Chapter 6

Inside the Mind of Brutus

Good gentlemen, look fresh and merrily,
Let not our looks put on our purpose;
But bear it as our Roman actors do,
With untired spirits and formal constancy.
—Brutus, in Julius Caesar

6.1 Where Are We In History?

Though the diagnosis is brutally simplistic, we see three stages in the history of story generation, namely, (i) Meehan's TALE-SPIN and the systems that spring from a reaction to it, (ii) Turner's MINSTREL, and — no surprise here — (iii) a stage starting with the advent of Brutus. We believe that stage (ii), which encapsulates the view that even cases of awe-inspiring creativity are at bottom based on standard, computable problem-solving techniques (and as such marks the natural maturation of stage (i)), will stubbornly persist for some time alongside (iii). As you will recall, we cheerfully operate under the belief that human (literary) creativity is beyond computation — and yet strive to craft the appearance of creativity from suitably configured computation. Stage (ii) is fatally flawed by the assumption that human creativity can in fact be reduced to computation. When the years tick on and on without this reduction materializing, more and more thinkers will come round to the approach Brutus is designed to usher in: that which shuns the search for such reduction in favor of clever engineering. Hollywood will
want machines to co-write screenplays, even if the scientists haven't figured out how reduce point of view to computation. (Remember Chapter 3.)

6.1.1 TALE-SPIN, and Reaction

In the previous chapter we saw some sample output from TALE-SPIN, the world’s first noteworthy story generator. How does TALE-SPIN work? Basically as follows:

1. Identify a character from a pre-defined set;
2. Give that character a problem from a pre-defined set;
3. Create a micro-world from a pre-defined set;
4. Input 1-3 to a simulator; document the character's attempt to solve his/her problem;
5. Either stop or Goto 1.

TALE-SPIN was seminal, but no one, save perhaps for the pathologically charitable, would ascribe literary creativity to the system. It is probably uncontroverted that a system warranting such an ascription needs to be in command of a loop like this (and indeed
BRUTUS₁ has the power to run and capitalize on such a loop, but a lot is missing, as is clearly revealed by the stories TALE-SPIN spins (recall Chapter 5). What’s missing? A number of authors have stepped up to try to answer this question: De Beaugrande and Colby [64] noted long ago that TALE-SPIN doesn’t handle more than one character (in any genuine sense), and has no notion of what a good story is like (above and beyond a character trying to reach a goal). Soon thereafter, Dehn [65] rejected TALE-SPIN’s core loop altogether, claiming that what is missing is management of the author’s intentions. Lebowitz [146] complained that TALE-SPIN lacks the sort of knowledge that allows for rich, believable characters. Yazdani [252], looking back on all these complaints, designed his ROALD system to address them: ROALD includes a simulator like TALE-SPIN’s, a module for generating plots, a module for generating what Yazdani calls “worlds,” and modules for representing the narrator’s plans and generating the actual text. It isn’t clear that ROALD ever matured,¹ but at any rate, the overall design is still well short of what one must aim at in order to build an artificial author that can hold its own with human counterparts.

### 6.1.2 Turner’s MINSTREL

The best extant story generator we know of is Scott Turner’s MINSTREL. The best thing about MINSTREL is that it is accompanied by a rich theoretical base, delivered in Turner’s The Creative Process: A Computer Model of Storytelling [237]. Serious students of story generation would do well to add his book to their libraries. The book you’re holding, coupled with Turner’s, seems to us to make for a nice foundation on literary creativity — in no small part because of the informative contrast between his approach and ours.² For example, the core of Turner’s approach is the position that creativity is problem solving, while we start by facing up to at least the possibility that creativity, at least the varieties that look like what we have called raw origination, is super-computational. Turner gives this example to anchor his treatment of creativity as problem solving:

¹Yazdani said in conversation with one of us (Bringsjord) in 1990 that the (interestingly enough — given our use of FLEX and Prolog for BRUTUS₁) Prolog-based ROALD crashed constantly, and was probably beyond repair.
²And no such library, of course, should lack [160].
One day, while visiting her grandparents, Janelle was seated alone at the dining room table, drinking milk and eating cookies. Reaching for the cookies, she accidentally spilled her milk on the table. Since Janelle had been recently reprimanded for making a mess, she decided to clean up the spill herself.

Janelle went into the kitchen, but there were no towels or paper towels available. She stood for a moment in the center of the kitchen thinking, and then she went out the back door.

She returned a few minutes later carrying a kitten. The neighbor’s cat had given birth to a litter about a month ago, and Janelle had been over to play with the kittens the previous day. Janelle brought the kitten into the dining room, where he happily lapped up the spilled milk. ([237], pp. 22–23)

If one accepts Janelle’s impressive reasoning here as a paradigmatic case of creativity, one from which general principles of (literary) creativity can be induced, then one might well end up where Turner does, namely, at the view that creativity is a matter of solving problems by adapting known solutions to solutions that can crack new problems. (Given the current state of AI, if you take Janelle to be representative, you also end up, as Turner does, affirming specific tools, such as case-based reasoning.) Those who have read our book to this point will be unsurprised by our reaction to the Turnervian approach. The bottom line is that there is no evidence whatever that belletristic fiction is the product of standard problem solving; the evidence runs quite the other way. (As we saw in Chapter 5, there is powerful evidence for the view that even judgments about whether or not a story is interesting aren’t the result of algorithms applied to previous cases.) Our approach is to stare unblinkingly into the eyes of the most mysterious literary muse, and to engineer a system that can hold its own — by capitalizing on an architecture, Brutus, that is designed from the outset to match the output of this muse. Turner’s approach, on the other hand, is to commit at the outset to a particular account of creativity that by its nature is implementable, but, relative to, say, Tolstoy’s powers, humble.

As we said in the Preface, it seems to us, in large part for reasons explained in previous chapters, that an artificial storyteller able to compete against inspired authors must satisfy seven “magic” desiderata that go miles beyond what Meehan and his successors, and Turner as well, have sought.
6.1.3 The Seven "Magic" Desiderata

MD1 *Do violence to proposed accounts of creativity.* An impressive storytelling AI is one that satisfies proposed sophisticated accounts of creativity. *Brutus* does this: recall that he (it? — we confess to finding it irresistible, for reasons having perhaps to do with the gender of the late, non-digital Brutus, to regard *Brutus* as male) qualifies as creative according to the definitions examined in Chapter 1.

MD2 *Generate imagery in the reader's mind.* An artificial agent aspiring to be counted among the literati must be able to spark significant readerly imaging. Recall Chapter 2.

MD3 *Story in "landscape of consciousness."* A good storytelling AI must produce stories having not only a landscape of action, but also a landscape of consciousness, that is, a landscape defined by the mental states of characters. Recall Chapter 3.

MD4 *Mathematize concepts at core of belles-lettres fiction.* No artificial agent will lay claim to being counted literarily creative unless it processes the immemorial themes at the heart of literature; and such processing can presumably come only if the themes in question have been formalized. This desideratum was discussed in Chapter 4.

MD5 *Generate genuinely interesting stories.* A true artificial storyteller must produce genuinely interesting stories — which may well mean, given what we learned in Chapter 5, that the AI in question must appear to do something that is uncomputable (viz., decide the set $S^T$ of interesting stories).

MD6 *Tap into the deep, abiding structures of stories.* Any truly impressive artificial author must be in command of story structures that give its output an immediate standing amongst its human audience. Such structures are discussed in this chapter, in the form of story grammars.

MD7 *Avoid "mechanical" prose.* Last but not least, there is a challenge we have yet to face up to: An artificial author must produce compelling literary prose. We cover this (daunting!)
6.2 Story Grammars Resurrected

Once upon a time story grammars were alive and well in AI. Then they died — in large part because of a three-pronged denunciation delivered by Black and Wilensky [14]. Their three objections are that story grammars

- Are formally inadequate;
- Don’t constitute necessary and sufficient conditions for something’s being a story;
- Don’t assist in a computational account of story understanding.

Given our methods and objectives, these objections are anemic: We don’t for a minute intend story grammars to in any way constitute a set of necessary and sufficient conditions for something’s being a story. (We are inclined to believe that an interesting story is at least a partial instantiation of one or more story grammars.) Our work has next to nothing to do with story understanding; it of course has everything to do with story generation. And as to whether a grammar is formally inadequate, well, inadequate for what? Our aim isn’t to devise and implement story grammars that function as all-encompassing programs for producing stories (an aim which would presumably require that such grammars be Type 0 in the Chomsky Hierarchy). What we want is simply a formalism that will enable us to pull off a certain specific trick, namely, represent and process story structures of a sort that are sorely missing in TALESPIN. Once we have this formalism in hand, we can then mold it to fit our purposes.

Let’s be a bit more specific. The notation we prefer is Thorndyke’s [231]. (The primogenitor of story grammars in connection with literature was Vladimir Propp [186], and his *Morphology of the Folktales.* In fact, we not only like the notation, but we like the specific structure. BRUTUS is capable of generating some stories that conform to Thorndyke’s constraints (see the grammar shown in Figure 6.2).

