Story Understanding in Genesis:
Exploring Automatic Plot Construction through Commonsense Reasoning

by

Harold William Capen Low, IV

S.B., C.S. M.I.T., 2009

Submitted to the Department of Electrical Engineering
and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

May, 2011
[June 2011]

Author: _____
Department of Electrical Engineering and Computer Science
May 24, 2011

Certified by: _____
Patrick Winston, Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor
May 24, 2011

Accepted by: _____
Dr. Christopher J. Terman Chairman, Masters of Engineering Thesis Committee

Story Understanding in Genesis:

Exploring Automatic Plot Construction through Commonsense Reasoning

by

Harold William Capen Low, IV

Submitted to the
Department of Electrical Engineering and Computer Science

May 24, 2011

in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Whether through anecdotes, folklore, or formal history, humans learn the lessons and expectations of life from stories. If we are to build intelligent programs that learn as humans do, such programs must understand stories as well.

Casting narrative text in an information-rich representation affords AI research platforms, such as the Genesis system, the capacity to understand the events of stories individually. To understand a story, however, a program must understand not just events, but also how events cause and motivate one another. In order to understand the relationships between these events, stories must be saturated with implicit details, connecting given events into coherent plot arcs.

In my research, my first step was to analyze a range of story summaries in detail. Using nearly 50 rules, applicable to brief summaries of stories taken from international politics, group dynamics, and basic human emotion, I demonstrate how a rendition of Frank Herbert's *Dune* can be automatically understood so as to produce an interconnected story network of over one hundred events.

My second step was to explore the nuances of rule construction, finding which rules are needed to create story networks reflective of proper implicit understanding and how we, as architects, must shape those rules to be understood. In particular, I develop a method that constructs new rules using the rules already embedded in stories, a representation of higher-order thinking that enables us to speak of our ideas as objects.

Acknowledgements

Patrick Winston *for taking me in when no one else would, and for showing me that there are still those who believe in Artificial Intelligence*

Brett van Zuiden, Adam Kraft, and Matthew Fay *for reminding me that ideas are worth talking about.*

Heavy Metal *for courage.*

Audrey *for love, and for sanity.*

My family, especially my parents, *for giving so much and asking only my happiness in return. You shall have it tenfold.*

Table of Contents

List of Figures

"'But dash it all, those two were the only ones with a shadow of a motive!'

'Yes', said Poirot. '*Motive*. It was that which has led us astray. If A has a motive for killing C and B has a motive for killing D -- well, it does not seem to make sense, does it, that A should kill D and B should kill C?'

Spence groaned. 'Go easy, M. Poirot, go easy. I don't even begin to understand what you are talking about with your A's and B's and C's.'

'It is complicated,' said Poirot, 'it is very complicated. Because, you see, you have here *two different kinds of crime*-- and consequently you have, you *must* have, two different murderers. Enter First Murderer, and enter Second Murderer.'

'Don't quote Shakespeare,' groaned Spence. 'This isn't Elizabethan Drama.'

'But yes, it is very Shakespearian -- there are here all the emotions -- the human emotions -- in which Shakespeare would have revelled -- the jealousies, the hates -- the swift passionate actions. And here, too, is successful opportunism. '*There is a tide in the affairs of men which taken at its flood leads on to fortune*...' Someone acted on that, Superintendent.'

- Agatha Christie, *Taken at the Flood*

I. Introduction

> "You already have the information. All the names and dates are inside your head. What you want – what you really need – is a story."
>
> - William Rookwood, *V for Vendetta*

I.1 Vision

Stories matter.

Stories matter because individual events, no matter how richly laden with deep structural architecture or encyclopedic meta-knowledge, are just events, cognitively barren without an understanding of how those singular pieces fit together. To understand characters' motivations, to detect familiar narrative patterns, to predict what comes next – these all require an understanding not just of events, but also of the relationships between events, of cause and consequence, of a story.

This thesis, the product of my graduate research of the past two and a half years, contributes to and explores the automatic generation of such relationships, transmuting simple lists of events into rich networks of interlaced plot points, ripe for the detection of common patterns both within and between stories. Such pattern-matching has myriad applications in as many domains, but it is my belief that this technology's greatest value lies within its integral role in creating a larger consciousness, that the capacity to learn from narrative analogy and apply that learning to other situations is the very root of advanced cognition.

I.2 Fiction: Reality Distilled

Literature and stories are grounded in human experience. Stories operating upon rules that do not closely approximate our own seem incorrect or unbelievable. Stories that speak to our own nature, that simulate what feels, to the reader, like a 'real' situation, however, make the bulk of what society might consider humanity's most enduring works. Consider the works of Shakespeare. Despite their age, his plays are no less tangible to

the modern reader for it. The centuries-ago affairs of the heart, of passion, of war, and of sorrow are no different from those people experience today.

Fiction might be considered a controlled simulation of the real world, or, to borrow from the terminology of machine learning, the training data to reality's test material. As a corpus of study, fiction works well because it was constructed specifically to guide, to educate, to entertain, and to caution. Folklore and mythology explain basic moral lessons (e.g. don't break rules, don't steal from others, don't wander into the forest alone) while more complex stories help us to understand human nature from a range of perspectives and situations or to convey our society's ideals and values. Indeed, fiction's proliferated use often makes its lessons indistinguishable from those of reality.

It is important to recognize that fictional stories operate as a close approximation of historical or current real world events. Such approximations are necessary because, typically, only incomplete information is available regarding real world events, particularly when the event is recent. Fiction, on the other hand, is specifically constructed to tell its readers a coherent story. After reading a fictional story, a reader can typically tell you which characters did what and often why. Stories of real world events, particularly those in the anonymously digital era, are frequently less articulate about their actors, their actions and motivations, and the resulting consequences.

In what I believe follows the original purpose of stories, I have analyzed fictional stories of known structure in order to better understand the parameters of real-world events and of life. My exploration of story understanding, its nuances, its limits, and its potential has centered on the continued development of Patrick Winston's AI research platform, the *Genesis* system.

I.3 Overview

This thesis is divided into seven chapters.

In Chapter II, I introduce the internal representations of the Genesis system, explain how meaning is attributed to words and sentences, and illustrate how this meaning allows enables the construction of stories from sequences of text.

Chapter III presents the stories that I have developed and explored in the course of my study, from a six-line interpretation of *Macbeth* to the crown jewel of my work, a sprawling rendering of Frank Herbert's science fiction epic *Dune* spanning over one hundred inter-connected events.

Chapter IV describes the collected rule set used in Dune, category by category, outlining the purpose behind each group of rules and the reasons for their present incarnation.

Chapter V discusses the numerous theoretical and mechanical obstacles that have arisen in the development of functional stories, ranging from the painfully specific to the hazy theoretical.

Chapter VI introduces the Self-Reflection Engine, a system of higher-order thinking that enables a significantly more powerful rule base through abstract compression. This introduction is followed by a necessarily complex discussion of a single example metarule, its merits, and its potential failings.

Chapter VII summarizes the contributions of this thesis.

II. The Language of Genesis

In this chapter, I describe the tools principally involved in building stories: mechanisms by which words and sentences are assigned meaning, processed, and understood. Together, these form the building blocks of language in Genesis: rich internal representations of words, events, and inferences, the very stuff of stories.

II.1 WordNet and Thread Memory

WordNet (Miller, 2006) is a lexical database of English language developed at Princeton University, cataloguing nouns, verbs, adjectives, and adverbs into groups of cognitive synonyms. When Genesis reads plaintext English, a parsing engine queries WordNet for the collective possible meanings of each parsed word. From each word's stored meaning classes, Genesis constructs a bundle of *threads* (Vaina and Greenblatt, 1979). Each thread in turn represents a different telescoping hierarchical definition or context for the word, beginning with highly abstract categories and slowly narrowing its scope with each successive word/category until ending with the thread's seed word. Consider the bundle of threads tied to the word "hawk:"

thing entity physical-entity object whole living-thing organism animal chordate vertebrate bird bird-of-prey hawk
thing entity physical-entity object whole living-thing organism person adult militarist hawk
thing entity physical-entity object whole artifact sheet board mortarboard hawk
action act interact transact deal peddle hawk
action get capture hunt hawk
action exhaust expectorate cough clear-the-throat hawk

Fig. 1. Bundle of threads meaning "hawk".

Notice that "hawk" may refer to the hunting bird, a militarist person, or the act of peddling goods. Exactly which definition is meant by an instance of "hawk" must be construed by context or explicitly given.

II.2 The Frame Representation

While threads may define the semantic meaning of a given concept, the thread itself is simply text. In order to understand and manipulate the information contained within a

thread, these concepts must be cast into relational representations usable by the Genesis system. These representations are called *frames*.

Patrick Winston (2008) has described four frame types for use in Genesis: *things*, *derivatives*, *relations*, and *sequences*:

- The *thing* frame reifies a single thread; it is the representation of a simple object or idea, such as 'book' or 'bird'.
- The *derivative* frame is used for those concepts that 'derive' from a single thing or object. For example, the location '*in* the library,' derives from 'the library'
- The *relation* frame defines a relationship or connection between two objects. For example, classification relation define an is-a relationship between two types, while most actions are represented as a relation between an actor and a target.
- The *sequence* frame acts as a simple list, containing any number of other frame objects. Rules frequently use a sequence to group a series of conditional events as the rule's antecedent.

Upon parsing a sentence of plaintext English, Genesis produces a frame representing the sentence semantically, an object the system can understand and manipulate. By nature, sentence frames are deeply structural. Though humans likely form similar structures in the process of understanding phrases, the concept is difficult to express with words and frequently must be visualized for full comprehension. Figure 2 illustrates a pair of frames representing statements from a story. Each vertical bar represents an individual frame, and each frame contains, as arguments, those frames pictured to its immediate right. Frame types may be identified by color: relations are red, derivatives blue, things gray, and sequences black.

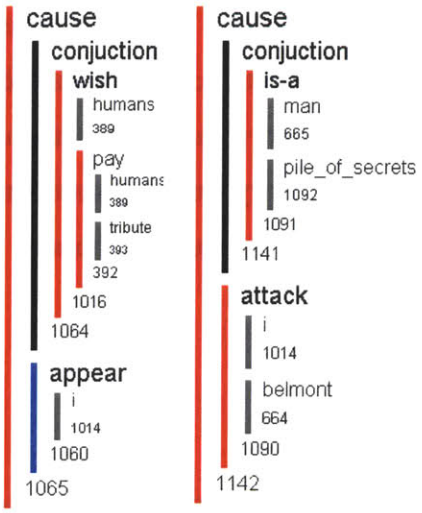


Fig. 2. Frames, often heavily nested, are powerful structural interpretations of sentences.

II.3 Rules

The central focus of automated story construction is the formulation of *rules*. Rules define relationships between a story's events, generate new implicit information, and explain motivation. Rules transform a series of independent sentences into a well-connected, cogent network of events and motivations. Rules create stories.

In the last section, I discussed how Genesis parses raw sentences to create frames. When a sentence describes an action, a classification, or an occurrence, this is called an *event*, and its corresponding frame expresses deep structural and semantic meaning. But understanding the individual events of a story is meaningless without an understanding of how those events relate to one another.

The principal purpose of rules is to automatically create and explain such relationships. To this end, rules perform two main operations based on the events of a story: rules generate new events, sometimes called *implicit* events or *inferences*, and rules generate connections between events, sometimes called *chains*.

Inferences serve as implicit intermediate events that express the effects of some events and refine the motivations of others. When a character is harmed by another, it is implied

that the victim will resent his aggressor. Where there is resentment, the desire for revenge may arise. Implicit events do not contain special defining content; indeed, any implicit event might be included in a story explicitly without altering its impact. Rather, implicit events tend to be those that a story simply does not need to express explicitly, the products of commonsense reasoning no capable human reader would not infer for themselves.

Consider this simple rule:

```
Harold dislikes William because William conquers Harold.
```

The portions of the rule preceding `because` are called the *consequent*, and the portions following, the *antecedent*. Notice that rules are written in a *Then because If* format. With this rule, a story that includes the rule's antecedent event

```
William conquers Harold.
```

generates the corresponding consequent event `Harold dislikes William` implicitly. Such a connection is visualized in a story as follows:

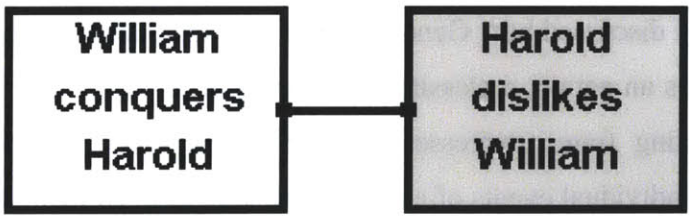


Fig. 3. Visualization of a simple prediction. Explicit events are pictured with a white background, while implicit events are given a gray background. Chains connecting events lead from the right side of an antecedent to the left side of a consequent.

