Text to Text: Plot Unit Searches Generated from English

by

David Douglas Nackoul

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 2010

[June 2010]

Author [illegible]
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by [illegible]
Patrick Winston, Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted by [illegible]
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Text to Text: Plot Unit Searches Generated from English

by

David Douglas Nackoul

Submitted to the
Department of Electrical Engineering and Computer Science
May 21, 2010
In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The story of Macbeth centers around revenge. World War I was started by an act of revenge. Even though these two stories are seemingly unrelated, humans use the same concept to draw meaning from them. Plot units, revenge included, are the common set of structures found in human narrative. They are the mistakes, the successes, the revenges and the Pyhrric victories. They are the basic building blocks of stories. In order to build a computational model of human intelligence, it is clear that we must understand how to process plot units. This thesis takes a step in that direction. It presents an English template for describing plot units and a system that is capable of turning these descriptions into plot-unit searches on stories. It currently processes 26 plot units, and finds 10 plot units spread out over Macbeth, Hamlet, the E-R Cyber Conflict, and a collection of legal case briefs.

Acknowledgements

This work is partially funded by the Office of Naval Research under award number N00014091059. Any opinions, findings, and conclusions or recommendations expressed in this thesis are those of the author and do not necessarily reflect the views of the Office of Naval Research.

Professor Patrick Winston, for serving not only as my supervisor, but also as my mentor.

My family, especially my mother, for their unwavering support throughout my life.

Contents

List of Figures

1. INTRODUCTION

1.1 My First (Painful) Experience with Plot Units

My very first memory in life is a painful one. When I was four years old, I was obsessed with the movie *Roger Rabbit.* Particularly, I was obsessed with how Roger, the animated rabbit and main character, could run into walls, become completely flat, and then "inflate" himself back to normal. Using perhaps typical four year old reasoning, I determined that I too could flatten myself if I ran into a wall hard enough. So, I ran as hard as I could and launched myself into the living room wall. I woke up in the emergency room with 24 stitches in the middle of my forehead.

I will never forget that day. I still have a giant scar on the middle of my forehead that serves as a constant reminder that running into walls is a bad idea. However, I also learned something else. In the emergency room I asked my mother how I got there. She replied "You ran into a wall, honey. You made a mistake. You're a person, not a cartoon." There was one word in her reply I did not understand: mistake. I asked her what a mistake was, and she explained that "a mistake is something you do that is wrong. It makes you sad instead of happy." A few weeks later, I was at the local park playing on the slides when the boy in front of me decided to go down the slide head first. He bumped his head on the mulch at the end of the slide and began to cry. I ran to my mother and told her, "Mommy that boy made a mistake! He's sad because he went down the slide the wrong way and hit his head!"

That statement was the beginning of my ability to recognize *plot units.* Plot units are the basic recurring themes in human narrative. Successes, mistakes, Pyrrhic victories, revenges, broken promises, double crosses; These are all essential plot units that shape the way that humans understand stories. Anyone who has read Macbeth knows that the plot revolves around revenge. Anyone who has studied world history knows that World War I was started by an act of revenge. Even though these two narratives are vastly different, humans still refer to the same basic plot unit when trying to make sense of their events.

To build a computational program that approaches human intelligence, it is clear that one must build a program that is capable of recognizing plot units. I was able to deduce, based on what my mother had told me, that the child who went down the slide head first made a mistake. Likewise, a program should be able to deduce, based on a textual description, whether or not a narrative contains a given plot unit.

Developing this kind of intelligence is the focus of this thesis. During the past year I have developed an extension to the Genesis system that is capable of recognizing plot units, experienced in stories, written in English. Much like my younger self, the extension generates all of its plot unit searches from English descriptions. However, before taking a deep dive into the details of computational plot unit searches, it is important to understand both the long-term and the immediate vision.

1.2 Understanding Cyber Policy, Shakespeare and Beyond

"Today's technology has made it possible for us to get our news and information from a growing range of sources. We can pick and choose not only our preferred type of media, but also our preferred perspective... that exposes us to an unprecedented array of opinions, analysis, and points of view"

-Barack Obama

This morning, I started my day by reading my email, checking CNN.com for the latest news stories, perusing Facebook to find out what is happening with my friends and then finally looking over some of my old thesis notes. After looking at my thesis notes, it dawned on me that I had been staring at plot units all day. I read about governments taking revenge on banks. I waded through wall posts where people trumpeted their successes and wallowed in their failures. I even made a mistake; I sent an email to someone that it should not have been sent to.

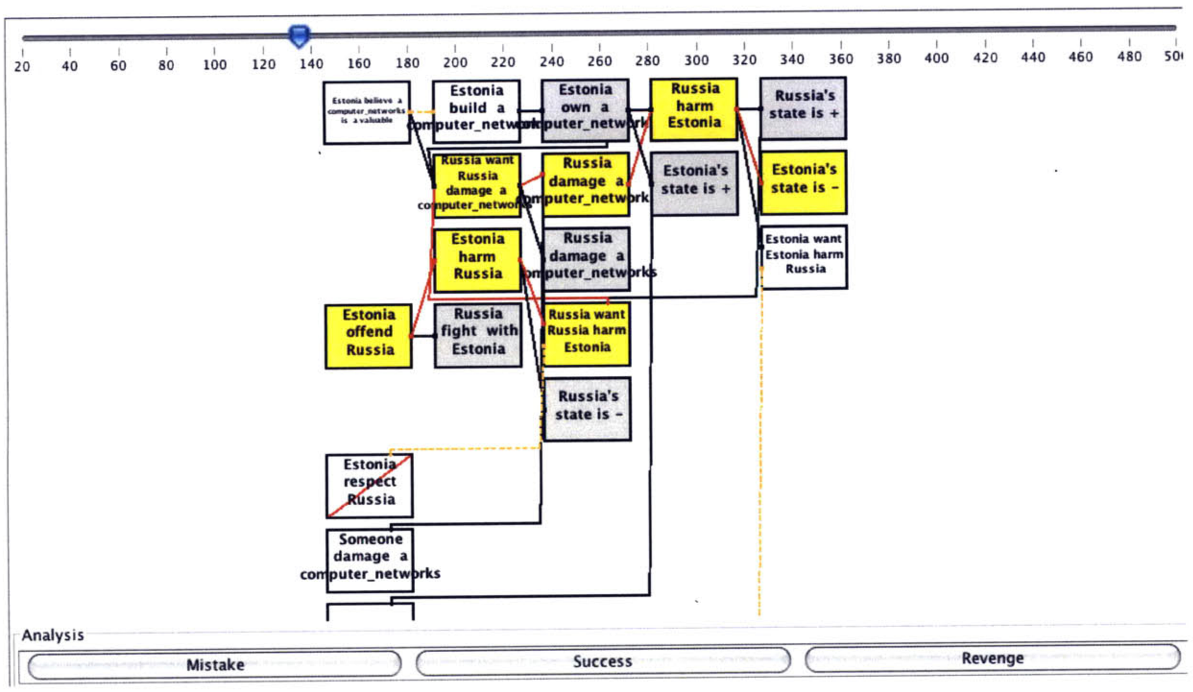
Given the speed and variety of current communication, it is not surprising that we are bombarded by stories and story like information. Whether it is a long news article or a collection of Twitter posts, we have to process many stories, and consequently many plot units, daily. Ideally, an intelligent system would help us make sense out of this mountain of data. In the medium and long term, the Genesis group is looking at several applications. Here are two of them.

1.2.1 ECIR and Assessing Cyber Relations

Over the past year, I have had the pleasure of discussing my work with members of the Explorations in Cyber International Relations, or ECIR, research program. Broadly, ECIR seeks to bring together experts in both Artificial Intelligence and Political Science in order to develop systems that can help understand cyber relations. That problem is very difficult and it makes sense to attack it from all fronts, one of which is to better understand the goals and intentions of other countries, and eventually practice law at cyber speeds, based on available evidence. This leads naturally to the need for rapid story understanding.

