UNIVERSITY OF CALIFORNIA
SANTA CRUZ

NARRATIVE ENCODING FOR COMPUTATIONAL REASONING AND ADAPTATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Sarah M. Harmon

March 2017

The Dissertation of Sarah M. Harmon
is approved:

Professor Arnav Jhala, Chair

Professor Michael Mateas

Jill Denner, PhD

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Narrative Encoding for Computational

Reasoning and Adaptation

by

Sarah M. Harmon

Narratives provide a powerful means of making sense of our world. They are cognitive tools that we use to examine and shape ourselves, as well as the environment around us. Every day, we seek out and tell stories to reflect on our past, navigate our present, and direct our future.

Because narratives embed the complexity of the universe as we know it, it is challenging to develop a complete representation of narrative meaning. Some proposed theories focus on the perceptions and interactions of narrative agents. Others emphasize the communicative impact that a narrative has on its audience. Many more interpretations have been explored, but relatively little work exists that seeks to reconcile these multiple definitions and applications of narrative.

Narrative models may reflect more about their creators and their unique scenarios than the elusive tangles of meaning they are meant to capture. They are, after all, stories we are telling about stories. Each presents a lens in which to view, and subsequently define, narrative for a particular context. Regardless, perhaps we can learn from these diverse perspectives. Their congruence illuminates key features of story, while their discrepancies progressively carve out the rich, explorable space of all

possible narratives. Although we may never reach the boundaries of that space, our explorations could give rise to new insights in narrative understanding.

This thesis seeks to determine whether we can devise a framework that can usefully encode multiple narrative representations in a human-readable format. To do so would promote annotation, comparison, and sharing of narrative data among researchers. Further, a framework that can express distinct narrative representations should also reveal methods for automatically transforming their fundamental units. I will provide examples for how this type of adaptation may be useful for personalizing human-computer interaction scenarios which rely on narrative structures.

Dedicated to Mary, David, Katharine, and Peter.

Without you, my story would be incomplete.

Acknowledgments

I must first thank my friend and advisor, Arnav Jhala, for his guidance and continuous support. I am grateful for his firm confidence in my potential, and for the freedom he gave me to pursue my goals. I would also like to extend my gratitude to the other members of my advancement and dissertation committees: Marilyn Walker, Gina Lepore, Ellen Riloff, Jill Denner, and Michael Mateas. Through their feedback and outstanding support, I have learned how to develop the ideas in this document and grow as a scholar.

This dissertation would not have been possible without funding from the National Science Foundation and the Jack Baskin and Peggy Downes Baskin Fellowship. Additionally, I would like to recognize the supportive community at the University of California at Santa Cruz. Noah Wardrip-Fruin has both provided and encouraged thoughtful discourse, and has developed a positive research environment along with Michael Mateas, Marilyn Walker, and Arnav Jhala. I must also acknowledge Sonali Somyalipi for her encouragement and generosity, and Jo Mazeika, for her conviction that the Rensa framework enables a formal theory of style.

Finally, I would like to thank my family and close friends outside of the university. Their kindness, courage, and strength have never failed to inspire.

Chapter 1

Introduction

"People think that stories are shaped by people. In fact, it's the other way around."

— Terry Pratchett

The customer ahead of me was a kindly gentleman, although any bystanders probably wouldn't have thought so. He had a sour look on his face, and was harrumphing up and down about the crowded nature of the grocery store.

"Students!" he grumbled to me. "That's the problem, you know. They take up too much space in this town! And not one of them knows a decent day's work!"

That was more or less what he said, although I have spared you the expletives. Rest assured that he was passionate enough about the matter to elaborate in a lengthy and colorful way.

I expressed my sympathies, and, in an attempt to lighten the mood, then directed the topic toward his own career. That was a gamble, of course. For all I knew, he might have despised his job as much as the students in town.

Still, I had to try something. The customers at thumb's length were beginning to stare.

"What is it you do, sir?" I ventured.

"Oh, I'm an author. I write books!"

The anger in his face subsided, and his chest noticeably puffed out of a slouch. Encouraged, I pressed on.

"What kind of books?"

"A bit of everything," he said. "Children's books, mostly. Lately, though, I've become more of a novelist."

"A novelist!" I said, and now it was his turn to see my delighted expression. Stories were a not-so-secret love of mine, as I had traveled to Santa Cruz to learn about them. When he mentioned he was looking for a new editor, I offered my services accordingly, and he agreed my qualifications appeared satisfactory for the job. At that point, he rightly inquired of my present vocation. I replied that I was, in fact, one of the dreadful students that he so abhorred.

"Oh!" the gentleman said simply, and grew quiet. Cogs spun inside his head, revising the situation. Only after a lengthy pause did he speak again.

"To have an editor who is a student at the same time - why, that would be great!" he said. "The main character in my story could learn a lot from you!"

I was amazed, and somewhat nonplussed. Here was a gentleman who had gone from condemning students to considering them worthy and helpful. I entertained the possibility that he was simply being polite, but his tone and manner were genuine, and

- as I confirmed through later discussions - he was not one to hide his true feelings. Perhaps it was enough to get to know one of these bothersome students, and put a face to the abstract concept of them in his mind. The mental representations he had - indeed, the stories he was telling himself about students - were rewritten, and reread.

Soon thereafter, he began to recall his own fond memories of school. In a matter of seconds, hatred had transformed into respect and wistful nostalgia. The gentleman and I became friends.

Such is the power of the stories we tell ourselves, and each other. To match our surroundings, we adapt and transform our mental narratives of the world. These narratives, in turn, serve as guiding lights for our everyday decisions. In so doing, they can transform our thoughts, feelings, and behaviors.

How do these stories work? What are they made of, and how do we transform them to help us as we go about our daily lives? How might we change our lives for the better through new and adaptive storytelling technologies? These are the questions I have sought to answer with this thesis. To begin, let's take a closer look at the functions and effects of storytelling.

1.1 How Stories Transform Us

Narratives are powerful modes of communication, and influence our daily lives in many ways. For instance, stories help us to understand and organize our personal experiences [18]. Through telling and attending to stories, we gain insights for inter-

preting the world. We can also use stories to express and entertain ourselves. They allow us to convey our personal emotions, experiences, and identities. Simultaneously, they can delight and immerse us, and help us strengthen social bonds.

Narratives would not be useful if they could not emboss lasting impressions on our minds. We are able to map story information to analogous real-world scenarios, resulting in facilitated problem-solving [47, 48]. We are also mentally and emotionally affected by stories, which has led to several discoveries in using narrative to promote human health. Notably, research suggests that writing and narrative therapy are effective treatments for abuse [88], depression [77, 78], PTSD [1], and psychosis [95], as well as useful for resolving family, workplace, and school conflicts [14]. Narratives can empower us and inspire action as well. As an example, Perrier et al. demonstrated that disability narratives encouraged individuals with spinal cord injury to pursue rehabilitation [99]. Ultimately, we use stories to inform our decisions, construct our identities, and reflect on our past mistakes and possible futures. As anthropologist Mary Catherine Bateson would say, we use our experiences with narratives to compose our own lives [10].

By studying and building resources for narrative adaptation, we learn how to better support each of these functions in the real world. In this thesis, we will focus on how narratives can be used to organize information to interpret a variety of contexts. By exploring how to represent and re-organize narrative content in multiple contexts, we provide a foundation for new systems that aid in positive life composition.

1.2 Problem

Researchers often divide the study of narrative into content (what is told) and presentation (how it is told). This idea is echoed in Todorov's terms *histoire* and *discours* [117], Benveniste's *l'enonce* and *l'enonciation* [12], Genette's *diegetic level* and *extradiegetic level* [46], as well as the terms *fabula* and *syuzhet* in Russian formalism. Bal defines three levels [5]. First, she names the *fabula*, which specifies the precise events and which are "logically and chronologically" related. She then suggests two levels of presentation: *story*, which results when a narrator applies a lens to and organizes a fabula, and *text*, which are the signs produced to convey the narrative itself.

In this work, I will consider *narrative* to be a report of at least two connected events presented in sequence [73]. The *fabula* are these events, and the *story* is, similar to Bal, how this fabula is organized, selected, and interpreted by the narrator. The story is not constrained by medium, and may be told by any manner of creativity, including (but not limited to) speech, literature, gameplay, music, or transmedial combinations thereof. The *storyworld* is the contextual model of the universe within which the story takes place.

Given these definitions, we now begin to understand that narratives can be adapted in many ways. One might make adjustments to the fabula itself, by inserting, removing, or changing the connected events. One may also influence the presentation of the fabula, by shifting the point of view, voice, structure, style, theme, form, genre, or other elements of expression. One may also make adjustments to the storyworld, which,

although not necessarily a narrative itself, can influence the story and its telling.

There have been many proposed representations for narratives in computer science, but none are hailed as satisfyingly all-encompassing and complete. As van Dijk and Kintsch have stated, searching for an all-purpose representation is an "elusive and deceptive goal" [28]. As such, this thesis does not pretend to enable the encoding of all possible narratives. Instead, we will define a means to encode and adapt the concepts behind a given fabula, story, or storyworld for many distinct classifications of narrative. This work enables comparison among representations of meaning, and demonstrates how a multitude of narratives can be adapted for useful purposes automatically.

1.3 Contributions

This thesis seeks to provide several resources and analyses of use to researchers. The primary contributions are:

1. The construction of a cross-platform knowledge representation framework ("Rensa") which enables adaptation for multiple classifications of narrative.

2. An in-depth analysis of how Rensa can be used in the example subdomain of narrative simplification. This analysis includes:

   (a) A specification of conceptual relations which organize narrative events, actions, and beliefs and how they are connected (*SimpREL*).

   (b) A set of algorithms for reasoning over these relations, empowering the system to make inferences based on its understanding.

3. A review of approaches for automatically extracting narrative concepts from natural text, which can then be used with the Rensa framework.

4. A discussion of existing systems which use Rensa, demonstrating how the framework can be used in various contexts.

5. A corpus of Rensa encodings, which includes concepts present in fairy tales, film scripts, games, poems, music, recipes, and short fiction.

The pursuit of this work has led to a number of steps toward improved narrative understanding. The Rensa framework can encode a variety of narrative structures, which opens the door for the comparison of models and systems. Rensa encodings are also written in a human-readable format in line with modern goals of information retrieval. Consequently, humans and machines should be well-equipped to annotate Rensa-like data. The included corpus provides additional examples for how to write SimpREL encodings, and also functions as a dataset for future projects in machine learning.

This work motivated the development of the first system that measures narrative skill progress [60] as specified by Petersen et al.'s measure [100]. SimpREL can also be used to transform narratives from one skill level to another, as will be discussed in Chapter 7. I am hopeful that these contributions will act as stepping stones, and raise awareness in the computer science community about the need for these kinds of tools. Improved narrative comprehension and assessment resources would greatly benefit not only our schools, but our society in general.

Beyond supporting methods for assessing and transforming narrative complexity, Rensa has inspired a number of human-computer interaction applications that aim to benefit society in some way. These include a responsive, educational game generator as well as various systems that produce customized narrative experiences.

1.4 Outline

This thesis is structured as follows: Chapter 2 will describe the theory and related work which provide the foundation for this dissertation. Chapter 3 explores the advantages and limitations of a pre-existing narrative representation designed for accessibility and expressive power (the Story Intention Graph). The Rensa framework is introduced in Chapter 4. A formal specification of one Rensa encoding format (SimpREL) is described in Chapter 5, with supplementary SimpREL examples in Appendices A and B. Chapter 6 makes use of SimpREL to demonstrate how Rensa can support existing narrative representations. In Chapter 7, I demonstrate how Rensa can be applied to the problem of narrative simplification. An exploration of other existing Rensa applications is presented in Chapter 8. Lastly, Chapter 9 offers directions for future work and closing remarks.

Chapter 2

Theoretical Foundations

"Aren't you a computer scientist? So, why are you talking about stories?"

— Graduate student at the University of California, Santa Cruz

In order to adapt narratives, we must understand the primitive units of fabula, story, and storyworld. To do so, we will first consult with cognitive and social psychology to understand how we process narrative. Then, we will discuss a history of how narrative models have attempted to formalize this process.

2.1 Cognitive and Social Psychology

2.1.1 Levels of Discourse Comprehension

Researchers in cognitive psychology and narratology seek to define the cognitive mechanisms which underlie narrative production and comprehension. Most define five levels of discourse comprehension: *surface code*, *textbase*, *situation model*, *text genre*, and *communication level* [55, 96].

The *surface code* layer refers to the exact wording and phrasing used in the narrative, and is analogous to Bal's text layer [5]. In most cases, readers will retain only the most recent clause processed in memory [56].

The *textbase* is a set of "explicit text propositions in a stripped-down form that preserves meaning, but not the exact wording and syntax", as well as "a small number of inferences that are needed to establish local text coherence" [56]. These propositions are states, events, or actions, and may be real or imaginary. Each proposition contains a predicate (such as a main verb, adjective, or connective) and one or more arguments (such as nouns or embedded propositions). Each of the arguments has a functional role in the narrative (e.g., it is an agent, location, object, or patient). Graesser et al. present the following sentence as an example [55]:

> A mushy, brown peach is lifted from the garbage and placed on the table to pinken.

Given this narrative, the textbase would be:

```
PROP 1: lift (AGENT = X, OBJECT = peach, SOURCE = from garbage)
PROP 2: brown (OBJECT = peach)
PROP 3: mushy (OBJECT = peach)
PROP 4: place (AGENT = X, OBJECT = peach, LOCATION = on table)
PROP 5: pinken (OBJECT = peach)
PROP 6: [in order] to (PROP 4, PROP 5)
PROP 7: and (PROP 1, PROP 4)
```

The meaning of Props 1, 4, and 7 is that an unknown agent is lifting the peach object from the garbage, and placing it on the table. Props 2 and 3 specify

properties of the peach ("brown" and "mushy"). Props 5 and 6 specify that the agent's action in Prop 4 was done to cause the peach to pinken. Representing meaning with propositions and their relationships in this way is a well-known format accepted by linguists and philosophers in addition to psychologists. Similar representations have also been explored by researchers in computer science, which we will discuss in the next section.

Another important level of discourse comprehension is the *situation model*. The situation model is a level of memory representation for the storyworld, which the reader constructs by interpreting the propositional content. This representation typically includes events, individuals, actions, and - broadly - the narrative situation. It may include previous experiences and textbases to describe the story world [28].

The remaining two levels are both factors of presentation. The *text genre* is its structural category, e.g., narration, exposition, persuasion, and so on. *Communication level* refers to the context of how the text is conveyed, and is reminiscent of Bal's story layer [5].

All discourse comprehension models assume that, at minimum, readers will generate the textbase and situation model to cognitively represent a text [96]. Neuroscientific research has also suggested that specific regions in the brain correspond to the textbase (e.g., [62]) and situation model (e.g., [30, 39]) levels.

2.1.2 Narrative Interpretation

There are a number of other well-founded psychological theories with significant implications for narrative comprehension. Theory of mind (ToM), the ability to attribute and understand the mental states of others, is believed to be closely tied with narrative cognition. There is considerable evidence that language and social development give rise to ToM skills [4, 44]. In turn, brain regions involved in ToM and working memory appear to be heavily involved in story comprehension and production [83, 84].

Theories that describe how we organize data are also of great relevance to narrative modeling. Schemata theory, for instance, was introduced by Bartlett in 1932 to describe reconstructive memory and perception [9]. He showed that participants would interpret and recall narratives based on expectations derived from their mental models (*schemata*). Crucially, he demonstrated that these schemata evolve with experience.

Schemata are often compared with frames, another influential concept in psychology. Gregory Bateson popularized the term *frame* as a concept that defines or delimits a set of messages. According to Bateson, frames are *metacommunicative*: they provide signals to the receiver as to how to interpret the messages in-frame. Frames can communicate information by being *exclusive* (including certain messages implies that others are excluded) or *inclusive* (excluding messages means others are included). He related these ideas to figure-ground organization in Gestalt psychology, which states that we separate an image into figure (the main object(s) in the image) and ground (the remaining background of the picture) during perception. Bateson imagined that

frames around messages - not unlike a frame around a picture - directed the receiver to attend to certain aspects of the messages. Researchers in the social sciences as well as in artificial intelligence found framing theory attractive, as we shall see in Section 2.2.3 of this chapter.

2.2 Computational Models

2.2.1 Narrative Functions

Propp is well known as one of the early pioneers in modeling narrative. In 1928, he defined a set of 31 event and 7 character *functions* (stereotypical patterns or roles) [102]. He then described how Russian folktales could be described as a combination of these functions. While highly influential, Propp's functions are restricted to the Russian folktale domain and do not scale to sufficiently express other types of narratives.

2.2.2 Grammars

Later work in narrative representation included the development of a context-free grammar model for connecting discourse [105]. In Rumelhart's model, stories consist of a *setting* (time, place, and introduction to the main characters) and an *episode* (an event followed by some reaction to that event). This grammar was extended by several researchers [27, 66, 116], but ultimately abandoned due to limited expressivity and the fear that story grammars could not adequately model cognitive mechanisms [41]. Lang revisited these ideas during the implementation of his own declarative grammar, which

could describe goal-directed behavior and used temporal logic to represent states and events [74]. He acknowledged that, while novel, his model suffered the same inflexibility in terms of representing a wide range of stories.

2.2.3 Frames

Minsky is recognized as the first to relate Gregory Bateson's frames to general artificial intelligence. For Minsky, a frame was a set of nodes and relations that represented a "stereotyped situation" - essentially, a template that an AI could fill in when certain events or scenarios occurred. He described four levels of frames: *surface syntactic* ("Mainly verb and noun structures. Prepositional and word-order indicator conventions."), *surface semantic* ("Action-centered meanings of words. Qualifiers and relations concerning participants, instruments, trajectories and strategies, goals, consequences and side-effects."), *thematic* ("Scenarios concerned with topics, activities, portraits, setting."), and *narrative* ("Skeleton forms for typical stories, explanations, and arguments. Conventions about foci, protagonists, plot forms, development, etc., designed to help a listener construct a new, instantiated Thematic Frame in his own mind."). Unlike the dynamic, adaptive schemata proposed by Bartlett, a given Minsky frame was always fixed for some scenario.

The concept of frames generated a considerable buzz in the artificial intelligence community. Fillmore independently developed *frame semantics* as an extension of his earlier case grammar, using Minsky-like frames in terms of how words in a sentence are characterized and interact with each other [40]. Schank and Abelson also looked to

use Minsky-like frames in a more localized context. They found the notion of Minsky's frames promising, but believed the concept was too generic for real-world application. Together, they developed a frame-based representation that is now termed the Schankian script [108]. Scripts encode situations with slots that can be filled like frames, but focus expressly on encoding social events rather than broad directions for AI processes.

2.2.4 Plot Units

In contrast to the event-centric approach of scripts, Bremond [16] devised an elementary sequence representation based on Propp's [102] folktale analysis that included character motivations and emotional impacts. This work inspired the development of new interactive storytelling systems [107, 19], as well as new representations of story.

Bremond's model is now likened with the *plot unit* formalism developed by Lehnert [76]. Similar plot units are used in Elson's Story Intention Graph, a model founded in ToM that is meant to represent narrative in the context of analogy and social connectedness [32]. Representations like the plot unit tend to be semantically rich and expressive. Elson's model, in particular, draws attention to the overlap between ToM and narrative comprehension seen in neuroscience research. However, because such models require implicit information (such as components of character mental models) for a complete encoding, they are typically difficult to extract from natural text. Researchers are currently exploring how these types of representations might be automatically learned from corpora [54].

Model	Main Idea	Focus	Purpose	ToM
Narrative Functions [102]	Russian fairy tales contain a stereotypical set of characters and events.	Patterns	Comparison	N
Story Grammar [105]	Stories can be described with syntactic (structure) and semantic (interpretation) rules.	Events, Characters	Summarization	N
Scripts [108]	Stories can be an event sequence containing slots to fill.	Events	Understanding	N
Elementary Sequence [17]	Stories always have a goal, an action/event toward the goal, and an outcome.	Characters, Patterns	Comparison	Y
Plot Unit [76]	Events can be represented as mental and affect states for each character involved.	Characters, Patterns	Comparison, Summarization	Y
Story Intention Graph [34]	Stories can be compared based on patterns of relations.	Patterns	Comparison	Y
Doxastic Preference [79]	Stories are about time, preferences, and beliefs.	Characters	Characters	Y
Narrative Event Chain [20]	Stories are modeled as sets of temporally-ordered events that share a protagonist.	Events	Understanding	N
Social Network [3]	Novels can be represented as social networks.	Characters	Comparison	N
Formal Ontology [8]	Stories can be described as a formal ontology.	Events, Concepts	Comparison, Information Retrieval	Y

Table 2.1: A comparison of proposed narrative models and their philosophies. All models shown feature some manner of concepts and the relations between them, but differ in terms of their underlying principles. The last column indicates explicit support for theory of mind (Y=support, N=no explicit support).

