable strings

A.2. Narrative Structures



ID	Category	Domain Label	Description	Sequence Terms	Visibility	Status
1	Villainy1	fantasy_default_01		Villain injures person Villain kidnaps person Villain kills person Villain steals item from Person	PUBLIC-READONLY	ACTIVE
2	Villainy1a	fantasy_default_01		Villain injures person Villain kidnaps person Villain kills person	PUBLIC-READONLY	ACTIVE
3	Villainy2	fantasy_default_01		Villain injures person Villain kidnaps person Villain kills person Villain steals item from Person	PUBLIC-READONLY	ACTIVE
4	Villainy2a	fantasy_default_01		Villain injures person Villain kidnaps person Villain kills person	PUBLIC-READONLY	ACTIVE

FIGURE A.2.1: Narrative Structures - Sequence Terms Category list display



Add Sequence Term ×

A sequence term may have multiple intentions.

Main label	New sequence term	Sub-label	
Intention Actor	*author [admin]		
Intention Predicate	has		

Not	☐
Predicate Label	has
Predicate String	(has ?healer1 ?oracleobj)
Parameter: ?somecreature	?healer1
Parameter: ?someitem	?oracleobj

Add Intention

Current Predicates

Series	Predicate
1	(intends ?author (has ?hero1 ?item1))

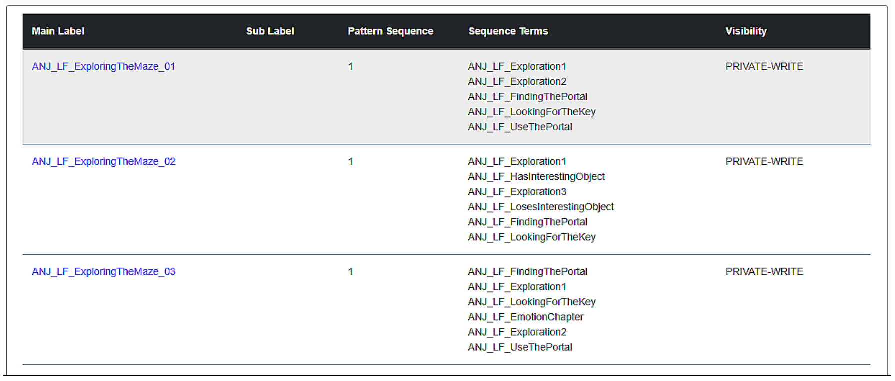
Required Goal Predicates: --Select one--

Add Required Goal

Required Goal Predicates

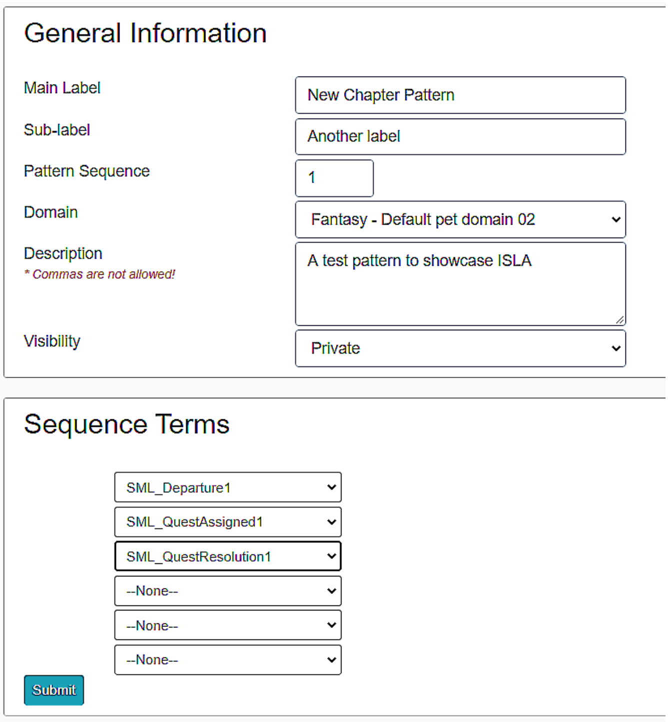
Series	Predicate

FIGURE A.2.2: Narrative Structures - Sequence Terms definition interface



Main Label	Sub Label	Pattern Sequence	Sequence Terms	Visibility
ANJ_LF_ExploringTheMaze_01		1	ANJ_LF_Exploration1 ANJ_LF_Exploration2 ANJ_LF_FindingThePortal ANJ_LF_LookingForTheKey ANJ_LF_UseThePortal	PRIVATE-WRITE
ANJ_LF_ExploringTheMaze_02		1	ANJ_LF_Exploration1 ANJ_LF_HasInterestingObject ANJ_LF_Exploration3 ANJ_LF_LosesInterestingObject ANJ_LF_FindingThePortal ANJ_LF_LookingForTheKey	PRIVATE-WRITE
ANJ_LF_ExploringTheMaze_03		1	ANJ_LF_FindingThePortal ANJ_LF_Exploration1 ANJ_LF_LookingForTheKey ANJ_LF_EmotionChapter ANJ_LF_Exploration2 ANJ_LF_UseThePortal	PRIVATE-WRITE

FIGURE A.2.3: Narrative Structures - Chapter Patterns list display



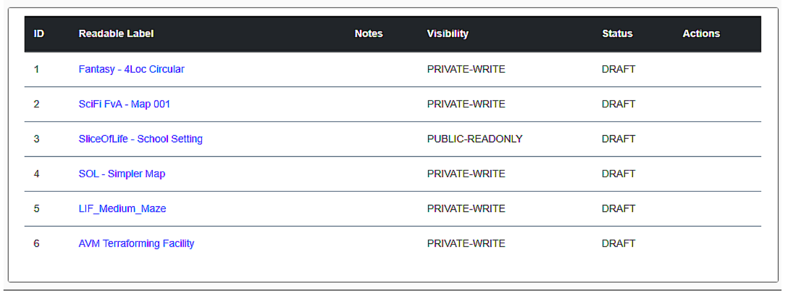
General Information

Main Label	New Chapter Pattern
Sub-label	Another label
Pattern Sequence	1
Domain	Fantasy - Default pet domain 02
Description * *Commas are not allowed!*	A test pattern to showcase ISLA
Visibility	Private