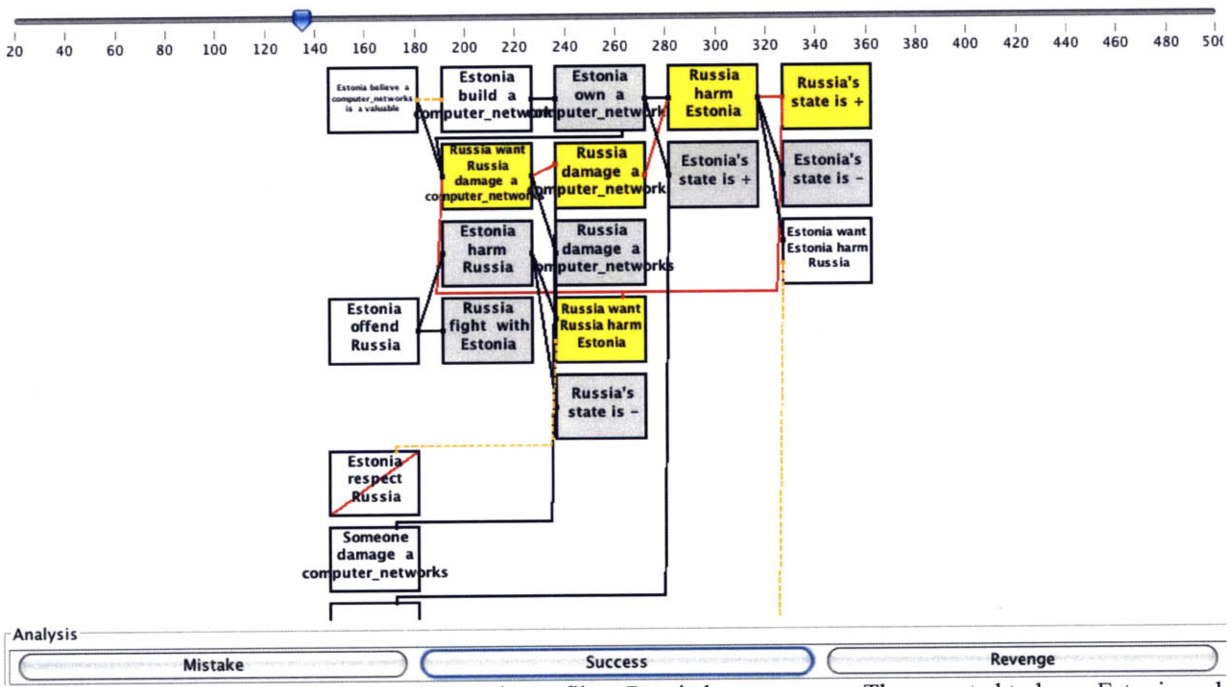
Much like any story, plot units are integral to understanding cyber relations cases. Plot unit analysis can be used to determine motivation, "Was Russia out for revenge?", to detect threats, "There might be an impending security breach.", and even to compare situations, "This situation looks like the Google-China debacle." Analysts only have so much time; computers can play a very important role in helping them manage the aforementioned barrage of data.

The Genesis group has already made some progress on understanding cyber relations between countries. The next page contains an analysis of a simplified version of a cyber conflict between Estonia and Russia. The analysis is presented in graphical form; a line from left to right means that the event on the left causes or explains the event on the right.



20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320 340 360 380 400 420 440 460 480 500
Estonia believe a computer_networks is a valuable
Estonia build a computer_networks
Estonia own a computer_networks
Russia harm Estonia
Russia's state is +
Russia want Russia damage a computer_networks
Russia damage a computer_networks
Estonia's state is +
Estonia's state is -
Estonia harm Russia
Russia damage a computer_networks
Estonia want Estonia harm Russia
Estonia offend Russia
Russia fight with Estonia
Russia want Russia harm Estonia
Russia's state is -
Estonia respect Russia
Someone damage a computer_networks
Analysis
Mistake
Success
Revenge

Figure 1: Genesis' analysis of the Estonia-Russia conflict. Estonia makes a mistake by offending Russia.



20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320 340 360 380 400 420 440 460 480 500
Estonia believe a computer_networks is a valuable
Estonia build a computer_networks
Estonia own a computer_networks
Russia harm Estonia
Russia's state is +
Russia want Russia damage a computer_networks
Russia damage a computer_networks
Estonia's state is +
Estonia's state is -
Estonia harm Russia
Russia damage a computer_networks
Estonia want Estonia harm Russia
Estonia offend Russia
Russia fight with Estonia
Russia want Russia harm Estonia
Russia's state is -
Estonia respect Russia
Someone damage a computer_networks
Analysis
Mistake
Success
Revenge

Figure 2: Genesis' analysis of the Estonia-Russia conflict. Russia has a success. They wanted to harm Estonia and they did.

1.2.2 Using Plot Units to Analyze Legal Cases

Whenever a judge or jury determines whether or not a defendant has committed a crime, they are looking to see if that defendant played a particular role in a plot unit. For example, "was Dave the thief who stole money from the bank?" Even though we may not realize it, crimes are essentially plot units. Murder, theft, arson, and fraud are recurring themes in narrative, just like mistake and double cross.

A plot unit analyzer would be able to give a quick answer to questions like "is this person guilty of arson?" or "is this third degree or second degree assault?" Of course, a judge or jury has the ultimate say. However, a plot unit system would be able to provide instant analysis over large amounts of cases.

The Genesis group has taken some initial steps towards this goal as well. Below is a graphical representation of a case brief. In the case, Francis, an immigrant, hired Graham, a lawyer, to fight his deportation out of the US. Graham informed Francis that he needed to pay a fine and offered to deliver the fine to the INS if Francis gave him the money. However, Graham claimed that the fine was greater than it actually was. He paid the INS and pocketed the rest. Genesis finds an instance of "larceny by deception" in the brief. Graham gained Francis' trust by helping him in a time of need and then stole money from him.

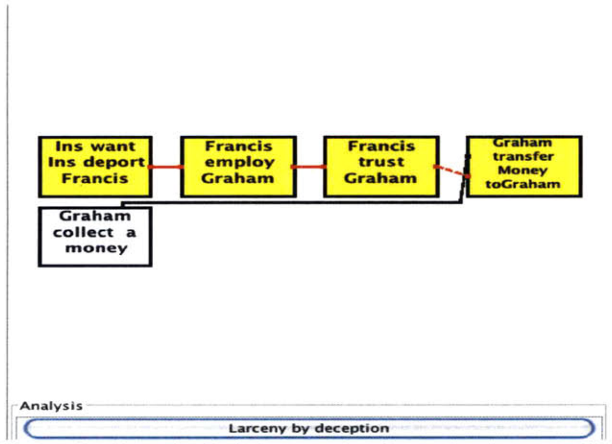


Figure 3: Genesis finds larceny by deception in a case brief. Stealing is defined as transferring money to yourself from another person. The from part does not fit in the box display.

1.2.3 The Larger Goal: Understand it All

While these two applications would prove immediately useful, the ultimate goal of the Genesis group is to develop a comprehensive view of human intelligence. Much like I can find plot units in places ranging from news articles to my Facebook wall, a truly intelligent system should be able to analyze stories regardless of type. So far, the plot unit analysis exhibited by my system has not proven to be limited by type. It finds units in Shakespeare, cyber conflicts, and legal briefs. This breadth suggests that the Genesis group is on the correct path with regards to story understanding. However, we recognize that this path is long and that we have only just begun to unravel human story processing.

2 Preparing to Process Plot Units

2.1 Wendy Lehnert Paves the Way

In this section I will explain the seminal work done by Wendy Lehnert on plot unit recognition. Dr. Lehnert's 's insights inspired the Genesis group's approach.

2.1.1 A Computational Structure

Even though humans talk about plot units often, it is not readily apparent what plot units are at a computational level. Stated another way, it is not immediately clear what internal representation a computer should use to represent a plot unit.

Fortunately, Wendy Lehnert provided an answer to this question in her 1984 thesis, *Plot Unit Recognition for Narratives* (Lehnert 1984). In her work, Lehnert describes a graph based representation for plot units: "A plot unit is a simple configuration built up of three primitive affect states... we need only distinguish generally positive reactions, negative reactions, and more neutral mental states associated with desires, goals, and plans." In her graph mental states are the nodes. The edges are word links that fall into one of four categories. A-links represent actualizing a goal, t-links represent terminating an object, e-links represent equivalence relations, and m-links represent motivations. Lehnert argues that these graphs can be used to represent basic plot units. For example, a success plot unit may consist of a person who has a mental state that is a goal; They have something they want to achieve. That mental state is linked to another positive mental state by an a-link. Together, these elements represent a success: A person has a goal and they achieve it (Lehnert 61).

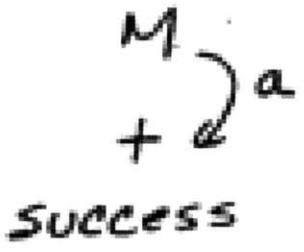


Figure 4: Lehnert's representation of success.

Lehnert then argues that these basic plot units can be combined to form even more complex plot units. For example, a success can be extended into a promise. A person's goal can be to help another person, and actualizing that goal can be viewed as fulfilling a promise (Lehnert 37).

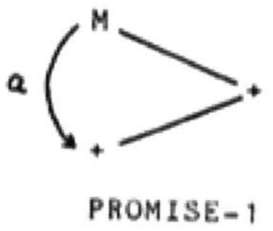


Figure 5: Lehnert's representation of a promise.

Near the end of her thesis, Lehnert compiles a list of over 50 distinct plot units of various complexity. These units range from primitives like success and mistake to complicated idioms such as “killing two birds with one stone.” It is clear that her representation is both efficient and generalizable. However, it does not completely solve the problem at hand. I needed an English based solution; a system that can take in an English description of a plot unit and an English description of a story and find instances of the unit found in the narrative. In order to accomplish that task, I worked within the framework of the Genesis system.

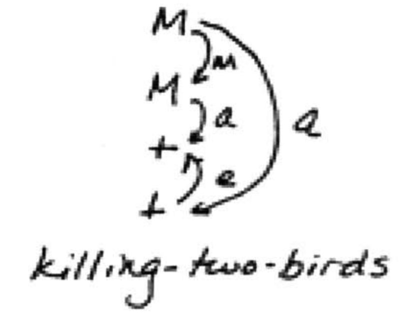


Figure 6: Lehnert's representation of killing two birds with one stone.

2.1.2 Why Lehnert's Representation Works

It was not apparent at the start of this thesis whether or not modifying Lehnert's approach would be the correct way to implement plot-unit searches in Genesis. However, after completing my work, I believe that Lehnert chose a great representation. There are two reasons I believe this.