2.2.5 Specialized Partial Models

Several models have been developed to represent one or more key aspects of narrative in particular. Specialized models tend to be much easier to extract from

natural text, but lack the rich implicit data other models might provide.

Social Graphs [3] are one example of a highly specialized narrative model. This representation considers only how characters interact over the course of a story. In particular, social graphs map the frequency of spoken interactions between characters and mutual presence (if characters occur in the same paragraph). The underlying principle of this model is that novels are akin to societies, which are representative of both the author and genre of the text.

Another specialized model is the Doxastic Preference Framework (DPF) [79]. This representation uses game trees to describe narratives. Tree nodes indicate character beliefs about preferences (their own, or those of others) for event outcomes. Although DPFs are similarly expressive and support ToM like plot unit representations, they are likewise difficult to interpret with high annotator reliability [82].

Unlike DPFs, the Narrative Event Chain (NEC) was developed to ignore implicit information completely [20]. NECs are simply event sequences that relate to a common protagonist. This representation was inspired by hand-coded scripts and built to efficiently predict and extract information relating to narrative events.

2.2.6 Formal Ontologies

Most recently, Bartalesi et al. have attempted to organize narratives into a formal ontology founded in description logic [8]. This ontology is composed of concepts and relations. According to their model, fabula events may be *real* (recorded information such as that from a witness), *hypothetical* (e.g., suggested events in an attempt to recon-

struct some part of history), or *fictional* (constructed using the writer's imagination). Events are connected by three types of relations: *mereological*, *temporal occurrence*, and *causal dependency*. A mereological relation signifies that one event is part of another (for example, the event of someone's birth is irrefutably part of their life). A temporal occurrence relation associates events by when they occur. The Bartalesi model employs Allen's formal theory of temporal logic, which includes a set of "mutually exclusive primitive relations that can hold between temporal intervals" [2]. These relations are depicted in Table 2.2. Finally, a causal dependency relation connects two events wherein one, either indirectly or directly, gives rise to another. This relation is deemed reflexive (each event causes itself) and transitive (if Event A causes Event B, and Event B causes Event C, then Event A causes Event C). A distinction is made between textual units that narrate an event (*narrative fragments*) and events of the fabula. The text itself is connected to its narrator by an *is authored by* relation.

Interestingly, the idea of concept relations is echoed in knowledge representations for artificial intelligence and interactive systems founded in rhetoric. Intelligent playable experiences, for instance, often seek to encode aspects about the game state or the player, and the relationships that occur as a result of interaction and time. Although game AI models tend to emphasize planning and prediction over a particular narrative structure, it is plausible that they bear similarities to those of narratology. We can tell stories about interactions that take place in a game, just as we might of a text-based narrative.

A prime example of using concepts and relations for general AI research is the

Relation	Symbol	Inverse Symbol	Pictorial Example
X before Y	$<$	$>$	XXX YYY
X equal Y	$=$	$=$	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Table 2.2: Thirteen primitive relations for temporal intervals [2].

OpenCog framework, which is meant to be an open source cognitive architecture [51]. The basic unit of the framework is the Atom, which loosely represents an atomic formula in mathematical logic. Atoms can have a variety of types, although there are two major classes: Nodes (an Atom with a name) and Links (an Atom with no name that connects other Atoms). Each Atom has a truth value (e.g. fuzzy, probabilistic, etc.), although the formalization of these types is currently not yet complete [50]. Generally, Atoms and their interactions with each other can be described as a *hypergraph*: a graph in which an edge can join any number of vertices. The hypergraph structure has also been proposed to represent computer narratives [121].

In order for these structures to be useful for an intelligent system, they must contain some module for logically reasoning over them. First-order logic is a common standard in this regard, and allows programmers to declare facts and concepts and rela-

tions between them. Unsurprisingly, computer science narratologists are often interested in using their models to advance research in natural language processing, which likewise involves a reasoning component. Elson built a corpus of SIG encodings in Prolog, a language designed for AI and linguistics, with this in mind [33].

Given that narratives are integral in human understanding and perception, it is unsurprising to see parallels between narrative and artificial intelligence models. As advances are made in each field, it is possible the region in which they overlap will increase further.

2.2.7 Surface-Level Adaptation

The representations discussed thus far are largely concerned with semantic encoding, reasoning, and adaptation of the fabula. While the primary focus of this thesis is also at the level of fabula, frameworks that enable adaptation at the surface-level are also useful. Curveship [93], for instance, is a system that allows interactive fiction authors to change certain aspects of a narrative's style. This includes the capability to specify *focalization* (the way that information available to the narrator is regulated), *time of narrating* (the temporal relationship between the narration and the story events), and the order of events. Another example is PERSONAGE [81], which can vary the personality features for a narrative speaker. The same type of variations that are possible with each of these systems could be encoded within the Rensa framework (introduced in Chapter 4) as future work.

2.3 Discussion

In this chapter, we have unearthed key structural units of narrative that are salient for readers. We also have described existing attempts for formalizing these units into a computational model. These foundations imply that any model for narrative adaptation must employ several key features. First, we must permit our model the capacity to conceptualize narrative primitives, including story characters, events, goals, actions, and affect. Secondly, we need a means for connecting these primitives with relationships. These relationships include temporal ordering, causal dependencies, and mereological relations. However, formalizations also demand relationships to classify primitives, such as whether they are real or imaginary. Relationships should be shareable and easily layered or embedded within each other. For example, goals may belong to multiple agents or be nested to form bigger goals. They also must be able to be related to actions, outcomes, and affectual impacts.

Additionally, narrative and artificial intelligence research suggest the need for a bridge between the representation and formal logic. Because specific categories and relationships for generalized representation have not yet been fully determined, it is likewise important that a computational model of narrative be as flexible as possible to support the encoding of the unknown.

Chapter 3

Case Study: Story Intention Graphs for Narrative Reasoning

> *"What kind of model can best capture the essence of a story? ...Can we define a set of intratextual relations specific to our idea of the essence of narrative discourse?"*
>
> — David Elson

By now, we have considered a brief history of narrative representations, with a particular focus on reasoning about the fabula. This chapter will examine one of these existing models - the Story Intention Graph - to explore the limitations of current representations in narrative processing.

The SIG is a relatively new model with considerable expressive power. Originally, SIGs were designed to express and compare thematic content [32]. Early work did not suggest their application as a formalism for narrative generation. Nevertheless, a later publication demonstrated that SIGs could be used to generate distinct tellings [104].

This fact highlights an impressive quality of the SIG formalism: its capacity to transcend predetermined boundaries. I decided to pursue several case studies to determine if the SIG would also be amenable to my goals for an expressive, human-accessible encoding format. Specifically, I was interested in how the SIG would fare in applications relating to narrative generation, adaptation, and assessment.

3.1 Creative Story Generation

Elson et al. had already demonstrated that the SIG formalism had potential for generating distinct tellings [104]. A natural course of action, then, was to explore how the SIG model could improve upon existing narrative generation systems.

An early system in modeling human authoring was Minstrel, which utilized a case-based reasoning approach to incorporate a model of computerized creativity [119]. To reconstruct old stories into new ones, Minstrel relied on twenty-five heuristics called TRAMs (Transform-Recall-Adapt Methods). As an example, Minstrel contained a default TRAM called Standard-Problem-Solving which simply looked for a pre-existing solution in memory. If no solution existed, the TRAM would fail. The TRAM also would fail if any found solutions had already been used, because such solutions were deemed boring by the Minstrel system. Whenever a given TRAM failed, the problem would be transformed and Minstrel had to search for a case that best matched the newly transformed problem.

I revisited this creative generation mechanism through the extension of a con-

temporary rational reconstruction of Minstrel called Skald [115]. As part of this work, I organized the underlying Skald narrative representation and mapped the result to the SIG formalism. These refinements enabled generated stories to be transformed at the level of thematic patterns, and thus expanded the creative authoring space. Three primary phases were necessary for the system: *event ordering*, *node construction*, and *chain construction*.

3.1.1 Event Ordering

Skald generates a natural language story with events told out of order. While not every narrative generation system may require event ordering, I included a module for this purpose so that any story generated by Skald would be told in the proper sequence. The Event-Ordering Module (EOM) used knowledge of preconditions and consequences to reorder events in the timeline.

3.1.2 Node Construction

The Node Constructor (NC) unit categorized each graph element as a Proposition (P), Goal (G), or Belief (B) node. Skald already labels frames as states, goals, and actions. This greatly simplified the labeling process, as P nodes included states and actions that occurred in the fabula timeline. Using these labels, every element of the output graph was translated into a discourse relation and annotated with the correct agents, objects, and any other related entities. Because Beliefs and Goals are frames containing content, they were labeled and filled with one or more Interpretive Propo-

sition (I) relations. In Skald, the affectual impact of a P node or actualized I node is merely implied based on event-consequence pairings and whether goals are achieved. To create a proper SIG encoding, Affectual (A) nodes were created for each character of the story.

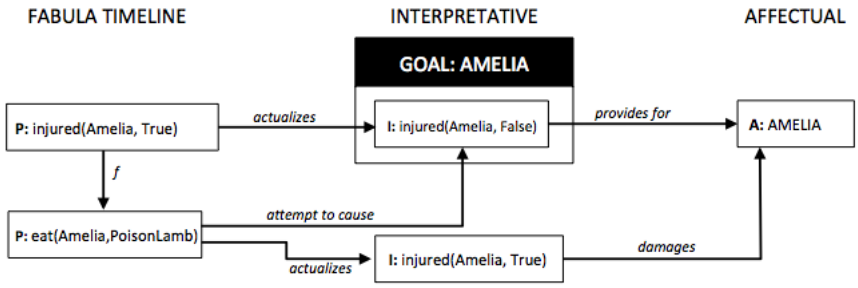

Figure 3.1: A SIG encoding derived from Amelia's Story. Amelia's Story follows the *Backfire* pattern shown, coupled with an ultimate gain after a new *Desire to Aid Self* goal. In the story, a wizard named Amelia is injured. To make herself well, she tries eating some lamb, but the lamb is poisoned. Undaunted, she uses magic to restore herself to full health.

3.1.3 Chain Construction

Once all nodes were established, they needed to be linked to complete the SIG encoding process. This process was ensured by the Chain Constructor (CC) module, which reviewed the given event-consequence pairings to make decisions about how P and I nodes (including Goals and Beliefs) were linked. Links were defined in accordance with Elson [32]. The system connected each I node to corresponding A nodes by considering the effects of that I on each agent's goals. If a goal was met for an agent when an I node was carried out, a *provides-for* link was established between an agent and that node. Conversely, a *damages* link was created when the current I node thwarted an agent's

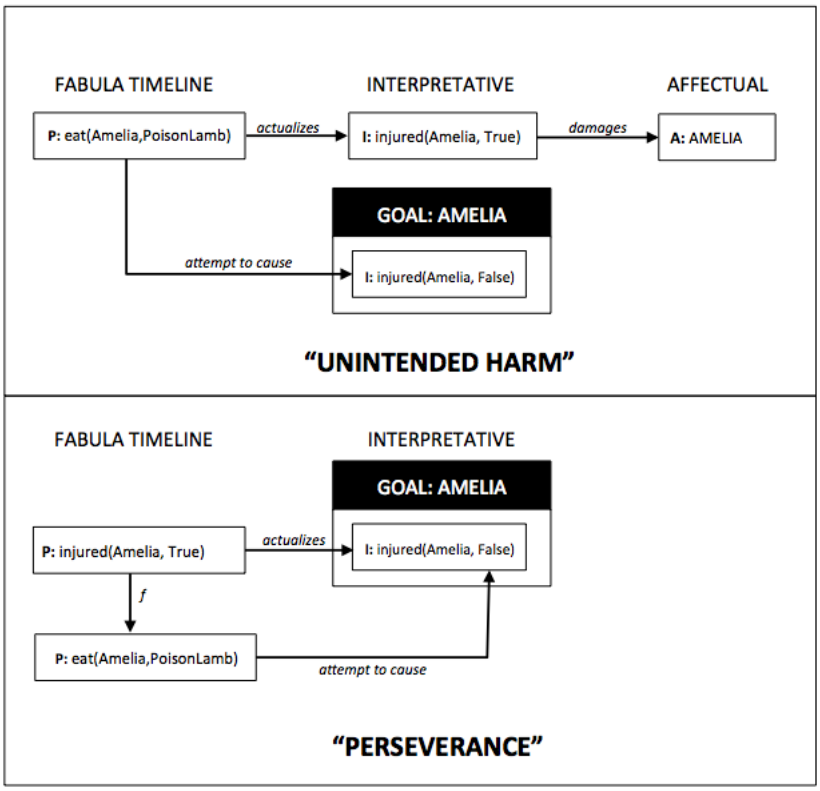

Figure 3.2: Alternate tellings of Amelia's Story.

goal. If any A nodes contained no links by the end of the chain construction process, they were removed from the final graph.

3.1.4 Results

By providing Skald with a SIG case library and specifying rules for SIG-based transformations, we can apply the TRAM procedure to the SIGs themselves. For instance, consider Amelia's Story (Figure 3.1). Nodes and links may be removed from the *Backfire* sub-encoding by following variations of *Ignore* TRAM templates, resulting in different tellings (Figure 3.2). In the *Unintended Harm* telling, the storyteller empha-

sizes only how Amelia mistakenly came to harm herself by eating the lamb. In contrast, the *Perseverance* encoding tells the story of how Amelia continued to eat poisoned lamb in an attempt to make herself well.

• • •

At the end of our third day in Midland City, Felix became tearful and risked the displeasure of Captain Julian Pefko by asking him if we could please, on the way to the main gate, have our purple school bus make a slight detour past Calvary Cemetery, so we could visit our parents' grave.

For all his rough and ready manners, Pefko, like so many professional soldiers, turned out to have an almond macaroon for a heart. He agreed.

• • •

Almond macaroons: Preheat an oven to three hundred degrees, and work one cup of confectioners' sugar into a cup of almond paste with your fingertips. Add three egg whites, a dash of salt, and a half teaspoon of vanilla.

Figure 3.3: An excerpt from Vonnegut's *Deadeye Dick* [122], which includes full recipes as part of the novel's telling.

3.1.5 Reflection

Regardless of whether SIGs are extended as a formal representation for creative narrative systems, they possess key properties that satisfy the provision of semantic plot information and a bridge towards linguistic generation. Most significantly, SIGs have inherent event-ordering properties and may be transformed, as demonstrated by this work. Any representation which provides this level of detail is useful for imaginative recall systems. SIGs also map well to plan-based and plot graph-based story generation

systems.

Scheherazade [34], an interface for manually annotating SIGs, is capable of realizing SIG encodings into natural text. However, this realization is not very customizable or guaranteed to be correct. It is important to also note that, like other representations, SIGs are designed with a certain type of narrative in mind. For example, authors such as Kurt Vonnegut have inserted recipes inside their novels as a storytelling technique (Figure 3.3). One might attempt to use SIGs to describe the result, but to neatly communicate the layers of meaning that emerge is challenging. Additionally, while we can transform stories or compare generations in terms of SIG patterns, the SIG may not enable us to comfortably express or perform all types of adaptations, such as altering the presentation of the fabula or medium. These observations reveal the SIG is not well suited for combining different storytelling techniques or models.

3.2 Measuring Narrative Complexity

While the SIG may not be able to model all modes of storytelling, it is capable of encoding deep semantic information present in narratives. This begs the question of whether SIGlike structures might be used to support the automation of intricate semantic or thematic narrative analyses.

Consider, for example, the fact that complex narratives are difficult to create and understand even for humans. Individuals may face additional obstacles that hinder their progress in narrative development, which, in turn, can deeply impact their aca-

demic, social, and cultural skills [90, 94] as well as their health [13, 15, 72]. It is thus important that we devise a suitable assessment procedure to provide insightful feedback for learners.

Some years ago, the index of narrative complexity (INC) was proposed to serve as a measure of narrative development for children and individuals with language disorders. The INC scoring system examines thirteen narrative elements: character, setting, initiating event, internal response, plan, action/attempt, complication, consequence, formulaic markers, temporal markers, causal adverbial clauses, knowledge of dialogue, and narrator evaluations [100]. Each category is associated with a weighted score which reflects its importance in terms of narrative production skills. Evaluation of the INC scoring system suggested that its assessments are consistent and strongly correlate with the Test of Narrative Language [49]. However, a rich understanding of the narrative text is required to perform each assessment, which is a time-consuming process for educators.

3.2.1 The Automated INC Scoring Process

To determine if SIGs could assist in the automation of INC assessment, I created a system that provided a score for twelve of the original thirteen INC narrative elements. The system required the full text and SIG encodings of the story. The following sections will explain how the system scored each element in more detail.

3.2.2 Character

A *character* is a reference to the subject of a clause. Following previous work [36, 97], part-of-speech tagging and WordNet categories were used to determine if a particular entity was displaying signs of animacy. For example, the subject of a clause may have engaged in dialogue, made a plan using a cognitive verb (e.g., "think"), or taken an animated action (e.g., "walk"). If the same subject contained a reference that used capital letters (such as "Tin Soldier" or "Little Tuk"), the system considered the reference a name.

If no character was found, or ambiguous pronouns ("he", "they") were used, the system assigned a score of zero points. One point was awarded if the narrative included at least one frequent actor with nonspecific labels ("Once, there was a knight."), and two if that character had a name ("Once, there was a knight named Mary."). If more than one named character took frequent action, three points were awarded.

3.2.3 Setting

The *setting* is any reference to a place or time. Setting references were identified using WordNet's location and time categorization lexnames. To improve the accuracy of the system, state abbreviations that have common duplicate meanings (such as "OH", "OR", "US", and "ME") which WordNet marks as locations were not counted as setting references. Zero, one, and two points were respectively awarded to no, one, or multiple setting references in the text.

3.2.4 Initiating Event

An *initiating event* is a reference to any event or problem that is likely to elicit a character response. Because identifying any sort of problem for a character is a complex narrative understanding problem, the system designated any event in the SIG timeline which prompted a character goal as an initiating event. To determine whether a goal was prompted by an event, the system checked if the two were connected by an *actualizes* link.

Critically, the system must also discover if characters reacted to the initiating event. These reactions may be *internal responses* or *attempts* (refer to the following sections). If an internal response was marked using WordNet, it was also marked in the SIG encoding during its portion of the timeline. If the internal response directly followed the initiating event in the graph, the system considered it to be a reaction.

If no initiating events occurred, zero points were awarded. One point was awarded if characters did not appear to respond to one or more initiating events. Two points were awarded if at least one event elicited a response, and three if two or more distinct initiating events elicited a response.

3.2.5 Internal Response

Internal responses encapsulate information about a character's psychological state, such as their emotions and desires. The system first used WordNet to determine whether each word expressed (or had a synonym that expressed) a basic emotion. This procedure identified explicit statements of emotion, such as "The video made him

angry". The Stanford parser [25] was used to determine whether these emotions were related to specific entities. If so, two points were awarded. Otherwise, zero or one points were respectively given to no references or a single reference to emotion.

3.2.6 Plan

A *plan* always includes a cognitive verb that indicates an intent by a character to act on or solve an initiating event. WordNet was used to identify cognitive verbs as well as several exceptions that indicate intention ("want","hope","desire"). The Stanford parser was also used to determine whether each instance of a cognitive verb was associated with a main character, i.e., the most frequent actor or speaker. Zero, one, two, and three points were respectively awarded to no, one, two, or three cognitive verb references in the text.

3.2.7 Action/Attempt

An *attempt* occurs when a main character acts in response to the initiating event. An *action* may be taken by a main character that is not directly related to the initiating event. The system examined the SIG encoding for instances of acts with the intent to influence goals associated with the initiating event. These acts must be subsequent to the initiating event itself. If links such as *actualizes*, *ceases*, *prevents*, *attempt to cause*, or *attempt to prevent* existed between the action and the goal, the system considered the character to be making an *attempt* with respect to the initiating event. If the character merely acted with no clear connection to the initiating event, it

considered these *actions*. If neither actions nor attempts were pursued, zero points were awarded by the system. Instances of only actions taken resulted in one point awarded. Two points were awarded if at least one attempt was made.

3.2.8 Complication

A *complication* is either (1) an event that prevents the execution of a plan or attempt, or (2) a second initiating event. To determine if the latter was present, the system checked for a second instance of an event that beget a character response (see *Initiating Event*). The system determined if the former occurred by analyzing the SIG encoding for the presence of a *ceases* (or *prevents*) link that was associated with a tagged plan or attempt. No, one, or two points were rewarded for zero, one, or two complications, respectively.