Sequence Terms

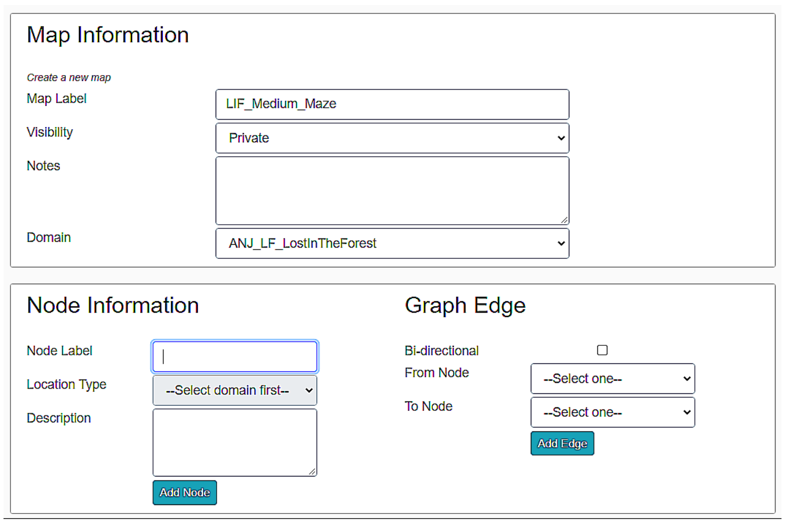
SML_Departure1
SML_QuestAssigned1
SML_QuestResolution1
--None--
--None--
--None--

FIGURE A.2.4: Narrative Structures - Chapter Patterns definition interface



ID	Readable Label	Notes	Visibility	Status	Actions
1	Fantasy - 4Loc Circular		PRIVATE-WRITE	DRAFT	
2	SciFi FvA - Map 001		PRIVATE-WRITE	DRAFT	
3	SliceOfLife - School Setting		PUBLIC-READONLY	DRAFT	
4	SOL - Simpler Map		PRIVATE-WRITE	DRAFT	
5	LIF_Medium_Maze		PRIVATE-WRITE	DRAFT	
6	AVM Terraforming Facility		PRIVATE-WRITE	DRAFT	

FIGURE A.2.5: Narrative Structures - Location Maps list display


Map Information

Create a new map

Map Label	LIF_Medium_Maze
Visibility	Private
Notes	
Domain	ANJ_LF_LostInTheForest

Node Information

Node Label	
Location Type	--Select domain first--
Description	

Graph Edge

Bi-directional	☐
From Node	--Select one--
To Node	--Select one--

FIGURE A.2.6: Narrative Structures - Location Map attributes editor

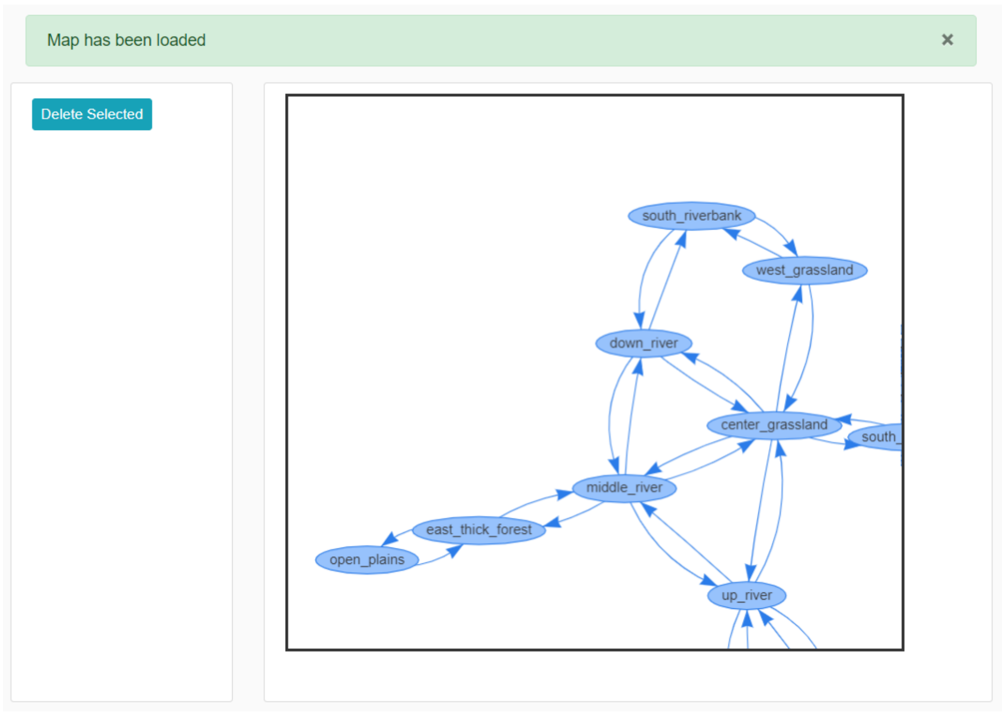
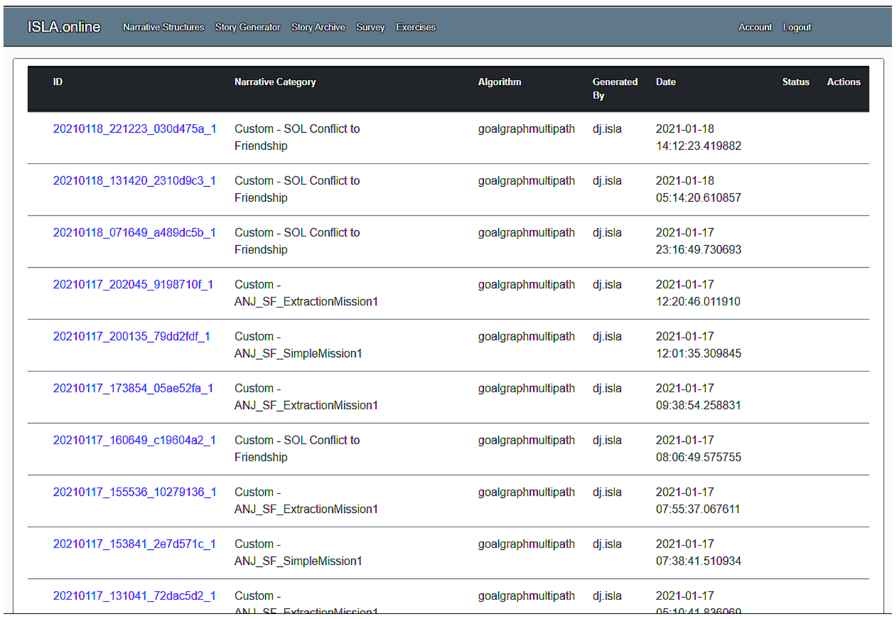