First, mental states are pivotal to analyzing plot units. The difference between a success and a mistake is not necessarily in the actions someone took. Instead, the difference lies in how those actions *affected that person emotionally.* In fact, what might be a mistake to one person might be a success to another. Without accounting for mental states, any plot unit search program would be severely lacking. Mental states, whether they are positive, negative, motivational, or even delusional, are what drive people to act.

Second, a graph based structure is ideal for computational systems. Whenever humans read stories, they almost certainly do not build an explicit graph structure in their heads. On the other hand, computational systems need a representation that lends itself well to computer science techniques. There is hardly a more studied, researched, and taught structure than graphs.

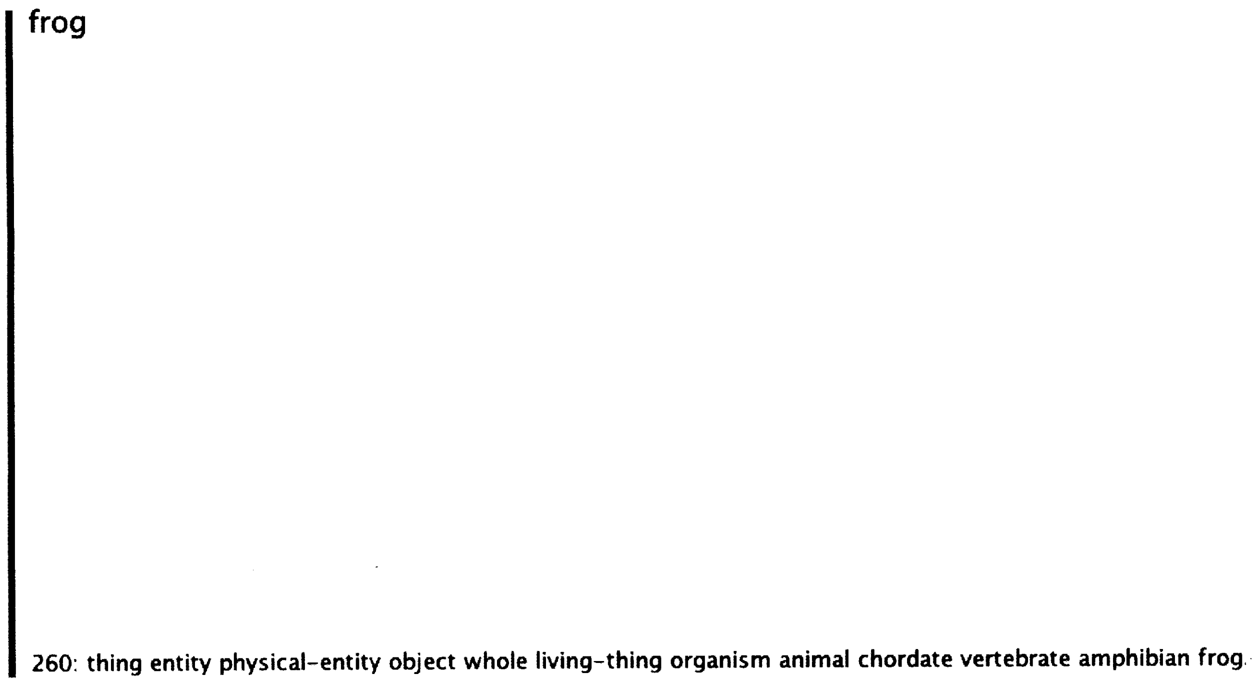
By transforming plot units into graphs, Lehnert created a representation that not only makes sense to humans, but also is easily manipulatable by computational systems.

2.2 Introducing Genesis

The Genesis group is currently developing a system that combines a good deal of research done in the field of Artificial Intelligence over the past 30 years. The system combines language, vision, and reasoning systems in an attempt to model some aspects of human intelligence. One of the language modules is a story processor that can read in typed English stories and make inferences about the events that occur, remember who the actors are, and even visualize certain aspects of the narrative. This module can also be used to form a structure that is very similar to Lehnert's plot units. This section will break down how Genesis transforms a story into the needed representation.

2.2.1 Parsing, WordNet and Threads

Whenever Genesis reads in a story, each sentence in the story is first analyzed syntactically using a parser. The parser marks each word in the sentence with the appropriate syntactical information. Then, WordNet (Miller 2006) is consulted in order to obtain categorical information for each word. This information is used to represent words as *threads* (Vaina and Greenblatt 1979). Threads are hierarchical memory representations of objects. An example of a thread for a frog is shown on the next page. The thread shows the type-hierarchy of a frog: it is an amphibian, which is a subtype of vertebrate, which is a subtype of chordate, which is a subtype of animal, and so on.


frog

260: thing entity physical-entity object whole living-thing organism animal chordate vertebrate amphibian frog

Figure 7: A screenshot of Genesis' thread memory for a frog. The bottom row represents frog's type hierarchy.

Each unique item in a story is represented as a thread. This includes concrete objects, such as frogs, as well as more abstract concepts, like harm. For plot-unit processing, threads hold a distinct advantage: They allow plot units to be encoded generally and found specifically. If revenge is defined as harm leading to harm, then a computational plot-unit program should be able to find revenges that consist of, say, a punching leading to a kicking. If punching and kicking are represented as threads, then this task is made easy. Without threads, many plot units would have to be encoded to represent the same basic concept.

2.2.2 Things, Derivatives, Relations, and Sequences

While threads are useful for representing a single object, Genesis needs a way to combine threads *in a meaningful way*. In order to accomplish this task, Genesis uses four different types of semantic objects: things, derivatives, relations, and sequences.

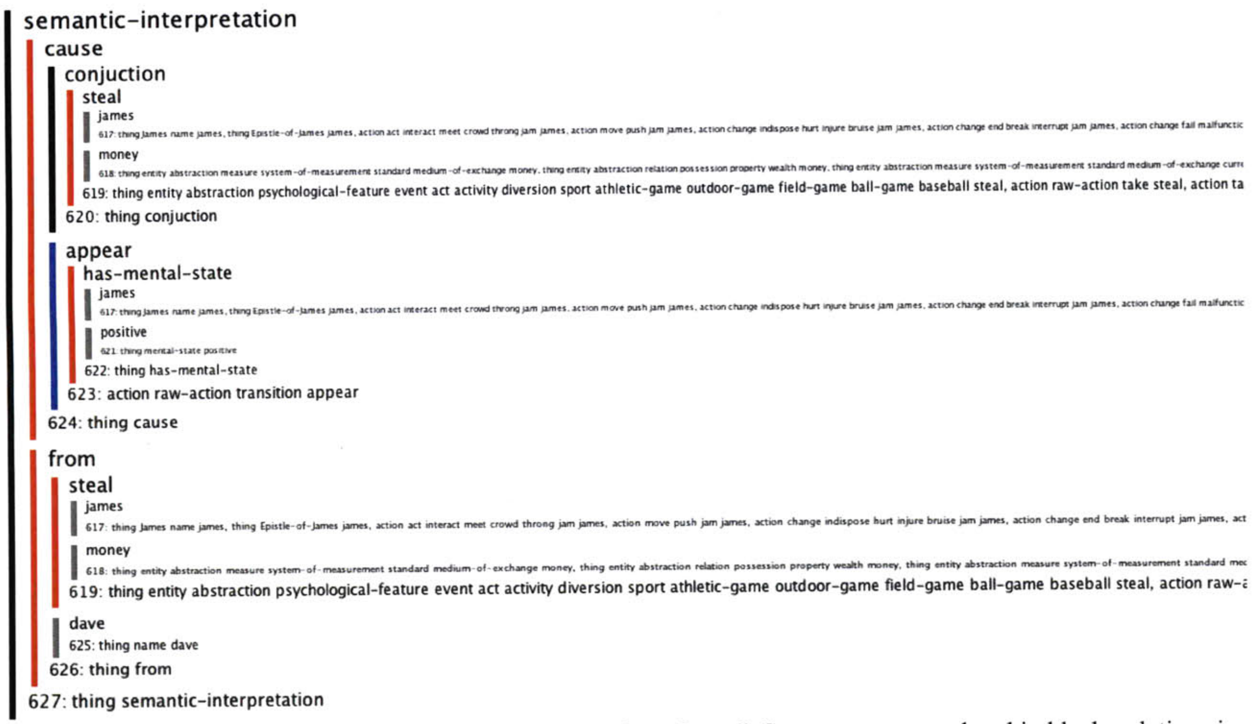
A thing is just a wrapper for an individual thread. This basic type of object stands alone and takes in no additional elements. An example of a thing is the concept "frog."

A derivative is a thing with a subject slot. For example, the concept "appear" is considered a derivative. Appear signifies that the attached subject appeared at some point in the story.

Relations are threads that represent relationships between subjects and objects. The concept "harm" is considered a relation. Whenever one person harms another, Genesis represents the harm as a relation where the subject is the person doing the harming and the object is the person being harmed.

Sequences are objects that group together multiple elements. An example is a path with multiple places. In this case, the path's elements would be the places, in order.