3.2.9 Consequence

A *consequence* is the result of an action or attempt on the initial problem. Such a consequence does not need to resolve the problem, but it must be related to the initiating event and explicitly stated. We can thus make the assumption that consequences arise because the plans of a main character have been achieved or thwarted. To identify such cases, the system searched for Proposition nodes that were connected with initiating event-prompted character goals via *actualizes*, *ceases* (or *prevents*), and *implies*. No, one, two, or three points were awarded for zero, one, two, or three consequences, respectively.

3.2.10 Formulaic Markers

Formulaic markers, such as “once upon a time” or “they lived happily ever after” indicate the beginning or end of a narrative. The system used pattern matching techniques to determine if a formulaic marker was present. Zero, one, or two points were assigned for no instance, a single instance, or two (or more) instances of a formulaic marker.

3.2.11 Temporal Markers

Temporal markers (e.g., “before”, “instantly”, “once”) indicate time. Like formulaic markers, they may be assessed by simply identifying these key words or phrases. For each type of these elements, the system assigned zero points for no instance of the element, one point for a single instance, and two points for two or more instances of the element.

3.2.12 Causal Adverbial Clauses

Causal adverbial clauses are words or phrases in a sentence which indicate cause (e.g., “because”, “since”, “so that”). For each type of these elements, the system assigned zero points for no instance of the element, one point for a single instance, and two points for two or more instances of the element.

Narrative Element	Accuracy
Character	80%
Setting	80%
Initiating Event	75%
Internal Response	55%
Plan	85%
Action/Attempt	85%
Complication	80%
Consequence	90%
Formulaic Markers	95%
Temporal Markers	95%
Causal Adverbial Clauses	100%
Knowledge of Dialogue	95%

Table 3.1: Evaluation of the automated narrative complexity assessment system.

3.2.13 Knowledge of Dialogue

To determine how many characters are engaging in conversation, we must identify the speakers in the text. In accordance with He et al.'s work [63], the system extracted speakers by targeting speech verbs ("say", "speak", "talk", "ask", "reply","answer", "add", "continue", "go on", "cry", "sigh", "think") proximal to the utterance. If none of these verbs occurred, any verb preceded by a name, personal pronoun, or animate character was chosen. If the story included named characters, the closest name preceding the pronoun or character label was selected as the speaker. The reference to the speaker was generally extracted by parsing the fragment containing the speech verb, and by following a deterministic method based on syntactic rules. This reference was then matched to known speakers and their associated labels. For example, if the current speaker was "Harry", and "he" was the next speaker, the system would

assume “he” referred to “Harry”. If no speech verb and character reference was found in the vicinity of the utterance, it was assumed that the quotation marks were used to denote an expression other than dialogue, such as a title. If no dialogue was present, zero points were assigned by the system. If one speaker was present, one point was awarded. If there were two or more characters engaging in conversation, two points were assigned.

3.2.14 Evaluation

To evaluate our system, twenty excerpts were randomly selected from open-domain fairy tale texts [61]. Each selection was at least 200 characters in length, and was manually annotated and scored in terms of its index of narrative complexity [100]. These scores were then compared with the system's assessments (Table 3.1). The number of points awarded by the system for each narrative element had to agree exactly with the annotation to be considered accurate.

3.2.15 Reflection

The development of a SIG-based narrative complexity assessment system revealed several properties of the SIG as an encoding scheme. First, SIGs are expressive enough to capture meaningful information about events and the character reactions they inspire. A future system might even use a SIGlike structure to improve automated assessment of internal response.

While researchers are working on automatic extraction of SIGs from natural

text, this achievement has yet to be realized. As a result, SIG encodings had to be manually annotated as input for the system presented here. Annotation options at the time this system was devised included using the Scheherazade interface [34] or programming in Prolog directly. These annotations had to also conform to the finite set of relations, entities, actions, qualities, and so on as detailed by the specification.

Because the SIG annotation process is relatively subjective, multiple annotations per source text would ideally be needed for this type of assessment system to compute a reliability score. This subjectivity further suggests that consistent results may not always be guaranteed when using SIGs as the basis for the system.

3.3 Discussion

This chapter examined the advantages and challenges that ensued when using SIGs in several narrative processing applications. Notably, it was shown that SIGs could be used to support reasoning in narrative generation systems and enable transformations on the level of narrative patterns. Moreover, these case studies revealed that SIGs are expressive enough to capture internal states of agents, as well as how events are related to other events, agent perceptions, and agent behavior. All are useful for assessing narrative complexity. Unlike other representations such as plot units, SIGs can also express hypothetical events, which serves to clarify how plans and goals are connected in the story.

Although SIGs were designed to be simple enough for lay readers to annotate

encodings, the annotation procedure was simultaneously awkward and restrictive. One had to either understand the Prolog language, or mentally reduce their model of the narrative to fit the vocabulary and elements known to Scheherazade. Elements of story that were based fundamentally in concepts, rather than traditional story elements, were difficult to express. Finally, SIG structures could be realized into natural text, but these realizations were not always accurate or sufficiently tunable within Scheherazade.

Taken together, these findings suggest that a representation with greater accessibility and expressive power is needed for our purposes. In the next chapter, we will use these insights to guide the design of a framework for encoding and adapting narrative structures.

Chapter 4

The Rensa Framework

"Here we go!"

— Mario

4.1 Goals

In Chapter 2, we explored the history of modeling narrative structures. Narrative representations tend to require, in some shape or form, the existence of *concepts* and how they are related to one another (*relations*). These units build into encodings of explicit text coherence and meaning (*textbase*) and the overall storyworld (*situation model*). Each representation, however, may have distinct assumptions and approaches surrounding these related concepts. There is no single accepted set of concepts, nor relations, and any we have established for either may continue to evolve as our language and narrative representations change.

Several existing resources have been designed to encode concepts and rela-

tions. ConceptNet [114] has been proposed as an extractable representation for semantic knowledge, but generally supports binary relations without nesting, and is meant for sharing a large wealth of commonsense data rather than encoding local, client-specific data. A more flexible project is OpenCog [51], which represents the idea of concepts and relations as *nodes* and *links*. However, OpenCog is a heavy development endeavor for general artificial intelligence, and requires its users to possess a substantial amount of knowledge in computer science and logic.

These observations, in addition to what we have learned from the previous chapters, highlight a number of goals for our adaptable narrative representation framework. Ideally, our representation should be designed in such a way that encodings can be easily extracted from input data. This feature would reduce the need for heavy manual annotation of narrative structures. The representation should also be accessible for research, application development, and classroom use. Programmers (and ideally, non-programmers) should be able to read and easily annotate or edit encodings manually. Encodings should be stored in a cross-platform format to further expand accessibility. Lastly, the representation should have high expressive range, and be able to perform basic adaptations for many types of stories.

The following sections will describe Rensa, a framework designed to meet these goals. We will define a general specification for Rensa and explore its applicability in Chapters 5-8.

4.2 Framework Overview

In Rensa, a *brain* contains a set of *assertions*, each of which are an unordered set of name/value pairs (*attributes*).[1] A name is a string literal that represents the label for the value. A value can be an array, a Boolean, a number, a string literal, or another assertion. These structures can be nested.

A Rensa *concept* represents some unit(s) of knowledge that can be related to others. When we use this term, we might mean the value (or part of the value) of an assertion, a subset of attributes inside an assertion, or an entire assertion.

Within these conditions, we can specify any name or value for a Rensa attribute. This helps us build assertions that are specific to a certain model or system. The assertions themselves are encoded using JSON, a lightweight and platform-independent format for storing and exchanging data. JSON is relatively straightforward for non-expert humans and machines to interpret, unlike other more formal logical languages.

4.3 Querying Brains

One may add, edit, or remove assertions associated with a particular brain, or check if a brain contains an assertion. Rensa also contains functions that enable:

- Brain comparisons ("Are these brains equal? What exact assertions do they share?")

[1] Please note I do not claim that brains in the real world work this way. I am simply borrowing the term.

- Retrieval of assertions that have a specified tag (e.g., "Find me all assertions that are tagged with *protagonist.*")
- Retrieval of assertions which contain certain attributes (e.g., "Find me all assertions that contain an attribute named *relation* with the value *father_of.*").
- Retrieval of all assertions that are related to a particular concept, and how they are related ("Find me all assertions related to *prince* and tell me which part of the assertion contains *prince.*")
- Retrieval of individuals that comprise a given group (e.g., "Find me all concepts that are in the group *male*" might return all male individuals known to the brain).

4.4 Bridge to Formal Logic

Rensa is meant to represent simple relations of narrative meaning as opposed to formal logic rules. However, there is some support for logical statements, which will be covered in the next chapter. Additionally, Rensa can translate assertions into first-order logic. Any JSON file of assertions can be converted into a Prolog file, and users can request certain queries of the file through Rensa. These queries might include whether a statement is true, what properties an entity has, the value of those properties, and so on. This functionality provides a bridge between logic reasoning systems and semantic narrative planners.

4.5 Surface Realization

Rensa provides some basic surface realization support for an encoding format which will be discussed in the next chapter (SimpREL). To realize an assertion, Rensa takes a Boolean flag and a brain as additional inputs.

The Boolean flag indicates whether the statement should be realized as a sentence fragment (True) or a standalone sentence (False). The brain parameter is a set of facts that Rensa can reference when trying to make sense of the assertion. To illustrate, consider if we were to ask Rensa to realize an assertion about an individual with a unisex name. To select a pronoun for the individual, Rensa first can refer to the given brain to determine if the person is known to be associated with a certain pronoun. Rensa can thus use knowledge based on some known set of facts to modify how a sentence is presented. This may especially come in handy if we would like to realize dialogue from the perspective of different actors with distinct brains.

4.6 Automated Visualization

Rensa can generate images that represent assertions or visualize assertions in the browser. When paired with the ability to translate logic code into semantic structures, Rensa can help developers visualize the underlying mechanics of interactive experiences (as in [85]). Beyond visualizing assertions and their interactions, Rensa can be coupled with an authoring interface for one to navigate and edit a given story, or create their own for the computer to understand. An example screenshot of this

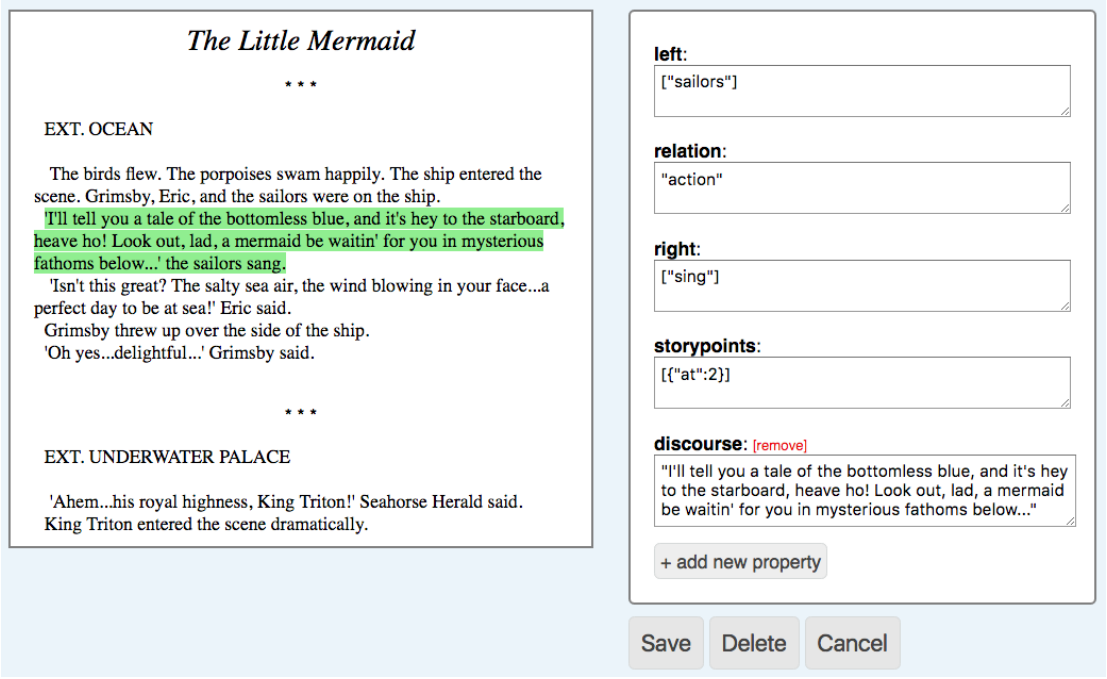

Figure 4.1: A screenshot of Rensa's story authoring interface. The information displayed is extracted from the film script for Disney's *The Little Mermaid*. The left section displays the story assertions realized into natural text. Users can click to edit any part of the story, and the realized text will update accordingly. Facts about concepts that aren't necessarily explicitly realized as part of the telling (such as an actor's gender) may be edited as well. Stories can be created from scratch, loaded from JSON files, or exported into JSON format for use in another system.

authoring interface is shown in Figure 4.1.

4.7 Summary

This chapter introduced the Rensa semantic framework, which stores knowledge about narrative concepts and concept relationships. Rensa can represent and adapt narrative data and automatically realize this knowledge into natural text. The data itself is stored in an open-standard format (JSON), which promotes accessibility

for developers. We will discuss the challenge of automated encoding extraction as well as how Rensa can represent other story representations in the next two chapters.

Chapter 5

SimpREL: A Simple Rensa Encoding Method

"It had been startling and disappointing to me to find out that story books had been written by people, that books were not natural wonders, coming up of themselves like grass."

— Eudora Welty

5.1 Encoding with Rensa

Rensa is designed to handle many different ways of encoding narrative structures. We will discuss how Rensa supports existing narrative representations in the next chapter, but let us first consider an example syntax for Rensa encodings. To do so will help us (1) become familiar with how Rensa can map concept relationships, and (2) provide a common language to which we can refer later on.

One such encoding method is SimpREL (pronunciation: [sɪm.ˈprɛl]), an optional syntax supported by the Rensa framework. The name refers to *simple relations*,

or - alternatively - the *Simple Rensa Encoding Language*. SimpREL can represent many common narrative structures that have appeared in novels, song lyrics, and recipes, among other works. A corpus of encodings is available at `https://github.com/RensaProject/simprel_corpus`.

The remainder of this chapter will outline the specifications for SimpREL. To begin, recall that a Rensa value may take the form of a string literal, array, Boolean, number, or another assertion. We will define SimpREL assertions in a moment, but for now, let's examine how we would express a name-value pair in formal notation.

▶ **Definition 1.** A *unary relation* expresses the idea that some Rensa attribute named ϕ has some value x, and is written $\phi(x)$.

Example. To specify the idea that we are using the interrogative grammatical mood, we can use the unary relation *grammatical_mood(interrogative)*.

Unary relations build into more complex expressions: *binary relations* and *assertions*. In SimpREL, a binary relation is a relationship between two concepts, and an assertion is a binary relation with additional attributes.

▶ **Definition 2.** A *binary relation* describes a relationship with value ψ existing between two values x and y, and is written $\psi(x, y)$.

Example. The statement *Bilbo lives in the Shire* may be expressed as *lives_in(Bilbo, the Shire)*.

▶ **Definition 3.** In SimpREL, an *assertion* is described as an unordered set of unary relations $\{\phi_1(x_1), ...\phi_n(x_n)\}$. It is typically understood as a logical conjunction of a

binary relation $\tau(\lambda, \rho)$ and zero or more unary relations.

$$\{l(\lambda), relation(\tau), r(\rho), \phi_1(x_1), \phi_2(x_2), ...\phi_n(x_n)\} \Leftrightarrow \tau(\lambda, \rho) \wedge \phi_1(x_1) \wedge \phi_2(x_2)... \wedge \phi_n(x_n)$$

Remark. Here, the symbols λ and ρ denote the values of the *left-hand* (l) and *right-hand* (r) concepts of the assertion, respectively. The symbol τ represents the value of the relationship between them, which is named *relation*.

Relation	**Definition**	**Example**
is_a / type_of	A hyponym λ is a type of hypernym ρ (*class inclusion*).	*is_a(bird, animal)*
instance_of	λ is a specific, named instance of ρ (*class membership*).	*instance_of(Charlotte, spider)* *instance_of(princess, hero)*[1]
part_of	A meronym (constituent) λ is a part of a holonym (entity) ρ (*composition*).	*part_of(wheel, automobile)* *part_of(book, library)*
member_of	A meronym (member) λ is a member of a holonym (container) ρ (*containment*). ρ cannot exist without λ.	*member_of(element, set)* *member_of(student, class)*
has_a	A holonym (whole) λ has a meronym (part) ρ. ρ can exist without λ (*aggregation*).	*has_a(library, student).*

Table 5.1: Classic SimpREL relation types.

5.2 Relations

We have established that each SimpREL assertion involves a unary relation $relation(\tau)$. In this section, we will discuss a number of relation types, i.e., standard values for τ.

[1] In this case, *princess* is the only name we have for the person in question. Many characters (including protagonists!) in old fairy tales lack proper names, but still have some keyword that describes their essence.

Relation types that correspond with traditional relationships in knowledge representation are called *classic* relations (Table 5.1). For example, an aggregation relationship may be expressed using a *has_a* relation. Beyond *has_a*, Rensa provides support for more specific possession relationships. Namely, any λ can *has_x* ρ, where x can be any noun phrase represented as a string with all spaces replaced with underscores. Examples are listed in Table 5.2.

Relation	**Definition**
has	λ has a prop ρ.
has_property	λ is ρ.
has_name	λ's name is ρ.
has_gender	λ's gender is ρ.
has_belief	λ has the belief ρ.
has_goal	λ has the goal ρ.
has_intention	λ has the intention ρ.

Table 5.2: Example SimpREL variable relations of the form *has_x*.

Any has_x relation can be paired with a *find_x* relation. For instance, consider *has_shoe_size(Cinderella,perfect)*. One might imagine that Prince Charming has the goal to *find_shoe_size* of Cinderella. As another example, perhaps someone wants to *find_identity* of a superhero. These types of expressions (*variable relations*) provide for greater expressive range and ease for automated interpretation. Further examples of variable relations include *set_x*, *x_for*, and *x_of* relations. Let us consider each of these categories in turn.

The *set_x* relation is used to specifically set the value of a property. We could *set_favorite_color* to *blue*, for example. If we were building a system that could be issued

commands, this assertion could be transformed into a has_x assertion (*has_favorite_color blue*).

The *used_for* relation is likely the most frequent example of *x_for*. The relation *used_for* specifies the function of a concept (e.g., a spoon is *used_for* scooping).

We have already seen several *x_of* relations such as *part_of* or *type_of*. One may use any *x*, of course, but several common *x_of* relations are *capable_of* (λ can ρ), *symbol_of* (λ represents ρ), and *reminds_of* (λ is reminiscent of ρ).

Another variable relation that can be used is *location_x*. In this case, *x* specifies how λ is located with respect to ρ. For example, *location_at*(λ, ρ) translates to λ *is located at* ρ, while *location_on_top_of*(λ, ρ) translates to λ *is located on top of* ρ.

Comparison relations help us form comparative structures. These often take the form of *x_than* or *more_x_than*, given some comparative x. For example, *greater_than*, *stronger_than*, and *more_interesting_than* are all examples of *x_than* comparison relations. All assertions with *x_than* relations may use attributes of *num_relation* and *relation_unit* to specify the amount and unit type of the distinction between λ and ρ. This allows us to encode constructions such as *the potato is heavier than the carrot by 50-150 grams* or *there are two more apples than pears.*

The *x_as* relation can be used when two concepts are similar. If λ and ρ are identical, for instance, we could say *same_as*(λ, ρ). If, however, we wish to specify how λ and ρ are similar, we could use a more precise term for *x*, such as *popular*, *soft*, or *intelligent.* With this relation, we can represent ideas like *he smiled as calmly as the moon in the midst of night.*

Beyond classic and variable relations, one may use the relation *action* to specify that λ is taking action ρ. Attributes that further explain the details of action assertions will be explored in Section 5.4.1. SimpREL also recognizes relations which correspond with directed arcs of discourse (Table 5.3) in a manner similar to Elson [32].

Relation	**Definition**
causes	The effect of λ is ρ.
ceases	λ prevents ρ.
implies	λ infers ρ.
interpreted_as	λ is equivalent to ρ.
precondition_for	λ is necessary for ρ's actualization.
precondition_against	λ is necessary for ρ's prevention / cessation.
harms	λ negatively impacts ρ.
helps	λ positively impacts ρ.

Table 5.3: Example SimpREL discourse relations.

5.3 Basic Concepts

The relations we have defined so far have given us great expressive power. With them, we can categorize concepts or link them together. Let's now consider some of the basic concepts we may want to define and relate over the course of a story.