If one wants other story structures, one has only to devise an appropriate grammar. For example, the following simple story is one
6.2. *STORY GRAMMARS RESURRECTED*

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Story → Setting + Theme + Plot + Resolution</td>
</tr>
<tr>
<td>(2)</td>
<td>Setting → Characters + Location + Time</td>
</tr>
<tr>
<td>(3)</td>
<td>Theme → (Event)* + Goal</td>
</tr>
<tr>
<td>(4)</td>
<td>Plot → Episode*</td>
</tr>
<tr>
<td>(5)</td>
<td>Episode → Subgoal + Attempt* + Outcome</td>
</tr>
<tr>
<td>(6)</td>
<td>Attempt → { Event* }</td>
</tr>
<tr>
<td></td>
<td>{ Episode }</td>
</tr>
<tr>
<td>(7)</td>
<td>Outcome → { Event* }</td>
</tr>
<tr>
<td></td>
<td>{ State }</td>
</tr>
<tr>
<td>(8)</td>
<td>Resolution → { Event }</td>
</tr>
<tr>
<td></td>
<td>{ State }</td>
</tr>
<tr>
<td>(9)</td>
<td>{ Subgoal }</td>
</tr>
<tr>
<td></td>
<td>{ Goal } → Desired State</td>
</tr>
<tr>
<td>(10)</td>
<td>{ Characters }</td>
</tr>
<tr>
<td></td>
<td>{ Location }</td>
</tr>
<tr>
<td></td>
<td>{ Time } → State</td>
</tr>
</tbody>
</table>

Figure 6.2: Thorndyke’s Story Grammar.
we quickly wrote on the basis of one of our home-grown grammars (see Figure 6.3), the gist of which is that the main character, suffering from some central character flaw, ventures out on his own and gets a number of opportunities to repent of this flaw. If the opportunity is refused repeatedly, eventually the character perishes. Repentance, on the other hand, produces a happy ending. (This story makes a good starting place for reverse engineering not only the underlying grammar we engineered, but other elements that we haven't striven for.)

Repentance

Peter the puppy lived in a warm, safe house, with a loving family, on the edge of a deep forest in which the hunters roamed. Sometimes their shotguns would echo like thunder through the air. At these times he saw sadness in his mother's eyes — sadness Peter found silly. The fact was, Peter was callous. His mother loved him greatly, but was hurt by the fact that her son was often downright cold.

The forest beckoned with adventure, and one day Peter scamp- pered off for it. Before long he came upon a baby bear whose furry leg was caught in a man-made trap.

"Won't you help me?" the bear asked. Peter didn't help. Hadn't he been taught not to talk to strangers? And didn't he have a lot of exploring to do?

At the edge of a pond, Peter met a young deer who announced: "I'm afraid I'm lost. Maybe if you bark loudly, my mother will come to this spot and find me. Will you help?"

"If you think your mother's within hearing range, how lost can you be?"

Not long thereafter, as darkness began creeping into the forest, and a chill came on, Peter concluded that he was himself utterly lost. At that moment he glimpsed a piece of steel glimmer in the fading sunlight, and he froze. When he looked down, he noticed that one false move on his part, and his own leg would be snared in a trap. Slowly, inch by inch, taking care not to trigger the cruel device, Peter moved out of danger.

Where was the baby bear? Where was the deer? Peter was sick with guilt.

Later, wandering hopelessly in the dark, he suddenly saw the lights of his house. His mother wept upon Peter's return, and
6.2. STORY GRAMMARS RESURRECTED

the next day saw that her son, after hearing a distant gunshot, wince as if in pain down to his very soul.

It's important to realize that story grammars aren't only applicable to children's stories, or even to adult fiction of the short form that BRUTUS prefers. Research that makes this obvious includes some carried out by Umberto Eco [83]: he shows that a story grammar is at the heart of at least 10 Ian Fleming novels in the 007 series. (Such a demonstration is one Eco can carry out for a lot of fiction. An interesting exercise is to attempt such a demonstration on Eco's own fiction, the overall structure of which strikes us as utterly mechanical.) The grammar can be constructed from 9 "moves." They are:

A M moves and gives a task to Bond;
B Villain moves and appears to Bond (perhaps in vicarious forms);
C Bond moves and gives a first check to Villain or Villain gives first check to Bond;
D Woman moves and shows herself to Bond;
E Bond takes Woman (possesses her or begins her seduction);
F Villain captures Bond (with or without Woman, or at different moments);
G Villain tortures Bond (with or without Woman);
H Bond beats Villain (kills him, or kills his representative or helps their killing);
I Bond, convalescing, enjoys Woman, whom he then loses.

The straight scheme, that is,

\[ A B C D E F G H I, \]

is found in Dr. No. But there are variations. For example, Goldfinger embodies

\[ B C D E A C D F G D H E H I. \]

In From Russia, With Love, on the other hand,

Figure 6.3: Our “Home-Grown” Ad
venture as Tough Teacher Gram
mar. A variant of this grammar was used in Bringsjord's MYTHLOGICAL system, described in his What Robots Can and Can't Be [40].
the company of Villains increases — through the presence of the ambiguous representative Kerim, in conflict with a secondary Villain, Krilunku, and the two mortal duels of Bond with Red Grant and Rosa Klebb, who was arrested only after having grievously wounded Bond — so that the scheme, highly complicated, is B B B B D A B B C E F G H I. ((83), p. 157)

It’s easy to cast Eco’s analysis as a formal grammar. It isn’t a grammar we find particularly useful.) But our point isn’t that such a thing can be carried out. Our point is only that plot structure can be engineered with help from story grammars, and that certain structures seem to “resonate” with readers — these would then be

---

Our readers (which we presume to be, on average, rather well-read) should not infer from the presence of fairly primitive structures at the “spine” of the 007 series that Fleming is a weak or “shallow” writer. The James Bond movies are one thing (an abominable cheapening of the texts from which they arose?); the books are (at least arguably) quite another. Here, for example, is Fleming’s description of the death of Mr. Big in Live and Let Die:

It was a large head and a veil of blood streamed down over the face from a wound in the great bald skull … Bond could see the teeth showing in a rictus of agony and frenzied endeavor. Blood half veiled the eyes that Bond knew would be bulging in their sockets. He could almost hear the great diseased heart thumping under the grey-black skin … The Big Man came on. His shoulders were naked, his clothes stripped off him by the explosion, Bond supposed, but the black silk tie had remained and it showed round the thick neck and streamed behind the head like a Chimân’s pigtail. A splash of water cleared some blood away from the eyes. They were wide open, staring madly towards Bond. They held no appeal for help, only a fixed glare of physical exertion. Even as Bond looked into them, now only ten yards away, they suddenly shut and the great face contorted in a grimace of pain. “Aaargh,” said the distorted mouth. Both arms stopped flailing the water and the head went under and came up again. A cloud of blood welled up and darkened the sea. Two six-foot thin brown shadows backed out of the cloud and then dashed back into it. The body in the water jerked sideways. Half of the Big Man’s left arm came out of the water. It had no hand, no wrist, no wrist-watch. But the great turnip head, the drawn-back mouth full of white teeth almost splitting it in half, was still alive … The head floated back to the surface. The mouth was closed. The yellow eyes seemed still to look at Bond. Then the shark’s snout came right out of the water and it drove in towards the head, the lower curved jaw open so that light glinted on the teeth. There was a horrible grunting scrunch and a great swirl of water. Then silence.
structures that an artificial storyteller would need to be in command of.

6.3 Brutus: Evolution of a System Architecture

6.3.1 Brutus and Brutus₁

Brutus is an architecture for story generation. Brutus₁ is an initial implementation of this architecture that demonstrates, among other things, the power of formal representations to capture and maintain interestingness through suitable plot, story structure and language generation.

This chapter focuses on system implementation; the how rather than the what. By our lights, in AI, discovering the how is just as important a result as the what. Generating an interesting story is the what, but how an agent goes about generating the story weighs significantly in one’s judgment of the agent’s intelligence and creativity. Consider a story generator that works by simply changing the names in an existing story to produce a “new” one. It isn’t very creative; the input and output in this case are for all intents and purposes the same; hence the agent in this case oughtn’t be deemed creative.

Likewise, a story generator that simply strings together canned paragraphs based on straightforward rules shouldn’t generate much ado, even if the system consistently produces narrative with the same level of interestingness as that which flowed from Ibsen’s immortal pen. Another trivial model that comes to mind is one inspired by the word game in which one person controls a paragraph with many words replaced with blanks, each of which is annotated with a part of speech. The other players are asked, in turn, to choose for each blank a word having the requisite part of speech. The “author” fills in the blanks based on the player’s choices. When the template is done, it is read aloud. Almost invariably the resultant story gets laughs and “rave reviews.” Might a program based on a vast database of similar templates indexed by theme, plot type, story structure, rhetoric, and so on, produce an interesting story? Sure. Should one therefore conclude that such a system is genuinely creative? Surely not.

We can render a subjective and humanistic judgment as to a
system's creativity only when we are privy to the details. If we look "under the hood" of a program and find it trivial for a human to transform the program's initial data to the program's output, we are less likely to consider the program creative. If, however, we, as humans, would find it challenging to map the program's initial data to a creative artifact (like a well-written and interesting story), then we are more likely to consider the program creative. We call the perceived difference between a program's initial data and its output **creative distance**.

The main goal behind the development of BRUTUS is to produce real, working systems which, by virtue of their internal logical structure (which implements the architecture) and implementation specifics, allow for generated stories to be sufficiently distant from initial, internal knowledge representations (called, again, **creative distance**) and to vary independently along different dimensions (called **wide variability**).