Subjects, objects, and sequence elements can be any of the four types. This allows Genesis to build recursive semantic representations of arbitrary complexity. The next page shows the semantic representation for "James stole money from Dave."



```
semantic-interpretation
  cause
    conjuction
      steal
        james
        617: thing James name james, thing Epistle-of-James james, action act interact meet crowd throng jam james, action move push jam james, action change indispose hurt injure bruise jam james, action change end break interrupt jam james, action change fail malfunctic
        money
        618: thing entity abstraction measure system-of-measurement standard medium-of-exchange money, thing entity abstraction relation possession property wealth money, thing entity abstraction measure system-of-measurement standard medium-of-exchange curre
      619: thing entity abstraction psychological-feature event act activity diversion sport athletic-game outdoor-game field-game ball-game baseball steal, action raw-action take steal, action ta
    620: thing conjuction
    appear
      has-mental-state
        james
        617: thing James name james, thing Epistle-of-James james, action act interact meet crowd throng jam james, action move push jam james, action change indispose hurt injure bruise jam james, action change end break interrupt jam james, action change fail malfunctic
        positive
        621: thing mental-state positive
      622: thing has-mental-state
    623: action raw-action transition appear
  624: thing cause
  from
    steal
      james
      617: thing James name james, thing Epistle-of-James james, action act interact meet crowd throng jam james, action move push jam james, action change indispose hurt injure bruise jam james, action change end break interrupt jam james, act
      money
      618: thing entity abstraction measure system-of-measurement standard medium-of-exchange money, thing entity abstraction relation possession property wealth money, thing entity abstraction measure system-of-measurement standard mec
    619: thing entity abstraction psychological-feature event act activity diversion sport athletic-game outdoor-game field-game ball-game baseball steal, action raw-a
    dave
    625: thing name dave
  626: thing from
627: thing semantic-interpretation
```

Figure 8: Genesis' representation of "James stole money from Dave." Sequences are rendered in black, relations in red, derivatives in blue, and things in grey.

2.2.3 The Story Processor

The story processor relates semantic objects by causality. Certain objects explain or predict other objects. For example, if I hit my roommate and he hits me back, one can say that he hit me because I hit him. In order to make these inferences, the processor relies on textually provided background knowledge (also known as commonsense). This mimics one way that humans attain commonsense knowledge. When you were younger, a parent probably informed you that you will be shocked if you stick a metal fork into an electrical socket. Likewise, in order to read a story involving silverware and power outlets, the processor needs to be fed certain background information.

The processors' causal linking can be used to build a graph structure that represents a narrative. The nodes are anything: actions, desires, wants, mental states, etc. The links represent a prediction or an explanation. Below is a graph representing the events in Macbeth.

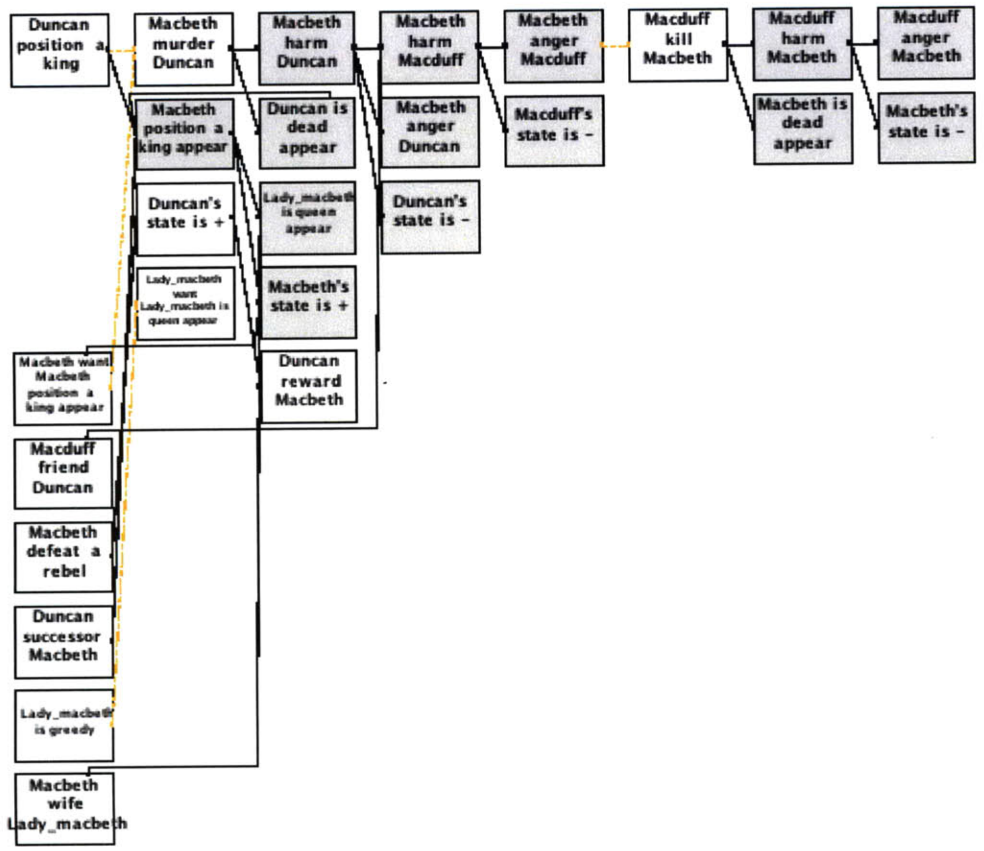


Figure 9: A graphical representation of Macbeth. White boxes are elements from the text. Grey boxes are inferences.

This structure is more general than Lehnert's, however, it lends itself better to an engineering approach. Instead of relying on links to represent things that are not mental states, the story processor makes them explicit nodes. This generalization makes it much easier to define plot units. Instead of worrying about what types of links are connecting mental states, one only has to worry about which things, mental states included, are linked together and in what way.

3. Building A Text Based Plot Unit Search Program

Prior to my joining the Genesis group, Professor Patrick Winston had built several custom plot unit search programs. He had code that would search for revenges, mistakes, successes, and Pyhrric victories and then display the plot unit trace. A sample output, the original Pyrrhic victory trace found in Macbeth, appears below. However, these programs were unwieldy. It would have been an enormous burden to write more than a few plot units in custom code. So, I developed a general solution; a solution that can take in plot unit descriptions written in English and both graphically and textually display the results.

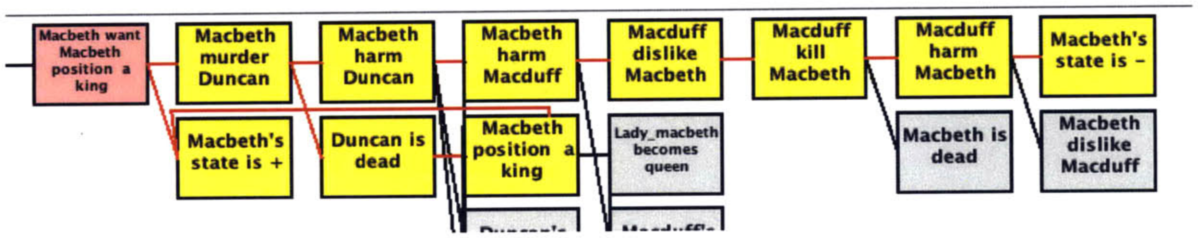


Figure 10: Genesis' original Pyhrric victory trace in Macbeth. Macbeth became king but got himself killed.

3.1 The Input English

In order to generate plot unit searches from text, the first thing that needed to be determined was the kind of input English the system would use. Based on Professor Winston's programs, I decided the system needs to know three different kinds of knowledge. First, it needs to know what roles exist in the plot unit. In the revenge example, there is a victim and a villain. The victim and villain can have many different types. They can be people, countries, groups, or even animals. Because Genesis has a type hierarchy built in, each role can be defined as the highest type in the hierarchy. For revenge, the victim and the villain are defined as entities.

Once the type of a role is defined and the role is given a name, it can be used as a variable throughout the description of the plot unit. The following input text used to generate searches for revenge provides an example:

```
Start description of "revenge".

xx is an entity.

yy is a entity.

xx's harming yy leads to yy's harming xx.

The end.
```
Figure 11: Input description of revenge.

In addition to roles, the system needs to know which events need to occur. For revenge, xx, or the villain, needs to harm yy, or the victim. Yy also needs to harm xx. The final piece of information is how the events relate to each other. We say that events "lead to" or "entail" other events. This instructs the plot-unit processor to search for a path in the graph from the first event to the second event. In the case of revenge, there is only one chain in the plot unit. However, the input English supports multiple chains. These chains may have overlapping events, which causes the plot unit to take on a graph structure. A graphical representation of a "Pyhrric victory" is shown on the next page. The input English is shown after.

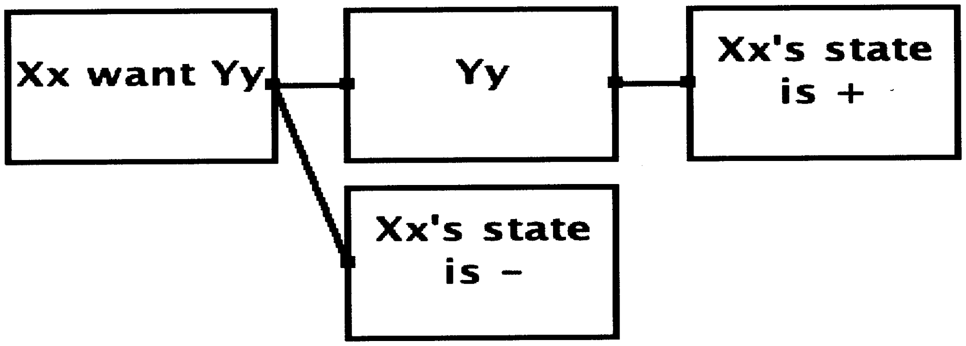


Figure 12: Graph formed by the Pyhrric victory description. A link from left to right represents a leads to chain.