Many stories are about notable people, animals, things, or places. To make this observation, we need look no further than the titles of popular books for young adults. *The Lord of the Rings*, *Harry Potter and the Sorcerer's Stone*, and *Charlotte's Web* all clearly state that they involve a central thematic character as well as one or more objects. *The Secret Garden* is an example of a narrative which centers around a

notable place.

In SimpREL, places are termed *locations*. People, animals, and things are all subclasses of *entities*. An *entity* is any thing that exists - abstractly or physically - in the storyworld. A person may be an entity, just as much as a book, or an idea. An entity is typically referred to by a string literal, which is either a name (such as "Smaug") or a keyword in lieu of a name (e.g., "dragon").

An *actor* is an entity that is capable of having intentions and performing actions, as in [69]. Notice that actors are not confined to being human: they may be a non-human species, or even something that we normally think of as an object (such as a gingerbread man).

To define an actor *a* in SimpREL, we can say:

```
{
    "l":["a"],
    "relation":"instance_of",
    "r":["actor"]
}
```

Rensa now will understand that there exists an actor with the name *a*, and will predict the actor's gender based on this information. You can override the gender prediction by specifying that *a has_gender g*, for some desired gender *g*.

A Story object recognizes that every actor is associated with a name, a set of pronouns (personal, object, and possessive), relationships with other actors, and assertions to which they are personally connected. Additionally, each actor may have their own brain containing assertions. This means that actors may understand facts

about each other and the storyworld.

5.4 Attributes

We can now define actors and other entities in our story, but we still lack the means to encode how, where, and when they interact. This section will define additional SimpREL attributes that let us add detail to our encodings. In other words, we will examine some standard values for ϕ and x in a SimpREL assertion.

5.4.1 Common Attributes for Actions and Conditions

Any given assertion may have the *owner* attribute, whose value is a string array containing all actors that contain this assertion in their brain. If the *owner* of an assertion is not specified, Rensa assumes it is common knowledge for everyone. Similarly, one may specify values for *x_owner*, given some other attribute named x. The value of this attribute is a list of actors to which x's value belongs. For example, an assertion with the following name-value pairs:

```
{
    "l":["shoes"],
    "relation": "has_property",
    "r": ["blue","suede"],
    "l_owner":["Carl"]
}
```

...would be realized as *Carl's shoes are blue and suede.*

As we've seen, one may use the relation *action* to specify that λ is taking

action ρ. The attributes listed in Table 5.4 may be used to provide more detail about the action being performed. Names used for the actions themselves are encouraged to be as simple as possible (as in all parts of a SimpREL encoding). Often, these are a verb in infinitive form without the "to" particle. The setting and quantitative conditions of an assertion, action or otherwise, are also important to encode. SimpREL attributes that further specify place, time, and amount are listed in Table 5.5.

Name	**Definition**	**Example**
with_property	Indicates how the action is performed.	*Ariel* **sadly** *gave her voice to Ursula.*[2]
action_object	Specifies the object to which the action is being applied.	*Ariel gave her* **voice** *to Ursula.*
action_object_owner	Specifies whose action object it is.	*Ariel gave* **her** *voice to Ursula.*[3]
action_recipient	Specifies the receiver of an action. These constructions often involve the preposition *to*.	*Ariel gave her voice to the* **sea witch**. *Ariel gave her voice to* **Ursula**.
***x*_property**	Indicates properties that are possessed by x, where x is any other attribute. Examples for x include *action_recipient*, *action_object*, and so on. (This attribute can also be used with non-action assertions.)	*Ariel gave her* **beautiful** *voice to Ursula.* *Ariel gave her voice to the* **terrifying** *sea witch.*
method	Specifies the means by which an action (or other assertion) is performed.	*Ariel gave her beautiful voice to Ursula by* **singing and signing the contract**.

Table 5.4: Common SimpREL attributes to specify action.

[2]The value of with_property can be expressed as either ["sad"] or ["sadly"] in the assertion, although the former is favored because it is concise. This attribute may also be used if the relation in question is a verb.

[3]Here, the action_object_owner would be specified as ["Ariel"]. Rensa then substitutes "her" in the realization by using knowledge about Ariel's possessive pronoun.

Name	Definition	Example
location_*x*	Similar to the *location_x* relation. Indicates the relative location(s), usually of an action, and the relative location object.	*We looked* **on top of the refrigerator**.
frequency	How often this assertion occurs (e.g., *often*, *always*, *twice*, *twice daily*).	*He smiled* **often**.
time_*x*	Specifies a relative time or duration. Examples for *x* include *begin*, *end*, *at*, *after*, *before*, and *for*.	*Boil the egg* **for three minutes**.
condition_*x*	Specifies conditions for the assertion, where *x* is a preposition or conjunction. Examples for *x* include *if*, *until*, *unless*, *without*, and *during*.	*Bake* **until the bread is golden brown**. *Boil the egg* **unless you have a boiled egg**.
num_*x*	Specifies the number of *x*, where x is any other attribute. Examples for *x* include *l*, *r*, and *action_object*.	*I have* **two** *strawberries*.
***x*_unit**	Specifies the unit for *x*, where x is any other attribute.	*Preheat the oven to 350* **degrees Fahrenheit**.

Table 5.5: Common SimpREL attributes to specify place, time, and amount.

5.4.2 Story Attributes

In SimpREL, we can organize assertions into Story objects. Story objects contain an array of known actors and a list of *scenes*. Each scene is associated with an optional setting, and actors present in the scene. The setting may contain information regarding time of day (such as "morning" or "9AM"), location (such as "the library"), and a more general location (such as "exterior", "interior", or both).

Narrative assertions may contain an optional name/value pair which specifies local event ordering within a scene. The name in this pair is called *storypoints*. Storypoints allow us to specify which assertions are added, removed, present, or absent in a particular scene. The value of a storypoints pair is a list of assertions, each of

which either specify a single time point or an interval of time in a scene. The end time point may be specified as "infinity", which indicates the assertion occurs until the end of the scene. If no storypoints are specified for a narrative assertion, it is assumed that assertion is true for all time points.

Other story-specific attributes include *discourse*, which exactly specifies what an actor says if they are speaking, and *text*, which contains the original natural language (or other signs) used to convey the encoding. When using *discourse*, there is an additional attribute *spoken* with a single Boolean value. If true, the discourse will be realized as speech (*"I do really wish to destroy it!" cried Frodo*). Otherwise, Rensa will realize the discourse as prose (*Frodo cried that he did really wish to destroy it*).

SimpREL also recognizes attributes that provide additional control over Story realization. The *grammatical_mood* attribute indicates the grammatical mood for a realized assertion. By default, the value of this attribute is *indicative*. However, one may specify further moods, including *imperative* and *interrogative*. Similarly, the *tense* attribute specifies the grammatical tense for a realized assertion. The value of this attribute is *present* by default. Other values for tense include *past* and *present participle*.

5.4.3 Belief Attributes

Over the course of a story, actors may be thinking about ideas (*beliefs*) or how ideas are connected (*linked beliefs*). In SimpREL, the *belief* attribute specifies a single belief object of an assertion. In the statement *Saruman wants Gandalf to believe* **he is trustworthy**, the bolded portion represents the value of the belief.

Linked beliefs are described using attributes named *linked_belief_nodes* and *linked_belief_edges*. The value of the former is a list of assertions representing a graph of beliefs, such that each assertion is associated with a node label. The value of the latter is a list containing each edge specification between linked belief nodes. For a detailed example of how belief attributes are used and realized, refer to Appendix A.

5.4.4 Miscellaneous Attributes

One useful SimpREL attribute we have yet to mention is *tags*, which lets us search assertions by non-hierarchical terms. The value of tags is a list of strings.

Another handy attribute is *prefix*. The value of this attribute is treated as if it were appended to the start of the relation. For example, any *has* relation can be preceded by *can* to indicate that λ has the capability to have ρ. Values for the prefix attribute include: *can*, *not*, *try*, and *might*. Multiple prefixes may be used in a single assertion.

Other attributes may be added to provide model-specific support using similar conventions to those mentioned here (as in the usage of variable relations). The primary rule of SimpREL is to be simple.

5.5 The Empty Slot Variable

Sometimes, we may want to specify that the value of an attribute is empty. To indicate this concept in SimpREL, we use the value *SVAR_x*, where x is the unique name of the variable. SVAR stands for slot variable or space variable.

It may be that we simply wish to use a SVAR as a filler value. A SVAR may also be used to indicate that something more concrete should fill the space. As one example, consider an actor named Hermione in the domain of the *Harry Potter* universe. We might wish to give Hermione the knowledge that *many wizards are afraid of saying Voldemort's name*, without stating precisely the wizards to whom this assertion applies. Listing 5.1 shows one SimpREL method for encoding this statement. Note that we can specify the qualities that apply to SVAR_1 using other attributes, such as the condition_x attribute.

Attributes that specify SVAR qualities also permit us to search, if we wish, for an appropriate concrete value to replace the SVAR. We could, for example, ask Hermione to think of all the actors that could fill the space of SVAR_1. The first search condition specifies that any SVAR_1 must be an actor, and - furthermore - a wizard. The condition that *SVAR_1 has_property plural* suggests that the answer demands either a name representing a group of actors, or a list of more than one actor. The remaining key search condition specifies that Albus Dumbledore and Harry Potter are not acceptable values for SVAR_1.

In this fashion, any SVAR may be filled with some response appropriate to the call (search conditions). This idea is quietly reflected in the term *svar* itself, which is the imperative form of *to respond* in Swedish.

Listing 5.1: A SimpREL encoding that translates to *Many wizards (except for Harry Potter and Albus Dumbledore) are afraid of saying Voldemort's name.*

```
{
    "l": ["SVAR_1"],
    "relation":"action"
    "r":["fear"],
    "action_object":[
      {
            "l":["SVAR_1"],
            "relation":"action",
            "r": ["say"],
            "action_object":["name"],
            "action_object_owner":["Voldemort"]
        }
    ],
    "condition_given":[
      {"l": ["SVAR_1"],"relation":"instance_of","r":["actor", "wizard
          "]},
      {"l": ["SVAR_1"],"relation":"has_property","r":["plural"]},
      {"l": ["SVAR_1"],"relation":"has_generic_num","r":["many"]},
      {"l": ["SVAR_1"],"relation":"has_name","r":["Harry Potter","
          Albus Dumbledore"],"prefix":["not"]},
    ]
}
```

5.6 Referencing Other Assertions

Rensa automatically assigns an ID to each assertion added to a brain. ID assignment takes the form of a linear sequence: the first assertion is assigned an ID of 1, the second, 2, and so on. Hence, the *nth* assertion added to the brain will have an ID of n. We can refer to IDs in a SimpREL assertion by using the syntax "$(x)", where x is the ID of the desired assertion.

We can also assign custom IDs to assertions by setting the value of an attribute named *myID* to the desired string. These custom IDs can be referenced in the same way as automated IDs.

5.7 Logic Support

Although SimpREL is meant to represent simple narrative concepts and their relationships, there is some support for more complex logical expressions. The value of the relation attribute could be any logical operator (e.g., "AND", "OR", "XOR"). Furthermore, some types of logical expressions can be represented more concisely. As you may have surmised, a list of values in SimpREL is equivalent to an expression that connects each value by the Boolean operator AND (Listing 5.2).

Listing 5.2: A SimpREL encoding for *blueberries and strawberries are types of fruit.*

```
{
    "l":["blueberries", "strawberries"],
    "relation": "type_of",
    "r": ["fruit"]
}
```

We can include inner list structures to represent each set of expressions to be connected by an OR operator. Any individual concept can be enclosed by a set of parentheses preceded by an exclamation point to signify the NOT operator. An example of OR and NOT usage is provided in Listing 5.3.

5.8 Automated Extraction Support

By now, we have investigated the SimpREL encoding scheme and seen a few examples of how we might use it. As mentioned in the previous chapter, Rensa helps users to automatically realize SimpREL encodings into natural text. The reverse process - extracting SimpREL assertions from text - is another useful task to consider.

Generally, the process of extracting information to use within the Rensa framework involves formalizing an input document D = $\langle C_d, R_d \rangle$ as a set of concepts $C_d = \langle c_{d0}, c_{d1}, ...c_{dn-1} \rangle$ and relationships $R_d = \langle r_{d0}, r_{d1}, ...r_{dm-1} \rangle$. Researchers in knowledge and open information extraction are currently studying how to optimize this process [37, 35, 124]. While the pursuit of a perfect information extraction system is well beyond the scope of this thesis, we will discuss some current achievements as well as baseline

algorithms included within the Rensa framework for the casual user.

Listing 5.3: A SimpREL encoding for *make a pie with blueberries, or a pie with strawberries and no rhubarb.*

```
{
    "l":["chef"],
    "relation":"action",
    "r":["make"],
    "action_object":["SVAR_2"],
    "condition_given":[
      {"l":["SVAR_2"],"relation":"instance_of","r":["pie"]},
      {
        "l":[["blueberries"],["strawberries","!(rhubarb)"]],
        "relation":"part_of",
        "r":["SVAR_2"]
      }
    ],
    "grammatical_mood":"imperative"
}
```

Several concept relation extraction tools exist, including Alchemy, CiceroLite, FRED, and ReVerb. Table 5.6 provides Gangemi's analysis of these tools [42]. There is no commonly-accepted technique for relation extraction; hence, these tools make different assumptions about the task and identify different types of relation patterns. Alchemy and ReVerb assume that relationships are always binary: *they gather more weapons* becomes *gather(they, more weapons)*. CiceroLite takes a slightly more complex approach, striving to first guess the relation's arity before producing *gather(they,*

Tool	p	r	$F1$	a
Alchemy	.69	.25	.37	.30
CiceroLite	.90	.20	.33	.25
FRED	.84	.82	.83	.82
ReVerb	.67	.23	.34	.27

Table 5.6: A comparison of relation extraction tools by one of the creators of FRED [42]. Each tool was assessed by computing precision (p), recall (r), F-measure ($F1$), and accuracy (a).

weapons) as the extracted relation.

In contrast to the other three tools, FRED relies on neo-Davidsonian semantics, which produces an event-centric graph representation. Using the same example of *they gather more weapons*, FRED defines the event *gather_1*, which is associated with an agent *thing_1* and theme *weapon_1*. Event, agent, and theme can then be associated with more concepts. For example, weapon_1 is defined as having the quality *More*, and has the type *MoreWeapon*, which is a subclass of *Weapon* (Figure 5.1).

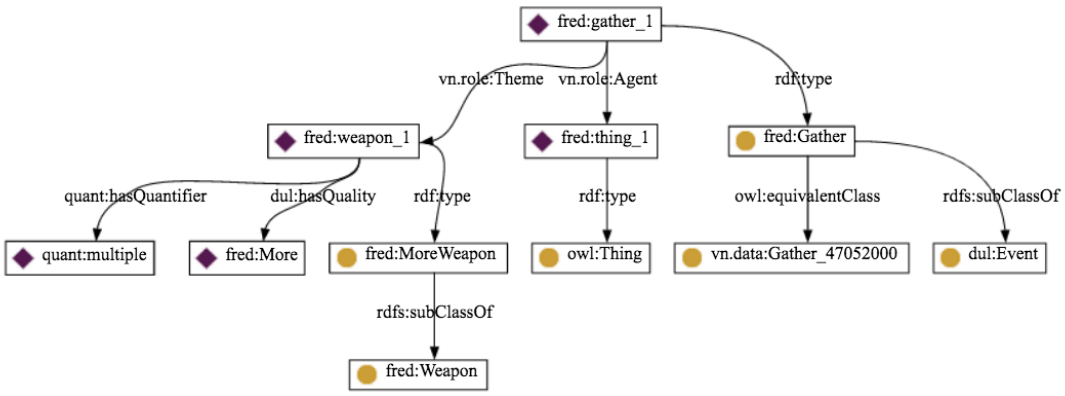

Figure 5.1: The graph produced by FRED [43] for the example *they gather more weapons*.

These types of tools may prove to be useful for Rensa or a Rensa-like framework in the future. For now, however, there is much to be improved, especially when more complex sentences are encountered. Consider Gangemi's proposed example for comparison: *it plans to blacklist the Nusra Front as a terrorist organization.* This sentence contains ambiguity and multiple arguments (among other complexities), making it difficult for tools to extract usable information. In this case, ReVerb does not extract anything. The results for the remaining tools are as follows:

- Alchemy:
  - *plans to blacklist(it,the Nusra Front as a terrorist organization)*
- CiceroLite:
  - *plans to blacklist(it,front,(AS) a terrorist organization)*
- FRED:
  - *experiencer(plan_3,United States)*
  - *theme(plan_3,blacklist_3)*
  - *agent(blacklist_3,United States)*
  - *patient(blacklist_3,NusraFront)*
  - *as(blacklist_3,organization_3)*
  - *TerroristOrganization(organization_3)*

Gangemi concluded that such tools were promising, but called for improved knowledge extraction benchmarks.

While these approaches and evaluation methods are being perfected, I have included several baseline functions in Rensa for SimpREL extraction from natural text. These functions scan the text for combinations of certain parts of speech in the form of rules. For example, assume the common linguistic tags for adverbs (RB), verbs (VB), possessive pronouns (PRP$), adjectives (JJ), and nouns (NN). Then, one rule is "n was (DT) (RB) JJ", where $n \in A$, the set of all actors identified in the text, and any tag in parentheses is optional. Actors are identified by searching the text for known proper names, or by recognizing that a noun has actor properties (e.g., gender or animacy). Tags in parentheses mean that an instance of that tag is optional in a matched sentence. This rule will match a construction such as *Samwise was brave*, resulting in the SimpREL assertion:

```
{
  "l": ["Samwise"],
  "relation": "has_property",
  "r": ["brave"]
}
```

Any extracted assertion is also given values for the *storypoints* and *tense* attributes. Tense is determined by assessing the original grammatical tense of the matched verb. In this case, the tense will be assigned a value of *past*. The assertion is also considered to be *at* a storypoint value equal to how far we are in the story by using a sentence counter.

A more complex rule is "n (RB) VB (PRP$) (JJ) NN to m", where $n, m \in A$. This example matches constructions such as *Marinette excitedly presented her incredible*

plan to Daniel. This construction is then translated into its corresponding SimpREL assertion:

```
{
  "l":["Marinette"],
  "relation": "action",
  "r":["present"],
  "action_object":["plan"],
  "action_object_property":["incredible"],
  "action_object_owner":["Marinette"],
  "action_recipient":["Daniel"],
  "with_property":["excited"]
}
```

The value for action_object_owner is only assigned if the possessive pronoun appears to match with what the system knows or can infer about Marinette. That is, if she has been explicitly assigned a gender, the system confirms that the pronoun matches that gender before making the assignment. Otherwise, it will guess Marinette's gender using historical data about baby names. If the prediction corresponds with the extracted possessive pronoun, the assignment will take place.

To demonstrate how this kind of simple concept extraction can be useful, I investigated how one might automatically classify protagonists from antagonists. Various prior work has focused on categorizing story actors by extracting story information. Some have proposed using frequency of occurrence to rank actors by importance [22], but this method, while useful and efficient, does not employ full semantic understanding. Declerck and others introduced a hand-crafted ontology-based method for character

type detection in fairy tales via analysis of nominal phrases, but these results are not easily generalized to other types of stories [26]. Meanwhile, Goh and others proposed identifying the main protagonist by means of syntactic features and a semantic lexicon, but did not extend the method to categorizing any other story actors [52]. In response, Karsdorp and Bosch introduced a system that ranks character importance by using direct and indirect speech [69]. However, their system did not categorize actor types - such as the distinction between a protagonist and antagonist - which is useful for understanding underlying intentions and behaviors over the course of the story. In contrast, Bamman and others proposed latent variable models for learning character types from movie plot summaries, which are packaged with actor descriptions such as "rebel leader Princess Leia" [6]. Their method included relying on agent verbs (verbs for which the entity is an agent argument), patient verbs (verbs for which the entity is the patient, theme, or other argument), and attributes (adjectives and common noun words that relate to the mention).

These approaches inspired me to explore how Rensa could support actor classification. I first labeled 207 actors from a corpus of 53 film scripts as "protagonist" (anyone the audience is supposed to support), and "antagonist" (any villain who gets in the way of a protagonist). For each script, I found the most frequent actor mentioned, which served as the baseline for protagonist detection (as in [22]). I then extracted attributes that were associated with those actors, similar to [6], to determine if such methods would apply to film scripts in addition to movie plot summaries. This process helped to identify clusters of adjectives that were commonly associated with each actor

type (Figure 5.2).

Figure 5.2: Generated word cloud that depicts the major clusters of adjectives associated with protagonists (blue) and antagonists (red) as extracted from 53 openly available film scripts in the action/adventure genre from *The Internet Movie Script Database* (www.imsdb.com). In this corpus, antagonists are often described as huge. Protagonists tend to be described with references to their age (such as "young"), suffering (e.g., "poor", "ragged"), or a temporary state of debilitation (e.g., "unconscious").