Map has been loaded

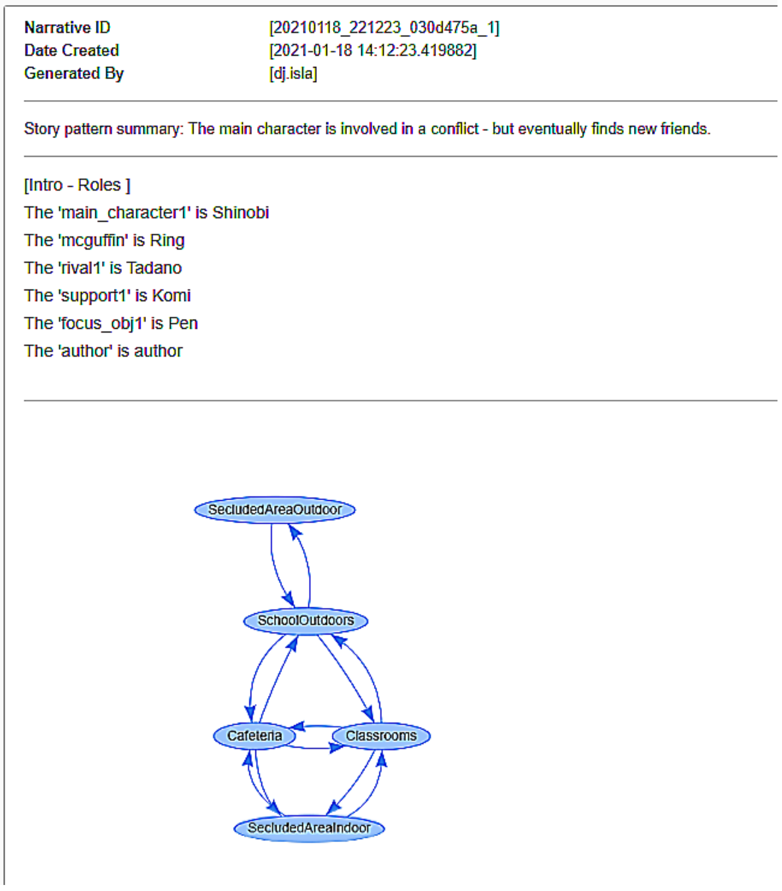
FIGURE A.2.7: Narrative Structures - Location Map visual display

A.3. Narrative Instances



ID	Narrative Category	Algorithm	Generated By	Date	Status	Actions
20210118_221223_030d475a_1	Custom - SOL Conflict to Friendship	goalgraphmultipath	dj.isla	2021-01-18 14:12:23.419882		
20210118_131420_2310d9c3_1	Custom - SOL Conflict to Friendship	goalgraphmultipath	dj.isla	2021-01-18 05:14:20.610857		
20210118_071649_a489dc5b_1	Custom - SOL Conflict to Friendship	goalgraphmultipath	dj.isla	2021-01-17 23:16:49.730693		
20210117_202045_9198710f_1	Custom - ANJ_SF_ExtractionMission1	goalgraphmultipath	dj.isla	2021-01-17 12:20:46.011910		
20210117_200135_79dd2fdf_1	Custom - ANJ_SF_SimpleMission1	goalgraphmultipath	dj.isla	2021-01-17 12:01:35.309845		
20210117_173854_05ae52fa_1	Custom - ANJ_SF_ExtractionMission1	goalgraphmultipath	dj.isla	2021-01-17 09:38:54.258831		
20210117_160649_c19604a2_1	Custom - SOL Conflict to Friendship	goalgraphmultipath	dj.isla	2021-01-17 08:06:49.575755		
20210117_155536_10279136_1	Custom - ANJ_SF_ExtractionMission1	goalgraphmultipath	dj.isla	2021-01-17 07:55:37.067611		
20210117_153841_2e7d571c_1	Custom - ANJ_SF_SimpleMission1	goalgraphmultipath	dj.isla	2021-01-17 07:38:41.510934		
20210117_131041_72dac5d2_1	Custom - ANJ_SF_ExtractionMission1	goalgraphmultipath	dj.isla	2021-01-17 05:10:41.836060		