Start description of "Pyrrhic victory".

xx is a person.

yy is anything.

xx's wanting yy leads to yy.

yy leads to xx's becoming happy.

xx's wanting yy leads to xx's becoming unhappy.

The end.

Figure 13: Input English for a Pyhrric victory.

3.1.1 Why does the English Form a Graph?

At a glance, it appears that the input English does not form a graph. Instead, it looks like it only specifies a set of independent links. However, the English does indeed form a directed graph. Genesis has modules that determine if different English descriptions refer to the same semantic object. By default, if Genesis sees the exact same English structure twice in a story, it will mark the two instances as having the same underlying semantic object. In the case of a Pyhrric victory, xx's wanting yy is mapped to the same underlying object both times. English markers such as "another" will override this default disambiguation.

3.1.2 Why Search for Paths?

The system searches for paths because a path represents causality without respect to the details. Depending on how thorough the analysis of the story is, there can be an arbitrary number of events between the desired cause and the desired effect.

Consider this example. Imagine a scenario where I steal a cookie from the cookie jar and I am scolded by my mother. At a higher level, I know that my act of stealing caused my mother to scold me. But I can also say that my stealing the cookie elicited emotions in my mother, which in turn caused her to scold me. I can then say that these emotions were triggered by her noticing the missing cookie. I can also say that she noticed the missing cookie because perceptual cells in her eye fired. There can be many causally related events that occur between any two events. It all depends on how much the story is broken down. By searching for paths, the plot-unit processor circumvents the issue of granularity of analysis.

3.1.3 Why not Give Genesis the Most Basic Story?

One of the options that was considered was to have the input English be the most basic story that encapsulates the desired plot unit. That way, Genesis could simply save that story's graphical representation and use that to generate searches. However, this approach was abandoned. One reason it does not work is that a user does not want a commonsense rule in the story processor to accidentally fire, filling in events that should not be in the plot unit. Having the user enter all of the chains automatically avoids this trap.

The second reason is that this template is easily extendable. For example, the input English also supports "does not lead to" chains. These chains occur whenever an event does not trigger another event. Consider the case of "unfulfilled desire."

```
Start description of "unfulfilled desire".
xx is a person.
yy is anything.
xx's wanting yy does not lead to yy.
The end.
```

Figure 14: Genesis' description of an unfulfilled desire.

An unfulfilled desire happens whenever a person wants something and does not achieve it. Without "does not lead to" chains, this plot unit would be impossible to express. The next section will cover some other additions that were made to enhance the input English. As this research continues, I expect that even more additions will happen. Keeping the English in template form will make these additions easy.

3.1.4 Reifying Actions

Let's take a closer look at what my mother told me when I barreled head first into that wall. She told me that I *did something* that caused me to be unhappy. Very rarely do we read a story where one actor is simply said to have done something. Instead, actions in narrative are concrete. We know if the actor struck someone, if he opened a door, or if he retrieved the newspaper. However, for the purposes of defining plot units, humans allow events to be abstract. One can simply say that a person did something.

This abstraction implies that we need to *reify* actions in our input English. That is, we need to be able to transform abstract concepts into concrete events. For the input English, we allow two reifications.

First, an action can be described simply as an action. A user can enter the chain "An action leads to xx's becoming unhappy," and the system will automatically map that action to something concrete in the story. For example, it may match an instance where someone is slapped and then becomes unhappy. The abstract concept "action" is then mapped to slapped.

The second reification that a user can enter is illustrated by the following example: "xx's performing an action leads to xx's becoming unhappy." The system will then look for some action where xx is the subject. This reification directly corresponds to the description of mistake that my mother told me when I was younger. In that case, the action I performed was throwing myself into a wall.

A third possible, but unimplemented, reification is to allow something like "xx's receiving an action leads to xx's becoming unhappy." This would imply that there was some action where xx was on the receiving end. However, this reification has not been needed to implement any of the plot units the Genesis group wanted to search for so far.

3.2 The Search Algorithm

This section describes how the plot-unit processor finds entered plot units in stories. But first, I will describe what it is looking for.

Given the graph of a plot unit built from an English description, the plot-unit processor searches for subgraph matches of that unit inside of story graphs. However, unlike normal subgraph matching, each edge in the plot unit graph is mapped to a path in the story. Like normal subgraph matching, each node in the plot unit graph is mapped to a node in the story. The following sections explains the details.

3.2.1 The Data Structures

The plot-unit processor uses several data structures in order to find plot units. The most basic data structure is a *story node.* A story node contains an individual semantic object found in a story. It has two sets of links. Parent links are links to other nodes that predict or explain the underlying semantic object. Child links are links to other nodes that are predicted or explained by the underlying semantic object. Whenever the plot-unit processor receives a story from the story processor, it builds all of the nodes and links. Together, these nodes form the *story graph.* The story graph is the graphical representation of the story discussed earlier.

Story nodes are used to build *story chains.* A story chain is a representation of a path in the story that matches an edge in the plot unit. It contains a source node, the beginning of the path, and a target node, the end of the path. Story chains also contain an ordered list of all the events that occur between the source and target node. Further, story chains contain an *actor map.*

An actor map is a hash map that maps the roles in the plot unit to the actors who are filling those roles in the chain. Figure 15 shows a story chain that matches the single edge found in revenge. The source node is "James punch Dave." The target

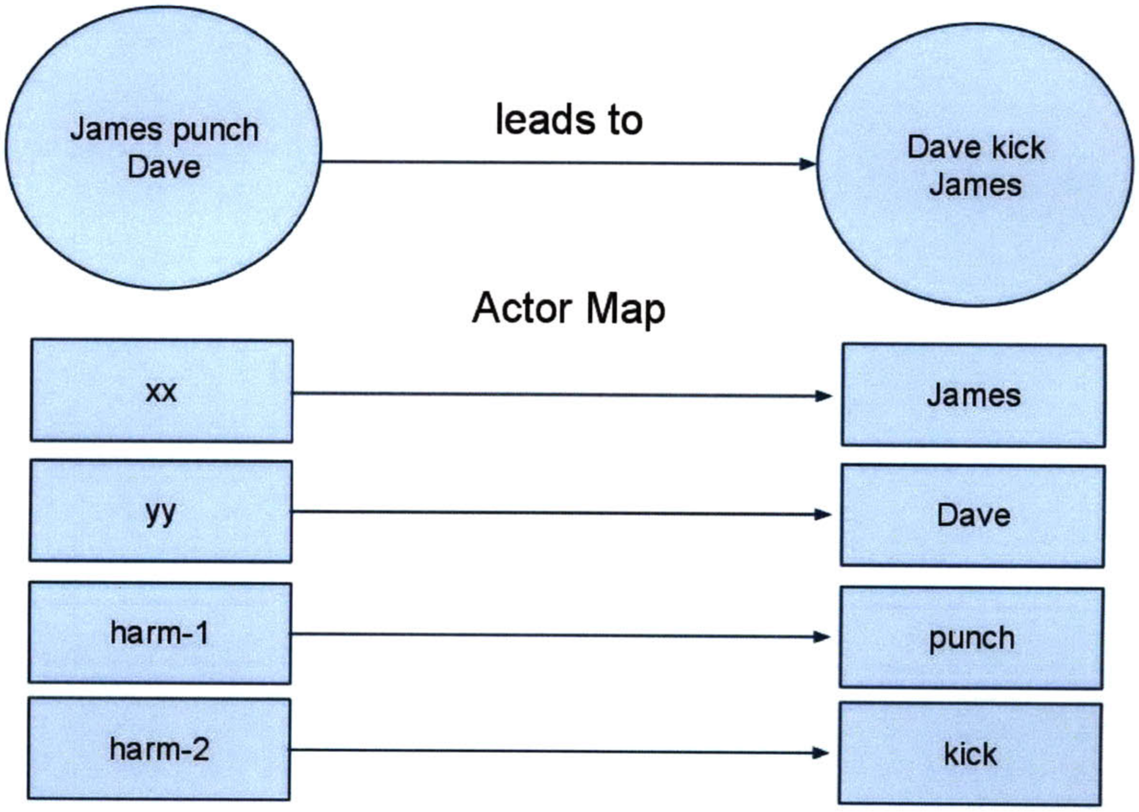


Figure 15: A story chain that maps to the single path in revenge. The top shows the source and target of the chain. The bottom shows the actor map for the chain.

node is "Dave kick James." Notice that there are more entries in the actor map than just xx and yy. The actor map also remembers which actions in the story correspond to which actions in the plot unit. Actually, it is more detailed than that, but that topic will be addressed in the next section.

Also note that the harms have a number attached to them. Genesis provides each semantic object with a unique identifier. These identifiers allow the plot-unit processor to distinguish between semantic objects of the same type.

In order to represent a plot unit, the plot-unit processor makes a plot unit object. The plot unit object contains a list of pairs of semantic objects. These pairs represent the leads to chains in the input English. They are the edges in the plot unit graph. Each pair also has a marker called *negated.* Negated signals whether or not each pair represents a does not lead to chain. The plot unit object also contains a *role map,* which is a hash map that maps user defined roles to their types.