There are many dimensions over which a story can vary. Plot is only one of them. Characters, settings, literary themes, writing style, imagery, etc. — these are other dimensions, and there are many more. (Recall our remarks in the Preface regarding the wide variability of Mark Helprin's belleslitrict fiction.)

Whether or not a story generator implementation can achieve wide variability hinges on what we call **architectural differentiation**. A story generation system has architectural differentiation if for each aspect of the story that can vary, there is a corresponding distinct component of the technical architecture that can be parameterized to achieve different results. While we owe many debts to the pioneers who have come before us in the field of story generation, it's safe to say that BRUTUS is a uniquely composite and differentiated architecture capable of enabling wider variability and greater creative distance than its predecessors. While our first implementation of this architecture, BRUTUS\textsubscript{1}, has very limited variability, ancestors will implement more and more of those parts of the architecture designed to secure wide variability. It is BRUTUS the architecture, not BRUTUS\textsubscript{1}, that we have toiled to create. With the architecture done, implementation is almost trivial.

Prior to the BRUTUS architecture, story generation projects experimented with various architectural approaches in relative isolation of one another. In the following sections we tour a categorization of
previous storytelling system architectures, namely:

1. Component configuration
2. Plot generation
3. Story structure expansion

We discuss how these architectural concepts in story generation have laid the foundation for Brutus.

6.3.2 Component Configuration

Sheldon Klein's Automatic Novel Writer ([129]; see also [209]) worked by following a flow-chart representing a typical mystery plot and randomly chose pre-written components, which were inserted into appropriate positions and then adjusted slightly (e.g., names were changed). The component repository from which selections were made was formed by deconstructing well-known short stories of the appropriate genre. A high-level architectural diagram for this approach is illustrated in Figure 6.4.

While writers may "borrow" a sentence structure from here or there to form components of a new story, the notion of simply configuring a story from canned components is not considered a creative approach to story generation. In our estimation, the distance from input to output achieved by this implementation could be improved
not only in enabling finer, more artful manipulation of written components, but also in more flexible plot generation.

The BRUTUS architecture does not reject component configuration in its entirety. BRUTUS\textsubscript{I} generates a scenario based on thematic descriptions. A story outline is constructed by selecting paragraph and sentence grammars from a classified collection; these grammars are treated like highly parameterized components. They are specialized on the basis of theme, plot, and literary dimensions. As we begin to detail the BRUTUS architecture we reveal its composite nature. BRUTUS’s design moderates and integrates a variety of techniques to produce a more compelling artificial creative agent.

6.3.3 Plot Generation

A variety of story generation systems focused on variability in plot. These systems used planning engines to generate plot descriptions. These descriptions were then mapped to natural language constructions. (As we saw earlier, this approach was the heart of TALE-SPIN.) The notion of interestingness was not guaranteed by explicit, separate, and well-bounded structures in the architecture. Implicit notions of interestingness were assumed to arise as by-products of the story generator’s planning process. Figure 6.5 illustrates a high-level architecture intended to produce interesting narrative merely through the process of plot generation. In this architecture, planning engines, like the one in TALE-SPIN, would implicitly ensure that the plot involved a character trying to achieve some goal. But the problem is that sometimes a character’s striving for a goal is tedious. A description of how one of us strove during a busy day of meetings to free up sufficient time to play a game of on-line solitaire might not exactly quicken your pulse.
CHAPTER 6. INSIDE THE MIND OF BRUTUS

Figure 6.6: Story Structure Expansion Architecture.

6.3.4 Story Structure Expansion

Another implementation approach based on the architecture illustrated in figure 6.6 relies principally on story grammars. Through iterative structural expansion of a story grammar, this type of system builds increasingly detailed outlines. In his excellent overview, "Storytelling by Machines," Sharples [209] illustrates grammars used by GESTER, a program developed in 1984 by Pemberton for his PhD thesis at the University of Toronto. GESTER used a story grammar to produce story outlines for the genre of old French epics. In this approach it becomes extremely difficult to represent the complex plot and literary variations in the declarative form required by story grammars. While we believe it may be theoretically possible for a story grammar to represent all the details required to generate a large variety of interesting stories, we agree with Sharples' assertion that it is highly unlikely that such a grammar will be found.

In BRUTUS, as previously indicated, story grammars are used where they are effective: to produce an outline for the story in a way that is largely independent of other storytelling dimensions.

6.3.5 BRUTUS: A Composite Architecture

The factorization of the BRUTUS architecture explicitly reflects a story generation process that starts with a symbolic encoding of a theme to anchor interestingness. The instantiation of the theme from a knowledge-base of characters and events is used to guide plot generation, story structure expansion, and natural language (= English prose) generation. The result is a quicker path to scenarios and stories that conform (necessarily) to the explicitly represented theme, which is likely to be interesting from the get-go. The idea of a composite architecture that "covers all the bases" with differentiated and independently varying components seems to have helped breathe
6.4. BRUTUS$_1$’S ANATOMY: AN INTRODUCTION

Figure 6.7: Brutus: A Composite Architecture

new life into the vision of a storytelling AI. Figure 6.7 illustrates the orchestration of BRUTUS’s different components, including thematic instantiation, story expansion, and plot and language generation, all working together with different classes of knowledge.

The rest of this chapter is decidedly “no-nonsense.” We describe enough of the “ guts” of BRUTUS$_1$ (our current implementation of the BRUTUS architecture) to enable readers to understand how the system works. This description is short on theory, but long on information about our actual implementation.

6.4 BRUTUS$_1$’s Anatomy: An Introduction

The BRUTUS technical architecture is decomposed into two distinct levels: the knowledge level and the process level.

The knowledge level comprises the different types of knowledge required to generate a written story about a particular theme. The process level contains the processes that together use the knowledge level to generate a story. The process level represents BRUTUS$_1$’s computational approach to story generation. In this section we describe the major aspects of the BRUTUS technical architecture (an overview of this architecture, highlighting the knowledge and process levels, is illustrated in Figure 6.8). We explain the representational
and computational approach used by the system to generate interesting stories. We first introduce the knowledge and process levels. In a separate sub-section we offer a brief overview of the implementation structures and methods provided by FLEX, a multi-paradigm AI programming system based in the programming language Prolog. FLEX was used to implement BRUTUS1, and the constructs available in this development system are used throughout the chapter to convey examples. In separate sub-sections, the knowledge level and the process level are explained in more detail; readers are exposed to the different types of knowledge (with corresponding representations and applications) in the BRUTUS1 system.

### 6.4.1 Introduction to the Knowledge Level

The knowledge level contains representations of different types of knowledge required to generate the written story:

1. Domain knowledge (e.g., people, places, things, events, goals, behaviors, etc.)
2. Linguistic knowledge (e.g., sentences, phrases, words, parts of speech)
6.4. BRUTUS' S ANATOMY: AN INTRODUCTION

3. Literary knowledge (e.g., thematic descriptions, literary structures, story grammars, etc.)

6.4.1.1 Domain Knowledge

In very general terms, a story is a natural language description of objects, their attributes, relationships, behaviors and interactions. It may or may not be centered around any particular theme or have any particular message. A story may or may not be interesting (and, as we've seen, a quick route to uninterestingness is through the absence of themes and structure). But all stories include a description of some set of objects and their interactions.

Domain knowledge encodes a formal representation of objects, attributes, relationships, goals, behaviors, and events — a formal description of a domain. Domain knowledge is not the story itself, but is a description of a collection of concepts about which some story may be written.

Domain concepts may include the fundamental elements that might compose a story's settings (e.g., places, things, and their attributes and interrelationships), its characters (e.g., people, their physical and psychological makeup, their goals and behaviors) and the events that occur (e.g., steal, kill, murder, give, take, run, sign, buy, borrow, loan, kiss, etc.).

6.4.1.2 Linguistic Knowledge

A story is a description of a set of objects and events. The description itself is of course encoded in a natural language. A story therefore takes the form of a sequence of words of some language strung together to conform to the language's grammar.

While domain knowledge may be considered a description of domain concepts, it doesn't qualify as a story because, among other reasons, the description is not encoded in a natural language. Linguistic knowledge is knowledge required to produce a natural language description of domain concepts from a formal, logical representation of domain knowledge.

Linguistic knowledge formally describes the linguistic structure of paragraphs, sentences, phrases, and words. It categorizes words of a lexicon as verbs, nouns, adjectives, adverbs, etc.
6.4.1.3 Literary Knowledge

With but a pinch of domain knowledge and a dash of linguistic knowledge, a story generation system can cough up a story. But chances are the story won't be particularly interesting; to the literate in our culture it may not appear to be a story at all — because most of the time the appellation 'story' is reserved for fairly polished narrative. (TALE-SPIN, though certainly seminal, would seem to be a perfect example of a system that spits out a "story" in only a most relaxed sense of that term.) A weak story may look more like a laundry list (of descriptions of places, things, people, and events) than engaging narrative. Such a story doesn't have a theme or a message, nor will it be told in a way that holds readers or steers them toward some intended meaning. In short, such defective stories fail to satisfy (all, or at least most of) our aforementioned magic desiderata. But of course we must admit that these seven points both aren't sufficient, and aren't sufficiently detailed. These points really only gesture at the mysterious elements of compelling narrative. The seventh point is particularly mysterious; it stands at the darkest part of what we gropingly label literary knowledge.