FIGURE A.3.1: Story Archive - List display

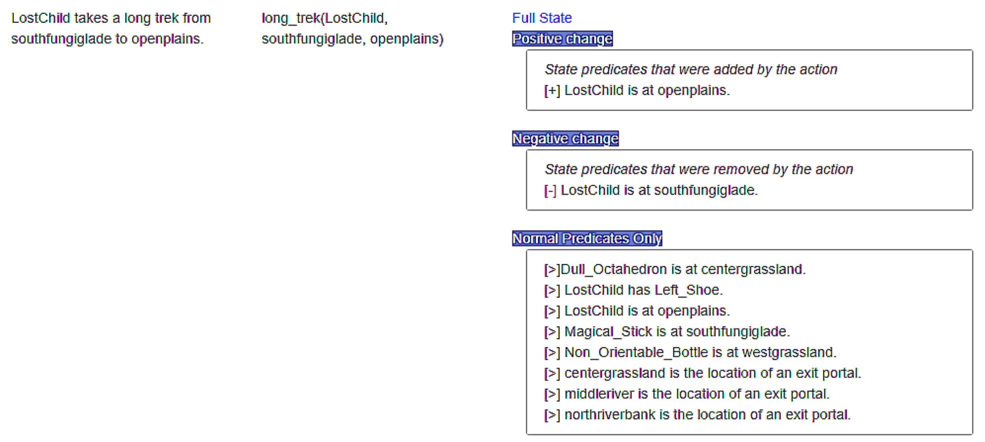


Narrative ID	[20210118_221223_030d475a_1]
Date Created	[2021-01-18 14:12:23.419882]
Generated By	[dj.isla]

Story pattern summary: The main character is involved in a conflict - but eventually finds new friends.

[Intro - Roles]
The 'main_character1' is Shinobi
The 'mcguffin' is Ring
The 'rival1' is Tadano
The 'support1' is Komi
The 'focus_obj1' is Pen
The 'author' is author

FIGURE A.3.2: Narrative Instance - Map display



LostChild takes a long trek from southfungiglade to openplains.	long_trek(LostChild, southfungiglade, openplains)	Full State Positive change *State predicates that were added by the action* [+] LostChild is at openplains. Negative change *State predicates that were removed by the action* [-] LostChild is at southfungiglade. Normal Predicates Only [>]Dull_Octahedron is at centergrassland. [>] LostChild has Left_Shoe. [>] LostChild is at openplains. [>] Magical_Stick is at southfungiglade. [>] Non_Orientable_Bottle is at westgrassland. [>] centergrassland is the location of an exit portal. [>] middleriver is the location of an exit portal. [>] northriverbank is the location of an exit portal.

FIGURE A.3.3: Narrative Instance - Additional state information

APPENDIX B

Demonstration to Industry Professionals

B.1. Survey Results

Part 1: Average score, with a range between -3 and +3

1. Did you find ISLA easy to use? **0.33 - "Neutral"**
2. Did you find ISLA easy to use? **0.33 - "Neutral"**
3. Did you find ISLA helpful in layouting stories? **1.33 - "Slightly Helpful"**
4. How sensible is this particular generated story? **1.33 - "Slightly Sensible"**

Part 2:Average score, with a range between 0 and 5

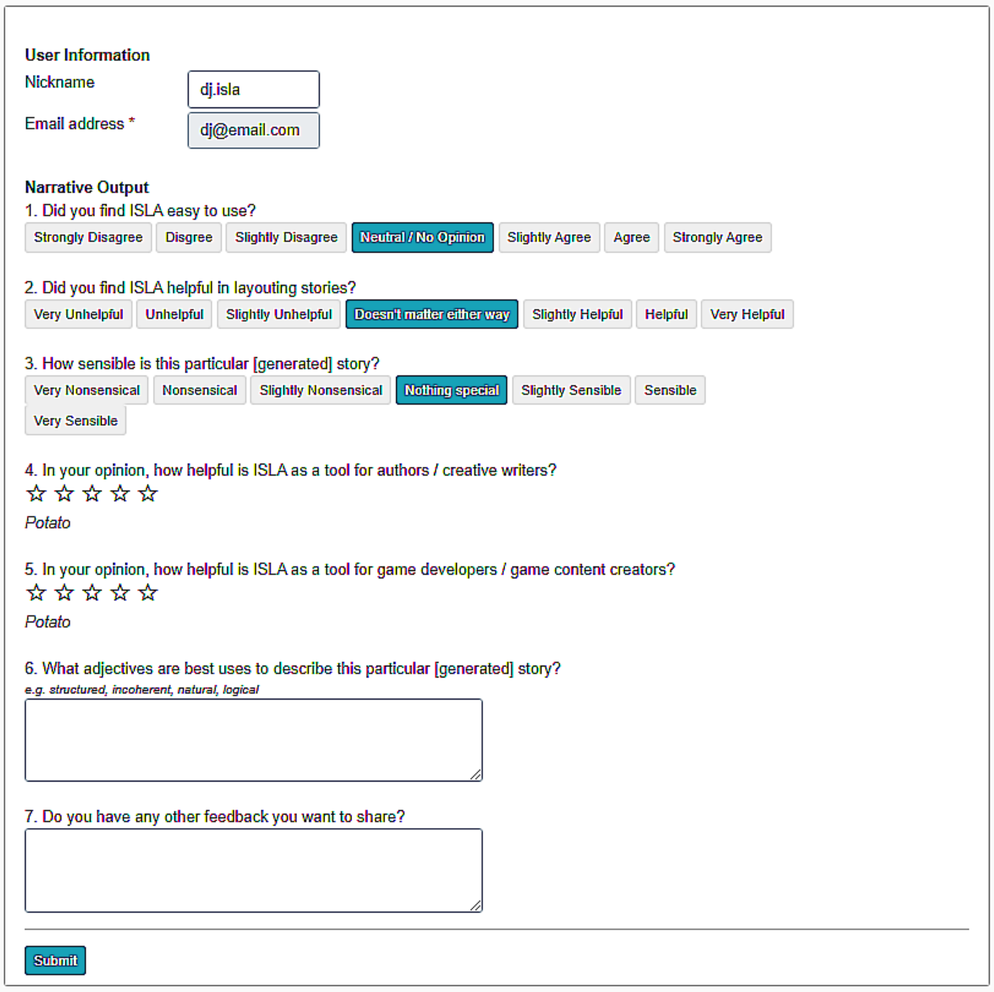
1. How helpful is ISLA as a tool for authors / creative writers? **3 - "Useful, but needs more improvements in many areas"**
2. How helpful is ISLA as a tool for game developers / game content creators? **2 - "ISLA has a tiny glimmer of potential"**