I considered eliminating the role map. The algorithm could derive all type information from the semantic objects each pair represents. However, including a role map allows the plot-unit processor to use some of its own types. Right now, a user can declare a role to be of the wild card type "anything." If a user inputs "xx is anything" into Genesis, Genesis will label xx as a "situation." By using a role map, the plot-unit processor circumvents instances where the type label provided by Genesis does not suit its needs.

3.2.2 Matching a Source and a Target

This section defines what it means for a semantic object in the story to match a semantic object in the plot unit. An object in the story matches an object in the plot unit whenever the story object contains a set of semantic objects that correspond to the set of semantic objects in the plot-unit object. Here, correspond is defined as being a subtype match. It is much easier to comprehend this matching visually. Figure 16 shows two semantic objects. The semantic object on the left represents "xx wants yy," where xx is a person and yy is anything. The semantic object on the right represents "Macbeth wants to become king." Note how matching is implemented recursively.

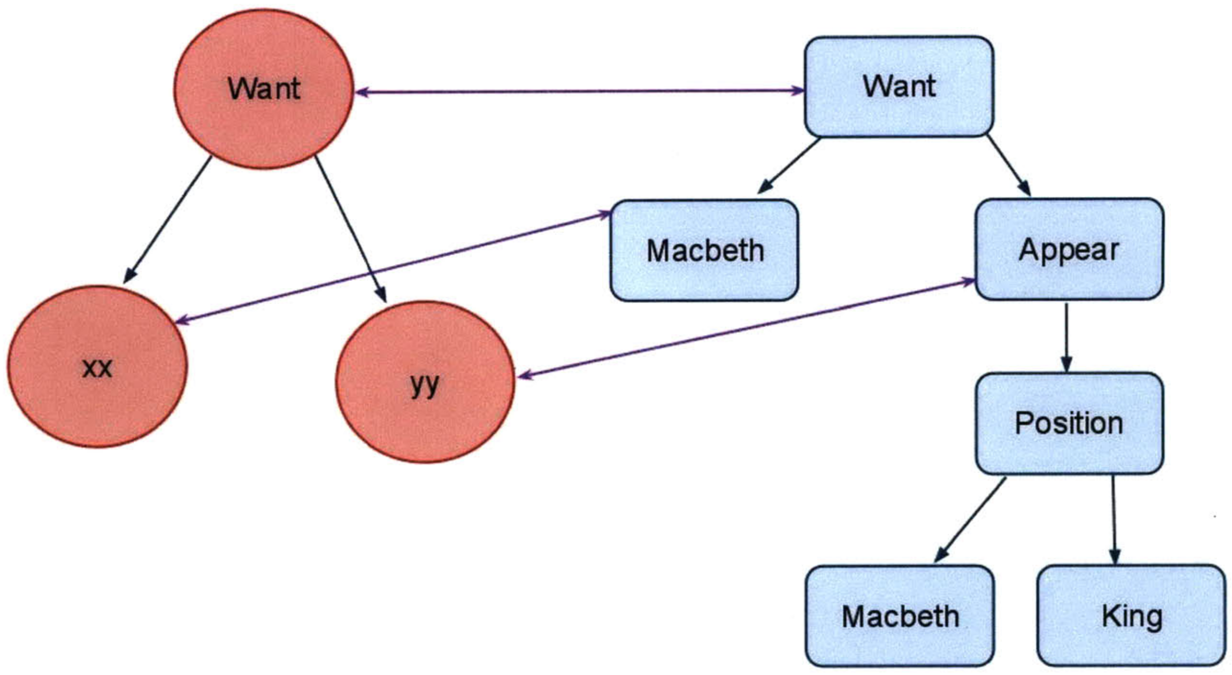


Figure 16: Matching "xx wants yy" to "Macbeth wants to become king." Blue links represent arguments to a semantic object. Purple links represent matching semantic objects.

First, the root types of each semantic object are checked. If they correspond, then each element in the plot-unit object is matched against each element in the story object. This process continues until all of the semantic objects in the plot unit have been matched or until an inconsistency is found.

Also note that a story object might contain more semantic objects than the plot unit object. This is because subtypes of objects may contain additional elements. In the above example, yy matches appear. Appear is a subtype of yy's type, however unlike yy, it requires an additional element.

The matching phase is also where actor maps are built. Whenever two semantic objects match, the object from the story is assigned the role of the object in the plot unit. Figure 17 shows the actor map that would be built from the above match.

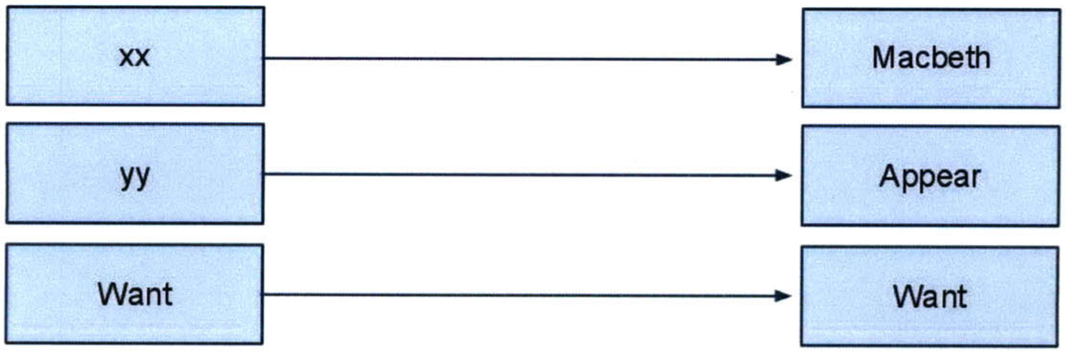


Figure 17: Actor map built from the match in figure 16. Objects in the plot unit are mapped to the objects in the story that match them.

3.2.3 Building Up Sets Of Chains

To find plot units, the plot-unit processor builds up sets of *candidate chains*. These sets represent all of the story chains found in the story that match a given pair of semantic objects, or leads to, in the plot unit.

Finding a candidate chain is simple given the data structures. First, the plot-unit processor loops over the story nodes in the story graph. If it finds a node that matches the first object in the plot unit pair, it will perform a depth first search to find a path from that node to a node that matches the second object in the plot unit pair. If it finds such a node, it will compare the actor maps made from each match. If the maps are found to be consistent, the processor will create a new story chain. Two actor maps are said to be consistent if each common key maps to the same object. Stated another way, they are consistent if they do not have different objects mapped to the same role. The actor map for the story chain is then the result of merging the actor maps of the nodes. Merging actor maps is simple. All of the key value pairs are combined to form a single, larger actor map.

Note that negated pairs are handled a little differently. A new story chain is created if the plot-unit processor does not find a match of the second plot unit object after finding a match for the first. In this case, the created story chain only contains a single object.

3.2.4 Building a Plot Unit Out of Consistent Chains

Once the plot-unit processor has built a set of candidate chains for each leads to defined in the plot unit, it will attempt to build plot units out of them. The plot-unit processor picks one story chain from each candidate set. Each chosen chain represents a different leads to in the plot unit. If the actor maps of each chain are consistent, that is, if all of the chains agree on which objects are filling which roles, the plot-unit processor will return that set of chains as an instance of the plot unit it is searching for. Any consistent set of chains has the following properties: It contains all of the leads to's in the plot unit. It contains all of the roles in the plot unit. Each role is filled by one and only one object with a correct type. Together, these three properties represent a plot unit.

The plot-unit processor does not need to build all of the possible sets in order to find all of the instances of a plot unit. Instead, it builds sets by depth first search. It begins by picking a story chain to represent the first "leads to" chain. Then, it scans the second set of candidate chains to find a chain that is consistent with the first. Then, it scans the third set of candidate chains to find a chain that is consistent with the first two. It repeats this process until it either completes a plot unit or cannot find a new chain that is consistent with the previously added chains. Once that happens, it backtracks and resumes searching. The search stops when all valid plot units have been found.

3.2.5 Aren't We in Theoretical Trouble? Depends on the Theory

There is one problem. The algorithm is doing subgraph matching, a known NP problem. It is possible that the plot-unit processor could take a very long time. Consider Genesis' graph of Hamlet. Hamlet contains 81 story nodes. Because a subgraph can be built out of any set of those 81 nodes, there are over 2.4 x 10^24 possible subgraphs. Yet, the

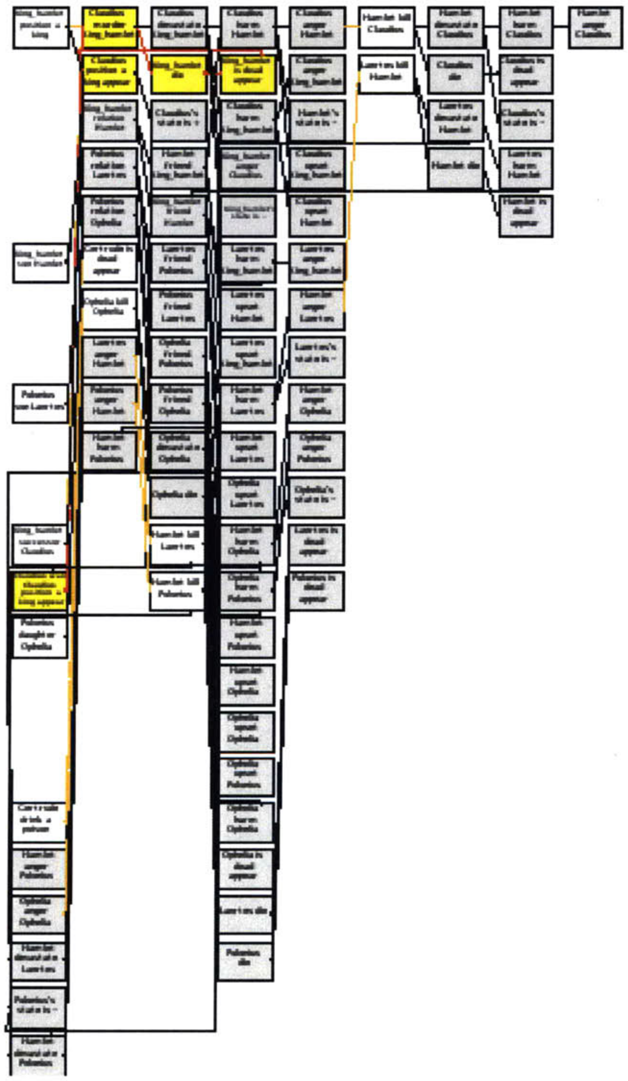


Figure 18: Genesis' representation of Hamlet with a highlighted plot unit.

plot-unit processor is able to find four plot units, a leadership achieved, a success, a revenge, and an answered prayer, in less than a second. Why?