Literary knowledge is independent of a story's domain content or grammatical integrity. It is knowledge of the high art of storytelling. Literary knowledge enables the compelling communication of interesting interpretations of domain knowledge through the medium of natural language. Specifically, BRUTUS encodes literary knowledge to generate stories that can achieve key literary objectives, including the objectives described in Chapters 2, 3, and 5, namely,

MD2 Trigger readerly imaging
MD3 Project P-consciousness
MD4 Engage readers in classic themes
MD6 Instantiate classic story structures

MD2 and MD3 are achieved through the characterization and selection of words, phrases, sentence, and paragraph structures in terms of their ability to trigger images and P-consciousness in the reader. MD4 is achieved through the formalization of thematic knowledge. (In Chapter 4, recall, the formalization of the thematic concept of betrayal was discussed.) This formalization is encoded as part of the literary knowledge-base in a structure referred to as
6.4. **BRUTUS'; ANATOMY: AN INTRODUCTION**

a thematic relation. This relation is an implementation structure used to gather essential ingredients for a story about betrayal from a domain knowledge-base. Thematic relations are described in some detail later in this chapter.

MD6, recall, refers to the familiar flow of a story — how it is structured at a high-level in terms of setting, character introduction, etc. Story grammars, as we explained earlier, are what we use to represent the high-level “look-and-feel” of story structure. These grammars are part of the literary knowledge-base (KB).

### 6.4.1.4 Knowledge Usage in Story Generation

One can view the domain knowledge-base as a pool of story elements (characters, places, things, events, etc.) that can be configured to produce any number of stories about any number of themes.

The literary knowledge-base's thematic relation defines a particular theme independently of any particular set of domain elements. It is used to identify a set of elements from the domain and their interrelated roles required to tell a story about that theme. The literary KB's story grammars are used to orchestrate the high-level structure of the story that determines which paragraphs to write first and what sentence forms they contain. Literary knowledge is also used to select the key words and phrases that best communicate the theme by triggering readily imaging and projecting P-consciousness. Finally, linguistic knowledge is used to configure grammatically correct English sentences.

### 6.4.2 Introduction to the Process Level

In **BRUTUS**, story generation is decomposed into four high-level processes:

1. Thematic concept instantiation
2. Plot generation
3. Story structure expansion
4. Language generation

Each process uses knowledge represented as part of the knowledge level and stored in **BRUTUS';** knowledge-base.
Thematic concept instantiation begins with a description of a theme. This description is independent of specific objects or events that might appear in a particular domain knowledge-base. Given a specific domain KB, the theme is instantiated, which serves to identify particular objects, events, characters, etc. from that KB to play the general roles required to realize the theme. The result of this process is called a stage.

The stage is input to plot generation. Plot generation, through planning and simulation, weaves the details of a specific plot for the cast of characters identified in the stage. Plot generation completes what is called a scenario. The scenario is the stage plus a completed set of events and the effects these events had on the state of world.

Story structure expansion is the root of a separate process thread in BRUTUS. A high-level story structure, represented in BRUTUS\textsubscript{1} as a story grammar, may be input or randomly selected. As already noted, story grammars describe how a story is organized with respect to introduction, character descriptions, conflict, resolution, conclusions, and so on. Story structures in BRUTUS\textsubscript{1} are independent of plot and theme. The process of story expansion recursively expands structural components of a story represented in the story grammar until a series of paragraph types are reached. Paragraph types are further expanded into a series of sentence types. The result is a detailed story template, or story outline, that may be entirely independent of story content. Literary constraints may be used to influence otherwise random choices in the generation of a story outline. It is through these constraints that decisions regarding theme or plot may carry over to achieve congruent influence over story structure.

The scenario developed in plot generation, along with the outline produced in story structure expansion, are fed into the final process: language generation. In this process, linguistic and literary knowledge are used to produce the written story. A high-level process model is illustrated in Figure 6.9.

6.4.3 Implementation Structures and Methods

BRUTUS\textsubscript{1} is implemented using a variety of knowledge representation and programming techniques available in a logic-programming system called FLEX, originally developed by Logic Programming Associates (LPA). FLEX is based in Prolog, a popular logic-programming
6.4. BRUTUS’S ANATOMY: AN INTRODUCTION

Figure 6.9: Brutus Process Model

language based on a general computational method that starts with a goal statement and searches a database of facts to find a proof for the goal. This method models a form of reasoning often referred to as "goal-directed reasoning." (For an in-depth treatment of Prolog we refer the reader to The Art of Prolog [223] and The Craft of Prolog [171]. The genealogy of logic programming is well-documented in Logic and Logic Programming [193]. We have ourselves written on Prolog’s genealogy, and its power [23].)

FLEX provides the developer with complete access to Prolog and enhances the paradigm with frame-based structures, relations, production rules, and an English-like syntax. The syntax and semantics of FLEX are documented in LPA’s FLEX technical reference [242].

6.4.3.1 Frames

FLEX’s Frames allow the developer to group sets of facts around a specific domain entity, enabling a more intuitive structuring of the knowledge. A frame is used to represent a domain entity, and its attributes are used to represent facts or properties about that entity. From a programmer’s perspective, frames are data structures that have a name and a set of attributes with default values. Frames can be organized in a generalization hierarchy via the relationship is-a. Attributes and default values are inherited from a parent in the frame hierarchy to its children. Instances are special kinds of
frames that have actual values rather than default values. Instances are leaves in the frame generalization hierarchy. A sample frame hierarchy, including some instances, is illustrated in Figure 6.10.

The following is part of a sample FLEX frame intended to represent the prototypical person. Specific people would be represented as instances with specific values that might override the default values inherited from the person frame. The value for an attribute in a frame can be a list of things. Lists are represented by { }'s.

```
frame person is an agent
default height is 5.5 and
default weight is 130 and
default gender is female and
default eye color is brown and
default goals are {eat, sleep} and
default beliefs are {"computers write interesting stories"}
```

6.4.3.2 Relations

Relations in FLEX are used to define abstract properties about a frame or a sets of interrelated frames. Relations are processed by Prolog's goal-directed reasoning mechanism to find the specific
frames and their elements that satisfy the relation. For example, a relation may be developed to find all frames in a knowledge-base that contain the necessary features to play the role of the betrayed in a story about betrayal. The following FLEX relation captures the abstract relationships between various persons, goals, plans and actions sufficient for BRUTUS 1 to identify the components of a simple story about betrayal.

```
relation betrayal.p
  if Evil is some goal whose plan is an EvilPlan
  and whose agent is a Betrayor
  and Saying is included in the EvilPlan
  and Saying is some say
  and Thwarting is included in the EvilPlan
  and Thwarting is some thwart
  and Betrayed.Goal is the prevented.goal of Thwarting
  and Betrayors.Lie is the theme of the Saying
  and Betrayors.Lie is some support of the Betrayed.Goal
  and Betrayed is some person
  whose goal is the Betrayed.Goal
  and whose beliefs include the Betrayers.Lie.
```

This relation is actual FLEX code. Strings starting with an uppercase letter are logic variables. The strings say and thwart refer to types of actions defined in the domain knowledge-base. Goals, agents, and plans are also concepts defined in the domain KB. The English-like syntax of this relation is translated by the FLEX system into standard Prolog programs. These programs drive a goal-directed search through the knowledge-base and assign the logic variables to specific domain concepts.

6.4.3.3 Production Rules

Production rules or condition-action rules have a condition and an action. If the condition is met, the actions are executed. These rules are processed in FLEX by a forward reasoning engine. They are used in BRUTUS 1 to represent the reactive behavior of agents. The forward reasoning engine is exploited in BRUTUS 1 to provide a computational facility for simulation that runs in the course of plot generation. A set of production rules is input to the forward reasoning engine. All conditions are checked; rules whose conditions
are satisfied are fired, that is, their actions are executed. The result
of the actions may change the state of the KB, potentially causing
other rules to fire. The process continues until all rules have fired or
failed to fire. FLEX provides a variety of control features for speci-
fying the order in which rules should fire and for resolving conflicts
if more than one rule can fire at a time.

6.5 The Knowledge Level

6.5.1 Domain Knowledge

Domain knowledge includes a static description of the world, or
domain, about which a story is written. Domain knowledge includes
descriptions of people, places, things, events, actions, goals, and be-
haviors. These descriptions are represented in BRUTUS\textsubscript{1} as FLEX
frames.

6.5.1.1 Agents and Events

Agents perform actions and participate in events. In stories about
people, people are obviously an important class of agents. People are
represented as agents with physical and psychological attributes. In
addition to the garden-variety physical attributes like height, weight,
and eye color, a person’s attributes include a set of goals and a set
of beliefs. The latter set is an important element in plot generation.
Consider frame excerpts describing the by-now-familiar characters
Hart and Striver as examples. It is in part through the actions
associated with characters’ goals that a plot is developed from a
stage. Hart’s evilGoal refers to a goal that includes actions intended
to prevent Striver from achieving his goal. The interplay of these
various intentions is integral to the betrayal theme.

```plaintext
instance hart is a person
    name is 'Hart' and
    gender is male and
    goals is {evilGoal}.
```
6.5. THE KNOWLEDGE LEVEL

\begin{verbatim}
instance striver is a person and
    name is 'Dave Striver' and
    gender is male and
    goals is {to_graduate} and
    beliefs is {harts.promise}.
\end{verbatim}

Events are concepts that relate people, places, and things. In BRUTUS, the focus is on the specific event attributes required to facilitate representation of key story elements.