To understand how these adjectives assist in actor role classification, consider the script of *The Lord of the Rings: Return of the King*. The leading protagonist, Frodo, is easily classified as a protagonist because his name appears the most frequently. However, the rest of the protagonists' names were not listed as often - some even less than unnamed antagonists, such as the Orcs. To identify their actor types, we can examine words associated with them. For example, Frodo calls his friend Samwise "dear". Sam also can be mapped to the *pitiable* cluster, since Frodo expresses his pity for him ("I'm so sorry, Sam...."). Orcs, in contrast, are associated with words that describe their enormous size ("huge", "massive"), which is a trait commonly linked with antagonists. In this way, a system relying on Rensa assertions was able to predict that, in the film, Sam is a protagonist while orcs are antagonists. Results of the actor role classification experiment are displayed in Table 5.7.

Actor Type	Baseline			Experimental		
	p	r	$F1$	p	r	$F1$
Protagonist	1.00	0.23	0.38	0.97	0.94	0.96
Antagonist	0.35	1.00	0.52	0.87	0.93	0.90

Table 5.7: Results of actor role prediction (p=precision, r=recall, $F1$=F-measure). The best results using the baseline and the attribute extraction method were found using Naive Bayes and a decision tree, respectively.

Beyond deducing actor roles, film scripts generally require setting information to be explicit, which is useful for building story objects automatically. Information particular to each scene in the film script corpus was obtained by extracting all descriptions and discourse that occurs between scene marking notes (e.g., "INT." and "EXT."). The lines containing such markers specify the setting, which includes the time and place of the scene. In this way, the Rensa-based system identified setting references by using text processing techniques combined with WordNet's time categorization lexnames to delineate scenes, locations, and times with accuracies of 99%, 100%, and 93%, respectively.

Film scripts, of course, are only one type of narrative from which we can learn information, and the usefulness of rule-matching approaches is limited. It is promising that researchers are interested in finding improved methods to derive concept relations from natural text. However, it is also encouraging that we can automatically encode and infer so much from simple extraction techniques. Beyond assisting in intelligent reasoning and prediction, this type of concept extraction enables Rensa assertions to

be encoded using a higher-level format than JSON, i.e., through simplified natural text sentences. Future research will likely expand and improve on existing machine learning methods for Rensa-supported concept extraction.

5.9 Discussion

We have discussed the SimpREL encoding scheme as one means of encoding knowledge with Rensa. SimpREL provides a simple means of representing both meaning (textbase) and storyworld (situation model). Theory of mind is supported through actor brains and assertion nesting. Furthermore, the goals of research in open information extraction are closely aligned with deriving structures that can be encoded in SimpREL format, which signifies that a robust bridge between Rensa encodings and natural text may be possible in the future.

Chapter 6

Support for Existing Representations

"A tremendous amount of knowledge is needed to understand even a simple sentence."

— David L. Waltz

We observed in Chapter 2 that there are many competing theories of narrative structure. One of the primary goals for Rensa's development is to be able to express and transform distinct narrative representations to advance research in understanding and comparing them. As such, the following sections will discuss how Rensa can support the encoding of some existing representations for narrative and natural text.

6.1 Grammars

6.1.1 Story Grammars

Each component of a story grammar may be expressed via Rensa relations. We will consider Thorndyke's approach [116] first because it is the simplest. His grammar described the parts that make up a *Story*:

```
Story -> Setting + Theme + Plot + Resolution
```

Similarly, he defined the *Setting*, *Theme*, *Plot* and *Resolution* in terms of their respective components:

```
Setting -> Characters + Location + Time

Theme -> (Event)* + Goal

Plot -> Episode*

Episode -> (Subgoal) + (DesiredState) + Attempt + Outcome

Attempt -> Event* or Episode

Resolution -> Event or State

Outcome -> Event* or State

State -> Characters or Location or Time

DesiredState -> Subgoal or Goal
```

In this notation, the symbol "+" indicates the combination of elements in sequential order. The parentheses around *Event* denotes that the element is optional. Finally, the asterisk suggests that the associated element may occur more than once. As an example, a plot is defined as a series of episodes.

Rules like these can be expressed using a *part_of* relation in SimpREL. For example, we could write the rule for *Theme* as:

```
{
    "l": ["(Event)*","Goal"],
    "relation": "part_of",
    "r":["Theme"]
}
```

Similar assertions can be constructed for the remaining element definitions.

An *Attempt*, for example, could be represented as:

```
{
    "l": [["Event*"],["Episode"]],
    "relation": "part_of",
    "r":["Attempt"]
}
```

Story grammars beyond Thorndyke's approach include semantic relations in addition to the syntactic relations we have just discussed. For example, Mandler and Johnson's method requires that Attempts cause Outcomes. In SimpREL, we could express this as:

```
{
    "l": ["Attempt"],
    "relation": "causes",
    "r":["Outcome"]
}
```

Because Rensa allows any string to encode a relation type, we are free to use any further name for a relation as we wish.

6.1.2 Universal Dependencies

The notion of *dependency* in linguistics is the idea that words - or other units of language - are connected to each other with directed links (*dependencies*). Rensa generally supports this idea by allowing strings or sets of strings to be connected by relations, and permits elaboration of these relationships via the addition of optional model-specific attributes. The direction of the relation is assumed to be from the left-

hand concept to the right-hand concept, but could be further specified in an additional attribute.

Dependencies are heavily used in narrative representations. As one example, the Universal Dependencies (UD) project was founded to advance research in multilingual parsing and learning [24]. Generally, a UD tree or graph can be reduced to a set, where each member of the set is a relation between left-hand and right-hand concepts.

6.1.3 DSyntS

The DSyntS (*Deep-Syntactic Structure*) is a unordered syntactic dependency tree that can be realized into natural text using the RealPro system [75]. Through their PERSONAGE system, Walker et al. have produced surface-level distinctions in DSyntS format [104]. Thus, if our framework can support DSyntS encodings, it may also support surface-level narrative transformations.

The DSyntS tree is composed of labeled nodes and arcs that may be nested. Each node structure can be represented as a Rensa assertion, with labeled syntactic relations as attributes. Each node may have an attribute *nodes* which contains a list of inner nested nodes represented as assertions. An example is shown in Listings 6.1 and 6.2.

Listing 6.1: A DSyntS representation for the utterance *I don't believe it.*

```
<dsyntnode class="verb" lexeme="believe"  polarity="neg">
  <dsyntnode lexeme="<PRONOUN>" number ="sg" person="1st" rel="I"/>
    <dysntnode lexeme="it" rel="II"/>
</dsyntnode>
```

Listing 6.2: The SimpREL equivalent for Listing 6.1.

```
{
  "class": "verb",
  "lexeme": "believe",
  "polarity":"neg",
  "nodes": [
    {"lexeme":"<PRONOUN>", "number":"sg", "person":"1st", "rel": "I"},
    {"lexeme":"it", "rel", "II"}
  ]
}
```

6.2 Frame-Based Models

Frames and scripts (a sub-type of frames) rely on the assumption that for a given situation, there is a template with slots that must be filled in order to understand the situation. A *frame* is a data structure that can be represented as a network of nodes and relations [91], which can neatly map to Rensa encodings. Each slot in a frame template can have values including entities (references to an object in the world) and pointers to other frames. In SimpREL, we can represent entities by their name or

keyword, and reference other assertions by their ID number. Listing B.1 in Appendix B provides a complete example of how a SimpREL-like format could encode the restaurant script, a classic example of Schank's script theory.

6.3 Plot Unit Representations

6.3.1 Bremond's Approach

Bremond suggested that narratives contain at least one *elementary sequence* consisting of a goal, an action or event towards the goal, and an outcome [16, 17]. Similar to grammars, we can use the *part_of* relation to describe this idea in SimpREL. A narrative may then take place by finding and requesting each part of the narrative to execute.

Agents may have goals and can take actions that may affect the outcome. Given some $a \in A$, where A is the set of all actors, one may represent the former in SimpREL by *has_goal(a,b)*, where *instance_of(b,goal)*. An actor a taking an action c can be represented as *action(a,c)*, and c can be associated with some new outcome state via a *causes* relation. Agents also have affect states and relationships, which can be designated via *has_x* relations. They can interact with each other in a number of ways, including intervention, negotiation, aggression, retribution, sacrifice, and punishment. These types of interactions can be specified via *action* assertions that indicate an *action_recipient* and any associated consequences.

6.3.2 Plot Units

Plot units contain structure hierarchies which build toward narrative patterns [76, 53]. The lowest level components are three types of affect states: positive (+), negative (-), and mental or neutral (M) states. Each affect state can be categorized as such with a *type_of* relation and associated with an entity of type *actor* who *has* that state at any given storypoint. These affect states build into primitive structures, which "consist of two affect states associated with the same character and a causal link between them" [53]. Such primitives may be represented in Rensa as a binary relation *causes(a,b)* where *has_affect_state(c,[a,b])*, given that *instance_of(c, actor)* and also that *type_of([a,b], affect state)*. If a single event affects multiple characters, the affect states may be associated with a list of entities of type *actor*, rather than a single actor. These primitive structures may then be connected by additional assertions to construct the full story graph.

6.3.3 Doxastic Preference Frameworks (DPFs)

Doxastic preferences represent character expectations over the course of a narrative [80, 79]. A game tree structure is used to describe which outcomes the characters believe they and others would prefer. This representation demands a theory of mind approach as well as a means of expressing nested beliefs, which can be achieved in Rensa by nesting assertions that are attributed to actors. Refer to Listing B.2 in Appendix B for an example of how a DPF analysis might be represented using the SimpREL encoding format.

6.3.4 Story Intention Graphs (SIGs)

A SIG consists of three subgraphs (*layers*): *textual*, *timeline*, and *interpretative*. The textual layer represents the original utterances as a linear vector of nodes, which is then translated to a list of events and statives (*proposition* nodes) that occur over time in the timeline layer. The interpretative layer encodes the cognitive events of the text, as in the situation model recognized by cognitive psychology. Here, nodes and relations encode beliefs, intentions, goals, plans, and affectual impacts based on how the story is understood by some audience. Connections between nodes may occur within or between layers.

To support SIG encodings, SimpREL enables any generic assertion (or set of nested assertions) to be related to another assertion or set. Actors may be related to cognitive events via *has* relations (*has_belief, has_goal,* etc.). Temporal connections may be described via relations (e.g., *followed_by*) or through storypoints for additional flexibility. Many of Elson's original discourse relations are also recognized as standard, including *causes*, *precondition_for*, *attempt_to_cause* (by combining the prefix *try* with the *causes* relation), and *provides_for* (which is written as *helps*). There is also a bridge to first-order logic (as discussed in Chapter 4, Section 4), which is particularly useful in the case of the SIG because Elson created a corpus of SIG encodings in Prolog (*DramaBank*) [33]. An example SIG analysis is presented along with its SimpREL interpretation in Appendix B.

6.4 Narrative Event Chains (NECs)

A NEC is a partial temporal ordering of narrative events that share a common protagonist [21], inspired by frame-based models. Although NECs do not contain implicit information like plot unit representations (beliefs, goals, plans, etc.), they can be easily extracted from natural text [20].

Formally, a NEC is a set of narrative events $e_1, e_2, ..., e_n$. Each narrative event is a tuple with typed dependencies that specify an event (the simplest representation of which is a verb) and its participants. Any pair of events (e_i, e_j) may be connected by a temporal relation B, which is true if e_i occurs before e_j in time. SimpREL can represent the simple NEC events proposed by Chambers and Jurafsky via providing the capability to relate two sets of actors (which may be distinct, overlapping, or equivalent) with some string representing the event. Events (represented by their name, or assertion containing the full event) may then be connected temporally via a B relation. The NEC representation also makes use of an empty slot, which is represented by a SVAR in SimpREL.

6.5 Social Graphs

Ardanuy and Sporleder extracted characters and generated networks that described when interactions occurred between them [3]. In similar social graphs, each character may be represented as an entity of type *actor* in SimpREL. The frequency and type of interaction can be expressed by using the *num* and *unit* attributes, respec-

tively. These encodings may then be used to produce frequency network graphs.

6.6 Bartalesi's Ontology and OWL

Bartalesi's formal ontology uses the Web Ontology Language (OWL), which has also been used to support user modeling [64]. In OWL, we define a *class* to be a set that contains members (*individuals*) which all share one or more properties. We can state relationships between individuals or from individuals to values with *properties*. This idea is similar to the *has_property* relation in SimpREL.

To create class and property hierarchies, we can establish *subClassOf* and *subPropertyOf* relationships. Additional relations can be used to describe classes (*equivalentClass*, *unionOf*, *complementOf*, *intersectionOf*, *disjointWith*, *oneOf*), properties (*equivalentProperty*, *range*, *domain*, *hasValue*, *inverseOf*, *TransitiveProperty*, *SymmetricProperty*, *FunctionalProperty*, *InverseFunctionalProperty*), and individuals (*sameAs*, *differentFrom*, *AllDifferent*). Relations also exist to restrict how properties can be used with respect to a class (*allValuesFrom*, *someValuesFrom*, *minCardinality*, *maxCardinality*, *cardinality*, *intersectionOf*). Each of these types of relations are supported by SimpREL's left-relation-right encoding format.

Listing 6.3: An OWL encoding stating that two URI references are the same people.

```
<rdf:Description rdf:about="#Theodor_Seuss_Geisel">
  <owl:sameAs rdf:resource="#DrSeuss"/>
  <rdfs:comment>Theodor Seuss Geisel and Dr. Seuss are the same person
      .</rdfs:comment>
</rdf:Description>
```

Listing 6.4: A SimpREL equivalent for Listing 6.3.

```
{
    "l":["Theodor Seuss Geisel"],
    "relation":"same_as",
    "r":["Dr. Seuss"],
    "l_rdf: "about",
    "r_rdf: "resource",
    "rdf": "Description",
    "text": "Theodor Seuss Geisel and Dr. Seuss are the same person."
}
```

6.7 ConceptNet

ConceptNet is a semantic network that represents commonsense relationships between words and phrases, along with associated metadata. These relationships take the form of binary relations, which map exactly to the left-hand concept, right-hand concept, and relation attributes in SimpREL. The metadata is in the form of name-value pairs, and can also be neatly mapped to a Rensa assertion as model-specific attributes.

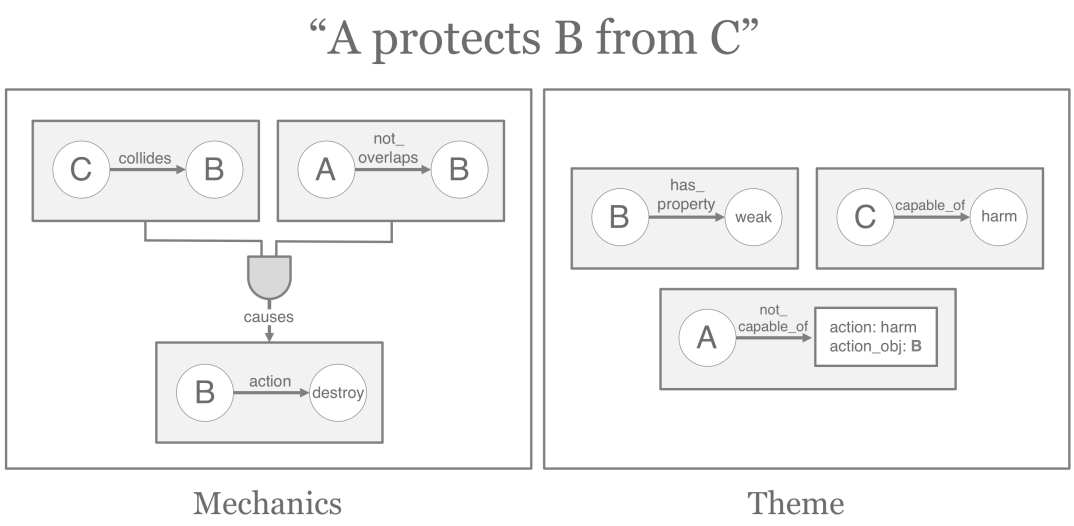

Figure 6.1: Example SimpREL concept mapping for one of the micro-rhetorics used by Game-o-Matic [118].

6.8 Game-o-Matic

Thus far, this thesis has focused largely on models of *narratology* (the study of narrative and its underlying structure) rather than *rhetoric* (the art of discourse). Rather than commit to a particular model of narrative structure, rhetoricians concern themselves more deeply with the interaction between storyteller and audience. This perspective is significantly distinct from the narratology view, but is useful to consider because it often plays a role in the development of interactive systems. Hence, let's examine how Rensa might be used to support a project founded in the theory of rhetoric: Game-o-Matic.

The Game-o-Matic project defined a game generation process that involved combining the rhetorical affordances of game mechanics with thematic elements to create

micro-rhetorics [118]. These micro-rhetorics represent a unit of the gameplay which could be supported by a *meaning derivation*: a formal hierarchical justification for the unit's interpretation and representation as an idea. Figure 6.1 shows how one might map an example micro-rhetoric to a SimpREL representation. Together, micro-rhetorics can be assembled and organized to form a larger game *recipe*, which can be expressed by linking micro-rhetoric concept objects together and specifying consequences of assembly.

6.9 Discussion

Rensa assertions can encode a variety of narrative representations as well as structures that are useful beyond traditional narratology, including game mechanics, user modeling, and ontology knowledge. This expressiveness permits the development of a translator between Rensa and other story representations. As a result, we can potentially expand a given annotation library automatically, and provide a source of comparison between representations.

Chapter 7

Narrative Simplification

"Simplicity is about subtracting the obvious and adding the meaningful."

— John Maeda

We have already established that narrative supports us in many different ways, such as by helping us to interpret the world around us and to express our unique perspectives. Among all of the use cases for narrative representations, we have not yet investigated *narrative simplification*: the process of reducing a narrative's complexity while preserving meaning. This is an important type of narrative reorganization that can help us communicate information in a more understandable and memorable way.

In computer science, narrative simplification is an important area of research to pursue because simplified texts are generally easier for automated systems to understand. Simplifying texts automatically also helps us gain insight into the most important units of narrative information to extract, which is useful for personalized summarization. Additionally, current models of machine translation perform worse with increased

complexity, and information retrieval systems that rely on content-based indexing may benefit from improved text simplification techniques. Text simplification has also been argued to be useful in increasing parser throughput and reducing timeouts by eliminating ambiguity [113].

Although text simplification systems exist, simplifying narratives on par with a human requires a rich interpretation of the original text, a representation of the target text, and a mapping between the two. Hence, narrative simplification is a compelling operation for us to consider, especially with a framework that combines the expressive power of multiple narrative representations in our toolbox.

7.1 The Index of Narrative Complexity

Research by Petersen et al. suggests that we can evaluate the progress of narrative learning by examining thirteen narrative elements in a student's story [100]. These elements have been combined into a formal measure called the Index of Narrative Complexity (INC). Enabling a computer to understand these elements and how they can be complex or simple is useful not only to eliminate redundancies for a machine, but to aid in automatically measuring narrative development. As detailed in Chapter 3, I demonstrated this idea by building the first narrative skill assessment tool based on the INC measure [60]. This system used the Rensa framework to encode and reason over Story Intention Graph representations of input text. Parts of the story were tagged using these encodings in addition to the original text. The relationships between

the resulting constructed networks were used to identify cognitive complexities such as actions, consequences, and complications.

To review how the system works, let's examine how it identifies several narrative elements recognized by the INC measure. First, we must define a few key terms. According to Petersen et al., an *initiating event* is an event or problem that elicits a response from an actor [100]. We will assume that an initiating event is always an instance of an event. Further, we will assume that for an actor to respond to an initiating event, there must be some *inspired goal* which prompts the response.

► **Definition 4** (Initiating Event, Inspired Goal)**.** Let E and G be the set of all events and goals in the input narrative, respectively. Assume $e \in E$, $g \in G$, and the SIG assertion $actualizes(e, g)$. Then, e is an *initiating event* and g is an *inspired goal*.

Remark. If distinct initiating events exist beyond e, these (excluding e) are each considered *narrative complications*. Complications can also arise in several other ways, as detailed by Figure 7.1.

Experts in natural language processing have extensively addressed surface-level lexical and syntactic complexities [11, 23, 45, 65, 68, 71, 98, 101, 103, 106, 111, 112, 123]. However, work that considers more abstract, cognitive complexities such as the INC narrative elements is considerably more infrequent. To do so is difficult because it requires a deeper understanding of the story beyond traditional analysis of explicit discourse features. The pursuit of understanding these complexities is fruitful, especially because these basic features exist even in relatively simple, well-known stories, such as fairy tales and Aesop's fables.