APPENDIX C

ISLA as a Teaching Tool

C.1. Student Surveys

C.1.1 Survey Forms



User Information

Nickname: dj.isla

Email address *: dj@email.com

Narrative Output

1. Did you find ISLA easy to use?

Strongly Disagree | Disgree | Slightly Disagree | [✓ Neutral / No Opinion] | Slightly Agree | Agree | Strongly Agree

2. Did you find ISLA helpful in layouting stories?

Very Unhelpful | Unhelpful | Slightly Unhelpful | [✓ Doesn't matter either way] | Slightly Helpful | Helpful | Very Helpful

3. How sensible is this particular [generated] story?

Very Nonsensical | Nonsensical | Slightly Nonsensical | [✓ Nothing special] | Slightly Sensible | Sensible | Very Sensible

4. In your opinion, how helpful is ISLA as a tool for authors / creative writers?

☆ ☆ ☆ ☆ ☆

Potato

5. In your opinion, how helpful is ISLA as a tool for game developers / game content creators?

☆ ☆ ☆ ☆ ☆

Potato

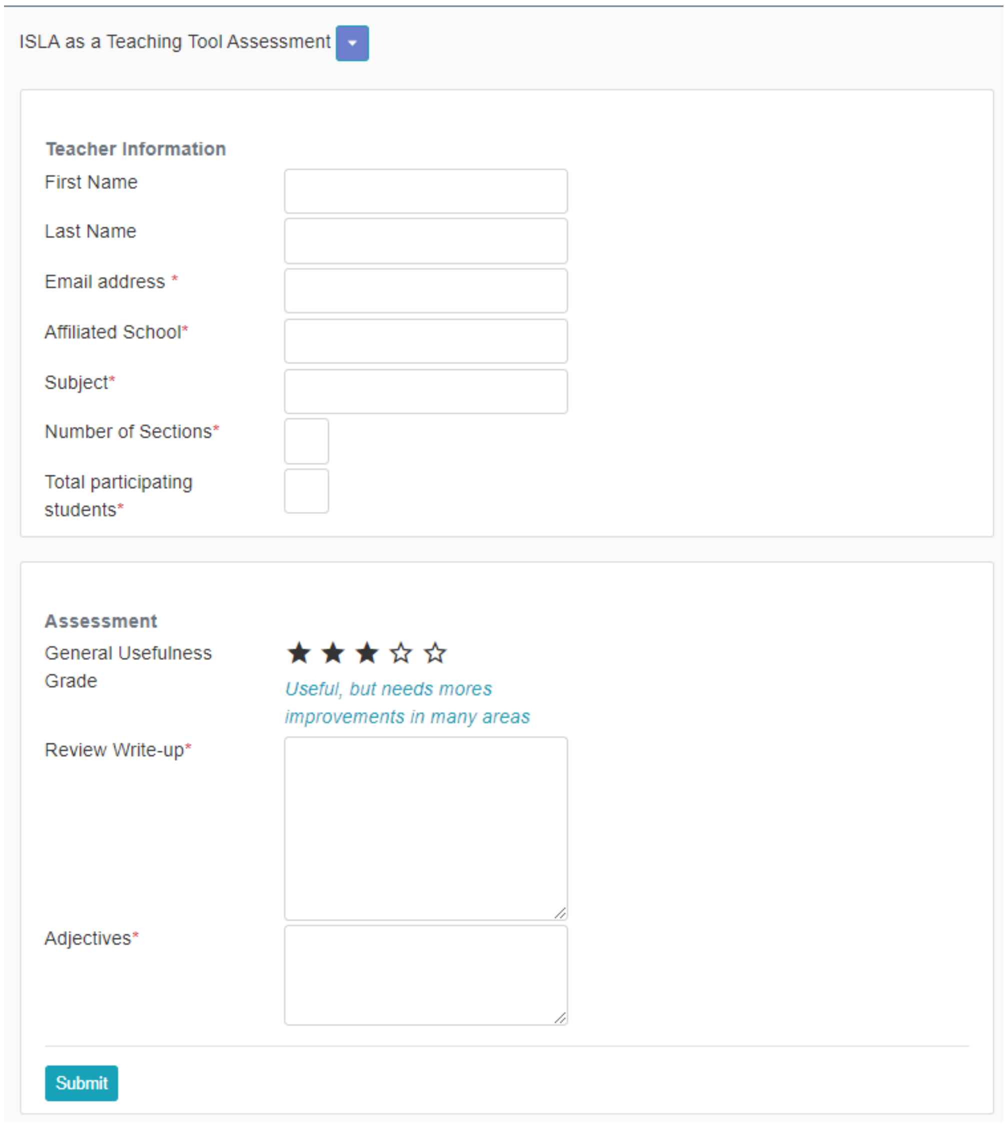
6. What adjectives are best uses to describe this particular [generated] story?

e.g. structured, incoherent, natural, logical

7. Do you have any other feedback you want to share?

Submit

FIGURE C.1.1: Narrative Instance - Survey to measure the quality of the story outline output


ISLA as a Teaching Tool Assessment

Teacher Information

First Name	
Last Name	
Email address *	
Affiliated School*	
Subject*	
Number of Sections*	
Total participating students*	

Assessment

General Usefulness Grade	★★★☆☆ *Useful, but needs mores improvements in many areas*
Review Write-up*	
Adjectives*	

FIGURE C.1.2: Teaching Tool Survey - Survey to measure the viability and usefulness of ISLA as a teaching tool

C.1.2 Survey Results

Average score, with a range between -3 and +3
N = 27

1. Did you find ISLA easy to use? **0.33: "Neutral"**
2. Did you find ISLA helpful in layouting stories? **1.30: "Slightly Helpful"**
3. How sensible is this particular generated story? **1.26: "Slightly Sensible"**

Average score, with a range between 0 and 5
N = 27

1. In your opinion, how helpful is ISLA as a tool for authors / creative writers? **3.63 - "Useful, but needs more improvements in many areas"**
2. In your opinion, how helpful is ISLA as a tool for game developers / game content creators? **3.85 - "Useful, but needs more improvements in many areas"**