Hart and Striver are described as people with physical and psychological attributes. They are interrelated in the domain KB through their roles in common events. For example, Striver's thesis defense is a thesis defense, which is a type of event, where Striver is the examined and Hart is a member of the examining committee. Among other attributes, the examined and the committee are the key elements of this event, which link Striver and Hart.

\begin{verbatim}
instance strivers.defense is a thesis.defense
    thesis is strivers.thesis and
    where is university_of_rome and
    examined is striver and
    committee is {hart,meter,rodgers,walken} and
    status is scheduled.
\end{verbatim}

6.5.1.2 Beliefs

The set of beliefs associated with a person describes in part what that person believes is true about the world. That which a person believes, however, need not be true. In BRUTUS, a person, by default, is assumed to believe all that is true about the physical state of the world as represented in the domain KB (from now on simply the world). This excludes knowledge about other people's psychological attributes or beliefs. A person is not assumed to have knowledge about other people's beliefs, goals, or behavior patterns — unless otherwise explicitly stated. For example, if Hart states publicly that he supports Striver's success, then Striver, by default, believes this statement was made. The system does not conclude by default, however, that Striver believes that Hart is telling the truth (or is lying). Statements about another person's beliefs must be explicitly inferred
in BRUTUS. Explicit belief statements must be included by the designer of the knowledge-base or may be inferred as the result of a rule.

Consider Striver's belief about Hart's promise. A promise is a type of statement. The frame representing Hart's promise follows.

```
instance harts.promise is a promise
agent is 'Prof. Hart' and
utterance is 'I will sign your thesis at your defense.' and
promised is 'Dave Striver' and
intended.state is signatures of strivers.thesis include 'Prof. Hart'.
```

The intended.state associated with a promise describes some aspect of the state of the world that would be true if the promise were kept. The following frame demonstrates the representation of Striver's belief about the truth of Hart's promise.

```
instance striver is a person
    name is 'Dave Striver' and
gender is male and
goals is {to_graduate} and
    beliefs is {harts.promise}.
```

Persons may also believe statements that contradict the current state of the world. For the person with the false belief, these contradictory statements override that actual state of the world. Contradictory or false beliefs must be stated explicitly in the list of beliefs associated with a specific person. For example, if the saying of harts.promise, represented by the following frame say101,

```
instance say101 is a say
agent is 'Prof. Hart' and
statement is harts.promise.
```

existed in the domain knowledge-base and Striver did not believe that Professor Hart made the promise (i.e., that Hart uttered the statement), then Striver's beliefs must contain an explicit negative belief. The following frame represents that Striver does not believe that say101 occurred.
instance Striver is a person
    name is 'Dave Striver' and
    gender is male and
    goals is to graduate and
    beliefs is \{not(say101)\}.

6.5.1.3 Proactive Behavior: Goals, Plans, and Actions

Goals, plans, and actions are used to represent a character's proactive behavior. Once a character is set in motion, the character attempts to achieve its goal by executing the actions associated with the goal's plan. Each action has pre-conditions and state-operations. The pre-conditions ensure that the setting is right to execute the action; the state-operations attempt to change the state of the domain. If, for some reason, a state-operation fails to execute, then the action fails.

If all the elements of a plan succeed, then the goal is successfully completed by the corresponding agent. Goals, plans, and actions are considered part of the proactive behavioral model because they are initiated by the character as part of his or her explicit intentions.

A goal is a concept that has three main attributes:

1. Agent: someone or something that can achieve the goal, typically a person.
2. Plan: a list of actions.
3. Success state: a condition that if true would indicate that the goal has been satisfied.

For example, Hart may have the goal to destroy Striver's hope to graduate. To achieve this goal there is a sequence of actions that the agent, Hart, must execute. This sequence is the plan. It may include, for example, conveying to the examining committee that Striver's ideas are worthless by refusing to sign his thesis. Hart may execute all elements of the plan, but there is no guarantee that his goal will succeed. A goal's success state is a condition — a condition which, if true of some elements of the domain, indicates that the goal has been achieved: the agent has succeeded. In this case a success state may be the failure of Striver's thesis defense.

The following frame represents Hart's goal to thwart Striver's goal to graduate.


<table>
<thead>
<tr>
<th>instance evilGoal is a goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent is hart and</td>
</tr>
<tr>
<td>plan is {lie101, refuse_to_sign101} and</td>
</tr>
<tr>
<td>success is status of strivers_defense is failed.</td>
</tr>
</tbody>
</table>

lie101 and refuse_to_sign101 are specific actions that will execute during plot generation on Hart's behalf when their preconditions are met.

6.5.1.4 Reactive Behavior: Production Rules

**Reactive behavior** describes how a character reacts to changes or events in the world. Reactive behavior is represented by a set of condition-action rules (or production rules) associated with agents. If the condition is met, then the actions are executed. With reactive rules, BRUTUS₁ allows character behavior to be described in terms of how a character may respond to different events independently of the character's specific goals. Reactive behavioral knowledge is ideal for describing basic elements of a character's default behavior independently of specific roles in a theme. It provides a representation mechanism for capturing particular behavioral dimensions that "trademark" a character.

Rules associated with a character's behavior may be as specific or as general as the knowledge-base designer feels is appropriate. For example, Professor Hart might have specific behavior rules that describe exactly how he may react to a given state. Alternatively, without a specific rule, Professor Hart might react according to a default behavioral rule associated with all persons.

Reactive behavior in BRUTUS₁ functions both to increase the variability of a plot's outcome during simulation and to produce dialogue.

BRUTUS₁ combines proactive and reactive behavior in simulation in order to vary plot generation while still maintaining a theme-driven focus. The thematic frame sets key characters in motion by triggering their proactive behavior. Effects may, in turn, trigger the reactive behavior of other characters who can twist the plot by helping or hurting a character's proactive agenda.

The following production rule describes the typical reactive behavior of a member of a thesis committee. In summary, this rule states that if a member of a thesis committee is asked by the chair
of the committee to sign the candidate's thesis, the member reacts by signing the thesis.

```
rule committee.members_behavior:
  IF
  Candidate is some person and
  Thesis is the thesis of Candidate and
  the committee of the Candidate includes Member and
  Request.To.Sign is some request and
  Member is the requestee of Request.To.Sign and
  the requester of Request.To.Sign is the chairman of the committee and
  Thesis is the document of subject of Request.To.Sign and
  status of Request.To.Sign is pending
  THEN
  do answer(Member, Request.To.Sign) and
  do sign(Member, Thesis).
```

Proactive behavior, by default, overrides reactive behavior. In our first story of betrayal, “Betrayal in Self-Deception” (shown in Chapter I and again at the end of this chapter), Professor Hart intends to thwart Striver's goal to graduate by refusing to sign his thesis. Professor Hart does not react like a typical committee member. His proactive refusal to sign Striver's thesis must override the reactive behavior imposed by the production rule previously described. This is accomplished in BRUTUS; by ensuring that all actions associated with plans are given a chance to execute in response to changes in the knowledge-base before production rules are fired in response to the same changes. In this case the fact that Professor Hart refused to sign the thesis would occur first, ensuring that the reactive rule would not execute for him, since the request directed to Professor Hart would no longer be pending when the reactive rule fired.

By changing behavioral rules and/or actions, and activating different sets of rules, the simulation process will unfold different plots about the same theme and the same stage. For example, varying Professor Hart's reaction to pressure from the chairman to sign the thesis can result in a happy or sad ending for the candidate, Dave Striver. While the story is still about betrayal, the difference revolves around whether or not the betrayer succeeds in the quest to thwart the betrayed's goals.
6.5.2 Linguistic Knowledge

While domain knowledge, the thematic relation, and the results of simulation determine what BRUTUS₁ will write about, how BRUTUS₁ will express the story is determined by linguistic and literary knowledge. Linguistic knowledge is knowledge about natural language. The written story is formed from "legal" expressions in a natural language; in BRUTUS₁ the natural language is (as readers by now certainly know) English.

Linguistic knowledge is explicated through observations of how humans express internal knowledge in distinct language structures. Linguistic theory describes language structures at a number of different levels. These levels include:

1. Discourse
2. Paragraph
3. Sentence
4. Phrase
5. Word

The discourse level may be viewed as the most aggregate level. It is composed by configuring elements from the paragraph level. The paragraph level, in turn, is composed by configuring elements from the sentence level, and so on.

Words are defined as the smallest independent units that can be assigned descriptive meaning; however, words can be further decomposed into sets of letters that act as prefixes, roots, and suffixes. These can be combined to form different words. The study of how these combinations occur is called word formation morphology. The morphology of words may be influenced by semantic properties relating to number, gender, case, or tense. The study of the forms of words as they relate to these properties is called derivational morphology. Finally, words may take on different forms based on agreement with the forms of other words which occur simultaneously in a phrase. For example, the number of a noun influences the form of the verb in a noun phrase. The study of the types of agreement and how they influence the form of words is called inflectional morphology.

Encoded in the linguistic knowledge-base are a lexicon containing a set of words and the morphological rules necessary to demonstrate
6.5. **THE KNOWLEDGE LEVEL**

story generation examples. For details on encoding morphological rules see *Natural Language Computing* by Dougherty [77].