The answer lies in which plot units humans are able to recognize. As I entered plot units, I noticed a common theme: A significant amount of overlap in the variables used by each leads to. This overlap means that the events that happen in plot units tend to center around the same actors. It also means that once a few events are filled in, the rest of the plot unit becomes clear. All of the defined roles are filled, so the remaining events that have to happen in order to form a plot unit become explicit.

This overlap also makes it easy for the plot-unit processor to find sets of story chains that form plot units. Usually, once it has built up a partial match with one or two chains, other existing chains that do not lead to completion become incompatible. These incompatible chains are inconsistent with the roles that have already been filled in.

This common theme suggests a computational constraint on the ability of humans to recognize plot units. Humans might have an easier time recognizing plot units where finding one or two pairs of causally related events immediately signals what other events to look for. In fact, this constraint might underpin the difficulty of significant literary analysis. One reason that insightful literary criticism is considered so difficult might be that finding complicated recurring plot structures in narrative is computationally intensive. Therefore, the ability to find these structures becomes praiseworthy.

3.2.6 Potential Improvements

When I was in high school, I had an English teacher who would administer nearly impossible quizzes. The quizzes tested details about the story that I am confident no one could possibly remember. “What color was Lucy's blouse in chapter 4?” He claimed to give these tests to make sure the class had actually read the work, not just the Cliff Notes. I always read the work. It did not matter. However, whenever I participated in class discussions, I could always recall the events being discussed. These events were things like betrayals, harms, and achieved goals. In other words, they were the events that formed plot units.

As Genesis evolves, it is possible that sets of typical plot-unit events, such as types of harm, could be developed. If Genesis had such a set, then the computational complexity of searching for plot units would be reduced. Instead of looping over every node in a story, the plot-unit processor could only look at the nodes that are relevant to that plot unit.

Additionally, the plot-unit processor could be smarter about searching for chains. If a partially matched chain fills in all of the roles, then the processor could simply search for the remaining events. This approach is similar to the human approach discussed in the previous section.

3.3 Outputs

This section will briefly describe how plot units are displayed to the user.

3.3.1 Text Output

The text output is simple. The underlying objects of each chain are run through a simple language generator and printed to the screen. Figure 19 shows an example.

```
Found instance of "leadership_achieved"

Macbeth position a king appear . Macbeth's state is +.
```

Figure 19: Text output for a "leadership achieved." Macbeth becomes the king and leader.

The text output has not changed much over the last year. It was abandoned after realizing that it is more of a language generation problem and is less informative than the graph based output.

3.3.2 Graphical Output

The graphical output has already been presented many times in this thesis. The graph output shows Genesis' graph of the underlying story. A button appears at the bottom of the screen showing any found plot units. Clicking the button will highlight all of the events that form the found plot unit.

4 Where do we go from Here?

This section will briefly describe some extensions that can be made to this thesis. The Genesis group expects that these enhancements will be made by future students.

4.1 Deeper Story Analysis

This section presents a two ways to perform more detailed analysis on stories.

4.1.1 Analogies Made Simple

Actor maps make analogies simple. Because actor maps contain a map of which actor is playing which role, they can be used to compare actors across stories. For example, in Macbeth, Macbeth makes a mistake by murdering Duncan. In the Estonia-Russia conflict story, Estonia makes a mistake by offending Russia. As a first pass, one can say that Estonia is like Macbeth because they both played the same role in a mistake.

It is hard to believe that analogical analysis can be produced so simply. However, it is not really that simple. There is quite a lot of machinery that goes into building up the actor map. However, once actor maps are built, the amount of computation needed to produce basic analogies is almost zero.

The Genesis group expects additional work in the area of analogies. There are many additional ways to compare plot units. One can compare the types of each actor, the actions they took, or even the events that lead from one action to another. Analogies fit neatly into our vision of helping analysts deal with cyber relations issues and legal cases because they can quickly find insightful similarities between two narratives.

4.1.2 Getting in the Other Guy's Head

One thing that the Genesis group noticed while examining plot units was that, depending on the commonsense rules used, different plot units occurred. From a human perspective, this makes sense. Humans interpret stories differently. One person's success might be another's failure. What one person might interpret as an act of revenge another might interpret as an act of temporary insanity.

What the Genesis group wants to develop is a way to view stories through multiple *perspectives*. There has already been some research done towards this goal. Figure 20 on the next page shows a legal case brief viewed through two different perspectives. The case centers around a man, named Ceballos, who had some property stolen from his garage by Steven. In response, Ceballos created a pulley and spring system that attached to a handgun. He connected the pulley to his garage door so that whenever someone opened the door, the pistol fired. Steven was shot by the trap when he tried to break back into Ceballos' garage. Is Ceballos guilty of manslaughter? It depends on your perspective. If you believe that Ceballos' act of creating a trap makes him responsible for Steven's death then, yes, it is manslaughter. However, if you do not believe that rule, then it is not manslaughter. By loading in different sets of commonsense knowledge, a user can simultaneously view a story through two different lenses.

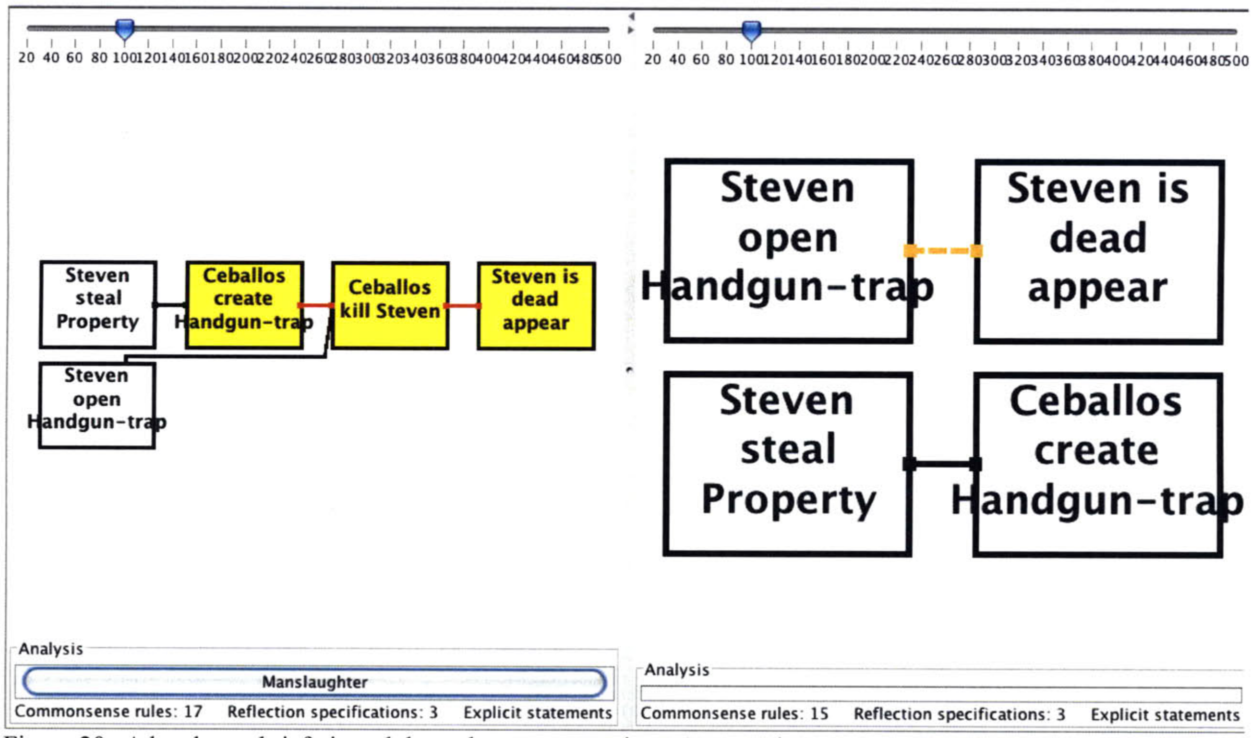


Figure 20: A legal case brief viewed through two perspectives. Perspective one contains a trace of manslaughter. Perspective two does not.