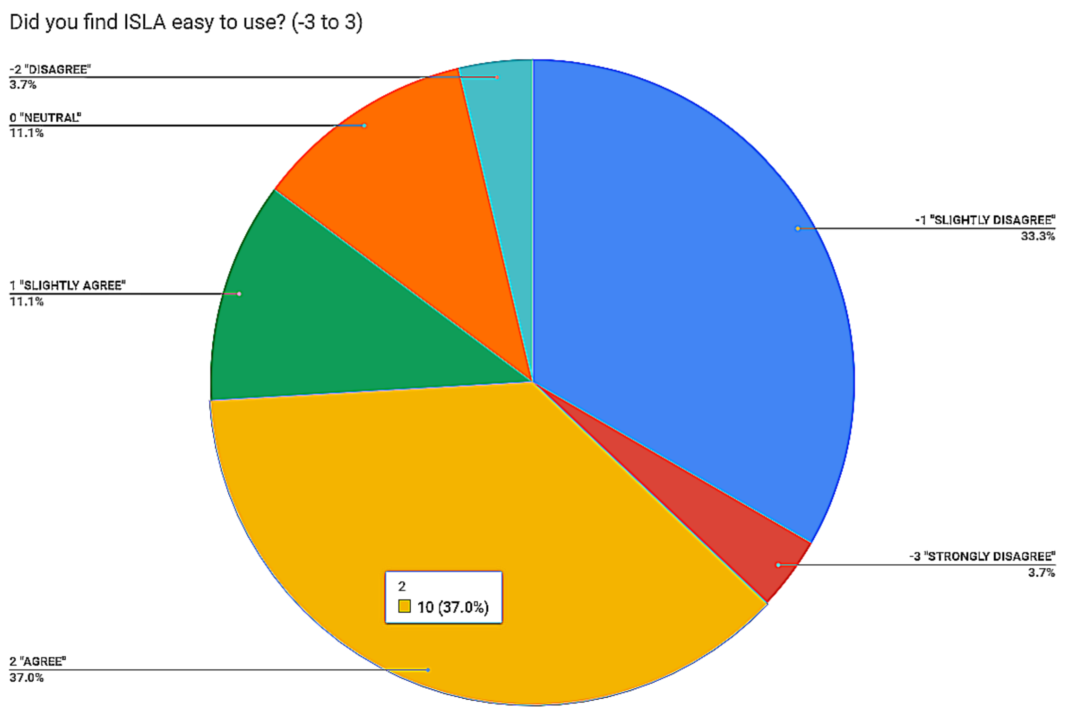


FIGURE C.1.3

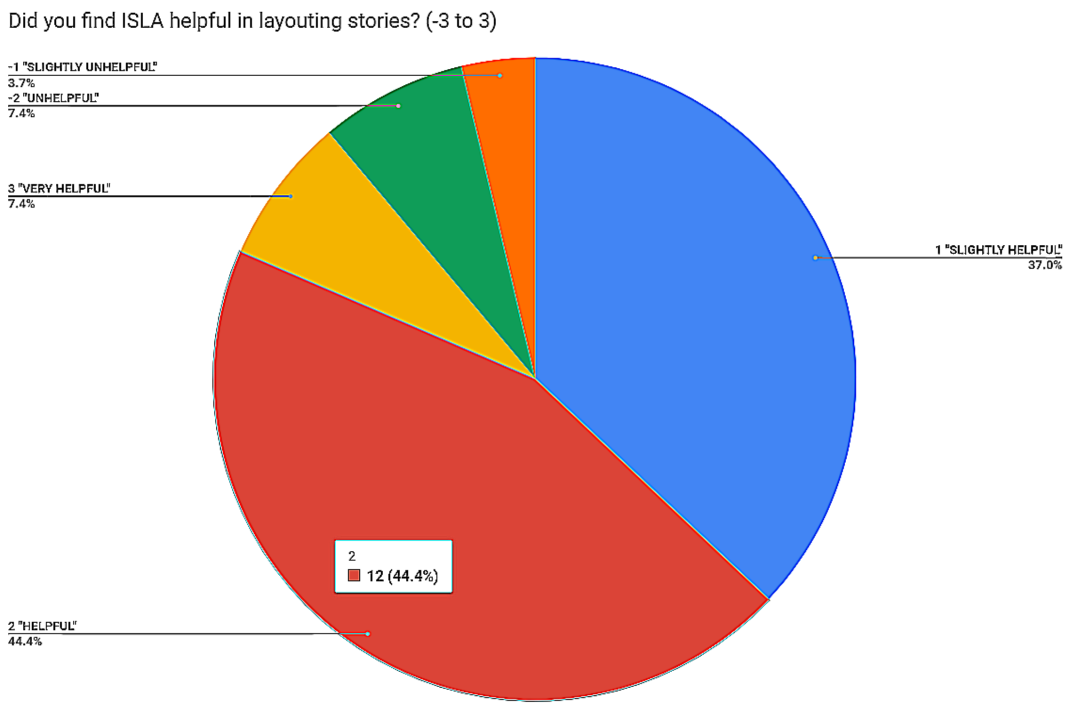


FIGURE C.1.4

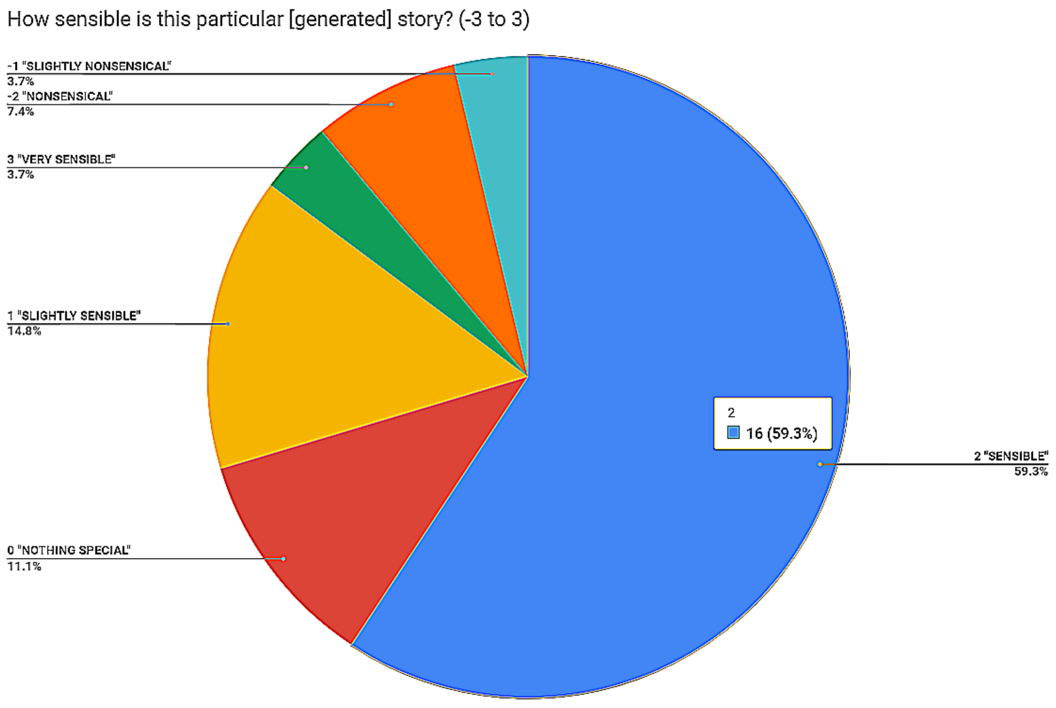


FIGURE C.1.5

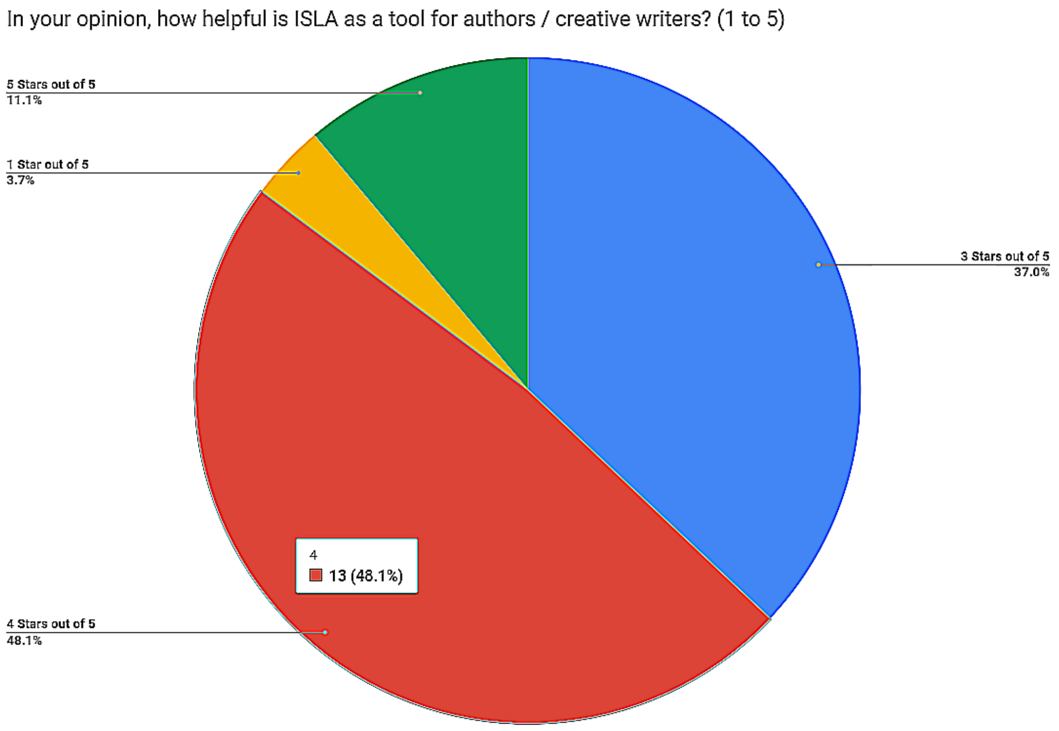


FIGURE C.1.6

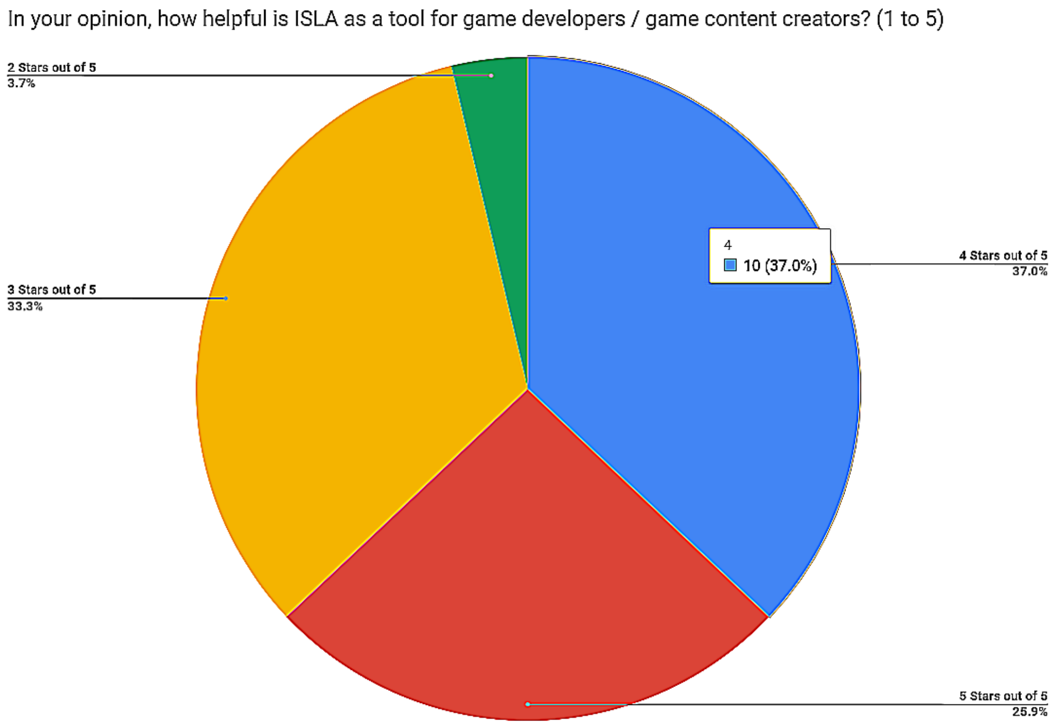


FIGURE C.1.7

Bibliography

[Agu11] Maniel Aguirre. An Outline of Propp's Model for the Study of Fairytales. *The Northranger Library Project*, 2011.

[Bal99] Mieke Bal. *Narratology: Introduction to the Theory of Narrative, Second Edition*. University of Toronto Press Incorporated, 1999.

[BBY15] Julio César Bahamón, Camille Barot, and R. Michael Young. A Goal-based Model of Personality for Planning-based Narrative Generation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 4142–4143, 2015.

[BCFC11] Anne-Gwenn Bosser, Pierre Courtieu, Julien Forest, and Marc Cavazza. Structural Analysis of Narratives with the Coq Proof Assistant. In *Interactive Theorem Proving - Second International Conference, ITP 2011, Berg en Dal, The Netherlands*, pages 55–70, 08 2011.

[Ber17] Leonid Berov. Towards a Computational Measure of Plot Tellability. In *10th International Workshop on Intelligent Narrative Technologies*, pages 169–175, 10 2017.

[Col07] Andrew Ian Coles. Heuristics and Metaheuristics in Forward-Chaining Planning. *Ph.D. Dissertation, University Strathclyde*, 2007.

[Die13] Leslie Dietiker. Mathematical texts as narrative: Rethinking curriculum. *For the Learning of Mathematics*, 33(3):14–19, 2013.

[GLM15] Pablo Gervás, Carlos León, and Gonzalo Méndez. Schemas for Narrative Generation Mined from Existing Descriptions of Plot. In *Proceedings of Computational Models of Narrative*, May 2015.