Notice that the implicit event `Harold dislikes William` has been generated and added to the story, chained to the explicit event `William conquers Harold.` Because a rule's antecedent phrase was fulfilled (appeared in the story), the rule *fires*, automatically generating a new implicit event connected to its causal parent by a chain. This new event is, in turn, treated as a newly parsed event and may fire new rules of its own.

Chains lead from one event to another. While chains do not necessarily signify a strictly causal link between pairs of events, they do represent a related, if more abstract, connection. A chain between two events implies that one event has somehow lead to the other. Without saying for certain that the antecedent event has directly caused the consequent to occur, a chain states that it has nevertheless contributed. Any one event can lead to several others, and can be led to by several sources, as well.

As of this writing, two major categories of rules exist: *strong* and *weak* rules.

Strong rules, or *predictions*, absolutely fire whenever their antecedent is matched. When the appropriate consequent does not yet exist in a story, strong rules automatically generate the consequent as an *implicit* event. These sorts of rules typically define commonsense effects of certain actions, e.g.

```
Abel becomes dead because Cain murders Abel.
```

Weak rules, or *explanations*, do not necessarily fire once their antecedent is matched. Rather, weak rules are used to explain how explicit events might have connections with earlier events. When the antecedent of a weak rule is matched, a rule daemon observes the story and waits for the appropriate consequent to appear. Should this consequent later appear in the story, a chain is generated between the consequent and antecedent events, thus *explaining* the consequent's appearance.

Weak rules are typically used to describe actors' motivations, as in the following example:

```
Cain may murder Abel because Cain dislikes Abel.
```

I cannot be certain (and therefore use a strong rule to predict) that Cain will murder Abel because he dislikes him. However, if I already know that Cain dislikes Abel, and Cain later murders him, I can easily infer that Cain's dislike bore some connection with Abel's murder. Remember that a weak rule connection is not explicitly a *causal* connection; it is *motivational* and supports a story's coherence by establishing how events are interrelated.

III. Stories

In this chapter, I present some of the stories I have analyzed during my time working with the Genesis research group. These pieces began as quite short blocks of text, but grew in size with each passing story. Each story builds off the work of those that came before, beginning with mere six-line representations of Shakespeare's *Macbeth* and culminating in this thesis's master story, Frank Herbert's *Dune*.

III.1 Macbeth

My first experience with story understanding centered on an analysis of Shakespeare's *Macbeth* in light of Wendy Lehnert's (1981) research on plot units as a representation for stories. Lehnert theorized that stories could be rendered as sequences of interconnected positive, negative, and emotionally neutral 'mental' states for characters over time. My research attempted to cast *Macbeth* into such a framework, seeking out the rules required to automatically generate the kinds of relationships that Lehnert had described.

Being the first story that I approached in this manner, my earliest rendition of *Macbeth* was greatly compressed. In a scant six lines and ten rules, I described only a handful of the play's motivating relationships and murders, yet this proved sufficient information to produce patterns that Lehnert defined as a 'success', a 'mistake', and a 'revenge':

Story:
- S1. Duncan is King.
- S2. MacBeth is Duncan's successor.
- S3. Duncan is MacDuff's friend.
- S4. MacBeth wants to be King.
- S5. MacBeth kills Duncan.
- S6. MacDuff kills Macbeth.

Rules:
- R1. If a friend is harmed, your friend's harmer harms you.
- R2. If the king dies, the king's successor becomes king.
- R3. If you want X, then you are happy when X, and you are unhappy when not X.
- R4. If you are harmed, you become unhappy.
- R5. If you are harmed, then you dislike your harmer.
- R6. If you dislike X, you want to harm X.
- R7. Murder – If X murders Y, then: · X harms Y and · Y becomes dead
- R8. Procedural Blaming – traces up chain of causal events (if any) to blame instigator.
- R9. If you die, you are not alive.
- R10. People are alive by default. [Assume at introduction of character, until death]

Fig. 4. Early rendition of *Macbeth* story.

Although written in preliminary pseudocode, many of these rules would later be translated into modern analogues for use in the Genesis system. During such translation, however, some of the more complex rules (e.g. *R3*) revealed assumptions, some explicit and some subtle, that I had made about rule construction. Several of these conceptual obstacles are discussed in greater length in Chapter IV in their relevant sections.

III.2 Dune

In the years following my foray into Shakespeare's works, I studied numerous other stories, striving to explore interesting domains that we might generate interesting rules. From conflicts in Darfur and Estonia to the digital sparring of Google, China, and East Asian figure skating fans, I have distilled a powerful battery of potential rules outlining the interplay of nations, the dynamics of groups, and fundamental aspects of human action. That battery and more come together to form the cognitive backbone of this thesis's core work: *Dune*.

III.2.1 The Story of Dune – Background

The plot of Dune is set in the far-distant future in a period where space travel is ubiquitous, Earth is a forgotten cinder, and the known universe is ruled through a House system, presided over politically by an Emperor and economically by the Spacing Guild. The central feature of this universe surrounds a precious resource: the spice mélange (or simply "spice"), which uniquely enables space travel. The spice can only be found on one planet, Arrakis, better known as Dune.

From a high-level international perspective, the events of the book revolve around a multi-house struggle to control Arrakis, the unrivaled production of spice, and consequently, all investments in space travel. Dune's operating form of 'states' are the major and minor Houses of the Landsraad. At the time of Dune's telling, there are three Houses of serious consequence: Houses Atreides (home to the story's hero, Paul), Harkonnen (their sworn enemies), and Corrino (the House of the reigning Emperor Shaddam IV). Although Shaddam holds the title of "Ruler of the Known Universe," the Spacing Guild hold great sway over the Houses of the Landsraad, and are generally free

to dictate terms to the Houses. It is the Guild's highest priority that one of the Houses control Arrakis and maintain spice production, enabling space travel.

III.2.2 The Story of Dune – Plot

Act I

Dune begins with an audience between the Emperor Shaddam and a Navigator of the Spacing Guild. The Guild holds great sway over the operations of the universe, and as such, the Guild intimidates House Corrino (as well as other bodies). The Navigator, being a representative of the Guild, thereby intimidates Emperor Shaddam as a member of Corrino. Navigators possess the ability of prescience, and as such, have foreseen that young Paul Atreides, son of Duke Leto, the leader of House Atreides, will someday threaten the production of spice. As it is the Guild's highest priority to ensure spice production, the Guild wishes for Paul to be killed before this threat can manifest. Given the Guild's intimidation, the Navigator compels the Emperor to seek Paul's death.

Act II

This second act contains little story advancement, but rather defines the necessary relationships between individuals, leaders, states, and blocs to activate rules about international relations.

The information concerning government structure can essentially be expressed as a tree:

- Guild [state]
    - Represented by: Navigator
    - Intimidates Corrino
- Landsraad [bloc]
    - Values: Spice
    - Governed by: Corrino
    - Corrino [state]
        - Led by: Emperor (Shaddam IV)
    - Harkonnen [state]
        - Led by: Baron
            - Baron hates Atreides
        - Other members: Feyd
        - Controls: Dune, GiediPrime

- Atreides [state]
  - Led by: Leto
  - Other members: Paul (successor)
  - Controls: Caladan

Dune, Caladan, and Giedi Prime are all planets, but are rendered here simply as *regions*, as each planet acts roughly atomically as territory. It is further noted that the region Dune contains Spice.

Acting on his newly compelled motivations and exercising his power as the leader of the governing body of the bloc that contains the current controllers of Dune (trust me), Emperor Shaddam seizes control of Dune from the Harkonnen and grants it, instead, to Atreides. Because of the Baron's (and consequently the entire House Harkonnen's) hatred for the Atreides, Shaddam knows that this slight in favor of the Atreides will anger the Harkonnen and fuel their vendetta. This may result in endangering members of Atreides, which of course includes Paul.

Acts III and IV

As I am sticking to a purely international scope in the telling of Dune, I skip the bulk of this story except to note the repeated turnover of Dune, several attacks, and the murder of a few key characters during those same assaults.

Following the transfer of Dune from Harkonnen to Atreides hands, the Harkonnen launch an assault on the Atreides. In the ensuing struggle, Duke Leto is killed, but Paul escapes into the desert. Victorious, the Harkonnen regain control of Dune and the spice production continues.

In the desert, Paul meets the nomadic tribes of the Fremen, a desert people intimate with the ways of Dune and the spice, but longing to free their planet from the tyrannical grasp of the Landsraad Houses. Paul seizes upon the opportunity to avenge his father and destroy the Harkonnen by joining the Fremen and ultimately becoming their leader. Under Paul, the Fremen conduct a great assault upon the Harkonnen, kill the Baron and

Emperor Shaddam, and seize control of Dune for themselves. As the Fremen now control Dune and are not members of the Landsraad, spice production ceases.

III.2.3 Dune Text

The following is a slightly altered version of the plaintext story file representing Dune. Code comments, flags, and unused alternate forms have been removed.

```
Clear text.
Clear story memory.

Insert file macbeth reflective knowledge.
Start commonsense knowledge.

XX, YY, and ZZ are entities.
Group1 is group. Group2 is group.
BB is bloc.
RR is region.
SS is state.
RSC is resource.

//Desire
XX becomes happy because XX wanted an action to occur and the action occurred.
XX becomes unhappy because
        XX wanted an action to occur and the action did not occur.

//Failure
XX fails action because
        XX wanted the action to occur, the action did not occur, and XX became dead.

// Intimidation and Compulsion
 // Intimidate → Compel
 XX may compel YY to want an action because XX intimidated YY.

 // Intimidation via Grouping
 XX may compel YY to want an action because Group1 intimidated YY, XX is a
        member of Group1, YY is an entity, and Group1 is group.
 XX may compel YY to want an action because Group1 intimidated Group2, XX is a
        member of Group1, and YY is a member of Group2.
 XX may compel YY to want an action because XX intimidated Group2, YY is a
        member of Group2, YY is an entity, and Group2 is group.
 // Compulsion Semantics
 YY wants an action because XX compels YY to want the action to occur.
```

// Delegation
XX may want YY to want an action because XX wanted the action to occur.

//Agency
XX murders YY because Agent1 is XX's agent, and Agent1 kills YY.
Agent1 is XX's agent because XX leads Agent1.

//Leadership and Representation
XX becomes a member of YY because XX joins YY, and because YY is group.
XX is a member of YY because XX represents YY.
XX is a member of YY because XX leads YY.
YY wants an action because
    XX leads YY and XX wants the action to occur and XX is an entity.
XX wants an action because
    XX represents YY and YY wants the action to occur and YY is an entity.
Group1 hates YY because XX hates YY, and XX leads Group1.

//Death
XX becomes dead because XX dies.

//Government
XX governs YY because XX governs BB, BB includes YY, and BB is bloc.
XX governs YY because XX leads Group1, Group1 governs YY, and Group1 is group.

SS values RSC because SS wants RSC.
SS values RR because SS values RSC, and RR contains RSC.
SS wants to control RR because SS values RR.
SS wants to control RSC because SS values RSC.
SS controls RSC because RR contains RSC, and SS controls RR.

XX values RSC because BB values RSC, and BB includes XX.

// Land Seizure
XX may seize RR from YY because XX governed YY, and YY controlled RR.
XX controls RR because XX seized RR from YY.
Group1 controls RR because XX controls RR, and XX leads Group1.
XX may grant RR to ZZ because XX controls RR, and XX governs ZZ.
ZZ controls RR because XX granted RR to ZZ, and XX governs ZZ.

//Rivalry and Land Seizure
YY harms ZZ because
    XX seized RR from ZZ, XX granted RR to YY, and ZZ dislikes YY.

//Hatred
XX dislikes YY because XX hates YY.

//Assaults
XX may murder YY because XX attacks YY.
YY may murder XX because XX attacks YY.
XX may murder YY because
    XX attacks Group1, YY is a member of Group1, and Group1 is group.
YY may murder XX because
    XX attacks Group1, YY is a member of Group1, and Group1 is group.
XX may murder YY because Group1 attacks Group2,
    XX is a member of Group1, and YY is a member of Group2.
YY may murder XX because Group1 attacks Group2,
    XX is a member of Group1, and YY is a member of Group2.

// Basics of disliking those who harm you or your friends
Insert file Hamlet commonsense knowledge.

// Additions to Hamlet
XX may attack YY because XX dislikes YY.
XX may murder YY because YY harmed XX.

Start story.

// ACT I - IMPERIAL CONSPIRACY
Guild, Corrino, and Atreides are nations.
Emperor, Navigator, Leto, and Paul are persons.
Emperor leads Corrino.
Navigator represents Guild.
Paul is member of Atreides.

Guild intimidates Corrino.
Navigator wants Paul to die.
Navigator compels Emperor to want Paul to die.

// ACT II - LANDSRAAD GOVERNANCE
Harkonnen is nation.
Baron and Feyd are persons.
Baron leads Harkonnen.
Baron hates Atreides.
Feyd is member of Harkonnen.
Leto leads Atreides.
Leto is Paul's father.

Landsraad is bloc.

Landsraad includes Atreides, Corrino, and Harkonnen.
Corrino governs Landsraad.