In general, a sentence in a natural language, \( L \), is a string of words taken from a fixed list of words that constitute \( L \)'s lexicon. A natural language grammar for \( L \) is a device that distinguishes the grammatical from the ungrammatical sequences of words; this defines the valid (i.e., grammatically correct) sentences of \( L \). For an introductory but impressively detailed treatment of formal grammars for a subset of English, and their use in mechanized communication, see [201].

**BRUTUS**_1_ encodes, as Prolog goals, a variety of sentence-level grammars, which include phrase-level grammars, for generating grammatically correct sentences. These generative grammars presuppose lower level linguistic procedures to ensure, among other things, verb/subject agreement, punctuation, etc.

Sentence grammars are categorized in **BRUTUS**_1_ to represent classes of sentences that have a particular structure; these sentences vary around pivotal elements. **BRUTUS**_1_ uses numerous sentence grammars. Examples include sentence types designed to describe a setting, to describe a character, to reveal a character's goals, to narrate different classes of events, and so on. Variations can result in the generation of a negative or positive tone, or the selection of words to produce different classes of imagery.

As an example, we illustrate a sentence grammar for producing a sentence type called an **independent parallel setting description**:

- **INDPSSD** → **SETTING** verb **FP**
- **FP** → 'its' **FEATURE** **FP** | 'its' **FEATURE**
- **SETTING** → noun_phrase
- **FEATURE** → noun_phrase

In sentence grammars, uppercase words in the grammar are non-terminals. Words in quotes are literals used exactly as is in a generated sentence. Lowercase words are terminals in the sentence grammar. The terminals are phrase or word types that are selected and/or constructed from the available lexicon when a sentence is generated. Sentence generation procedures use sentence grammars to produce individual sentences that conform to the grammar. Typically, certain elements are fixed in the application of a sentence generation
procedure. \textsc{brutus1} supports different generation procedures for the same grammar. A generation procedure pivots around the elements which it allows to be fixed. For example, a procedure that pivots on \textsc{setting} will generate all possible sentences that can be built from the lexicon where \textsc{setting} is fixed to a particular noun. Other sentence generators for a given grammar may pivot on more than one element.

An example of an independent parallel setting description sentence where \textsc{setting} is fixed to the noun \textit{university} is as follows:

\begin{quote}
The university has its ancient and sturdy brick, its sun-splashed greens, and its eager youth.
\end{quote}

Sentences of this type are of course fairly simple. A noun that is a setting is chosen, and some possessive verb is used to link the setting to a list of feature phrases that describe a set of objects considered characteristic of the setting.

This simple sentence grammar is inadequate, however. For sentences of this form to have literary effect, elements of the sentence must be somehow associated with literary knowledge. In the section of this chapter on linking literary and linguistic knowledge, augmented grammars that enhance linguistic structures with literary objectives are described.

A description at the discourse level in \textsc{brutus1} is realized by a structure called the \textit{story grammar}, which we visited earlier in this chapter. Such a grammar describes, at a high level, how the story will be laid out in terms of paragraphs that describe the setting, characters, and events that make up the story. For us, stories are best viewed as literary concepts as opposed to linguistic concepts, since they are ultimately language-independent. In fact, the same story grammar can be used to both lay out the frames in a silent movie and generate a written story.

\section{Literary Knowledge}

While the linguistic knowledge-base identifies types of words, phrases, and sentences as they function in the English grammar, literary knowledge describes different ways to use words and phrases together to achieve a variety of literary objectives.
6.5. THE KNOWLEDGE LEVEL

Literary objectives may include generating imagery in the reader's mind (recall Chapter 2 and MD2), suggesting a character's landscape of consciousness (recall Chapter 3 and MD3), and producing a positive, secure mood, or a negative, anxious one for the reader. Literary objectives like these may be achieved by selecting appropriate words and sentence structure combinations. Literary concepts, found in BRUTUS\textsubscript{1}'s literary knowledge-base, define attributes of linguistic concepts as they relate to domain concepts so that text generation procedures can produce expressions that achieve literary objectives.

6.5.3.1 Literary Associations

Objects described in the domain knowledge-base are interrelated and linked to linguistic concepts (words, phrases, etc.) as one way of capturing literary knowledge. The resulting associations between concepts and language elements are called literary associations. They are used to generate sentences that satisfy specific literary objectives.

Currently, BRUTUS\textsubscript{1} includes three types of literary associations:

1. Iconic features
2. Literary modifiers
3. Literary analogs

6.5.3.1.1 Iconic Features: Positive and Negative Objects are related to other objects in BRUTUS\textsubscript{1} via an attribute called iconic features. A object's iconic features are represented as a list of other objects that are typically associated with the first object in literary settings. For example, ivy and clocktowers might be iconic features of a university (as indeed they are for BRUTUS\textsubscript{1}). Wheels, engines, and speed might be iconic features of a motorcycle. The list of iconic features associated with an object is further specialized with respect to general notions of positive or negative imagery. This classification is highly subjective and may vary with respect to context; however, it can be very effective for imaging.

Consider the following frame fragment describing a university.
In this frame ivy is listed as a positive iconic feature of a university, while tests is listed as a negative one. Though the subjectivity of these classifications is apparent, the representational framework allows for capturing and configuring a variety of literary descriptions.

6.5.3.1.2 Literary Modifiers: Positive and Negative The concept of literary modifiers is an association between modifiers and the objects that they typically modify. In BRUTUS, these associations are grouped into positive and negative classes. The associations are linked to the frames representing the modified objects with the attributes

\[
\text{positive.literary.modifiers}
\]

and

\[
\text{negative.literary.modifiers}.
\]

As is the case with iconic features, the negative and positive classification of literary modifiers is entirely subjective, but nonetheless this classification is profitably configurable in the literary KB.

Given the frame for university and the following frame fragment for ivy

\[
\text{frame ivy is an object} \begin{align*}
\text{default positive.iconic.features is} & \{\text{leaves, vines}\} \text{ and} \\
\text{default negative.iconic.features is} & \{\text{poison}\} \text{ and} \\
\text{default negative.literary.modifiers is} & \{\text{poisonness, tangled}\} \text{ and} \\
\text{default positive.literary.modifiers is} & \{\text{spreading, green, lush}\}.
\end{align*}
\]

a text generation procedure for independent parallel setting description sentences can produce positive imagistic sentences like

\[
The \text{university has its lush, green ivy.}
\]

If the knowledge-base were reconfigured and ivy was also listed as a negative iconic feature of university, then the following negative imagistic sentence can be produced:
6.5. THE KNOWLEDGE LEVEL

The university is full of tangled poison ivy.

Sentence generators in BRUTUS₁ are parameterized and can be called to produce negative or positive phrases.

6.5.3.1.3 Literary Analogs Metaphors are used in order to support sentence types in BRUTUS₁. These literary devices are supported by an underlying representation structure called literary analogs. A literary analog identifies a single main object, another object called the analog that is used to represent the main object in a literary fashion, and the intended imagery suggested by the analog.

The following frame is an instance of an analog for the main object eye and one of its analogs, the sun.

```
instance analog1 is a literary_analog
  object is eye
  analogs are sun
  images are \{ warmth, power, trust \}.
```

6.5.3.2 Linking Literary and Linguistic Knowledge

To the extent that sentence grammars capture the grammatical rules of a natural language, they are considered linguistic concepts; however, in BRUTUS₁ these structures are augmented to contain elements that reflect the linkages between literary objectives and grammatical structure.

**Literary augmented grammars** (LAGs) are used in BRUTUS₁ to represent and use literary knowledge to produce compelling and interesting prose. Consider the following LAG for independent parallel description sentences:

- INDPsd → SETTING verb (isa possessive_verb) FP(n=3)
- FP → 'its' FEATURE FP | 'its' FEATURE
- SETTING → noun_phrase (has_role setting)
- FEATURE → noun_phrase (isa iconic_feature_of SETTING)

Elements on the right side of the LAG rules are annotated with expressions in parentheses. These expressions are called **literary constraints**. It is through literary constraints that linguistic and
CHAPTER 6. INSIDE THE MIND OF BRUTUS

literary knowledge interact. While sentence grammars drive the construction of classes of grammatically correct sentences in a natural language, literary constraints in LAGs are designed to shape the contents of the sentence by using literary and domain knowledge to achieve various literary objectives.

Sentence generation procedures limit the instantiation of terminals to words or phrases that satisfy the literary constraint. The constraints describe the literary roles that the elements of the sentence must assume independently and/or with respect to one another. Constraint processing relies on how words (and their associated objects) are classified and linked in the literary knowledge-base. The constraints are used to search the literary and linguistic KBs, where nouns, modifiers, verbs, etc. are categorized and associated with one another by a variety of classificational and associations. For example, brick, greens, and youth are nouns classified as iconic features of a university.

The constraint

\[(\text{isa possessive.verb})\]

in the preceding grammar constrains the preceding terminal, verb, to be instantiated from a certain class of verbs, namely, \text{possessive verbs}. Membership in this class is captured in the knowledge-base. Similarly, \text{(has role setting)} constrains the selection for the terminal \text{noun.phrase} in the \text{SETTING} rule of this grammar to be instantiated by a noun phrase whose subject can function as a setting according to the classifications contained in the literary knowledge-base. The constraint \text{isa feature.of SETTING} is used to narrow the selections for a noun phrase based on its relationship with a choice for \text{SETTING}. In this case, the constraint ensures that the noun in the associated noun phrase is represented as an iconic feature of whatever object is selected in the \text{noun.phrase} for \text{SETTING}.