[Gog99] Joseph Goguen. Social and Semiotic Analyses for Theorem Prover User Interface Design. *Formal Aspects of Computing*, 11:11–272, 1999.

[HN01] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14:253–302, May 2001.

[Jor12] Anna Jordanous. A Standardised Procedure for Evaluating Creative Systems: Computational Creativity Evaluation Based on What it is to be Creative. *Cognitive Computation*, 4(3):246–279, 2012.

[KB17] B. Kybartas and R. Bidarra. A Survey on Story Generation Techniques for Authoring Computational Narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):239–253, Sept. 2017.

[KMRW07] Fairouz Kamareddine, Manuel Maarek, Krzysztof Retel, and J. B. Wells. Narrative Structure of Mathematical Texts. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, pages 296–312, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Leh81] Wendy G. Lehnert. Plot Units and Narrative Summarization. *Cognitive Science*, 5(4):293–331, 1981.

[Man13] Inderjeet Mani. Computational Narratology. `https://www.lhn.uni-hamburg.de/node/43.html`, January 2013.

[McI11] Neil McIntyre. Learning to Tell Tales: Automatic Story Generation from Corpora. *Ph.D. Thesis, University of Edinburgh*, 2011.

[Mee77] 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.

[MP20] Ben Miller and Julie S. Park. Computing Narrative. In *Proceedings of the Workshop on Computational Humanities Research (CHR 2020), Amsterdam, The Netherlands, November 18-20, 2020*, volume 2723 of *CEUR Workshop Proceedings*, pages 182–190, 2020.

[RY09] Mark O. Riedl and R. Michael Young. Narrative Planning: Balancing Plot and Character. *J. Artif. Int. Res.*, 39(1):217–268, September 2009.

[SHCK17] Rushit Sanghrajka, Daniel Hidalgo, Patrick P. Chen, and Mubbasir Kapadia. LISA: Lexically Intelligent Story Assistant. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2017*, pages 221–227, 2017.