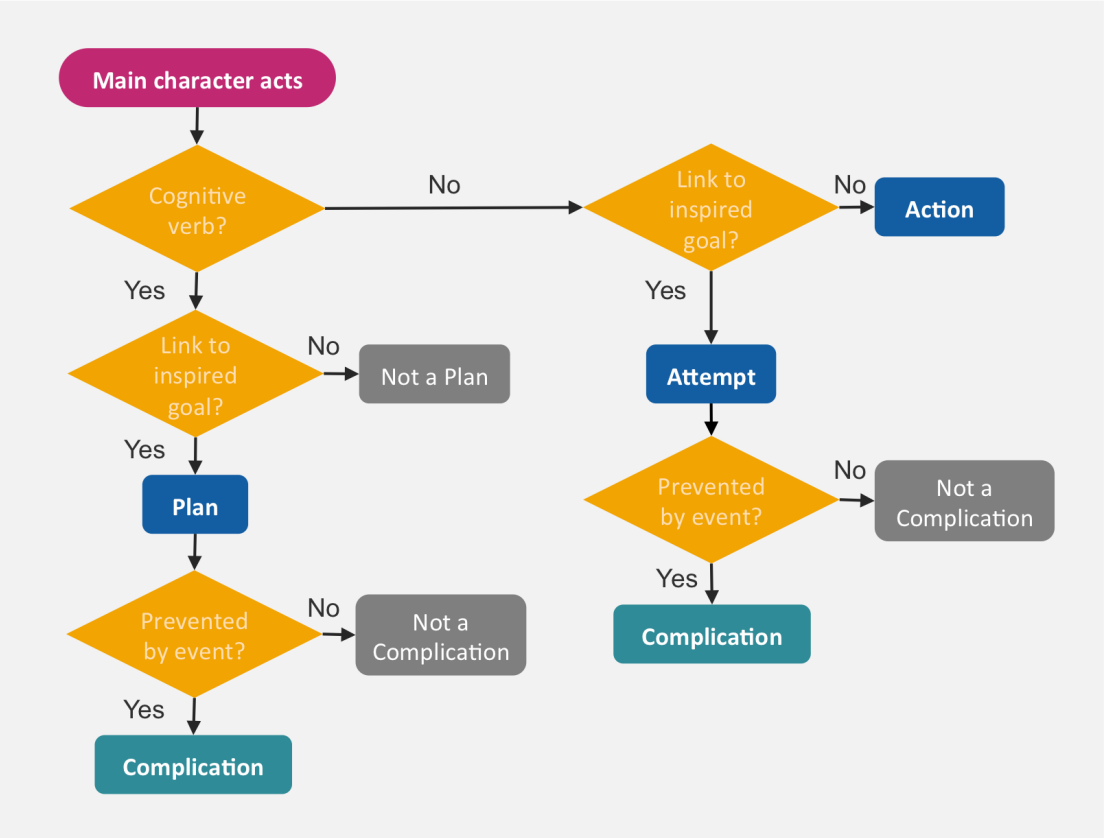

Figure 7.1: A simplified flowchart for identifying the occurrence of plans, attempts, actions, and certain kinds of complications. Let r be a relation such that r is one of *actualizes, ceases, prevents, attempt to cause, or attempt to prevent.* If some story character performs an act a, we must then consider if the description of the act contains a cognitive verb. If so, and if the act is related to an inspired goal h such that $r(a, h)$, then the act is tagged as a *plan.* If not, but $r(a, h)$, then a is an attempt; otherwise, it is an action. If a is an attempt or plan, any event f such that $prevents(f, a)$ is deemed a complication.

In the next sections, we will examine how Rensa might be used to identify and reduce these cognitive complexities. Specifically, we will consider two types of redundancies based on narrative elements recognized by the Index of Narrative Complexity: *characters* and *complications.*

7.2 Narrative Simplification with Rensa

7.2.1 Characters

A *narrative character* is a reference to the subject of a clause in a narrative. The INC measure specifically evaluates how a storyteller includes *main characters* with a story. This is done by noting how many main characters there are, and how the storyteller refers to their main characters. One may refer to a main character by ambiguous pronouns (such as "he"), nonspecific labels (e.g., "boy"), or specific names (e.g., "Harry"). We can represent main characters in Rensa by using entities, or - more specifically - *actors*.

As I hinted in the introduction to this chapter, we can find examples of narrative complexity even in very short stories, such as fables. For example, consider Aesop's fable of *The Lion, the Fox, and the Beasts*:

> *The goat, sheep, and calf believed the lion was dying. They all visited the lion. The lion asked the fox to visit him, too. The fox refused because he said the goat, the sheep, and the calf didn't return.*

Note that the goat, sheep, and calf all serve the same purpose for the story. This fable is an example of *actor redundancy*, i.e., a scenario in which the entire set of an actor's (or several actors') contributions are contained within that of another actor. More generally, it is evidence of greater *entity density*, which is related to text readability [38]. Let us now consider how we can use Rensa to eliminate redundant actors from a story.

★ **Algorithm 1** (Simplify Redundant Actors)**.** *Let H be the set which contains all*

actors in a narrative N with assertions $[n_0, ...n_m]$, and let $h_k, h_g \in H$ such that $h_k \neq h_g$. Let G and K be the set of assertions that contain h_g and h_k, respectively. Replace any instances of h_g with a SVAR in each element of G, and label the resulting set G'. Similarly, replace all instances of h_k with the same SVAR in every element of K to form K'. If $G' \subseteq K'$, then h_g is a redundant actor*, and G may be removed from N.*

Remark. A non-redundant actor may not be essential to the plot, but participates in one or more assertions in a unique way.

When applied to *The Lion, the Fox, and the Beasts*, Algorithm 1 results in the following simplified story:

> *The goat believed the lion was dying. He visited the lion. The lion asked the fox to visit him, too. The fox refused because he said the goat didn't return.*

In other words, we eliminate all actors whose only associated assertions are a subset of another actor's assertions. This general method for simplifying redundant actors has been validated by Mazeika [89]. We will discuss her Rensa-based system in the next chapter.

7.2.2 Complications

As previously mentioned, a *complication* is either an extra initiating event, or an event that prohibits the execution of a plan or attempt. Once all initiating events are identified, reducing the former is trivial. To reduce the latter type of complications, we must find all pairs of story concepts that *cease* the same plan or attempt. If one of the concepts in the pair is only associated with assertions that are a subset of the other concept's assertions, we can eliminate the former concept.

★ **Algorithm 2** (Simplify Redundant Complications)**.** *Let G be the set of all plans and attempts in a narrative N, and let g be an element of G. For all g, if $a, b \in$ the set of all story concepts C such that ceases(a, g) and ceases(b,g), then let A and B be the set of assertions that contain a and b, respectively. Replace any instances of a with a SVAR in each element of A, and label the resulting set A'. Similarly, replace all instances of b with the same SVAR in every element of B to form B'. If $A' \subseteq B'$, then a is a* redundant complication*, and A may be removed from N.*

Aesop's fable of *The Lion in Love* provides an example of redundant complications:

> A lion fell in love with a maiden, and asked to marry her. The girl's father agreed, on the condition that the lion would cut its claws and remove its teeth. When the lion did so, the father chased it away.

In this story, the lion's teeth and claws are both complications which prevent the father from chasing away the lion. A simplified version might include only the lion's teeth (or only its claws) being an obstacle for the father.

7.3 Discussion

In this chapter, we have demonstrated that the Rensa framework is expressive enough to represent and reason about factors which contribute to narrative complexity. Rensa's expressive power enables researchers to isolate narrative features and determine how each contributes to narrative complexity, as well as how to transform each to simplify narratives.

Eliminating redundancies does not always imply clarity and reader satisfaction, and simplification techniques should always be applied thoughtfully. As an illustration of why this is important, consider *The Lion in Love*. One might decide claws and teeth are both dangerous aspects of the lion, and thus, both integral to the story.[1] In this case, the reader might be confused or unsatisfied if we removed either redundant complication.

Another example is Aesop's fable of *The Fox and the Cat*. In this story, a Fox imagines several different ways he might escape a pack of hounds, and takes so long in deliberation that he does not escape them. Had the Fox only imagined one means of escaping the hounds, he might have had time to escape. To prevent a loss of coherence, at least two redundant assertions and their subgraphs must be preserved during iteration through these hypothetical events. Alternatively, if only one imagined method is preserved, the story should emphasize that the Fox took enough time in his contemplation to result in his downfall. Regardless, a system that is looking to eliminate redundancies must not do so blindly.

Repetition may also be used to create rhythm, supply emphasis, or provide a sense of unity in narratives. Previous work has suggested the perceived clarity and preference of generated text may be correlated [58], but there is no clear algorithm to ensure them consistently. There are likely more factors at play, such as the reader's expecta-

[1]Note that these types of decisions may be highly subjective. Another reader might decide that removing one of the complications would be acceptable. Alternatively, perhaps yet another reader might conclude that the lion would still be dangerous until its muscles are weakened, or its size is somehow reduced. Neither of these qualities are described in the original story. Adaptation and generation systems alike must decide how to manage these user-specific design challenges.

tions [29] and previous experience with similar stories or language. Further investigation is needed to determine precisely which narrative features influence perception, and how they do so. This, in turn, will lead to improved narrative adaptation systems.

Chapter 8

Existing Applications

"Stories...are the making of something, and that something is soul. Anytime we feed soul, it guarantees increase."

— Clarissa Pinkola Estés

One of the best ways to assess a new computer science tool is to examine its use in solving real-world problems. In this chapter, I will enumerate several existing applications of the Rensa framework.

8.1 Fairy Tale Adaptation

Mazeika [89] used Rensa to adapt fairy tales into Walt Disney-style narratives. Each narrative included a set of actors with a name, gender, narrative role (e.g., protagonist, villain, sidekick, etc.), and miscellaneous attributes. Throughout the narrative, actors could perform actions, move to a new location, acquire a prop, or gain or lose a property. Time steps were encoded using the *storypoints* attribute.

To transform a fairy tale into one with the style of Disney, Mazeika used a

rules-based system. These rules were:

1. Every actor must have a name.
2. The hero must have an animal companion as a sidekick.
3. A maximum of two actors who serve identical narrative functions can exist.
4. The villain must be defeated.
5. No excessively violent actions may occur.

The first two rules possess relatively simple implementations. For the first, an actor is granted a proper name if one is not encoded. Similarly, the second rule is followed by checking if a sidekick is already encoded. If not, and a set of animals exists within the narrative already, one is chosen at random to be the hero's sidekick. Otherwise, a random animal is chosen.

To obey the third rule, all actors who perform the same action at the same storypoint are identified, and all but two are selected for removal. All assertions that are associated with the selected actors are removed as well. This method was inspired by my approach for simplifying actors presented in Section 7.2.1, and represents a limited version of the same method. If the villain is defeated in the original fairy tale, the fourth rule is satisfied. Otherwise, an action is chosen that would result in the villain's defeat. The system then selects an actor to perform the action, who must be in an appropriate state and time to be able to do so. If the action requires any prerequisite assertions, the system revises the narrative until they are successfully added.

The system enforces the fifth rule by identifying actions that are tagged as violent. If one is found, a replacement action with the same set of encoded consequences is identified. As with the fourth rule, any prerequisites for this replacement action must be added to the narrative.

Although dependent on a hand-crafted ontology, Mazeika's work demonstrates that Rensa can be used to apply new styles to short stories by transforming the fabula. Future work should be able to identify and apply these kinds of rules to encodings without relying on manual authoring.

8.2 Game Generation

In [85], Martens et al. used Rensa to visualize and support the formalization of proceduralist argumentation as logic programs. Proceduralist readings are a method of deriving meaning from games based on the underlying interactions between game mechanics, dynamics, aesthetics, and cultural knowledge. Rensa was used to encode and represent these interactions, resulting in improved understanding of what makes for meaningful game generation.

By coupling Rensa with a constraint solver that generates games, we can automatically visualize the interactions as well as the results. We can also realize the underlying rules into an actual game. This method was explored in Martens et al.'s later work [86]. During game generation, an answer set programming (ASP) solver was used to determine the basic ideas behind a proposed game. Rensa encoded these ideas

and reorganized their content based on specified constraints. The transformed assertions were realized into JavaScript code for the game. While the game was played, Rensa was used to communicate ideas between the gameplay and the narrative modules.

We accomplish two main goals by encoding the concepts behind a generated game using Rensa. First, Rensa helps developers to better understand, visualize, and debug their code and its outputs. Second, Rensa encodings provide a means for representing and reasoning over the game. This assists with initial game generation and adaptation of the game mechanics, aesthetics, and dynamics, as well as any associated narrative during play. Taken together, these accomplishments illustrate the value of using Rensa to adapt narrative structures beyond those found in natural text.

8.3 Dilemma Generation

A *dilemma* is a literary device which forces a person to make a difficult decision between two or more alternatives. Dilemma scenarios have been used extensively in classical and modern literature, and have become an important component of digital games and game design [125, 87]. These scenarios may be *encountered* or *observed*. A player that *encounters* a dilemma during play is able to analyze real-world conflicts while avoiding stereotypical thought patterns [70]. Conversely, someone who *observes* a second party undergo a dilemma personalized to their moral values may gain empathy and respect for that person [120]. The development of a system which is able to reason about and generate personalized dilemmas with identifiable values in real time should

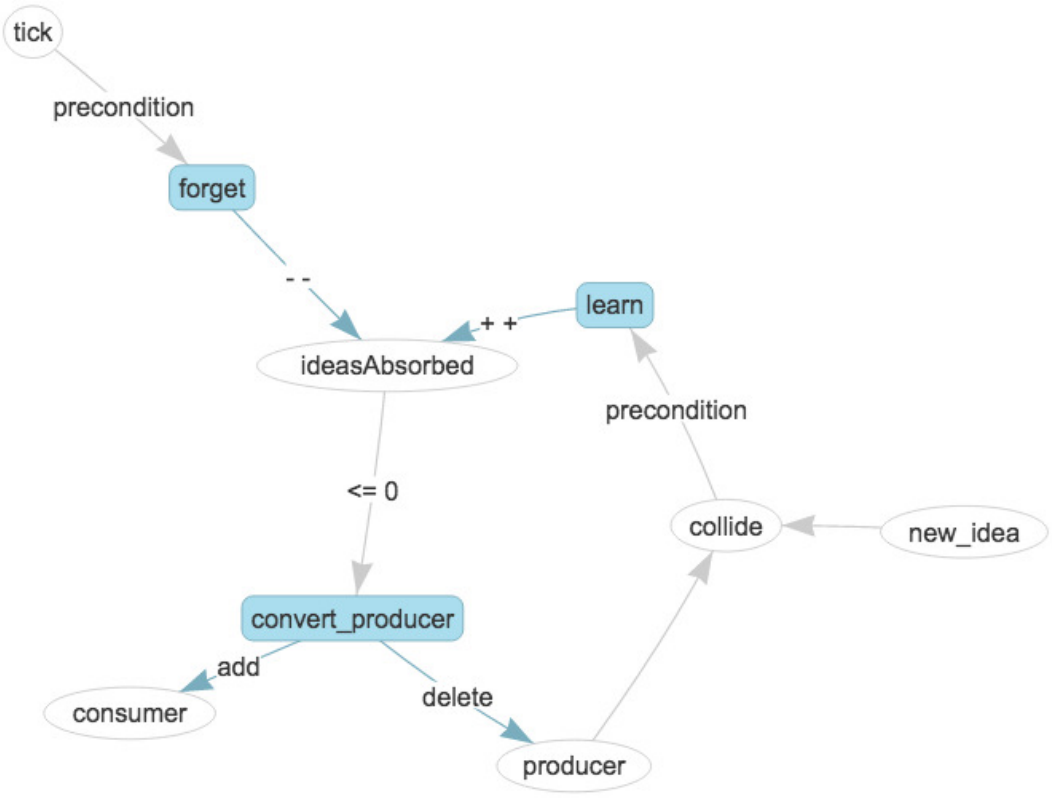

Figure 8.1: A generated visualization that represents several mechanic interactions in *The Free Culture Game* [92] according to developer specifications [85]. In this game, entities that represent new ideas roam the screen freely as floating particles. There are three types of agents: producers, consumers, and the vectorialist. Throughout the game, the vectorialist takes ideas out of the creative commons to commodify them for consumers. The player's goal is to herd the new ideas toward the producers to keep them inspired, which causes new ideas to form. If a producer becomes uninspired, they turn into a passive consumer.

be able to support these dynamics.

In [59], I describe an expressive dilemma generation model that extends prior work [7]. This system combines Schwartz's universal theory of values [109, 110] with an analysis of dilemma scenario factors derived from examining the dilemma's producer(s), audience, context, and the space of possible dilemmas. We can use Rensa encodings

to embed the resulting model within an interactive narrative. We will briefly present one method here, although it should be noted that there are many different ways of encoding the model with Rensa.

For a dilemma to take place, there must exist a *decider* (the entity that resolves the dilemma) and a *giver* (the entity that presents the dilemma). Let us assume that the player is always the decider in our interactive experience, which could be represented as *instance_of(player, decider)*. The giver could be any entity in the narrative, including the player. We can further characterize the player and their properties, either permanent or temporary, via *has* relations. We can also model the surrounding actors and environment to provide a sense of context.

During the dilemma, the decider can choose to *help*, *harm*, or *ignore* a set of values, which may be in the form of an entity or another action. The system can understand the player's choice by specifying each of these as action relations, where the left-hand concept is the player, the right-hand concept is the action taken, and the set of values can be expressed within *action_object* or *action_recipient* attributes. Each action the player takes can lead to a set of consequences, which may be expressed as assertions that are linked to the action assertion by discourse arc relations.

Through a preliminary study, a Rensa representation appeared to reasonably express contemporary game dilemmas to the satisfaction of human judges, especially when compared with a prior dilemma categorization model. Taken together with the preceding sections of this chapter, we see that not only can Rensa provide a means for expressing different aspects of a narrative, but it can express that narrative using

multiple lenses of detail. One may encode high-level information about a story, or dig deeper, even to examine the minute details of a single literary device. This kind of authorial control exemplifies Rensa's robust nature as a narrative adaptation framework.

Chapter 9

Conclusion

"We're all stories, in the end."

— Steven Moffat

9.1 Summary

This thesis has presented a new framework for modeling narrative: Rensa. Although many narrative representations exist, limited progress has been made toward human-readable encodings that can express and easily transform the inherent storiness of diverse media and use cases. This work is a first step toward addressing this challenge.

Rensa is designed to possess several key properties as a narrative transformation framework. First, Rensa is meant to be *accessible* for computers, programmers, and non-programmers. Assertions are stored using JSON, which is an open-standard format that is easy for both humans and machines to read. An online annotation interface exists that helps users upload, visualize, edit, and save SimpREL encodings. Basic Sim-

pREL encodings may also be constructed automatically from natural language inputs, which promotes accessibility for non-programmers. Rensa is also *expressive* in that it can support many distinct models of text and narrative, which we have demonstrated by using the example SimpREL encoding format. This permits its *applicability* to diverse human-computer interaction scenarios. Finally, Rensa encodings were designed to be *retrievable*. Research in open information extraction seeks to derive Rensa-like structures, supporting a bridge from natural text to encoded narrative meaning. Together, these qualities promote the encoding and adaptation of many kinds of narrative.

9.2 Future Work

Overall, we have explored how Rensa can be useful for a wide variety of applications. Future work should determine its robustness in these and other scenarios which demand the processing and adaptation of meaning.

Ongoing work in using Rensa for text generation includes the personalization of medical forms. I have designed an online service currently in development for this purpose. In addition to automatically performing lexical and syntactic simplifications, this service is capable of using Rensa to extract concepts from input documents to determine the final adaptation. These concepts might include the narrative content, such as the message a section of text is meant to convey. The system may also attempt to extract knowledge about the contents of the document itself. For example, consider the fact that an input document in PDF format may not contain styling information.

To support readability, the service identifies header text and uses Rensa assertions to describe how it should be displayed (e.g. in boldface, titlecase, and larger font than body text) in the final result. It also can identify text that represents key sections of typical consent documents (such as the suggested treatment, risks of the procedure, and signature sections), and can use this information to stylize and reorganize the document.

Beta testers found the system itself to be *"exciting"* and *"a good step forward."* One of the users, a health policy expert, noted that the system had the capacity to *"increase compliance with treatment advice...build trust between doctors and patients...[and] to establish a foundation for dialogue."* These comments suggest that this system (and/or similar systems) may be of use to the medical community in the future.