The addition of perspective allows the Genesis group to *model how entities think*. Each perspective represents a different interpretation of the same events. By studying these interpretations, several important questions can be answered: What is important to this entity? Why do they not think the way I do? What would motivate them to act a certain way? These questions are very important and the Genesis group expects that much work will be done on perspective in the upcoming year.

4.2 Predicting the Future

One of the main goals of the Genesis project is to provide tools that can help predict future events. I believe that plot units might be one such set of tools. Whenever Genesis detects a partially filled in plot unit, it can make predictions about upcoming events. If it sees a partially filled in revenge unit where China attacks Iran, it might predict that Iran will attack China in order to complete the revenge. Some work has already been done by Adam Belay on plot unit prediction in an unpublished term paper.

He designed a system that makes predictions based on plot units that are partially filled in. Here, partially filled in is defined as having a full set of actors and an almost full set of chains. One of the things he has found during his work is that some partial plot units are ubiquitous. For example, almost any action can lead to a mistake. Others like leadership achieved are not. We expect that the relative scarcity of potential plot units will play a factor in how useful they are for prediction. It would be unwise to make predictions based on a unit that is almost always partially complete.

Right now, we do not have enough stories in our database to resolve the question of prediction empirically. However, we have a theory that we are going to explore. The theory is that, like facial features, plot units of intermediate complexity will be the best for prediction(Finlayson and Winston 2006). We postulate that units such as mistake and success are, like eyes and noses, building blocks of other, more complicated units. We also recognize that plot units that are too specific and complex are probably unlikely to be useful for prediction. These units probably do not occur often. Using a complexity based model would allow the Genesis group to extend Adam Belay's work by ranking predictions based on the complexity of the partial unit. Prediction is very important to our group, so expect work in this area to progress quickly.

5 Results and Contributions

This section will present the results and contributions of the thesis.

5.1 Results

Currently, the plot-unit processor finds seven plot units in Macbeth: success, mistake, Pyhrric victory, insane violence, revenge, answered prayer, and leadership achieved. It find four plot units in Hamlet: leadership achieved, success, mistake, and revenge. It finds three plot units in the Estonia-Russia cyber conflict: mistake, success, and revenge. It finds three plot units spread out among seven legal case briefs: act of insanity, manslaughter, and larceny by deception. There are currently 26 different plot unit descriptions written for Genesis.

An important result is that so far, the type of story has not affected plot unit analysis in any way. The same underlying machinery finds plot units in Shakespeare, cyber security, and law. This result suggests that the machinery used in narrative processing is the same amongst different types of stories.

5.2 Contributions

In this thesis I made three main contributions.

1. I presented an English template for describing plot units. I explained several units written using the template and showed reasons why the template is a beneficial and extendable format.
2. I developed a computational system that is currently used to find plot units in human

narrative. The system works off of English descriptions and matches units in three different types of narrative. It also creates a simple and powerful data structure, the actor map, that can be used for deeper story analysis.

3. I explored the underpinnings of human narrative processing. I presented evidence that a common set of machinery might be used to process all types of stories. I also presented evidence that humans might be computationally limited in the set of plot units that they can find.

References

1. Wendy Lehnert. “Plot Unit Recognition for Narratives.” 1984.

2. George A. Miller. Wordnet 3.0, 2006. http://wordnet.princeton.edu/.

3. L. Vaina and R. Greenblatt, “The Use of Thread Memory in Amnesic Aphasia and Concept Learning.” MIT AI Lab: 1979.

4. Mark Alan Finlayson and Patrick Henry Winston, “Analogical Retrieval via Intermediate Features: The Goldilocks Hypothesis.” 2006.

Appendix

The 26 Plot Units

Start description of "perseverance". xx is an entity. xx's wanting an action leads to
xx's wanting the action. The end.

Start description of "success". xx is an entity. yy is anything. xx's wanting yy leads
to xx's becoming happy. The end.

Start description of "failure". xx is an entity. yy is anything. xx's wanting yy leads
to xx's becoming unhappy. The end.

Start description of "enablement". xx is an entity. yy is anything. The first action
leads to xx's becoming happy. The first action leads to xx's wanting second action.
The end.

Start description of "motivation". xx is an entity. yy is anything. xx becomes unhappy
because the first action occurred. A first action leads to xx's becoming unhappy. The
first action leads to xx's wanting a second action. The end.

Start description of "recursion". xx is an entity. ll is an action. mm is an action.
xx's wanting ll leads to xx's wanting mm. The end.

Start description of "positive tradeoff". xx is an entity. An action leads to xx's
becoming happy. Another action leads to xx's becoming happy. xx's becoming happy leads
to xx's stopping becoming happy. The end.

Start description of "negative tradeoff". xx is an entity. An action leads to xx's
becoming unhappy. Another action leads to xx's becoming unhappy. xx's becoming unhappy

leads to xx's stopping becoming unhappy. The end.

Start description of "loss". xx is an entity. An action leads to
xx's becoming happy.
Another action leads to xx's becoming unhappy. xx's becoming unhappy
leads to xx's
stopping becoming happy. The end.

Start description of "resolution". xx is an entity. An action leads
to xx's becoming
unhappy. Another action leads to xx's becoming happy. xx's becoming
happy leads to
xx's stopping becoming unhappy. The end.

Start description of "change of mind". xx is an entity. ll is
anything.
xx's wanting ll leads to xx's stopping wanting ll. The end.

Start description of "Positive co-reference". xx is an entity. yy is
an entity. An
action leads to xx's becoming happy. The action leads to yy's becoming
happy. The end.

Start description of "Negative co-reference". xx is an entity. yy is
an entity. An
action leads to xx's becoming unhappy. The action leads to yy's
becoming unhappy. The
end.

Start description of "Mixed blessing". xx is an entity. yy is an
entity. An action
leads to xx's becoming happy. The action leads to xx's becoming
unhappy. The end.

Start description of "Hidden blessing". xx is an
entity. yy is an entity. An action leads to xx's becoming happy.
The action leads to
xx's becoming unhappy. The end.

Start description of "Mistake".
xx is an entity.
xx's performing an action leads to xx's becoming unhappy.

```
The end.

Start description of "Leadership achieved".
xx is an entity.
yy is a position.
xx's becoming ruler leads to xx's becoming happy.
The end.

Start description of "Success".
xx is an entity.
yy is anything.
xx's wanting yy leads to yy.
yy leads to xx's becoming happy.
The end.

Start description of "Revenge".
xx is an entity.
yy is a entity.
xx's harming yy leads to yy's harming xx.
The end.

Start description of "Insane violence".
xx is an entity.
yy is a entity.
xx's being not sane leads to xx's killing yy.
The end.

Start description of "Pyrrhic victory".
xx is a person.
yy is anything.
xx's wanting yy leads to yy.
yy leads to xx's becoming happy.
xx's wanting yy leads to xx's becoming unhappy.
The end.

Start description of "Answered prayer".
xx is a person.
xx's wanting an action leads to the action.
The end.
```

```
Start description of "manslaughter".
xx is a person.
yy is a person.
xx's performing an action leads to yy's becoming dead.
The end.

Start description of "larceny by deception".
xx is a person.
yy is an entity.
An action leads to yy's trusting xx.
yy's trusting xx leads to xx's stealing money from yy.
The end.

Start description of "act of insanity".
xx is an entity.
yy is a entity.
xx's being not sane leads to xx's killing yy.
The end.
```

Representative story - Macbeth

Start story.

```
Macbeth, Macduff, Lady Macbeth, and Duncan are persons.
A thane is a kind of noble.
Macbeth is a thane and Macduff is a thane.
Lady Macbeth is Macbeth's wife and Lady Macbeth is greedy.
Duncan, who is Macduff's friend, is the king, and Macbeth is Duncan's
successor.
Macbeth defeated a rebel.
Macbeth's defeat caused Duncan to become happy.
Witches had visions and talked with Macbeth.
Duncan rewarded Macbeth because Duncan became happy.
Lady Macbeth is greedy.
Macbeth wants to become king.
Lady Macbeth, who is Macbeth's wife, wants to become the queen.
Lady Macbeth, who is Macbeth's wife, persuades Macbeth to want to
become the king.
Macbeth murders Duncan.
Lady Macbeth becomes crazy.
Lady Macbeth dies.
Dunsinane is a castle and Burnham Wood is a forest.
```

Burnham Wood goes to Dunsinane.
Macduff had unusual birth.
Macduff fights with Macbeth.
Macduff likes Macbeth.
Macduff kills Macbeth.

The end.

Representative Commonsense - Macbeth

Start commonsense knowledge.

Henry, George, James, and Mary are persons.

First perspective.

James may kill Henry because James is not sane.

Second perspective.

Henry may want to kill James because Henry is angry at James.
James may kill Henry because James is angry at Henry.

Both perspectives.

// If a friend is harmed, your friend's harmer harms you.
If James harmed George and George is Henry's friend, then James harmed Henry.

// If the king dies, the king's successor becomes king.
If George is king and Henry is George's successor and George becomes dead, then Henry becomes king.

// If you are harmed, you become unhappy.
If James Harms Henry, then Henry becomes unhappy.

// If you are harmed, then you dislike your harmer.

If James harms Henry then James angers Henry.

If someone kills you, then you become dead.

James harms Henry because James kills Henry.

James becomes happy because James became the king and James wants to become the king.

Mary may want to become the queen because she is greedy.

James may murder Henry because James wants to become king and because Henry is the king.

Mary becomes the queen because George becomes the king and Mary is George's wife.