GiediPrime is region.
Caladan is region.
Dune is region.
Spice is resource.
Dune contains spice.

Landsraad values spice.

Harkonnen controls GiediPrime.
Atreides controls Caladan.
Harkonnen controls Dune.

// ACT II.V -- SEIZURE OF ARRAKIS
Emperor seizes Dune from Harkonnen.
Emperor grants Dune to Atreides.

// ACT III - HARKONNEN TREACHERY
Harkonnen attacks Atreides.
Baron murders Leto.
Harkonnen controls Dune.

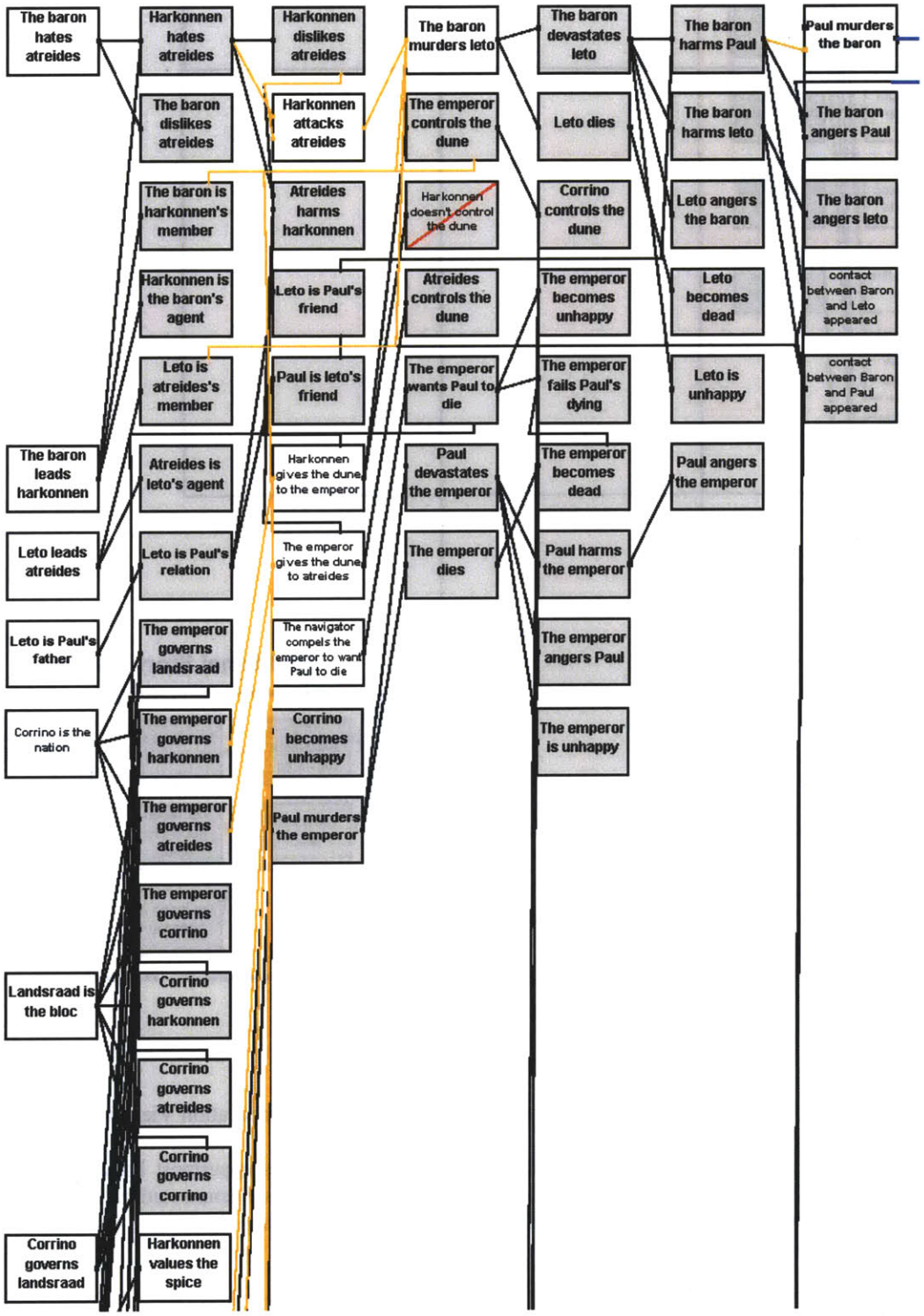
// ACT IV - FREMEN RISING
Paul does not die.
Paul joins Fremen.
Paul leads Fremen.
Fremen attack Harkonnen.
Fremen kills Baron.

Fremen kills Emperor.
Fremen controls Dune.

The end.

III.2.4 Dune Visualized

The following images comprise the visual representation of Dune, as viewed by the Genesis Elaboration Graph. White cells represent explicit statements, gray cells represent implicitly generated statements, and wires signify a "leads to" relationship.



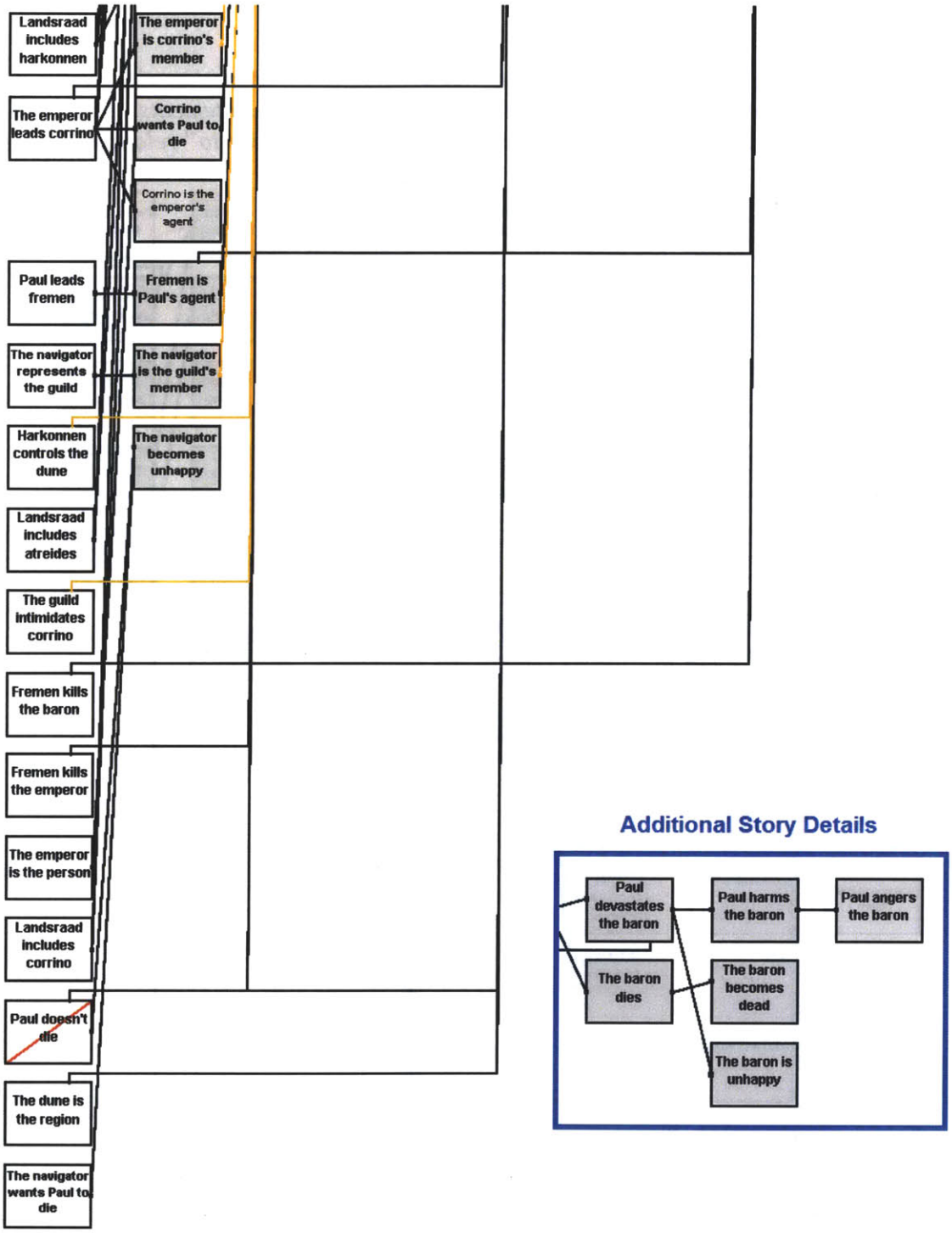


Fig. 5. *Dune*. Story of Dune as visualized by Genesis' Elaboration Viewer.

For the sake of resolution, some extra details from the story's uppermost branch have been removed and pictured within the inset. The group of details is connected to the main story graph via the upper-right blue lines wired to "Paul murders the Baron" and "The Baron angers Paul."

IV. Rules

In this chapter, I present the blocks of rules shown in Dune. Being that Dune effectively encompasses the rules of each milestone story before it, this same set of rules may be understood as the current complete rule set for Genesis stories. Within each block, I discuss the motivations in writing the rules as they appear, the issues that appeared surrounding the rules, what I have done to accommodate those issues (where applicable), and any further extensions that these rules might need. These issues are often heavily entangled with back-end processing issues, which are described in Chapter V. I provide pointers to the relevant sub-sections of Chapter V for clarification where needed.

IV.1 Desire

```
XX becomes happy because XX wanted an action to occur and the action
occurred.
XX becomes unhappy because XX wanted an action to occur and the action
did not occur.
```

Fig. 6. Desire. Rules describing goal fulfillment and failure.

The desire rules form the foundation for actualizing actors' goals. "Become happy" and "become unhappy" are explicitly translated into "<XX>'s mental state becomes +/-", some of the atomic story pieces used for creating and finding plot units. The language of mental states is inspired by Wendy Lehnert's (1981) work on plot unit architecture, with some differences between our representations. Whereas Lehnert's plot units cast actions as being either 'positive' or 'negative' actions, the Genesis representation focuses on characters' 'positive' and 'negative' mental states, psychological states of mind that in turn inspire and motivate the actions of a story.

See also: Variables, Reification

Further Thoughts: Perhaps I ought to include a new rule that links wanting to perform an action and doing that action. The principal actor of the action must be the same actor that wants the action to begin with, however, or this connection cannot exist. This may be as simple as:

```
XX may perform an action because XX wanted to perform the action.
```

However, I suspect that this concept may need be written as a *metarule*, as described in Chapter VI: Self-Reflection.

IV.2 Failure

XX fails action because XX wanted the action to occur, the action did not occur, and XX became dead.

Fig. 7. Failure. Rule describing when an action fails to occur.

Whereas Desire captures actors' happiness with satisfied wants, Failure captures their disappointments. A failure should typically occur when someone wants something to happen, and the desired events fail to occur. However, this definition is tricky at best, for when can I truly say that something has not happened? Where can I safely define a cut-off point, when there might always be some 'next event' that includes the failing action? These are difficult questions to answer, and I posit two preliminary cases for when actions may be said to not occur:

Firstly, an action fails to occur if, at the end of the story (or, in future, a smaller chapter), the action is not contained within the list of story events, explicit and implied together. In this case, I search only for actions which have been explicitly mentioned previously in the story, as these are the only ones likely to have any impact. (Obviously, I cannot and should not analyze every possible permutation of the set of all actions for each story.) For now, I define such relevant actions to be those that any actor wanted to occur, but did not occur at any point in the story up to that moment (post-implicit processing).

Secondly, I may also consider that characters who want an action fail that action upon their death. For individual characters, death is similar to a personal "The end" statement and serves the same concluding purpose. This case benefits greatly from expressing failure by blaming a particular actor with failure (XX fails...).

However, it is critical that the failure of an action be analyzed only after the implicit effects of the character's death have been fully calculated. In cases where a character, for example, explicitly wants to die or wants their successor to inherit their title, the

character's death actually fulfills this desire, and it would be wholly false to say that the actor had failed.

Further Thoughts: I think I might well distinguish from two types of failure. Consider the following cases, which I define with their natural linguistic forms: the action "... did not occur" and "... has not occurred". It may become helpful to have some rules operate on both forms separately.

"Did Not Occur": Some actions might explicitly fail; that is, a story might contain "George did not mow the lawn." I can explicitly declare that the failure of an event is itself an event, but to do so for every possibility is both infinite and unhelpful. These events "did not occur."

"Has Not Occurred": Some actions might have implicitly failed at a certain point. That is to say, I might check at a given point whether a particular event has yet occurred within the storyline. If a search fails to turn up a matching event, then this event "has not occurred."

IV.3 Intimidation and Compulsion

XX may compel YY to want an action because XX intimidated YY.
XX may compel YY to want an action because Group1 intimidated YY, XX is a member of Group1, YY is an entity, and Group1 is group.
XX may compel YY to want an action because Group1 intimidated Group2, XX is a member of Group1, and YY is a member of Group2.
XX may compel YY to want an action because XX intimidated Group2, YY is a member of Group2, YY is an entity, and Group2 is group.
YY wants an action because XX compels YY to want the action to occur.