The constraint (n=3) in the first rule of this grammar instructs the sentence generation procedures to produce exactly three descriptive features in any generated sentence.

6.5.3.3 Imagistic Expertise

As discussed in Chapter 2, Esrock identified four techniques for triggering images in the minds of the reader. They are:
6.5. THE KNOWLEDGE LEVEL

1. Exotic or bizarre material
2. Visual perception and P-consciousness-related verbs
3. Familiar reference
4. Voyeurism

BRUTUS\textsubscript{1} includes a framework in the literary KB for classifying linguistic elements to enable the automation of these techniques.

6.5.3.3.1 The Bizarre To capture the bizarre, modifiers are linked with objects in frames named \texttt{bizarro\_modifiers}. Consider the following instance describing the \texttt{bizarro} modifier \texttt{bleeding}.

\begin{verbatim}
instance bleeding is a bizarro\_modifier
    objects are \{sun, plants, clothes, tombs, eyes\}.
\end{verbatim}

An action analogy LAG may be augmented with constraints to stimulate bizarre images in the mind of the reader. The following LAG for action analogies,

- \texttt{BizarreActionAnalogy} $\rightarrow$ \texttt{NP VP} like \texttt{ANP}
- \texttt{NP} $\rightarrow$ \texttt{noun\_phrase}
- \texttt{ANP} $\rightarrow$ \texttt{modifier (isa bizzaro\_modifier) noun (isa analog of NP)}

in conjunction with \texttt{bizzaro\_modifiers} can be used to generate the following sentence.

\textit{Hart's eyes were like big bleeding suns.}

6.5.3.3.2 Perception & P-Consciousness BRUTUS\textsubscript{1} triggers images with the technique of visual perception by constraining the principal verb in the appropriate LAG to be a member of a class of verbs called \texttt{vp\_verbs}. These include verbs for seeing, looking, glancing, and so on.

Similarly, BRUTUS\textsubscript{1} narrates stories from a particular character's point of view using verbs selected from a special class of verbs represented in the literary KB called \texttt{pc\_verbs}. These include verbs for feeling, thinking, understanding, wanting, etc. PC verbs give the
reader the sense that the subject of these verbs has a psychological life.

A variation of the parallel setting description sentence is a sentence that describes a setting from a particular character's point of view. The following is the LAG for that sentence type. It includes a literary constraint that forces the use of a PC verb to convey the character's consciousness.

- $POV \rightarrow Agent \ (is \ a \ person) \ Verb \ (is \ a \ PC \ Verb) \ FirstPP$
- $FirstPP \rightarrow Setting \ FEATURE \ FP$
- $FP \rightarrow its \ FEATURE \ FP \ | '.$
- $FEATURE \rightarrow noun\_phrase \ (is \ a \ feature \ of \ SETTING)$
- $SETTING \rightarrow noun\_phrase \ (is \ a \ setting)$

6.5.3.3.3 Familiar Reference Stories that appeal to a reader's familiar experiences are known to quickly engage their audience. Triggering images of familiar things in the reader's mind keeps him or her interested and gives the author opportunity to manipulate the reader's expectation, better enabling the effect of plot twists and emotional turns. BRUTUS$_1$ can trigger images with familiar reference by classifying settings, objects and features as familiar to a class of readers and ensuring that they are used in stage and language generation through the elaboration of the thematic relation and the application of literary constraints.

In our examples of betrayal, the university setting is a familiar reference to a wide class of readers and is expected to quickly form familiar and comforting images in the minds of these readers. These images are reinforced with the choice of particular modifiers and features. Of course, in our examples, the imminent betrayal is quickly suggested, shifting the focus from the positive reference of the university to the universal image of classic betrayal.

6.5.3.3.4 Voyeurism Another literary "trick" known to trigger images and spark the reader's interest is voyeurism. As we noted in Chapter 2, Kafka, for example, often includes in stories scenarios in which one person sneaks a peak at the behavior of others, or even places the reader in the position of outright voyeur. Consider an elaboration of betrayal, voyeuristic betrayal. BRUTUS$_1$ may
include a thematic relation for voyeuristic betrayal that requires a few new stage elements, including a new character, the voyeur, and a new action secretly watch. The relation may compose a stage where the voyeur is envious of the betrayer and engages in regular acts of secretly watching the betrayer's activities. The stage will initialize the story with the potential for an additional subplot involving the voyeur secretly witnessing the betrayer reveal his plans to betray the betrayed to some confidant. Plot generation will run the stage through a simulation, as described earlier, working out the details regarding the voyeur's experience and decision to tell what he witnessed to the betrayed. Regardless of how the plot plays out, the point is that well-crafted sentences and word choices that describe scenes involving the secret witnessing of another's behavior will serve to tickle the emotions and imaginations of human readers.

6.6 The Process Level

6.6.1 Setting the Stage: Thematic Instantiation

The process of thematic instantiation selects components from the domain knowledge-base sufficient to generate a story about a particular theme. The process of thematic instantiation uses a FLEX relation, based on a formal mathematization of a thematic concept like betrayal, and the goal-directed reasoning mechanism underlying Prolog, to search the domain knowledge-base and assemble the story elements (characters, goals, events, settings, etc.) required to construct a story about the theme.

The following FLEX relation represents the components and their interrelationships sufficient to generate a story about the theme of betrayal.
A rough intuitive translation of how this particular relation is interpreted from a procedural perspective runs as follows.

- Look in the domain knowledge-base and find some goal with an associated plan and agent, where the plan includes a saying action and a thwarting action. This agent might be engaged in an act of betrayal; the agent might be the betrayer.

- Look to find that the alleged betrayer’s thwarting action is intended to thwart someone else’s goal. The person who has this goal might be vulnerable to a betrayal. That person might be the betrayed.

- Look to find that the alleged betrayer’s statement is in support of the betrayed’s goal and that the alleged betrayed believes the statement.

The search of the domain KB described here is performed automatically by the Prolog inference engine. If the search is successful, then the thematic instantiation process builds the stage. The stage, implemented by a FLEX frame, represents the key roles in a story about the designated theme.

In BRUTUS1, thematic instantiation captures the essential characteristics of betrayal by building the following frame:

relation betrayal_p(A.Betrayal)
  if Evil is some goal whose plan is an EvilPlan
   and whose agent is a Betrayor
   and Saying is included in the EvilPlan
   and Saying is some say
   and Thwarting is included in the EvilPlan
   and Thwarting is some thwart
   and Betrayeds.Goal is the prevented_goal of Thwarting
   and Betrayers.Lie is the theme of the Saying
   and Betrayers.Lie is some support of the Betrayeds.Goal
   and Betrayed is some person
      whose goal is the Betrayeds.Goal
      and whose beliefs include the Betrayers.Lie.

frame betrayal is a concept
  default betrayer is a person and
  default betrayed is a person and
  default betrayers.goal is a goal and
  default betrayeds.goal is a goal and
  default betrayers.lie is a statement and
  default betrayal.location is a place and
  default betrayals.evil.action is a action.
6.6. THE PROCESS LEVEL

While the domain elements of the story are selected as a result of thematic concept instantiation, the story generation process has yet to develop a specific plot. The domain concepts are static. They represent a snapshot in time, one capturing different characters and events that are the principal ingredients for a story. In plot generation the ingredients are cooked; the details of plot are produced based on the behaviors of the characters.

In BRUTUS1, thematic instantiation requires that the domain knowledge-base include many of the specific objects required to instantiate the theme. In future versions of BRUTUS the process of thematic instantiation will use general domain concepts as a basis for generating specific objects so that less domain knowledge is required as input into the story generation process.

6.6.2 Developing the Plot Through Simulation

The process of plot generation enables BRUTUS1 to produce a variety of plots around a particular theme and a particular stage. Through a forward reasoning process provided by FLEX, BRUTUS1 simulates character behavior. The principal knowledge used in plot generation is behavioral knowledge. As discussed earlier, characters have proactive behavior, represented by goals and plans, and reactive behavior represented by condition-action rules. These rules represent how characters react to particular states. Once set in motion, they result in actions that affect new states, and more rules fire; this process simulates a chain of reactive behavior. Eventually the process of forward-reasoning halts, indicating that a final state has been achieved and the simulation is over.

Changes made to behavioral knowledge lead BRUTUS1 to produce different plots given the same theme and stage. In this way BRUTUS1 may be used to write different stories about the same theme with the same cast of characters, events, and initial states.

Plot generation is very much a computational device for dynamically extending the domain knowledge-base through time. Initially the domain KB contains basic story elements that may interact in a variety of ways. The use of rules and simulation enables the process of plot generation to play out a scenario by simulating the behaviors of the selected set of characters and recording the results of the simulation in terms of new events, states, and dialogue (i.e., the results of speaking actions) in the domain KB. At the end of the simulation,
the plot is developed and a particular outcome is realized. The new domain KB contains the completed plot or the scenario.