Intelligent game generation is another domain in which Rensa has already made a significant contribution. In the future, researchers could use Rensa not only to guide development and interpret game knowledge, but to provide player feedback. As one example, Rensa could help supply instructions to the player by reasoning over the current state of the game. For example, if the system notices that *Looking under the hat might cause the player to win*, i.e.:

```
{
  "l": [
    {
      "l": ["player"],
      "relation": "action",
      "r": ["look"],
```

```
        "location_under":["hat"]
      }
    ],
    "relation": "causes",
    "r": [{"l": ["game"], "relation": "set_mode", "r": ["win"]}],
    "prefix": "might"
  }
```

...it might generate the following assertion as a hint for the player:

```
  {
    "l": ["player"],
    "relation": "action",
    "r": ["look"],
    "location_under: ["hat"],
    "grammatical_mood": ["imperative"]
  }
```

Once automatically realized into natural text, this assertion reads as *Look under the hat.*

Many more applications are possible for the Rensa framework. Other example domains include:

- **Data visualization**. Rensa has already helped developers to automatically visualize data in real time. Future systems may also be able to use this bridge to better inform or understand their work.
- **User modeling and interaction**. It is impossible to predict the requirements of all users, and understanding the complexities involved in interaction demands

considerable social and contextual knowledge. More responsive systems could be developed by encoding and modifying narrative structures that represent a particular user and interaction scenario.

- **Meaningful generation and analysis**. Knowledge engineering and computationally creative systems may be able to benefit from a framework like Rensa that can encode and reason over many types of concepts and their relationships.

- **High-level encoding of complex data**. Some researchers have proposed using a high-level narrative language for more efficient encoding of formal notations and mathematical expressions (e.g., [57, 67]). By providing support for these types of encodings, Rensa may be able to provide a foundation for more intuitive coding environments for scientists.

9.3 Closing Remarks

Rensa was built to fulfill several design goals. First, it can encode, delete, and edit narrative knowledge and, in so doing, provides a foundation for systems that require responsive intelligence. Second, it is expressive enough to encode multiple narrative representations, including formal schemata founded in theory of mind such as the Story Intention Graph. Finally, it is sufficiently understandable and able to be extracted from natural text to the point that other researchers, developers, and non-technical experts can annotate data. As a result, narrative data can be more easily realized and shared.

Narrative is fundamental to human experience: our lives are weaved together

by stories, and much of who we are would cease to be without them. It is my hope that the work presented in this thesis enables researchers - including those who specialize in a field other than computer science - to find value in encoding narrative structures, and to learn about the ways that narrative can heal and transform us.

Bibliography

[1] Ashraf Al-Hadethe, Nigel Hunt, Ghaffar Al-Qaysi, and Shirley Thomas. Randomised controlled study comparing two psychological therapies for Posttraumatic Stress Disorder (PTSD): Emotional Freedom Techniques (EFT) vs. Narrative Exposure Therapy (NET). *Journal of Traumatic Stress Disorders and Treatment*, 4(2), 2016.

[2] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[3] Mariona Coll Ardanuy and Caroline Sporleder. Structure-based clustering of novels. In *Proceedings of Computational Linguistics for Literature (CLFL)*, Gothenburg, Sweden, 2015. Association for Computational Linguistics.

[4] Janet Wilde Astington and Jennifer M Jenkins. A longitudinal study of the relation between language and theory-of-mind development. *Developmental Psychology*, 35(5):1311, 1999.

[5] Mieke Bal. Notes on narrative embedding. *Poetics Today*, 2(2):41–59, 1981.

[6] David Bamman, Brendan O'Connor, and Noah A. Smith. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

[7] Heather Barber. *Generator of Adaptive Dilemma-based Interactive Narratives.* PhD thesis, University of York, 2008.

[8] Valentina Bartalesi, Carlo Meghini, and Daniele Metilli. Steps towards a formal ontology of narratives based on narratology. In *OASIcs-OpenAccess Series in Informatics*, volume 53. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[9] Frederic Charles Bartlett and Cyril Burt. Remembering: A study in experimental and social psychology. *British Journal of Educational Psychology*, 3(2):187–192, 1933.

[10] Mary Catherine Bateson. *Composing a life.* Grove Press, 2001.

[11] Jan De Belder and Marie-Francine Moens. Text simplification for children. In *Proceedings of the SIGIR Workshop on Accessible Search Systems*, volume 155, pages 19–26, Geneva, 2010. ACM.

[12] Emile Benveniste. *Problèmes de linguistique générale.* Gallimard, Paris, 1966.

[13] Nancy D. Berkman, Stacey L. Sheridan, Katrina E. Donahue, David J. Halpern, and Karen Crotty. Low health literacy and health outcomes: An updated systematic review. *Annals of Internal Medicine*, 155:97–107, 2011.

[14] Tina Besley. Foucauldian influences in narrative therapy: An approach for schools. *The Journal of Educational Enquiry*, 2(2), 2009.

[15] Mehret S. Birru, Valerie M. Monaco, Lonelyss Charles, Hadiya Drew, Valerie Njie, Timothy Bierria, Ellen Detlefsen, and Richard A. Steinman. Internet usage by low-literacy adults seeking health information: An observational analysis. *Journal of Medical Internet Research*, 6(3):e25, 2004.

[16] Claude Bremond. La logique des possibles narratifs. *Communications*, 8(1):60–76, 1966.

[17] Claude Bremond and Elaine D. Cancalon. The logic of narrative possibilities. *New Literary History*, 11(3):387–411, 1980.

[18] Jerome Bruner. The narrative construction of reality. *Critical Inquiry*, 18(1):1–21, 1991.

[19] Nathanael Chambers and Dan Jurafsky. Dialogue generation in character-based interactive storytelling. In *Proceedings of the First Annual Artificial Intelligence and Interactive Digital Entertainment Conference*, 2005.

[20] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of Narrative Event Chains. In *Proceedings of ACL-08*, volume 94305, pages 789–797, Hawaii, USA, 2008. Association for Computational Linguistics.

[21] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the*

47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 602–610, Suntec, Singapore, 2009. Association for Computational Linguistics.

[22] Yun-Gyung Cheong, Arnav Jhala, Byung-Chull Bae, and Robert Michael Young. Automatically generating summary visualizations from game logs. In *Proceedings of the Fourth Artificial Intelligence for Interactive Digital Entertainment (AIIDE) Conference*, 2008.

[23] William Coster and David Kauchak. Simple English Wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, volume 2. Association for Computational Linguistics, 2011.

[24] Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal Stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–92, 2014.

[25] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC 2006*, 2006.

[26] Thierry Declerck, Nikolina Koleva, and Hans-Ulrich Krieger. Ontology-based incremental annotation of characters in folktales. In *Proceedings of the 6th EACL*

Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, pages 30–34, 2012.

[27] Teun A. Van Dijk. Recalling and summarizing complex discourse. *Text Processing*, pages 49–93, 1979.

[28] Teun Adrianus Van Dijk and Walter Kintsch. *Strategies of discourse comprehension.* Academic Press, New York, 1983.

[29] Adrian Dimulescu and Jean-Louis Dessalles. Understanding narrative interest: Some evidence on the role of unexpectedness. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 1734–1739, 2009.

[30] Giovanna Egidi and Alfonso Caramazza. Cortical systems for local and global integration in discourse comprehension. *Neuroimage*, 71:59–74, 2013.

[31] David K Elson. *Modeling narrative discourse.* PhD thesis, Columbia University, 2012a.

[32] David K. Elson. Detecting story analogies from annotations of time, action and agency. In *Proceedings of the LREC 2012 Workshop on Computational Models of Narrative*, Istanbul, Turkey, 2012b.

[33] David K. Elson. Dramabank: Annotating agency in narrative discourse. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, 2012c.

[34] David K. Elson and Kathleen R. McKeown. A tool for deep semantic encoding of narrative texts. In *Proceedings of ACL-IJCNLP 2009 Software Demonstrations*, pages 9–12, Suntec, Singapore, 2009.

[35] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, volume 11, pages 3–10, 2011.

[36] Richard Evans and Constantin Orăsan. Improving anaphore resolution by identifying animate entities in texts. In *Proceedings of the Discourse Anaphora and Reference Resolution Conference*, pages 154–162, 2000.

[37] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[38] Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters (COLING '10)*, pages 276–284, 2010.

[39] Evelyn C Ferstl, Jane Neumann, Carsten Bogler, and D Yves Von Cramon. The

extended language network: a meta-analysis of neuroimaging studies on text comprehension. *Human Brain Mapping*, 29(5):581–593, 2008.

[40] Charles J Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976.

[41] Alan M. Frisch and Donald Perlis. A re-evaluation of story grammars. *Cognitive Science*, 5(1):79–86, 1981.

[42] Aldo Gangemi. A comparison of knowledge extraction tools for the semantic web. In *Proceedings of the Extended Semantic Web Conference*, pages 351–366. Springer Berlin Heidelberg, 2013.

[43] Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, and Misael Mongiovì. Semantic Web Machine Reading with FRED. *Semantic Web*, Preprint(Preprint):to appear, 2016.

[44] Jay L Garfield, Candida C Peterson, and Tricia Perry. Social cognition, language acquisition and the development of the theory of mind. *Mind & Language*, 16(5):494–541, 2001.

[45] Caroline Gasperin, Lucia Specia, Tiago Pereira, and Sandra Alusio. Learning when to simplify sentences for natural text simplification. In *Proceedings of ENIA*, pages 809–818, 2009.

[46] Gérard Genette. Figures iii. 1972.

[47] Mary L. Gick and Keith J. Holyoak. Analogical problem solving. *Cognitive Psychology*, 12(3):306–355, 1980.

[48] Mary L. Gick and Keith J. Holyoak. Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1–38, 1983.

[49] Ronald B. Gillam and Nils A. Pearson. Test of narrative language. *Topics in Language Disorders*, 2004.

[50] Ben Goertzel. TruthValue - OpenCog, 2016. Accessed: 2016-11-18.

[51] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller. The OpenCog Framework. In *Engineering General Intelligence*, volume 2, pages 3–29. Atlantis Press, 2014.

[52] Hui-Ngo Goh, Lay-Ki Soon, and Su-Cheng Haw. Automatic identification of protagonist in fairy tales using verb. In *Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science*, volume 7302, pages 395–406. Springer Berlin / Heidelberg, 2012.

[53] Amit Goyal and Ellen Riloff. A computational model for plot units. *Computational Intelligence*, 29(3):466–488, 2013.

[54] Amit Goyal, Ellen Riloff, and Hal Daumé. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, Massachusetts, USA, 2010.

[55] Arthur C. Graesser, Keith K. Millis, and Rolf A. Zwaan. Discourse comprehension. *Annual Review of Psychology*, 48(1):163–189, 1997.

[56] Arthur C Graesser and Rolf A Zwaan. Inference generation and the construction of situation models. *Discourse comprehension: Essays in honor of Walter Kintsch*, pages 117–139, 1995.

[57] Maria Luisa Guerriero, Anna Dudka, Nicholas Underhill-Day, John K. Heath, and Corrado Priami. Narrative-based computational modelling of the gp130/jak/stat signalling pathway. *BMC Systems Biology*, 3(1):1, 2009.

[58] Sarah Harmon. FIGURE8: Generating and evaluating figurative language. In *The Sixth International Conference on Computational Creativity*, 2015.

[59] Sarah Harmon. An expressive dilemma generation model for players and artificial agents. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 176–182, 2016.

[60] Sarah Harmon and Arnav Jhala. Toward an automated measure of narrative complexity. In *Intelligent Narrative Technologies Workshop, Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 38–41, 2015.

[61] Michael Hart. Free ebooks - Project Gutenberg, 2014. Gutenberg.org. http://www.gutenberg.org/.

[62] Uri Hasson, Howard C Nusbaum, and Steven L Small. Brain networks subserving

the extraction of sentence information and its encoding to memory. *Cerebral Cortex*, 17(12):2899–2913, 2007.

[63] Hua He, Denilson Barbosa, and Grzegorz Kondrak. Identification of speakers in novels. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1312–1320, Sofia, Bulgaria, 2013.

[64] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, and Margeritta von Wilamowitz-Moellendorff. Gumo–the general user model ontology. In *International Conference on User Modeling*, pages 428–432. Springer, 2005.

[65] Hongyan Jing. Sentence reduction for automatic text summarization. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 310–315. Association for Computational Linguistics, 2000.

[66] Nancy S. Johnson and Jean M. Mandler. A tale of two structures: Underlying and surface forms in stories. *Poetics*, 9(1):51–86, 1980.

[67] Ozan Kahramanoğullari, Luca Cardelli, and Emmanuelle Caron. An intuitive automated modelling interface for systems biology. *arXiv preprint arXiv:0911.2327*, 2009.

[68] Sasikiran Kandula, Dorothy Curtis, and Qing Zeng-Treitler. A semantic and syntactic text simplification tool for health content. In *AMIA Annual Symposium Proceedings*, pages 366–370, 2010.

[69] Folgert Karsdorp, Peter van Kranenburg, Theo Meder, and Antal van den Bosch. Casting a spell: Identification and ranking of actors in folktales. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*. Edicoes Cilibri, 2012.

[70] Matthias Kemmer. The politics of post-apocalypse: Interactivity, narrative framing and ethics in Fallout 3. In Gerold Sedlmayr and Nicole Waller, editors, *Politics in Fantasy Media: Essays on Ideology and Gender in Fiction, Film, Television, and Games*, pages 97–117, Jefferson, NC, 2014. McFarland.

[71] Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. Text simplification for information-seeking applications. In *OTM Confederated International Conferences: On the Move to Meaningful Internet Systems*, pages 735–747. Springer Berlin Heidelberg, 2004.

[72] Michelle Koo, Ines Krass, and Parisa Aslani. Enhancing patient education about medicines: factors influencing reading and seeking of written medicine information. *Health Expectations*, 9(2):174–187, 2006.

[73] William Labov. The transformation of experience in narrative syntax. In *Language in the Inner City*, pages 354–396, Philadelphia, 1972. University of Pennsylvania Press.

[74] Raymond Lang. A declarative model for simple narratives. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*, pages 134–141, 1999.

[75] Benoit Lavoie and Owen Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing. Association for Computational Linguistics (ACL)*, 1997.

[76] Wendy G. Lehnert. Plot units and narrative summarization. *Cognitive Science: A Multidisciplinary Journal*, 5(4):293–331, 1981.

[77] Rodrigo T. Lopes, Miguel M. Gonçalves, Daniel B. Fassnacht, Paulo PP Machado, and Inês Sousa. Long-term effects of psychotherapy on moderate depression: A comparative study of narrative therapy and cognitive-behavioral therapy. *Journal of Affective Disorders*, 167:64–73, 2014.

[78] Rodrigo T. Lopes, Miguel M. Gonçalves, Paulo PP Machado, Dana Sinai, Tiago Bento, and Jo ao Salgado. Narrative therapy vs. cognitive-behavioral therapy for moderate depression: Empirical evidence from a controlled clinical trial. *Psychotherapy Research*, 24(6):662–674, 2014.

[79] Benedikt Löwe et al. Comparing formal frameworks of narrative structure. In *AAAI Fall Symposium: Computational Models of Narrative*, 2010.

[80] Benedikt Löwe, Eric Pacuit, et al. An abstract approach to reasoning about games with mistaken and changing beliefs. *The Australasian Journal of Logic*, 6:162–181, 2008.

[81] François Mairesse and Marilyn Walker. Personage: Personality generation for

dialogue. In *Annual Meeting-Association For Computational Linguistics*, volume 45(1), page 496, 2007.

[82] Inderjeet Mani. *Automatic Summarization*. John Benjamins Publishing Company, 2001.

[83] Raymond A Mar. The neuropsychology of narrative: Story comprehension, story production and their interrelation. *Neuropsychologia*, 42(10):1414–1434, 2004.

[84] Raymond A Mar. The neural bases of social cognition and story comprehension. *Annual Review of Psychology*, 62:103–134, 2011.

[85] Chris Martens, Adam Summerville, Michael Mateas, Joseph Osborn, Sarah Harmon, Noah Wardrip-Fruin, and Arnav Jhala. Proceduralist readings, procedurally. In *Proceedings of the Third Experimental AI in Games Workshop, at the Twelfth Artificial Intelligence and Interactive Digital Entertainment*, 2016.

[86] Chris Martens, Adam Summerville, Michael Mateas, Joseph Osborn, Sarah Harmon, Noah Wardrip-Fruin, and Arnav Jhala. From mechanics to meaning. *Manuscript in preparation*, 2017.

[87] Michael Mateas, Peter Mawhorter, and Noah Wardrip-Fruin. Intentionally generating choices in interactive narratives. In *Proceedings of the Sixth International Conference on Computational Creativity*, pages 292–299, Park City, UT, 2015.

[88] Joanne C. May. Family attachment narrative therapy: Healing the experience of

early childhood maltreatment. *Journal of Marital and Family Therapy*, 31(3):221–237, 2005.

[89] Jo Mazeika. A rules-based system for adapting and transforming existing narratives. In *Interactive Storytelling: 9th International Conference on Interactive Digital Storytelling (ICIDS 2016)*, pages 176–183, Los Angeles, CA, USA, 2016. Springer.

[90] Paras D Mehta, Barbara R. Foorman, Lee Branum-Martin, and William Patrick Taylor. Literacy as a unidimensional multilevel construct: Validation, sources of influence, and implications in a longitudinal study in grades 1 to 4. *Scientific Studies of Reading*, 9(2):85–116, 2005.

[91] Marvin Minsky. A framework for representing knowledge. Technical report, MIT-AI Laboratory Memo 306, 1974.

[92] molleindustria. The Free Culture Game, 2008.

[93] Nick Montfort. Curveship: an interactive fiction system for interactive narrating. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 55–62. Association for Computational Linguistics, 2009.

[94] Kate Nation, Paula Clarke, Catherine M. Marshall, and Marianne Durand. Hidden language impairments in children: Parallels between poor reading comprehension and specific language impairment? *Journal of Speech, Language, and Hearing Research*, 47:199–211, 2004.

[95] Abby J. Newberg. An analysis of a narrative therapy group work model for psychosis. Technical report, Master of Social Work Clinical Research Papers, Computer Science Department, St. Catherine University, Minneapolis, MN, 2016.

[96] Edward J. O'Brien and Anne E. Cook. Models of discourse comprehension. In *The Oxford Handbook of Reading*, page 217, 2015.

[97] Constantin Orăsan and Richard Evans. Learning to identify animate references. In *Walter Daelemans and Rémi Zajac, editors, Proceedings of CoNLL-2001*, pages 129–136, Toulouse,France, 2001.

[98] Gustavo Henrique Paetzold. Reliable lexical simplification for non-native speakers. In *Proceedings of the NAACL-HLT 2015 Student Research Workshop (SRW)*, page 9, 2015.

[99] Marie-Jose Perrier, Brett M. Smith, and Amy E. Latimer-Cheung. Narrative environments and the capacity of disability narratives to motivate leisure-time physical activity among individuals with spinal cord injury. *Disability and rehabilitation*, 35(24):2089–2096, 2013.

[100] Douglas B. Petersen, Sandra Laing Gillam, and Ronald B. Gillam. Emerging procedures in narrative assessment: The index of narrative complexity. *Topics in Language Disorders*, 28(2):115–130, 2008.

[101] Sarah E. Petersen and Mari Ostendorf. Text simplification for language learners:

A corpus analysis. In *Speech and Language Technology in Education (SLaTE 2007)*, pages 69–72, 2007.

[102] Vladimir Propp and Max Louwerse. *De morfologie van het toversprookje: vormleer van een genre*. Utrecht: Spectrum, 1997.

[103] Luz Rello, Ricardo Baeza-Yates nad Stefan Bott, and Horacio Saggion. Simplify or help?: Text simplification strategies for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13)*, page 15. ACM, 2013.

[104] Elena Rishes, Stephanie M. Lukin, David K. Elson, and Marilyn A. Walker. Generating different story tellings from semantic representations of narrative. In *Koenitz, H., Sezen, T.I., Ferri, G., Haahr, M., Sezen, D., C atak, G. (eds.) ICIDS 2013, LNCS*, volume 8230, pages 192–204. Springer, Heidelberg, 2013.

[105] David E. Rumelhart. Notes on a schema for stories. *Representation and understanding: Studies in cognitive science*, 211(236):45, 1975.

[106] Horacio Saggion, Sanja tajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. Making it simplext: Implementation and evaluation of a text simplification system for Spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):14, 2015.

[107] Leonie Schäfer, Agnes Stauber, and Bozana Bokan. Storynet: an educational game

for social skills. In *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pages 148–157. Springer Berlin Heidelberg, 2004.

[108] Rogert Schank and Robert Abelson. Scripts, plans, goals and understanding: An inquiry into human knowledge structures. *Interactive Storytelling*, 1977.

[109] Shalom H. Schwartz. Universals in the content and structure of values: Theory and empirical tests in 20 countries. In M. Zanna, editor, *Advances in experimental social psychology*, volume 25, pages 1–65, New York, 1992. Academic Press.