Fig. 8. Intimidation and Compulsion. Rules describing how goals can be compelled in others through intimidation and group dynamics.

Especially as I take to political analysis, I find it necessary to consider cases where entities can impose their motivations upon other actors. As I see it, compulsion can really only alter an actor's motivations. It is not correct to say Japan compels America to increase its research budget. Rather, Japan compels America to want to increase its research budget, and further, America then does this. While it may eventually be useful

to accept the first version as shorthand for expressing both sentiments at once, such a 'shortcut' is a more a relic of human conversation than a true narrative concept.

Consider this visualized example of compulsion at work from a compulsion test case:

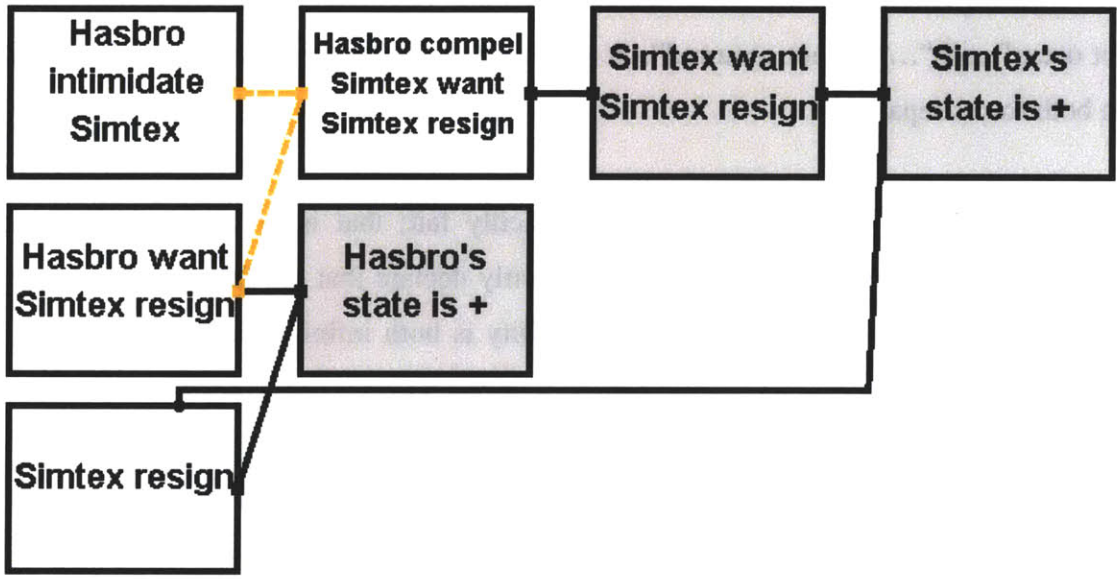


Fig. 9. Hasbro Take-Over. Test case representing early compulsion rule set in operation.

Hasbro's compulsion sets up an environment whereby Simtex could either resign or fail to resign, and both parties would be happy/unhappy, respectively. Remember that Hasbro's compulsion does not force the resignation; it is an independent event that must be declared explicitly.

See also: Variables

Further Thoughts: This is one of the earliest examples of groups transferring their properties to their members. It would be very useful to somehow abstract this process, but not every property of a group can be transferred to every one of its members. Until I can better define which powers transfer to which members, some elements of group dynamics will have to wait.

IV.4 Agency

XX murders YY because Agent1 is XX's agent, and Agent1 murders YY.
Agent1 is XX's agent because XX leads Agent1.

Fig. 10. Agency. Rules defining one instance of the agent relationship and one action whose responsibility is transferable.

Agency refers to the relationship between an actor (the leader) and some other entity that operates completely under that actor's control (the agent). In this relationship, almost anything an agent does, the leader does.

The present working definition of Agency is a poor description of its intended effect. Foremost, there are serious scoping issues with regard to which actions a leader repeats. I suspect that after accomplishing some powerful new changes (see Verb Abstraction), Agency will include a greater span of actions and become a powerful rule set. Secondly, "leads" is a poor choice of verb for defining an agent relationship, as leaders can potentially lead many non-agents, even under the operating language of Dune as it stands. For now, "leads" only suffices because I have limited the number of agency actions down to 'murder,' specifically because it only comes up once as a state's action in Dune. Again, I must find a better solution for framing agency.

The concept of Agency was first suggested by Patrick Winston in a hypothetical suggestion concerning Eisenhower's Army:

1. If an entity's agent does something then the entity does that thing.
2. If entity wants something to occur and it occurs, then entity is happy.
3. Eisenhower wants to invade France.
4. Eisenhower's army invades France.
5. 1 + 4 ==> Eisenhower invades France.
6. 2 + 3 + 5 ==> Eisenhower is happy

Fig. 11. Eisenhower's army: A hypothetical situation suggesting agency.

Line 1 encapsulates the crux of the agency problem. Only some actions should trickle up from the agent to the leader, not all actions. For now, I suspect that Verb Abstraction, the ability to refer to verbs by wide-spanning ancestor threads, will almost instantly solve this issue for us by allowing us to write the rule:

XX changes YY because Agent1 changes YY, YY is an entity, and Agent1 is XX's agent.

This will allow actions such as "move" and "kill" (both descendants of change) to trickle up, but not descriptive actions, such as "Eisenhower's army is large." Any interpretation of "Eisenhower is large" cannot rightfully be interpreted from the previous statement. I suspect that WordNet may reveal that English verbs are categorized such that a few high-level verbs essentially act as hierarchical ancestors to a wide breadth of other verbs. By identifying these hypothetical ancestral verbs, I may discover that a select few verbs can be isolated as the appropriate sorts of actions that ought to transfer properly by agency.

See also: Verb Abstraction

IV.5 Leadership and Representation

XX becomes a member of YY because XX joins YY, and because YY is group.
XX is a member of YY because XX represents YY.
XX is a member of YY because XX leads YY.
YY wants an action because XX leads YY and XX wants the action to occur and XX is an entity.
XX wants an action because XX represents YY and YY wants the action to occur and YY is an entity.
Group1 hates YY because XX hates YY, and XX leads Group1.

Fig. 12. Leadership and Representation. Rules describing group dynamics.

With leadership and representation, I explore the relationship between the motives of a group and its key players. These relationships are based around the transfer or sharing of motivations and goals between the group and the member. The leader of a group shapes the goals of the group, so a group inherits the goals of its leader. The representative of a group is meant to represent the group's interests, and so a representative inherits the goals of its group. Summarily, groups want what their leaders want, and representatives want what their groups want.

While this approach may lean towards a particularly dictatorial approach to group dynamics, my method serves as a strong approximation for most of the stories I have examined to date. Obviously, as different types of group/leader relationships are folded into Genesis' rule sets, these relationships will require their own models.

The last rule of this set, concerning hatred, is an example of a rule that could benefit heavily from Verb Abstraction. Really, the rules

> YY wants an action because XX leads YY and XX wants the action to occur, and XX is an entity.

and

> Group1 hates YY because XX hates YY, and XX leads Group1.

stem from the same abstract notion: If a leader feels/wants something, the group does as well. Capturing just which actions demonstrate this kind of transference will serve as a critical foundation for understanding and analyzing group dynamics in stories.

See also: Verb Abstraction

IV.6 Government

XX governs YY because XX governs BB, BB includes YY, and BB is bloc.
XX governs YY because XX leads Group1, Group1 governs YY, and Group1 is group.

Fig. 13. Government. Rules describing governing relationships.

The government rule block is fairly simple. The first rule merely distributes a "govern" relationship across the members of a bloc (a collection of states or other groups). The second is a similar transitive rule substantiating the roles of leaders in governing bodies.

IV.7 Regions and Resources

SS values RSC because SS wants RSC.
SS values RR because SS values RSC, and RR contains RSC.
SS wants to control RR because SS values RR.
SS wants to control RSC because SS values RSC.
SS controls RSC because RR contains RSC, and SS controls RR.
XX values RSC because BB values RSC, and BB includes XX.

Fig. 14. Regions and Resources. Rules describing the desire of national entities (states) for valuable property.

Drawing from Kaplan's (1957) political science theory, I developed these rules to describe the most fundamental elements of states' operation: demand for resources. States want to control resources and territory, but only some of these are valuable. Indeed, territory is rarely valued in and of itself. Rather, the region contains (valuable)

resources, has strategic worth, or possesses some alternative meaning (ancestral burial grounds come to mind). These rules ensure that states desire to control only those resources that they value and only those regions that have a reason to be considered valuable. (Here, I only specify those regions that contain valuable resources, but this qualification can easily be extended to include other conditions.)

Further Thoughts: Kaplan's defined parameters for varying states of international equilibrium provide distinct priorities for states in that system. These typically include attaining resources, preferring resources to fighting major wars, and fighting minor wars to major wars. To fully implement these sorts of priorities, I will need to develop a system of weighted goals whereby desires can be understood in balance with one another. However, this may require that I break free of the binary observation of mental states, and move away from "+" and "-" to some form of how positive or negative reactions are.

IV.8 Seizure of Territory

XX may seize RR from YY because XX governed YY, and YY controlled RR.
XX controls RR because XX seized RR from YY.
Group1 controls RR because XX controls RR, and XX leads Group1.
XX may grant RR to ZZ because XX controls RR, and XX governs ZZ.
ZZ controls RR because XX granted RR to ZZ, and XX governs ZZ.
//Rivalries
YY harms ZZ because XX seized RR from ZZ, XX granted RR to YY, and ZZ dislikes YY.

Fig. 15. Seizure of Territory. Rules describing the transfer of land control by seizure.

Dune's seizure rules are story-specific and need refinement before they might be deployed in another story. As they stand, they summarily declare that a governor may seize territory from one of the governed and grant it to another. These actions individually shift control of the territory in question, but unfortunately, without a working understanding of the chronological relationships of events, the understood result is that every house in Dune simultaneously controls Arrakis.

Seizures' role in rivalries is an unfortunately clumsy hard-coded rule. The rule is meant to capture how rivalries can be fueled when a third party harms one member of a feud and aids the other. In this case, the Harkonnen use the transfer of Arrakeen power to the

Atreides as an excuse to attack the Atreides, when traditionally one might expect more resentment towards the Emperor for taking the power from the Harkonnen in the first place. And yet this development does not feel unnatural. People frequently blame the colleague promoted over themselves or the neighbor who earns an unexpected windfall. Indeed, such feelings are heavily magnified when fierce rivalries are involved. It seems likely that a broader rule set will ultimately be required to truly capture the feelings of human competition.

See also: Variables, Chronology

IV.9 Assault

```
XX may murder YY because XX attacks YY.
YY may murder XX because XX attacks YY.
XX may murder YY because XX attacks Group1, YY is a member of Group1, and Group1 is group.
YY may murder XX because XX attacks Group1, YY is a member of Group1, and Group1 is group.
XX may murder YY because
        Group1 attacks Group2, XX is a member of Group1, and YY is a member of Group2.
YY may murder XX because
        Group1 attacks Group2, XX is a member of Group1, and YY is a member of Group2.
```

Fig. 16. Assault. Rules describing the potential effects of fighting.

The rules for assault and the potential for harm are limited in a number of respects, but sufficiently encompass the needed relationships for an understanding of Dune. With correct Verb Abstraction, I can allow for the possibility of any form of harm (to either party) to result from any form of attack. The greater bulk of this rule block, however, deals with expanding the initial definition to include groups correctly. That is, I allow that in the relationship of

```
[AA attacks BB] causes [CC may murder DD]
```

Both CC or any group of which CC is a member may substitute for AA, and similarly for DD and BB. These pairings may also be reversed as either party might be harmed.

See also: Verb Abstraction

V. Story Processing Concepts

In this chapter, I describe a variety of problems that I have encountered in the back-end processing of Genesis. These are problems centering on the active cognitive theory behind this research, the construction of rules, and the specific processing of Genesis itself. Sections V.1 through V.3 describe my solutions to various obstacles in attaining a proper understanding of the rules described in the last section and in constructing rules properly reflective of that understanding. Sections V.4 through V.6 similarly describe solutions to obstacles in story cognition, although these are particularly complex and as yet remain unimplemented.

V.1 Variables

I should first acknowledge that all rules in Genesis operate using variables. When I write "Bob may harm George because Bob dislikes George," Bob and George both act as variables. Instances of applied rules are detected by attempting to match the actors and objects of a story with Bob and George, mapping actors appropriately. I have since ceased using specific names in favor of the more consistent system of variables like XX, YY, "a person," and Group1. Such terms are what I mean when I say *variable*.