[Sim18] D. E. Simonson. Investigations of the Properties of Narrative Schemas. *Georgetown University - Graduate School of Arts and Sciences*, 2018.

[SWF17] Alireza Shirvani, Stephen G. Ware, and Rachelyn Farrell. A Possible Worlds Model of Belief for State-Space Narrative Planning. In *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17), October 5-9, 2017, Snowbird, Little Cottonwood Canyon, Utah, USA.*, pages 101–107, 2017.

[SWS$^+$18] Rushit Sanghrajka, Wojciech Witon, S. Schriber, M. Gross, and Mubbasir Kapadia. Computer-Assisted Authoring for Natural Language Story Scripts. In *30th Conference on Innovative Applications of Artificial Intelligence (IAAI-18)*, 2018.

[Tob12] Ronald B. Tobias. *20 Master Plots: And How to Build Them.* F+W Media, 2012.

[WWFC15] Aaron Weinberg, Emilie Wiesner, and Tim Fukawa-Connelly. The Narrative Structure of Mathematics Lectures. In *North American Chapter of the International Group for the Psychology of Mathematics Education, 2015*, pages 1306–1313, 2015.

[WY11] Stephen G. Ware and R. Michael Young. CPOCL: A Narrative Planner Supporting Conflict. In *Proceedings of the 7th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 97–102, 2011.

[WY14] Stephen G. Ware and R. Michael Young. Glaive: A State-space Narrative Planner Supporting Intentionality and Conflict. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'14, pages 80–86. AAAI Press, 2014.

[You99] R. Michael Young. Notes on the use of Plan Structures in the Creation of Interactive Plot. In *M. Mateas and P. Sengers (Eds.), Narrative Intelligence: Papers from the AAAI Fall Symposium*, pages 164–167, 1999.