[110] Shalom H. Schwartz. Refining the theory of basic individual values. *Journal of Personality and Social Psychology*, 103(4):663–688, 2012.

[111] Advaith Siddharthan. Syntactic simplification and text cohesion. Technical Report 597, University of Cambridge Computer Laboratory, Cambridge, UK, 2004.

[112] Advaith Siddharthan. Text simplification using typed dependencies: A comparison of the robustness of different generation strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG '11)*, pages 2–11, 2011.

[113] Advaith Siddharthan. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298, 2014.

[114] Robert Speer and Catherine Havasi. ConceptNet 5: A large semantic network for relational knowledge. *The Peoples Web Meets NLP*, pages 161–176, 2013.

[115] Brandon Robert Tearse, Peter A. Mawhorter, Michael Mateas, and Noah Wardrip-Fruin. Skald: Minstrel reconstructed. *IEEE Transactions on Computational Intelligence and AI in Games*, 6:156–165, 2014.

[116] Perry W. Thorndyke. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, 9(1):77–110, 1977.

[117] Tzvetan Todorov. Les catégories du récit littéraire. *Communications*, 8:125–151, 1966.

[118] Mike Treanor, Bobby Schweizer, Ian Bogost, and Michael Mateas. The micro-rhetorics of game-o-matic. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 18–25. ACM, 2012.

[119] Scott Turner. Minstrel: a computer model of creativity and storytelling. Technical Report CSD-920057, Ph.D. Thesis, Computer Science Department, University of California, Los Angeles, CA, 1992.

[120] Richard L. Upright. To tell a tale: The use of moral dilemmas to increase empathy in the elementary school child. *Early Childhood Education Journal*, 30(1):15–20, 2002.

[121] Clark Verbrugge. A structure for modern computer narratives. In *Proceedings of the International Conference on Computers and Games*, pages 308–325. Springer Berlin Heidelberg, 2001.

[122] Kurt Vonnegut. *Deadeye Dick: A Novel*. Dial Press, 2009.

[123] Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297, 2015.

[124] Zheng Xu, Xiangfeng Luo, Shunxiang Zhang, Xiao Wei, Lin Mei, and Chuanping Hu. Mining temporal explicit and implicit semantic relations between entities using web search engines. *Future Generation Computer Systems*, 37:468–477, 2014.

[125] Jos Zagal. Ethically notable videogames: Moral dilemmas and gameplay. In *Proceedings of DIGRA 2009: Breaking new ground: Innovation in games, play, practice and theory*, 2009.

Appendix A

Key SimpREL Examples

The following is a sampling of SimpREL encodings for common narrative structure types, and their natural text counterparts. Additional encodings in SimpREL format can be found at `https://github.com/RensaProject`.

A.1 States

Bilbo has Gollum's ring.

l	["Bilbo"]
relation	"has"
r	["ring"]
r_owner	["Gollum"]

Eric, Grimsby, and the sailors were on the king's ship.

l	["Eric","Grimsby","sailors"]
relation	"location_on"
r	["ship"]
r_owner	["king"]
tense	"past"

"Look out, lad!" the sailors sang.

l	["sailors"]
relation	"action"
r	["sing"]
discourse	"Look out, lad!"
tense	"past"
spoken	True

King Triton entered the scene dramatically.

l	["King Triton"]
relation	"action"
r	["enter"]
action_object	["scene"]
with_property	["dramatic"]
tense	"past"

A.2 Beliefs

Recall that if you load an assertion into an actor's brain, they are the owner of that assertion by default.

Frodo knows he has a ring.

owner	["Frodo"]
believer	["Frodo"]
l	["Frodo"]
relation	"has"
r	["ring"]

Frodo knows everyone knows he has a ring.

owner	["Frodo"]
believer	["SVAR_everyone"]
l	["Frodo"]
relation	"has"
r	["ring"]
condition_given	[{"l": ["SVAR_everyone"],"relation":"instance_of","r":["actor"]}, {"l": ["SVAR_everyone"],"relation":"has_property","r":["all"]}]

Sam knows everyone knows Frodo knows he has a ring.

owner	[“Sam”]
believer	[“SVAR_everyone”]
l	[“Frodo”]
relation	“has_belief”
r	[{l: [“Frodo”], relation: “has”, r: [“ring”]}]
condition_given	[{“l”: [“SVAR_everyone”],“relation”:“instance_of”,“r”:[“actor”]}, {“l”: [“SVAR_everyone”],“relation”:“has_property”,“r”:[“all”]}]

A.3 Goals and Intentions

Ariel wants to explore land.

l	[“Ariel”]
relation	“has_goal”
r	[{l:[“Ariel”], relation:“action”, r:[“explore”], action_object:[“land”]}]

Ariel needs to become human before she can explore land.

l	[{l:[“Ariel”], relation:“is_a”, r:[“human”]}]
relation	“precondition_for”
r	[{l:[“Ariel”], relation:“action”, r:[“explore”], action_object:[“land”]}]

If Ariel gave her voice to Ursula, she might become human.

l	[{l:[“Ariel”], relation:“action”, r:[“give”], action_object:[“voice”], action_object_owner:[“Ariel”], action_recipient:[“Ursula”]}]
relation	“causes”
r	[{l:[“Ariel”], relation:“is_a”, r:[“human”]}]
prefix	[“might”]

A.4 Imperatives

Mince the onion and the garlic.

l	[“chef”]
relation	“action”
r	[“mince”]
action_object	[“onion”,“garlic”]
grammatical_mood	“imperative”

Quickly pour the milk into the medium bowl that contains the sugar.

l	[“chef”]
relation	“action”
r	[“pour”]
action_object	[“milk”]
location_into	[“bowl”]
location_into_property	[“medium”]
with_property	[“quick”]
condition_if	[{l:[“sugar”], relation:“located_in”, r:[“bowl”]}]
grammatical_mood	“imperative”

Use the spoon for scooping the ice cream.

l	[“spoon”]
relation	“used_for”
r	[{l:[“spoon”], relation:“action”, r:[“scoop”], action_object:[“ice cream”]}]
grammatical_mood	“imperative”

Bake the bread until it is golden brown.

l	[“chef”]
relation	“action”
r	[“bake”]
action_object	[“bread”]
condition_until	[{l:[“bread”], relation:“has_property”, r:[“golden brown”]}]
grammatical_mood	“imperative”

Boil the egg for three minutes.

l	[“chef”]
relation	“action”
r	[“boil”]
action_object	[“egg”]
time_for	[“three minutes”]
grammatical_mood	“imperative”

Set the oven to 350 degrees Fahrenheit.

l	[“oven”]
relation	“set_value”
r	[350]
r_unit	“degrees Fahrenheit”
grammatical_mood	“imperative”

Appendix B

SimpREL Encodings that Demonstrate Support for Existing Representations

B.1 The Classic Schankian Restaurant Script

Listing B.1: An example SimpREL encoding for a variation of a popular Schankian script which describes scenes at a restaurant. In this example, the customer and server are entities of type *actor*, performing *actions* with optional *action_object* attributes for encoding props.

```
[
  {
      "l":["customer"],
      "relation":"action",
      "r":["enter"],
      "action_object":["restaurant"],
      "script_act":["PTRANS"],
      "scene":1,
      "storypoints":[{"at":1}],
```

```
    "text": "The customer enters the restaurant."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["look at"],
    "action_object":["tables"],
    "script_act":["ATTEND"],
    "scene":1,
    "storypoints":[{"at":2}],
    "text": "The customer looks at the tables."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["choose"],
    "action_object":["myTable"],
    "action_recipient":["customer"],
    "condition_given":[
      {
          "l":["myTable"],
          "relation":"instance_of",
          "r":["table"]
      },
      {
          "l":["myTable"],
          "relation":"has_property",
          "r":["available"]
```

```
      }
    ]
    "script_act":["MBUILD"],
    "scene":1,
    "storypoints":[{"at":3}],
    "text": "The customer chooses a table."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["move to"],
    "action_object":["myTable"],
    "script_act":["PTRANS"],
    "scene":1,
    "storypoints":[{"at":4}],
    "text": "The customer moves to the table of their choosing."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["sit"],
    "script_act":["PTRANS"],
    "scene":1,
    "storypoints":[{"at":5}],
    "text": "The customer sits."
},
{
    "l":["customer"],
```

```
    "relation":"action",
    "r":["take"],
    "action_object":["menu"],
    "script_act":["PTRANS"],
    "scene":2,
    "storypoints":[{"at":1}],
    "text": "The customer picks up the menu."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["read"],
    "action_object":["menu"],
    "script_act":["MTRANS"],
    "scene":2,
    "storypoints":[{"at":2}],
    "text": "The customer reads the menu."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["choose"],
    "action_object":["myFood"],
    "script_act":["MBUILD"],
    "condition_given":[
      {
          "l":["myFood"],
          "relation":"instance_of",
```

```
            "r":["food"]
        },
        {
            "l":["menu"],
            "relation":"has_entry",
            "r":["myFood"]
        },
        {
            "l":["customer"],
            "relation":"likes",
            "r":["myFood"]
        },
    ]
    "scene":2,
    "storypoints":[{"at":3}],
    "text": "The customer chooses food to order."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["signal to"],
    "action_recipient":["waiter"],
    "script_act":["MTRANS"],
    "scene":2,
    "storypoints":[{"at":4}],
    "text": "The customer signals to the waiter."
},
{
```

```
    "l":["waiter"],
    "relation":"action",
    "r":["move to"],
    "action_object":["table"],
    "action_object_owner":["customer"],
    "script_act":["PTRANS"],
    "scene":2,
    "storypoints":[{"at":5}],
    "text": "The waiter moves to the customer's table."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["request"],
    "action_recipient":["waiter"],
    "action_object":["myFood"],
    "script_act":["MTRANS"],
    "scene":2,
    "storypoints":[{"at":6}],
    "text": "The customer asks the waiter for food."
},
{
    "l":["waiter"],
    "relation":"action",
    "r":["move to"],
    "action_object":["location"],
    "action_object_owner":["chef"],
    "script_act":["PTRANS"],
```

```
    "scene":2,
    "storypoints":[{"at":7}],
    "text": "The waiter moves to the chef's location."
},
{
    "l":["waiter"],
    "relation":"action",
    "r":["request"],
    "action_recipient":["chef"],
    "action_object":["myFood"],
    "script_act":["MTRANS"],
    "scene":2,
    "storypoints":[{"at":8}],
    "text": "The waiter asks the chef for the chosen food."
},
{
    "l":["chef"],
    "relation":"action",
    "r":["give"],
    "action_recipient":["waiter"],
    "action_object":["myFood"],
    "script_act":["MTRANS"],
    "scene":3,
    "storypoints":[{"at":1}],
    "text": "The chef gives the food to the waiter."
},
{
    "l":["waiter"],
```

```
    "relation":"action",
    "r":["move to"],
    "action_object":["table"],
    "action_object_owner":["customer"],
    "script_act":["PTRANS"],
    "scene":3,
    "storypoints":[{"at":2}],
    "text": "The waiter moves back to the customer's table."
},
{
    "l":["waiter"],
    "relation":"action",
    "r":["give"],
    "action_object":["myFood"],
    "action_recipient":["customer"],
    "script_act":["ATRANS"],
    "scene":3,
    "storypoints":[{"at":3}],
    "text": "The waiter gives the food to the customer."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["eat"],
    "action_object":["myFood"],
    "script_act":["INGEST"],
    "scene":3,
    "storypoints":[{"at":4}],
```

```
    "text": "The customer eats the food."
},
{
    "l":["waiter"],
    "relation":"action",
    "r":["give"],
    "action_object":["check"],
    "action_recipient":["customer"],
    "script_act":["ATRANS"],
    "scene":4,
    "storypoints":[{"at":1}],
    "text": "The waiter gives the check to the customer."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["give"],
    "action_object":["money"],
    "action_recipient":["waiter"],
    "script_act":["ATRANS"],
    "scene":4,
    "storypoints":[{"at":2}],
    "text": "The customer gives money to the waiter."
},
{
    "l":["customer"],
    "relation":"action",
    "r":["exit"],
```

```
        "action_object":["restaurant"],
        "script_act":["PTRANS"],
        "scene":4,
        "storypoints":[{"at":3}],
        "text": "The customer leaves the restaurant."
    }
]
```

B.2 DPF Analysis of *Engagement*

Engagement is an example of a short, branching narrative. The story, as first presented by Lehnert [76], may be told as follows:

> John was thrilled when Mary accepted his engagement ring. But when he found out about her father's illegal mail-order business, he felt torn between his love for Mary and his responsibility as a policeman. When John finally arrested the old man, Mary called off the engagement.

Part of this story is shown in Figure B.1, and encoded using SimpREL (Listing B.2). Notice that, as in many story examples, there could be multiple ways to represent the *Engagement* story in SimpREL. For instance, we could also represent the tree using hypothetical assertions with *causes* relations.

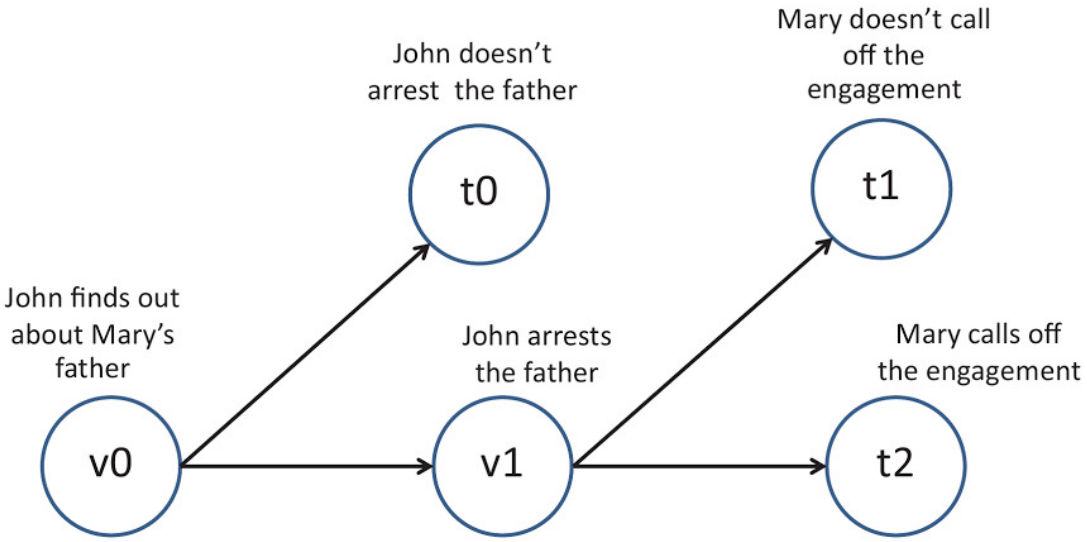

Figure B.1: DPF game tree of the *Engagement* story [82].

Listing B.2: An example SimpREL encoding for Figure B.1.

```
[
  {
        "myID":"v0",
        "l":["John"],
        "relation":"action",
        "r":["discover"],
        "action_object":["mail-order business"],
        "action_object_owner":["Mary's father"],
        "action_object_property":["illegal"],
        "children": ["$(t0)","$(v1)"]
    },
    {
        "myID":"t0",
```

```
    "l":["John"],
    "relation":"action",
    "r":["arrest"],
    "action_object":["Mary's father"],
    "prefix":["not"]
},
{
    "myID":"v1",
    "l":["John"],
    "relation":"action",
    "r":["arrest"],
    "action_object":["Mary's father"],
    "children":["$(t1)","$(t2)"]
},
{
    "myID":"t2",
    "l":["Mary"],
    "relation":"action",
    "r":["call off"],
    "action_object":["engagement"],
    "action_recipient:["John"]
},
{
    "myID":"t1",
    "l":["Mary"],
    "relation":"action",
    "r":["call off"],
    "action_object":["engagement"],
```

```
            "action_recipient:["John"],
            "prefix":["not"]
        },
    ]
```

B.3 SIG Analysis of *The Wily Lion*

Fables and folktales have been conventionally used as examples for narrative analyses because they are brief and straightforward. *The Wily Lion* is one of Aesop's fables, and can be told as follows:

> A Lion watched a fat Bull feeding in a meadow, and his mouth watered when he thought of the royal feast he would make, but he did not dare to attack him, for he was afraid of his sharp horns. Hunger, however, presently compelled him to do something: and as the use of force did not promise success, he determined to resort to artifice. Going up to the Bull in friendly fashion, he said to him, "I cannot help saying how much I admire your magnificent figure. What a fine head! What powerful shoulders and thighs! But, my dear friend, what in the world makes you wear those ugly horns? You must find them as awkward as they are unsightly. Believe me, you would do much better without them." The Bull was foolish enough to be persuaded by this flattery to have his horns cut off; and, having now lost his only means of defense, fell an easy prey to the Lion.

In this section, we will describe how to encode the SIG representation this story, as displayed in Figure B.2.

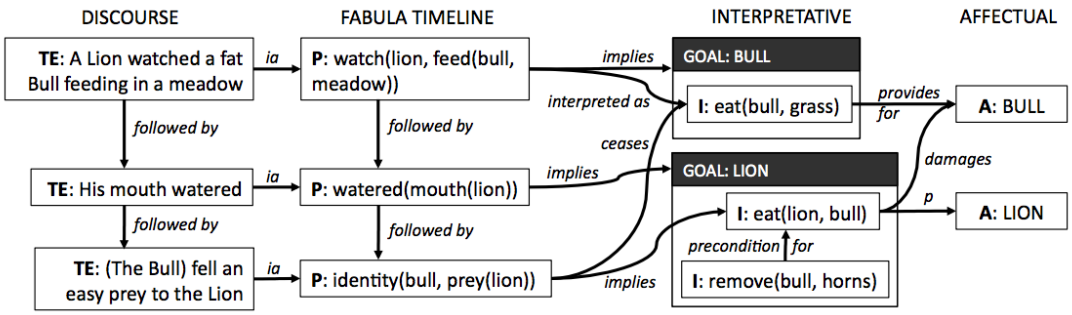

Figure B.2: SIG analysis of *The Wily Lion* [31].

Listing B.3: A SimpREL encoding for the SIG representation in Figure B.2. Assertion IDs are only shown for easier reading.

```
[
  {
        "id": 1,
        "node":["Proposition"],
        "l":["lion"],
        "relation": "action",
        "r": "watch",
        "action_object":[
          {
              "l":["bull"],
              "relation":"action",
              "r":["feed"],
              "location_in": "meadow"
          }
        ],
        "tense": "past",
        "storypoints":[{"at":1}],
```

```
    "text": "A Lion watched a fat Bull feeding in a meadow."
},
{
    "id": 2,
    "node":["Proposition"],
    "l":["mouth"],
    "relation": "action",
    "r": "water",
    "l_owner":["lion"],
    "tense": "past",
    "storypoints":[{"at":2}],
    "text": "His mouth watered."
},
{
    "id": 3,
    "node":["Proposition"],
    "l":["bull"],
    "relation": "instance_of",
    "r": ["prey"],
    "r_owner":["lion"],
    "tense": "past",
    "storypoints":[{"at":3}],
    "text": "(The Bull) fell an easy prey to the Lion."
},
{
    "id": 4,
    "l":["bull"],
    "relation": "has_goal",
```

```
    "r":["$(5)"]
},
{
  "id": 5,
  "node":["Interpretative"],
  "l":["bull"],
  "relation":"action",
  "r":["eat"],
  "action_object":["grass"]
},
{
    "id": 6,
    "l":["lion"],
    "relation": "has_goal",
    "r":[
      {
            "l": ["$(7)"],
            "relation": "precondition_for",
            "r": ["$(8)"]
    ]
},
{
    "id": 7,
    "node":["Interpretative"],
    "l":["bull"],
    "relation": "action",
    "r":["remove"],
    "action_object":["horns"],
```

```
    "action_object_owner":["bull"]
},
{
  "id":8,
    "node":["Interpretative"],
    "l":["lion"],
    "relation": "action",
    "r":["eat"],
    "action_object":["bull"]
},
{
    "l": ["$(1)"],
    "relation":"implies",
    "r": ["$(4)"]
},
{
    "l": ["$(1)"],
    "relation":"interpreted_as",
    "r": ["$(5)"]
},
{
    "l": ["$(2)"],
    "relation":"implies",
    "r": ["$(6)"]
},
{
    "l": ["$(3)"],
    "relation":"ceases",
```

```
        "r": ["$(5)"]
    },
    {
        "l": ["$(3)"],
        "relation":"implies",
        "r": ["$(8)"]
    },
    {
        "l": ["$(5)"],
        "relation":"helps",
        "r": ["bull"]
    },
    {
        "l": ["$(8)"],
        "relation":"harms",
        "r": ["bull"]
    },
    {
        "l": ["$(8)"],
        "relation":"helps",
        "r": ["lion"]
    }
]
```