V.1.1 Entities

Stories can more or less be defined by the actors who take part in them and the actions they take, coupled with an appropriate set of background knowledge. But what counts as an actor? My original analyses of Shakespeare's works (e.g. Hamlet, Macbeth) operated on the principle that human nature does not change, that stories that were compelling both four centuries ago and today must demonstrate the very essence of human nature. But I cannot limit human nature to only persons. Stories might equally well use nations, races, or proverbial animals, and any story analysis must be prepared to handle such cases. The appropriate high-level class that encompasses all these concepts is *entity*. Thus, most rules are defined using "entity" variables such as XX and YY.

V.2 Negation

Until recently, I had difficulty writing rules that properly reflected the semantic meaning of negation: that something not happening was the opposite of its occurrence. Consider the sentence:

Paul does not die.

This sentence would be parsed in a strange way. "Not" described how Paul dies, in the same manner that "Paul dies gruesomely" would describe Paul's death as being gruesome. Paul's death was 'not', but without the semantic impact of actual negation. Consequently, the negated sentence acts as a version of "Paul dies," and the typical events caused by death would fire (e.g. Paul became unhappy, his friends were upset). Negation-based sentences such as the "not die" example now fire properly.

V.3 Reification

Before I discuss Reification, I must discuss actions, how they exist, how they are modified, and how they ought to be constructed. I will draw from the example in V.2:

Paul does not die.

When I negate Paul's death, do I mean that 'Paul not-died', some action distinct from dying? Or instead, am I saying that 'Paul died' did not happen? In dealing with negation and many other cases, I have found that preserving the original core action significantly strengthens the understanding of the action's modifiers. Consider the following examples:

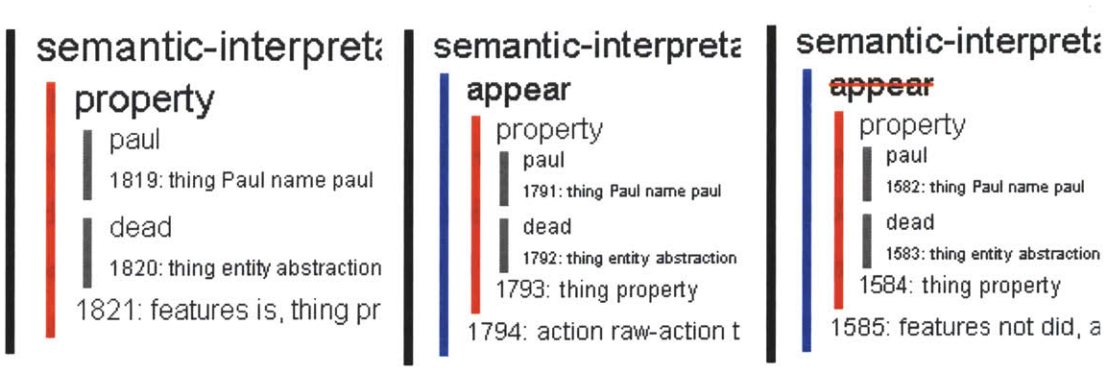


Fig. 17. Reifying actions through wrappers. The core action of Paul's death is never actually altered.

Figure 17 shows the basic semantic interpretations of three statements:

(a) *Paul is dead.*

(b) *Paul becomes dead.*

(c) *Paul does not become dead.*

Statement (a) is actually rather dangerous to use in a story, as it *describes* Paul as being dead, in much the same way that I might describe him as being tall. Afterward, the "dead" descriptor will become a new thread on the Paul object. Because this thread is also time-independent, every instance of Paul, past and present, will also be considered "dead," easily creating incorrect inferences.

Gary Borchardt's research (Borchardt, 1994) indicates that changes (such as [not] Appear, Disappear, Increase, Decrease, or Change) between states offer more salient information than descriptions of the states at discrete instances. With the value of denoting change in mind, I instead wrap the statement of Paul's death in an *appear* action, denoted in English by (b) "Paul becomes dead." The sentence thereby changes from a simple statement of fact to an event. Indeed, the event of someone's death is the more sensible trigger for other events. That Paul is dead may be sad, but surely it is when he becomes dead that his friends' sadness or anger most importantly manifests.

Statement (c) illustrates how I can use reifying techniques to handle other action modifiers. Rather than altering the core action of "Paul is dead," I attribute the negation of the statement to the enveloping 'appear.' Why is it useful to preserve the core action? Conceptually, actions are best described by their most elemental forms, with any

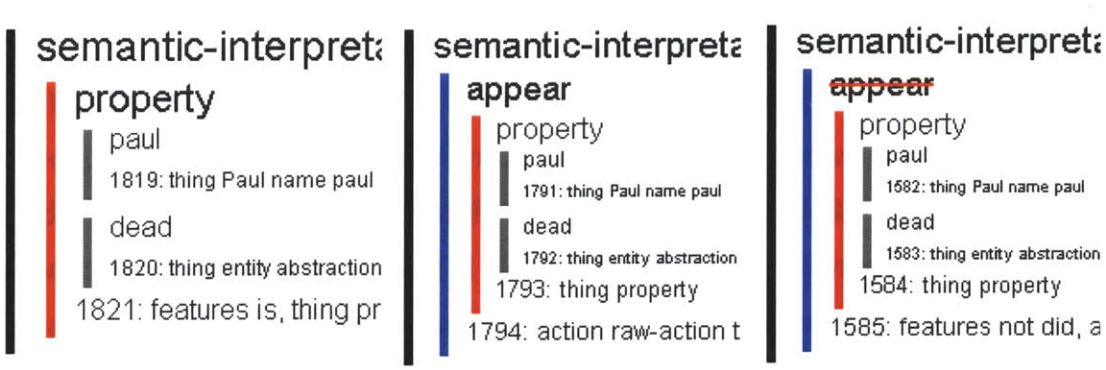


Fig. 17. Reifying actions through wrappers. The core action of Paul's death is never actually altered.

Figure 17 shows the basic semantic interpretations of three statements:

(a) *Paul is dead.*

(b) *Paul becomes dead.*

(c) *Paul does not become dead.*

Statement (a) is actually rather dangerous to use in a story, as it *describes* Paul as being dead, in much the same way that I might describe him as being tall. Afterward, the "dead" descriptor will become a new thread on the Paul object. Because this thread is also time-independent, every instance of Paul, past and present, will also be considered "dead," easily creating incorrect inferences.

Gary Borchardt's research (Borchardt, 1994) indicates that changes (such as [not] Appear, Disappear, Increase, Decrease, or Change) between states offer more salient information than descriptions of the states at discrete instances. With the value of denoting change in mind, I instead wrap the statement of Paul's death in an *appear* action, denoted in English by (b) "Paul becomes dead." The sentence thereby changes from a simple statement of fact to an event. Indeed, the event of someone's death is the more sensible trigger for other events. That Paul is dead may be sad, but surely it is when he becomes dead that his friends' sadness or anger most importantly manifests.

Statement (c) illustrates how I can use reifying techniques to handle other action modifiers. Rather than altering the core action of "Paul is dead," I attribute the negation of the statement to the enveloping 'appear.' Why is it useful to preserve the core action? Conceptually, actions are best described by their most elemental forms, with any

added another layer of appearance, these actions were not considered identical and the appropriate redundancy countermeasures did not fire properly. This bug has since been fixed by running actions through a cleaning procedure that strips the actions of all but one appear wrapper. It is also advised that future rules not include "to occur" in their consequent, although the aesthetic flaws of omitting this phrase are unfortunate.

V.4 Verb Abstraction

By using WordNet's threads, every parsed word is enriched with a deep hierarchical understanding. Using threads, I can refer to a wide variety of words by their high-level, perhaps abstract, forms. Rules that apply to chordates readily fire upon reference to either a fish or a human. The same goes for bodies-of-water, rivers and lakes. Indeed, this very principle forms the primary influence for my choice of *entities* in dealing with actor-based rules.

Sadly, English verbs do not demonstrate the same hierarchical richness boasted by nouns. Figure 19 illustrates how the thread representations of three seemingly related actions actually have very little in common.

action touch hit punch
action move propel hit shoot
action change affect hit strike

Fig. 19. Threads sharing common words might not share lineage.

Though all violent, potentially harmful words, Although all three words, in the given contexts, are descendants of the word "hit," each descends from "hit" by way of different interpretations of "hit" itself. Consequently, the three words ought not to be expected to match a rule for "hit" in the same way, and are not considered to be children of the same parent thread.

Consequently, building rules with the intention of abstractly matching many actions at once grows significantly more difficult. However, suppose special structures could be created to serve as these 'missing' mid-level verbs, perhaps standing in for many existing verb threads in parallel. As English simply does not have the words to express these

conceptually useful ideas, these nameless constructs must be represented symbolically. By creating such abstract verbs, sets of specific and appropriate actions otherwise unavailable may be targeted during rule construction.

Verb Abstraction particularly enables such rule sets as agency and group dynamics. In these relationships, a number of different actions performed by one member of the relationship are reiterated by their counterpart. By carefully selecting and populating an abstract verb concept that encompasses only those actions rightfully transferrable in such relationships, such rules grow both simpler and more powerful. Agents' actions are often attributed to their leaders, while group rules regularly transfer motivations between their members and the group as a whole. However, only some actions are meant to be transferred in these relationships. When Eisenhower's Army conquers a territory or kills a leader, these actions ought to be attributed to Eisenhower. Such attributions enable leaders to use their armies as extensions of themselves to fulfill goals such as revenge. Conversely, the descriptive actions of agents should not carry over to their leaders in the same manner. It would be a mistake to deduce that Eisenhower was large because his army was large.

V.5 Chronology

V.5.1 Starting and Stopping

Genesis has a limited sense of time. When stories are processed in Genesis, all events are treated as coexistent, frequently leading to collisions when contradictory events can create further contradictions. The greater problem here is that there exists no working concept of a "current state" as stories are stepped through. Facts cannot be only currently true, and previous information cannot be overridden. Such problems emerge most commonly when two opposing events occur in the same story – a frequent occurrence in violent stories where actors may be alive for part of a story and dead for the remainder. Unfortunately, any consequent rules contingent on the actor's life or death will be unable to distinguish between whether said actor is currently alive or dead. Indeed, such rules are handled with the assumption that the actor is, in fact, both alive and dead.

Admittedly, I have yet to encode any knowledge of mutually exclusive opposites in Genesis, and without such information, there is no reason to expect the system to conclude that "alive" and "dead" are conflicting descriptors. To Genesis, stating that an actor is both "alive" and "dead" is as equally permissible as describing Santa as both "fat" and "jolly". Although a system for opposite detection may seem an easy fix, problems arising from the lack of a "current state" are not always cases for opposites. Consider instead the problem of territorial control contained in Dune itself. The control of Dune shifts multiple times throughout the story, sometimes to the Atreides or Harkonnen, other times, the Fremen, and still others, no one at all. I might be able to capture this change in ownership if entities were made able to start and stop actions. Instead of having a notion of a current state, I might attempt to count how many times an action has been started and stopped, comparing the difference to determine whether the particular action had stopped or started. Yet, it would be more accurate (and faster) to simply check the newest instance of either of the pair.

This pending issue of redundancy might be solved by constructing a more refined sense of Chronology. At the moment, redundant events in a story are throttled to prevent rules from over-firing. This throttling is particularly useful when confronted with rules that may loop infinitely if given the chance (e.g., XX is YY's brother because YY is XX's brother). Unfortunately, full redundancy prevention stops *all* event repetition, including repeated instances of harm or mental states. When a character can become happy or unhappy at most once each every story, this is greatly disruptive to the overall understanding of a story.

V.5.2 Implementation

How might Chronology be introduced to Genesis? I propose the inclusion of local time-stamps in story frames as a story is processed. Assume that this time-stamp begins at $t = 10$ for the first line of the story, and I affix a time-stamp t to the first event of the story. When new events are automatically inferred from some firing event, these inferred events can be assumed to occur at the same time or very soon after the original. Let us then

assign these same events a time-stamp *t*. Such time stamps might well be stored alongside the already ubiquitous ID numbers coupled with all story events and actions.

Current redundancy measures should handle infinite looping with little alteration now so long as the time-stamp of an event is taken into account when checking redundancy. Theoretically, this would prevent an infinite loop generated in a single processing step from occurring, yet allow multiple harms and mental states to be generated at different times throughout the story as appropriate. Further, event time signatures are functionally non-intrusive; current processes will not need to be specially altered to perform their current operations.

Patrick Winston has recently developed a system of milestones that may provide some relief for this overzealous redundancy correction. Recognized by the lead-in "Then, ...", a milestone generates a separating border within a story, signifying a separation in time between groups of events. To the future 'right' of the separation, events are assumed to carry over from the past 'left,' but can be overruled or repeated on the 'right' side. Because most redundant events tend to appear in groups local in space and time, milestones can still prevent these redundancies while allowing identical events that are further apart in time to coexist coherently.

V.6 Grouping Rules by Parts

As Genesis evolves and more stories are added to its corpus, the total sum of available rules will grow to astronomical size. While one of the major goals of this thesis is to explore ways that many rules might be expressed by singular concepts, the range of human action is too great to expect a *complete* rule set to be anything but colossal. Theoretically, such size changes nothing about how stories are understood. Computationally, however, searching and re-searching over such a body at the introduction of any and all events is potentially crippling in time.

To combat the inevitable issues that a growing rule set will generate, it has been suggested that minor sets of rules be collected into *classes* of rules sharing relevant

common characteristics. Forming classes based on common antecedent or consequent could potentially smooth the process of forward or backward induction, respectively. Classes of rules that form predictive chains with one another enable an event to fire an entire series of predictions in one instant, rather than searching through a complete rule set *m* times to produce a chain of length *m*.

However, these class constructs are complicated in definition and ultimately unnecessary. A far simpler device will serve equal purpose with better efficiency: *Rule Hashing*. Suppose Genesis did not maintain its rules in a simple list, but instead stored rules in a hash map, each rule keyed by its antecedent parts. With this modified method of rule storage, the Genesis story processor can simply look up the set of rules whose keys match an incoming event, eliminating the need for a cumbersome linear search and removing the problem of having to define specific rule classes.

Because a given event can match many sorts of antecedents (e.g. `Hercules fights lions` matches both `man fights animals` and `XX fights YY`), a proper matching system must be designed cleverly. In a linear search, an incoming event is tentatively compared to the antecedents of every rule for a match at any level. A hash map, however, requires a very specific idea of what antecedent 'key' is needed, preventing the necessary deep matching. Therefore, a map-based matching system must preemptively generate all possible forms of a new event and query the rule map using these forms simultaneously.

While such a process may appear computationally expensive at the surface, one must recognize that a given event's alternate forms are entirely made up of permutations of each of the event's elements' threads. Mathematically speaking, an event composed of n elements, each defined by a thread of length t_n, possesses $t_1 * t_2 * \ldots * t_{n-1} * t_{n-2}$ total alternate forms. Given the generally short length of most events and the relatively shallow depth of most threads, I suspect this value is likely at least an order of magnitude less than the computational cost of a linear search over a complete rule set.

The expense of antecedent search may be further lessened by an implementation of Charles Forgy's RETE algorithm (Forgy, 1982), which maintains something like an active shortlist of already-witnessed antecedent events. Such a list potentially reduces the time spent searching for co-active antecedents when a multi-antecedent rule is first triggered by dramatically shortening search times for recently fired events. A discussion of such an implementation and its exact effects, however, are outside the scope of this thesis.

VI. Self-Reflection

> "Our ability to treat ideas as though they were objects goes together with our abilities to reuse our brain-machinery over and over again... once we replace a larger structure by representing it with a compact symbol-sign... we can build grand structures of ideas – much as we build great towers from smaller parts."
>
> - Marvin Minsky, *The Society of Mind*

In this chapter, I describe a mechanism for creating rules about rules, a system of higher-order thinking that I call the *Self-Reflection Engine*. By way of demonstration, I trace through the evolution of a self-reflective rule and its impact on the composition of a short story.

VI.1. Motivation

Simple story rules are patterns for events, capable of generating potentially infinite new connections and implicit detail, depending only on their antecedents. In the same way, *metarules* condense many rules into a single higher-level pattern, expressing many ideas as one more powerful idea. Cognitive science teaches that recognizing such patterns and constructing the rules to use them is a major part of human learning. Introducing self-reflective rules to Genesis takes the system one step closer to this ideal, while simultaneously reducing redundancy in the rule base and improving story coherence.

Consider the following rules and story:

```
XX harms YY because XX kicks YY.
XX dislikes YY because YY shoves XX.
XX may harm YY because XX dislikes YY.

Cain is a person.
Abel is a person.
Cain shoves Abel.
Abel kicks Cain.
```

Fig. 20. Trouble Brewing. A short story self-reflection may improve.

It seems clear that Cain's `shoving` will fire Abel's `dislike`. Similarly, Abel's `kicking` should fire Cain's `harm`. Would the weak rule fire? Where would the weak rule connection be drawn? Remember that weak rules draw connections between antecedents and consequents only after both have appeared. In this case, Abel's `dislike` and Cain's

`harm` have both appeared, so one might expect the resulting story to be visualized with a weak chain as in the following story graph:

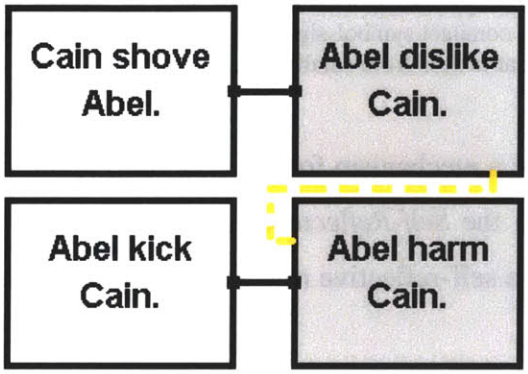


Fig. 21. Explanation of a Prediction.

But this interpretation of the story seems to have missed a critical distinction. Abel, like any actor, cannot simply `harm` other people. Rather, actors perform harmful *actions*, actions which *result* in harm. A more coherent story might state that Abel's `dislike` leads to Abel's `kicking`, with the full knowledge that the `kicking` leads directly to Cain's `harm`. Indeed, without this connection, a naïve interpreter may suggest that the `kicking` had no motivation whatsoever! Fortunately, this apparent error in composition may be quickly patched with a new rule:

```
XX may kick YY because XX dislikes YY.
```

With such a rule in place, the story can now be visualized as follows:

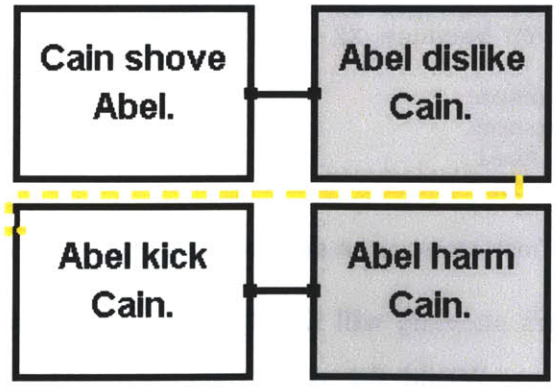


Fig. 22. Explanation of a Prediction's Antecedent.

But is adding a new rule the proper course of action? If this method of correction continues, every possible variety of harmful action will require a new rule to ensure a correct interpretation. Furthermore, such rules would only handle harmful actions motivated by a `dislike` particularly. Yet another rule would be required to handle other motivating factors, such as anger. As the active rule base expands to include more methods and more motivations, simply adding new rules swiftly becomes redundant, prone to error, and intractable in scale:

```
XX harms YY because XX kicks YY.
XX harms YY because XX shoves YY.
XX harms YY because XX shoots YY.
XX harms YY because XX insults YY.
XX harms YY because XX punches YY.
XX harms YY because XX fires YY from a comically large cannon.

XX may harm YY because XX dislikes YY.
XX may kick YY because XX dislikes YY.
XX may shove YY because XX dislikes YY.
XX may shoot YY because XX dislikes YY.
XX may insult YY because XX dislikes YY.
XX may punch YY because XX dislikes YY.
XX may fire YY from a device because XX dislikes YY.

XX may harm YY because YY angers XX.
XX may kick YY because YY angers XX.
XX may shove YY because YY angers XX. ...
```

Fig. 23. Too many rules for 'harm' alone.

In the rules listed in Figure 23, I have been forced to supplement `XX may harm YY because XX dislikes YY` with a multitude of different ways that that harm might be achieved. Each of these methods requires a rule explaining the connection between each of these "harmful" actions and the harm that they cause. Such rule lists explode in size as new harmful methods or motivations are introduced. Further, there is a risk of losing information by needing to write new methods in (at least) three different locations. I posit that a self-reflective rule can solve all these problems simultaneously by combining parts of an existing rule set to generate new rules automatically.

VI.2. Explaining Predictors: A New Idea

Think back to the short list of rules from the Trouble Brewing story from Figure 20. Without a rule connecting `kicking` with `dislike`, the story is doomed to exclude critical elements from its central plotline. Is it possible that such a connecting rule might be inferred from the original rule-set's parts? Consider a symbolic mapping of those original rules and the desired 'fix' rule:

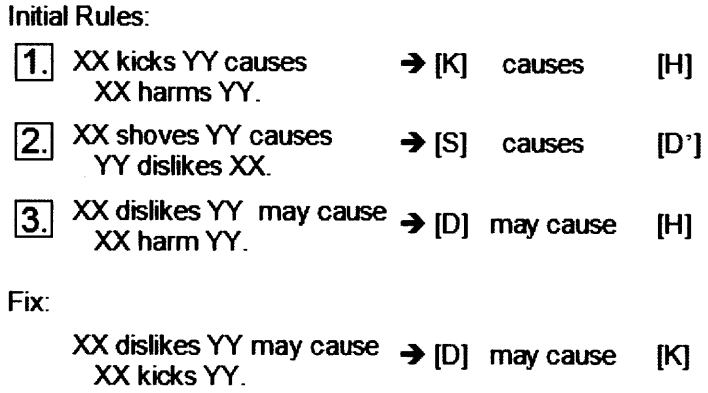


Fig. 24. A symbolic rule mapping.

Almost cyclically, the first and third rules share common symbolic language with the proposed 'fix' rule. That is, the two initial rules are relatable by their consequents ("H"), and their remaining parts comprise the elements of the needed 'fix' rule. I posit that these two rules can, coherently and justifiably, combine to create the intended rule. Further, I assert that this form of rule combination is appropriate for any set of rules in a similar pattern. Let a proposed metarule, which I will call *Explaining Predictors*, operate as follows:

```
Given the following rules:
    [A] may explain [C]
    [B] predicts [C]

Generate the new rule:
    [A] may explain [B]
```

Fig. 25. Representation of the *Explaining Predictors* metarule.

Recall that, in Genesis, strong "predicts" rules reflect an absolute and inexorable connection between two events, in this case meaning that the occurrence of *B*, without question, causes *C* to occur as well. This metarule, then, asserts that any event *A* which

can explain *C* necessarily explains *B* as well, for *B* is absolutely known to be causally inseparable from *C* itself. I shall discuss this metarule's merits in deeper detail in section VI.4 below.

With the *Explaining Predictors* metarule in place, the rule set first introduced in Figure 23 is suddenly and conveniently transformed, without loss of function, to a mere listing of only harmful actions and harmful motivators, its excess rules now automatically generated and no longer requiring manual introduction:

```
XX harms YY because XX kicks YY.
XX harms YY because XX shoves YY.
XX harms YY because XX shoots YY.
XX harms YY because XX insults YY.
XX harms YY because XX punches YY.
XX harms YY because XX fires YY from a comically large cannon.

XX may harm YY because XX dislikes YY.
   XX may kick YY because XX dislikes YY.
   XX may shove YY because XX dislikes YY.
   XX may shoot YY because XX dislikes YY.
   XX may insult YY because XX dislikes YY.
   XX may punch YY because XX dislikes YY.
   XX may fire YY from a device because XX dislikes YY.


XX may harm YY because YY angers XX.
   XX may kick YY because YY angers XX.
   XX may shove YY because YY angers XX...
```

Fig. 26. A minimized rule set. Rules in green are automatically generated by the *Explaining Predictors* metarule.

Using a single metarule, the rule set dictating 'harm' dramatically decreases in size, producing a compact list containing only rules possessing individual cognitive relevance.

VI.3. Implementation

VI.3.1 The Metarule Object

As the previous theoretical discussion has shown, metarules can be potentially expressed through a simple symbolic language, identifying mere patterns of antecedents, consequents, and rule types. Following this theory, metarules are efficiently created in Genesis using these same symbolic principles. In this section, I will describe how new metarules are created, processed, and understood.

Continuing with the overall running example, the *Explaining Predictors* metarule is declared in Genesis as follows:

```
ArrayList<Metarule> metarules = new ArrayList<Metarule>();

// Add metarules to search for here...
// Metarule -- Explaining Predictors
metarules.add(new Metarule(new MetaruleComponent(RuleType.EXPLANATION, "A", "C"),
                           new MetaruleComponent(RuleType.PREDICTION, "B", "C"),
                           new MetaruleComponent(RuleType.EXPLANATION, "A", "B")));
```

Fig. 27. Declaration of *Explaining Predictors* metarule in Genesis.

Metarule objects operate using an open constructor, taking some *n* (at least two) *MetaruleComponent* arguments, representing story rules. Each of these components has a rule type and a pair of symbol strings standing for the antecedent and consequent events of that rule, respectively. In the Metarule object itself, the first *n-1* component objects are cast as the metarule's antecedents, the final component becoming the metarule's consequent. Remember that each symbol string (e.g. "A", "B", or "C") acts as a variable capable of standing for any event or event phrase used in a simple story rule. In the case of the Cain/Abel conflict, "A" may be assigned to represent the event `Abel kicks Cain` or `Cain shoves Abel`. Which events are assigned to which symbols is determined by a search algorithm triggered every time a new rule is introduced to a story.

VI.3.2 Rule-Matching through Bindings

When Genesis is asked to 'read' a new story, each new rule or event of the story is parsed, converted into a rich system-readable format, and added to an active list of operant rules and events comprising Genesis' concept of the story. For each event that Genesis reads, that event is matched against the antecedents of all available operant rules to see whether it may fulfills those rules and cause new events to materialize.

The Self-Reflection Engine functions similarly. Whenever a new *rule* is read, that rule is matched against the antecedents of all existing metarules to determine whether any metarules will fire and cause a new rule to appear. However, because a metarule's antecedents and consequents are defined symbolically and essentially serve the same function as labels, such a matching process becomes more difficult. While the story rule

antecedent "XX harms YY" possesses a specific semantic meaning and describes an event, the metarule antecedent "A" is comparatively meaningless on its own and can match any event. Instead, metarule antecedents and consequents achieve meaning through their relative definition. That is, when "A" is used in *multiple* antecedents it is constrained by the need to match events in multiple rules simultaneously.

Metarule antecedent matching, then, is accomplished by using a persistent list of symbolic bindings. Recall the Cain and Abel story from Figure 20. The metarule matcher will find nothing useful about the first two rules, so I skip their analysis here. However, when the third rule, XX may harm YY because XX dislikes YY, is read, the matcher searches through its available metarules for an antecedent of the *Explanation* type that can bind XX dislikes YY to its antecedent symbol and XX harms YY to its consequent symbol. A match in the *Explaining Predictors* metarule is quickly found, instigating a search of the metarule's remaining antecedents with the following bindings:

```
Metarule Bindings:
      [A]: XX dislikes YY
      [C]: XX harms YY
```

Fig. 28. Initial bindings in matching a metarule.

In order to fulfill the *Explaining Predictors* metarule, all of its antecedents must be properly matched to existing rules. After the [A] may explain [C] rule has thus been discovered, the matcher checks whether any existing rules potentially match the remaining antecedent, [B] predicts [C]. Because "B" has not yet been defined in the active bindings list, it may yet match any event. However, "C" has now been constrained in meaning, and must match with XX harms YY. Despite the constraint, another rule is discovered as a match: XX harms YY because XX kicks YY. Having fulfilled all of the candidate metarule's antecedents, the matcher proceeds to generate a new consequent rule based on its final bindings:

```
Metarule Bindings:
      [A]: XX dislikes YY
      [B]: XX kicks YY
      [C]: XX harms YY
```

Fig. 29. Final bindings in matching a metarule.

As the consequent of the *Explaining Predictors* metarule effectively reads `[A] may explain [B]`, the matcher correspondingly generates a new rule: `XX may kick YY because XX dislikes YY`, precisely the rule needed to ensure a coherent and inclusive plot line in the original story.

Note that this new rule is resubmitted to Genesis' story processor and is itself used in a search for matching metarules. In this manner, a metarule such as *Explaining Predictors* may generate an entire series of new rules stemming from a long string of prediction rules and an explanation aimed at its base.

VI.3.3 Deep Merging and the Matching Process

There lies a certain danger in the matching process when one event subsumes another in scope. Although such a pair of events may not match one another identically, there is some level at which the two events overlap and share common meaning. For as a square is a kind of rhombus, rules about squares will apply to both shapes, yet rules about rhombi may not. In the language of stories, `Hercules fights lions` is a kind of `XX fights YY`. Rules about `XX fights YY` will apply when `Hercules fights lions`, but the same cannot necessarily be said of the reverse.

During the matching process, when a candidate event is compared to an existing binding, the matching system attempts to capture, in a single concept, all those events which may fire both: the common denominator between the incoming and the incumbent events. Determining whether such a denominator exists is a simple process that I call *Deep Merging*. The merging process steps through the two candidate events, comparing the active threads of parallel words. If every such pair of threads demonstrates an *is-a* relationship (e.g. a *penguin* is a *bird*, a *tiger* is not) and the two events maintain an equivalent structure, then a conceptual overlap for the two events exists.

Consider a deep merge of the following rules, here labeled for easy reference:

```
Police arrest XX causes Police restrain XX
        [A]      causes          [R]
George fights YY may cause YY restrict George
        [F]      may cause       [R']
```

Fig. 30. A pair of merge-able rules.

In attempting to fulfill the *Explaining Predictors* metarule, the R' event will be bound to the symbol "C". Subsequently, the R event will be compared to the contents of the "C" binding. If the two events R and R' are found to be potential matches, the *Explaining Predictors* metarule will be fired and a new rule will be generated. Being that restrain is a type of restrict, Police is a type of YY (entity) and George is a type of XX (entity), the two are found to match. However, the value bound to "C" must reflect only those events that strictly match both events; it cannot remain YY restrict George. Instead, "C" is bound to the deep-merged denominator of the R and R' events, constructed by taking the *most specific* of each thread pair from the original event comparison. In this example, the binding will ultimately read Police restrain George, describing all events that are the matching 'children' of both R and R'. By always replacing a binding with such denominators, metarules are guaranteed to use the maximal event space that safely describes every contributing rule.

VI.4. Examining the Explaining Predictors Metarule: Does It Work?

Self-reflective rules are powerful ideas. When working properly, a metarule must produce those rules which are intended, but it must also avoid the creation of unwanted rules, unintentional combinations of antecedent and symbolic value that suggest story connections that simply should not be. Anyone seeking to introduce metarules to a story understanding system must take great care that those metarules are correct, lest their stories become saturated with false rules.

I have shown that the *Explaining Predictors* metarule can produce the rules required to understand actions by virtue of those actions' consequences. But could this metarule produce incorrect rules? In this section, I analyze and ultimately dismantle two proposed counter-examples to the *Explaining Predictors* metarule. In the process, I will also

recognize and discuss a handful of ways in which story construction must be clarified and refined to avoid such pitfalls.

VI.4.1 The Rasputin Case

The Rasputin Case challenges *Explaining Predictors* in the case where two actions may be performed with identical intended consequents.

Consider this scenario: an Agent, harboring a dislike for Rasputin, wants to kill him. At first, the agent poisons Rasputin, but it fails to kill him. Instead, the Agent shoots Rasputin, killing him instantly. Such a scenario seems quite straightforward, but the *Explaining Predictors* metarule may discover a connection that ought not to exist.

The rules and plot describing this story could be written as follows:

```
Motives
    XX wants to kill YY because XX dislikes YY.
    XX may kill YY because XX wants to kill YY.

Methods
    XX may kill YY because XX poisons YY.
                    [For poison is a fickle thing.]
    XX kills YY because XX shoots YY.

Story
    Agent is a person.  Rasputin is a person.
    Agent dislikes Rasputin.
    Agent poisons Rasputin.
    Agent shoots Rasputin.
```

Fig. 31. The story of Rasputin.

Unfortunately, it seems that *Explaining Predictors* will generate an apparently false rule, linking the act of poisoning someone with that of shooting the same person.

```
Methods
       XX may kill YY because XX poisons YY.
       XX kills YY because XX shoots YY.

   Meta-Rule
     Given:
       [A]--may-->[C]
       [B]------->[C]

     Generate:
       [A]--may-->[B]
           or
       XX may shoot YY because XX poisons YY.
```

Fig. 32. *Explaining Predictors* generates a strange rule.

With the introduction of this new strange rule, the story will be visualized as follows:

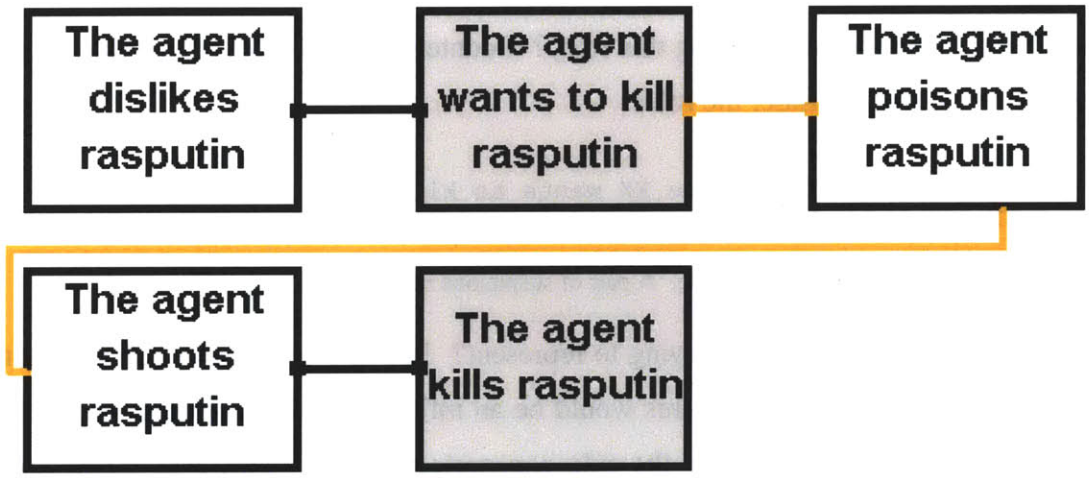


Fig. 33. A strange explanation of Rasputin.

While the flow of this particular plot may appear accurate in that these events happened in chronological order, this is a flawed way of representing the plot. Although the poison's failure motivated our `Agent` to seek another means for murder, the `shooting` itself ought not to derive its motivation from the `poisoning`. At the least, such a link must include an event and rules describing how the poison has failed, but such information is absent in this story. Good sense dictates that a proper representation of this story would show the poisoning and shooting to be forked branches stemming from the Agent's `wanting to kill Rasputin`, that both actions had equivalent motivation.

Ideally, then, the story ought to be graphed as illustrated in Figure 34.

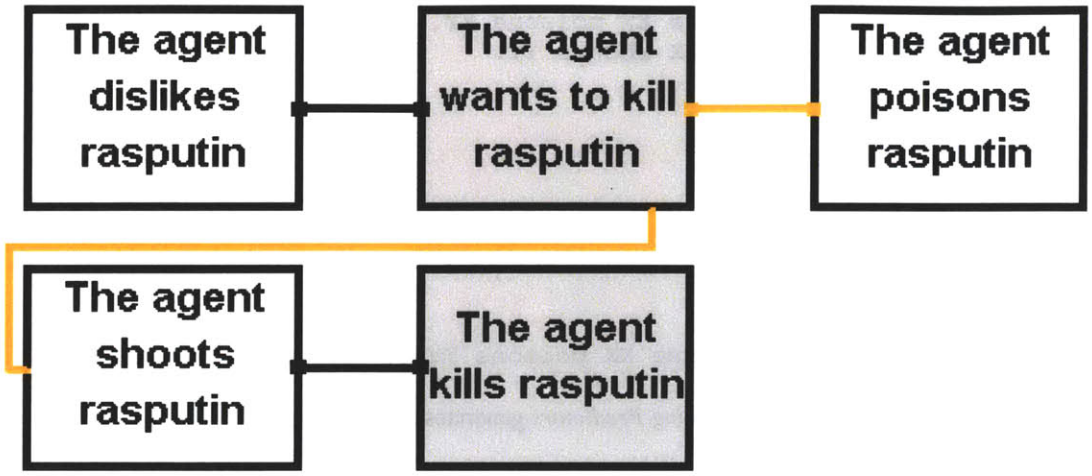


Fig. 34. The ideal plotline for Rasputin.

So what went wrong in constructing this story? I contend that the root of the issue lies in the choice of words. Consider the rules:

```
XX may kill YY because XX wants to kill YY.
XX may kill YY because XX poisons YY.
```

Fig. 35. A pair of suspicious rules.

What coherencies are these rules trying to represent? Because some actor XX wishes to kill some YY, it makes sense that this would be an influencing factor should XX indeed later kill YY. However, consider the rule concerning poison. This rule seems quite different from the rule stemming from a desire to kill. The poison rule is capturing the notion that poison could or could not lead to death, rather than XX's motivations in killing YY. Indeed, the poison rule seems more like it codifies a probability than a motivation.

I propose that a distinction can be made between these rules by altering their language:

```
XX may kill YY because XX wants to kill YY.
XX might kill YY because XX poisons YY.
```

Fig. 36. "Might" makes right: a distinction in language.

I define "might" to be a qualifying statement similar to "may," indicating a *weak* rule, or *explanation*. "Might" signifies a notion of probability, rather than motivation.

A vengeful ghost *may* haunt you; a jilted lover *may* ruin your date; a timid child *may* cry.

Poison *might* kill you; an earthquake *might* damage your house; black clouds *might* rain.

Must I explicitly assign defined probabilities to these events? Of course not. Such a course of action would be both presumptuous and unnecessary. Just as with *may* rules, *might* rules operate by awaiting the fulfillment of a given antecedent before firing. The difference between rules lies in what sort of uncertainties they address: probabilities or motivations.

Summarily,

May rules define **choice**.

Might rules define **chance**.

VI.4.2 The Arranged Marriage

This case was first presented in challenge by Karen Sittig.

The Arranged Marriage presents a problem where a generated weak connection seems ill-justified given other possible motivators within the story.

Consider the following example rules and story:

```
Rules
    XX loves YY because XX kisses YY.
    XX may marry YY because XX loves YY.
    XX marries YY because XX has arranged-marriage with YY.

Story
    Apu is a person.  Manjula is a person.
    Apu kisses Manjula.
    Apu has arranged-marriage with Manjula.
```

Fig. 37. The story of the Arranged Marriage.

Explaining Predictors will recognize a pattern match amongst this story's rules, and generate the new rule `XX may have arranged-marriage with YY because XX loves YY`. With this new rule in effect, the story is visualized as pictured in Figure 38:

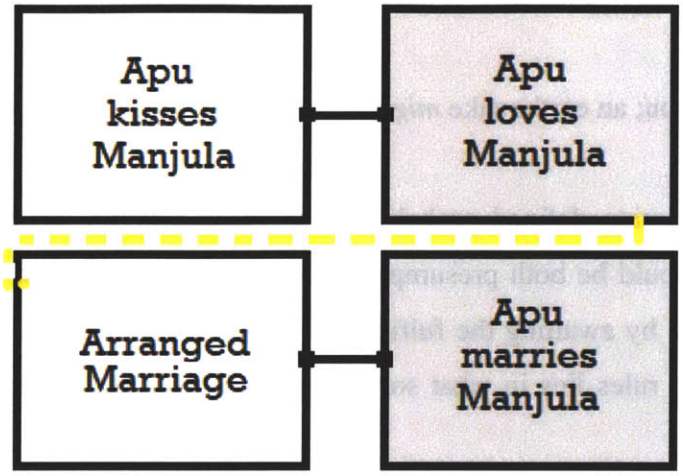


Fig. 38. The story of the Arranged Marriage visualized.

The cognitive disconnect here is that Apu's love should have no apparent bearing on the marriage, for an 'arranged marriage' is typically a marriage arranged by third parties - the relationship between the two being married is not an influencing factor in whether the marriage takes place.

I suggest that this encoding of the story is incorrect, failing for two reasons:

First, the concept of the arranged marriage carries with it a good deal of contextual information not included in this story's rules. Particularly, the story does not encode that a third party has compelled the actors to have a marriage. Without properly encoded rules, the story knows nothing about an arranged marriage's context. While the given rules include the knowledge that an arranged marriage is a kind of marriage (leads to XX marries YY), it is simply incorrect to exclude important contextual information for events and expect subsequent analysis to account for it.

Second, philosophical arguments aside, there is always a choice. Recall my discussion of *compulsion* rules earlier; compulsion does not generate events. Compulsion influences actors, generating motivations. Those motivations may, in turn, lead to events, but it is important to recognize that compulsion cannot absolutely force an action. Simply because the third party has negotiated the social construct of an arranged marriage, this

does not therefore mean that a wedding has taken place. Whether through obedience or love, Apu and Manjula must *choose* to marry.

As I have also established, plot webs are meant to assign as many valid, coherent explanations as possible when it is appropriate to generate any. That is, events that can be explained by multiple precursor events are, in part, explained by *all* those precursor events. The degree of these connections is irrelevant; it is enough that they exist. Even where the arrangement itself is the overwhelmingly motivating factor in a story, I consider it an error to drop love as a motivator altogether.

In summary, the given story is inappropriately condensed. Indeed, the actual wedding is included inside the Arranged Marriage block, with the following Apu marries Manjula statement merely serving to reiterate the fact of their marriage.

An appropriate re-write of the story might be illustrated as follows:

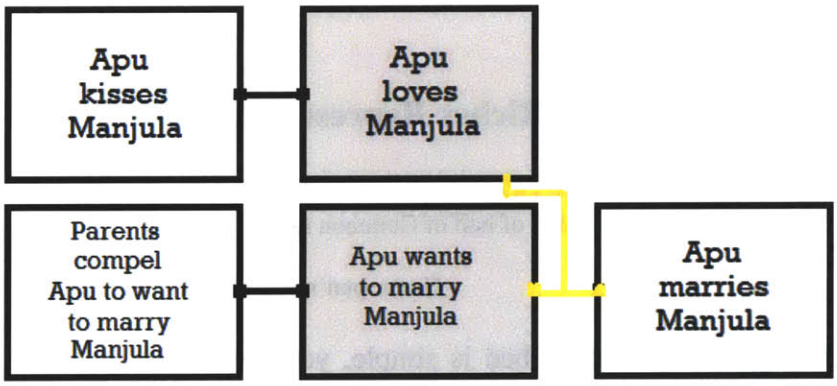


Fig. 39. Visualization of the rewritten Arranged Marriage.

One can see how this verbose version of the story in Figure 39 compares to the earlier condensed edition in Figure 38. By breaking the Arranged Marriage event into its component parts and explicitly including its assumed context, the Arranged Marriage into is separated into the compulsion of the third party ('parents') and the event of the wedding itself. In Figure 38, the statement Apu marries Manjula was made to explain that an Arranged Marriage is a type of marriage. With a verbose separation in place, this qualifying statement becomes unnecessary. When the two versions of the story are overlaid with one another, as shown in Figure 40, a curious structure appears.

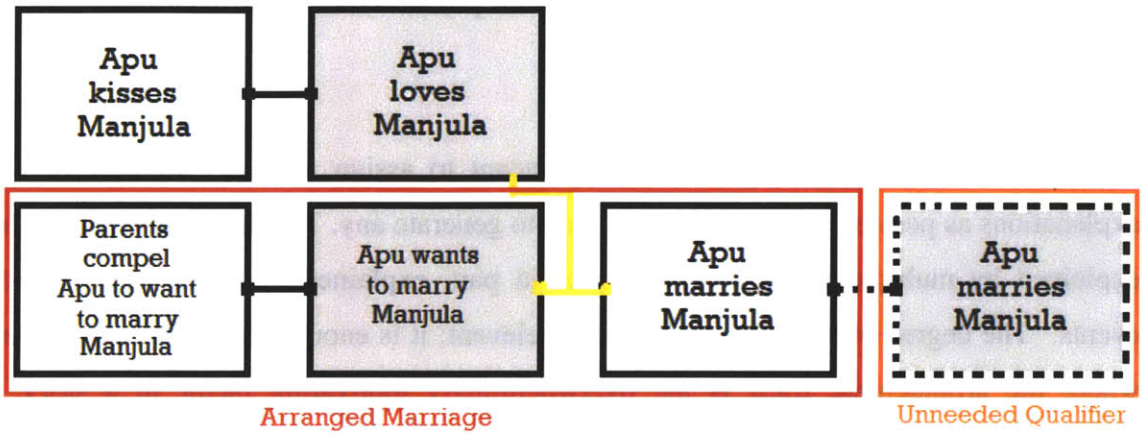


Fig. 40. Overlay of verbose and condensed versions of the Arranged Marriage.

Notice that the verbose version explicitly includes the actual wedding event and no longer needs a qualifying statement to underscore its presence within the Arranged Marriage. Intriguingly, the condensed version no longer even seems wrong; the couple's `love` event leads to the marriage *contained within* the Arranged Marriage itself. Because the original version compresses several contextual events into one, the proper motivators are forced to refer to that compressed event to establish their motivational connection.

VI.5. Component Formatting: A Richer Representation

> "Any sufficiently complicated C or FORTRAN program contains an ad-hoc, informally-specified bug-ridden slow implementation of half of Common Lisp."
>
> -Greenspun's 10th Rule of Programming

The Self-Reflection Engine I have described is simple, yet powerful. However, in its current form, this same simplicity surrenders the capacity for constructing deeper, more refined metarules. By collapsing entire antecedents and consequents into singular compact symbols, all contextual information *about* those events is lost.

The *Explaining Predictors* metarule operates well when describing events with a specific agent. That is, the metarule describes events in which some actor performs some action, becomes some state, or otherwise *does* something. However, when rules describe events without tracking their causal agents, the metarule can sometimes begins to break down. By way of example, consider the small set of rules listed in Figure 41.

```
Leader becomes angry because Follower becomes defiant.
Leader may become angry because Baby cries.
```

Fig. 41. Rules without strong agent connections.

Explaining Predictors will match these two rules and produce the nonsensical `Follower may become defiant because Baby cries`. Although this sort of configuration may seem common, one must realize that the established convention for rules in Genesis generally requires a strong agent presence. That is, the consequents of the rules in Figure 41 are preferably written in the alternative form `Follower angers Leader`, preserving the `Follower`'s role as the direct agent of the action. In rules concerned with murder, the consequent reads `XX kills YY`, not `YY dies`. (Note that although the consequent `YY becomes dead` is used in one rule, this is a particular state rule not used as a consequent in other rules.)

However, precedent and convention are by no means absolute. While the style may persist in the short run, such a constraint is unnecessary for the future and will likely limit further development. With this in mind, I propose that metarules may expand their symbolic language with a specialized formatting.

In *Explaining Predictors*, the component rule `[B] predicts [C]` assigns a strong rule's antecedent and consequent to "B" and "C", respectively, regardless of the content or structure of those events. As I have shown, this simplistic bindings mechanism can act in error when used on rules without strong agent cues. But suppose the symbolic language were extended to allow one to refer to such agency. Combining the syntax of Lisp with the semantic meaning of Genesis' frame structures, symbols could handily refer to multiple elements of an event with the same symbolic comparison mechanisms already used in Deep Merging.

An example will clarify the structure I envision. Let events be represented as parenthetical statements, their contents beginning with the inherent 'action' of the event,

followed by the various arguments of that action. (Typically, the agent of the action is listed first). Suppose that in place of `[B] predicts [C]`, the relevant component read:

```
[(B X Y)] predicts [(C X Z)]
```

If the bindings search algorithm is appropriately extended to handle such in-line symbolic language, then this line describes a strong rule that interprets roughly as "*X C*s *Z* because *X B*s *Y*". When the matching algorithm encounters this component, it will attempt to bind values to the *X*, *Y*, *Z*, *B*, and *C* symbols, but these values will be specific component elements rather than entire events. Pending prior bindings constraints, any strong prediction rule will match this component, but now with the constraint that *X*, whatever it may be, must be the acting agent of both events. This method of component formatting allows metarules like *Explaining Predictors* to specify constraints unavailable in the current simple design. Although as yet unimplemented, such formatting may dodge the problems of broken conventions while simultaneously enabling the construction of more specifiable, and therefore more powerful, metarules in the future.

VII. Contributions

In the course of this thesis, I have made the following contributions:

1. I have constructed, analyzed, and developed stories from multiple domains at a variety of levels of resolution, ranging from minor interpretations of Shakespearian works to intermediate conflicts in cyber-warfare to the story of *Dune*. Containing over a hundred events substantially generated implicitly by these rules, my analysis of *Dune* stands as a testament to the capabilities of story understanding in the face of complexity.

2. I have compiled a battery of rules describing relationships on the personal, group, and international level. These rules cover the space of international politics, valuation of resources, competition, persuasion, desire, and vendetta.

3. I have refined Genesis' story processing machinery in order to handle this very battery of rules and concepts, making both theoretical progress in the field of story understanding and mechanical progress in Genesis' inner operations.

4. I have introduced the Self-Reflection Engine, a means of producing and understanding metarules. By continuing the abstraction of concepts as objects, I have made the manipulation of rules and ideas simpler and enabled the expression of many thoughts in the space of a single common pattern.

5. I have presented and thoroughly discussed an example metarule, shown to be both operable and usefully powerful in the current Genesis environment. I have also introduced a method of extending the capability of such metarules, demonstrating how metarules need not stand on fragile precedent in order to be useful.

REFERENCES

Borchardt, Gary. 1994. *Thinking between the Lines, Artificial Intelligence series.* Cambridge MA: MIT Press.

Christie, Agatha. 1948. *Taken at the Flood.* New York, NY: Dodd, Mead, and Company.

Forgy, Charles L. 1982. "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, no. 1.

Herbert, Frank. 1965. *Dune.* Philadelphia, PA: Chilton Books.

Kaplan, Morton A. 1957. *System and process in international politics.* New York, NY: Wiley.

Lehnert, W. G. 1981. "Plot Units and Narrative Summarization," *Cognitive Science 4*, pp. 293-331

Miller, George A. 2007. *WordNet 3.0* [Website]. Princeton University 2006 [cited 5/24/2011]. Available from http://wordnet.princeton.edu/

Minsky, Marvin Lee. 1988. *The society of mind.* New York, NY: Simon and Schuster, Inc.

Vaina, L. and Greenblatt, R. 1979. *The use of thread memory in amnesic aphasia and concept learning.*

V for Vendetta. 2006. Screenplay by Andy Wachowski and Lana Wachowski. Dir. James McTeigue. Perf. Hugo Weaving and Natalie Portman. Warner Bros.

Winston, P.H. 2008. *Four world-reflecting representations and an implementation substrate* [Website]. MIT 2008 [cited 5/24/2011]. Available from http://groups.csail.mit.edu/genesis/frames.html