For example, suppose the process of thematic instantiation has produced, from the domain KB, a stage for the theme of betrayal, including the following elements (expressed informally for readability):

- betrayer: Professor Hart
- betrayed: Dave Striver
- location: University of Rome
- betrayed's goal: to get all members of thesis committee to approve and sign thesis
- betrayer's promise: to approve betrayed's thesis.

Plot generation will extend the story elements though time by executing actions associated with character goals, and will process behavioral rules; the final result is a detailed scenario. Specific actions that execute in the development of this stage include

- sign
- refuse-to-sign
- say
- answer
- request.signatures
- demand

Behavioral rules governing the typical behavior of committee members and the committee chairperson would execute during plot generation. For example, at the end of the thesis defense the following rule would trigger the committee chairperson to request the members to sign the thesis.

```
rule committee.chairs.behavior1
if Committee is a thesis defense committee and Chair is the chair of Committee and Defense is the subject of the Committee and the status of Defense includes completed and unjudged
then do(request_signatures(Chair.Committee)).
```
6.6. THE PROCESS LEVEL

The following rule would result in the typical committee member agreeing to sign the thesis when asked by the committee chair.

\[
\text{rule committee members behavior}
\]
\[
\begin{align*}
\text{IF} & \\
\text{Candidate is some person and} & \\
\text{Thesis is the thesis of Candidate and} & \\
\text{the committee of the Candidate includes Member and} & \\
\text{Request.To.Sign is some request and} & \\
\text{Member is the requestee of Request.To.Sign and} & \\
\text{the requestor of Request.To.Sign is the chairman of the committee and} & \\
\text{Thesis is the document of subject of Request.To.Sign and} & \\
\text{status of Request.To.Sign is pending} & \\
\text{THEN} & \\
\text{do answer(Member, Request.To.Sign) and} & \\
\text{do sign(Member, Thesis).} & \\
\end{align*}
\]

The result of plot generation produces a scenario that would for example include the following series of actions and states (expressed informally for readability):

- Prof. Hart tells Dave Striver "I will support your defense and sign your thesis" at T0.
- Dave Striver completed his thesis defense at time T1.
- Prof. Rodgers requests approval signatures of members of Dave Striver’s committee at time T2.
- All committee members except Prof. Hart sign Dave Striver’s thesis at time T3.
- Prof. Hart refuses to sign Dave Striver’s thesis at time T4.
- Prof. Hart says "Dave does not deserve to graduate." at T5.
- Prof. Rodgers insists that Prof. Hart signs at T6.
- Prof. Hart refuses to sign at T7.
- Dave Striver’s thesis is rejected.

All the actions that take place in this series are encoded in the domain KB and are related through the stage to the roles that participate in the theme of betrayal. This relationship allows \textsc{Brutus1} to infer, among other things, that Professor Hart’s refusal to sign Striver’s thesis is the specific act through which the betrayer successfully thwarted the goal of the betrayed (i.e., Striver’s goal to graduate).

\textsc{Brutus1} must now compose the language to write the story based on the scenario and its relationship to the theme.
6.6.3 Writing the Story: Outline and Language Generation

Thematic instantiation and plot generation produces a stage and then a scenario respectively. The scenario will be input into the process of language generation. In this process, sentences are constructed based on the characters, goals, events, etc. present in the scenario. The sentence types chosen and the sequence in which they appear depend on the story outline.

The process of Story structure expansion builds a story outline. Story grammars, discussed in some detail in the beginning of this chapter, are used to represent a variety of possible story structures. The process pursues a path through a grammar hierarchy, making either random or constrained choices as non-terminals in the grammars are expanded until a string of terminals, representing sentence types, is produced. The sequence of sentence types is the story outline.

BRUTUS captures the knowledge used by story structure expansion and language generation in a three-level grammar hierarchy (illustrated in Figure 6.11). We named this hierarchy the literary-to-linguistic grammar hierarchy because its successive levels takes story expansion from high-level story structure (literary knowledge) all the way down to English sentence grammar and word selection (linguistic knowledge). The top level of the hierarchy is composed of story grammars. Story grammars may be organized in a taxonomy, in which top-level story grammars are very generic and may apply to a wide variety of story types. Lower levels in the story grammar taxonomy may contain story grammars specialized for particular kinds of stories, like stories about betrayal, for example.

The terminals, or leaves of story grammars, are names of paragraph types. These are associated with structures in the second level of the grammar hierarchy called paragraph grammars. The terminals of these grammars are in turn sentence types. The final level in the grammar hierarchy is composed of literary augmented sentence grammars (abbreviated, recall, as LAGs). These are formal language grammars that represent components of English syntax augmented with literary constraints. The augmentation enables the generation of grammatically correct sentences that achieve particular literary objectives. The leaves of LAGs are variables instantiated to words that represent domain concepts. LAGs are described in
Figure 6.11: BRUTUS\textsubscript{1} Literary-to-Linguistic Grammar Hierarchy.
some detail in the section dedicated to BRUTUS's knowledge level.

Starting with a story grammar, the structures in the grammar hierarchy are expanded until a sequence of sentence types is generated. This entire process may be performed independently of the scenario produced by thematic instantiation and plot generation. However, augmenting the story and paragraph grammars in a manner similar to the way LAGs do for sentences can allow for shaping the story outline by constraining choice points in its expansion based on literary objectives. These objectives may be determined by different elements of the theme or plot. More generally, the BRUTUS architecture suggests that any parameter, for example author styles or user preferences, may be used to constrain choices in the expansion of a story outline. The current implementation, BRUTUS, uses the notion suggested by LAGs only to direct sentence generation, however.

**Language generation** takes the scenario and the story outline and uses literary and linguistic knowledge to generate a story in English. The language generation process is all about choosing words. The story outline is a map identifying the sequence of sentence types that will be used to tell the story. Each of these sentence types must be instantiated, that is, subjects, objects, verbs, adjectives, etc. must be chosen from a lexicon. These choices are directed by the story elements represented in the scenario and the constraints imposed by literary objectives captured in the LAGs.

Consider the following fragment (simplified to enhance readability) from a BRUTUS story grammar specialized for short stories about betrayal:

1. \( \text{Story} \rightarrow \text{Setting} + \text{Goals_and_plans} + \text{Betrayed's_evil_action} + \text{betrayed's_state} \)
2. \( \text{Goals_and_plans} \rightarrow \text{Betrayed's_goal} + \text{Betrayed's_promise} + \text{Betrayers_goal} \)
3. \( \text{Setting} \rightarrow \text{setting_description(betrayal_location,pov,betrayed}) \)
4. \( \text{Betrayed's_goal} \rightarrow \text{personal_goal_sentence(betrayed}) \)
5. \( \text{Betrayed's_goal} \rightarrow \text{personal_goal_sentence(betrayed}) \)
6. \( \text{Betrayers_promise} \rightarrow \text{promise_description(betrayer,betrayed}) \)
7. \( \text{Betrayers_evil_action} \rightarrow \text{narrate_action(betrayers_evil_action}) \)

This grammar specifies a structure that first exposes the reader to a description of the setting where the betrayal occurs from the point
of view of the betrayed. It then describes the betrayed's goal and the betrayer's promise to assist the betrayed in achieving that goal. Finally, the story ends with a description of the betrayer's evil action that ultimately thwarts the goal of the betrayed, and a statement about the betrayed's state. Non-terminals in this grammar begin with caps. Terminals begin with lowercase letters and indicate the names of paragraph grammars or LAGs. Arguments to grammars are roles contained in the stage-produced thematic instantiation and are assigned domain entities from the stage completed in plot generation.

The third rule in this story grammar leads to a paragraph grammar containing two sentences for describing a location from some character's point of view. Following from our sample stage, the paragraph grammar called in this third rule would be

```
setting_description(university_of_rome,pov,'Dave Striver').
```

The terminals in the following paragraph grammar lead to LAGs; and one is for the parallel description sentence explained earlier in the section on BRUTUS's knowledge level.

1. `Setting_description(Loc,pov,Person) -> pc.of(Person,Loc) + parallel_descrp(Loc,pov,Person)`

A brief sample story generated from the simple grammars just described and the theme and stage produced earlier in our example follows.

```
Dave loves the university of Rome. He loves its studious youth, ivy-covered clocktowers and its sturdy brick. Dave wanted to graduate. Prof. Hart told Dave, "I will sign your thesis at your defense." Prof. Hart actually intends to thwart Dave's plans to graduate. After Dave completed his defense and the chairman of Dave's committee asked Prof. Hart to sign Dave's thesis, Prof. Hart refused to sign. Dave was crushed.
```

The brief examples here taken from the BRUTUS grammar hierarchy illustrate how story grammars can drive a variety of story structures that lead to paragraphs and sentence structures, which are in turn tied directly to domain entities and literary objectives through LAGs. Significant storytelling and literary variability can be achieved by altering, adding, or selecting different story, paragraph, or LAGs. Content variability can, of course, be achieved by creating or modifying thematic relations, behavioral rules, and domain knowledge.
6.6.4 Variations on a Theme

In preceding sections of this chapter, the betrayal relation and frame, and the process by which BRUTUS generates stories, starting with an interesting theme, are described. Variations can be realized by adjusting the thematic relation to describe different outcomes. For example, while the betrayal relation previously described allows for success or failure of the betrayer's goal, a betrayal frame for generating specialized stories about betrayal where the betrayer necessarily succeeds can be built (see the definition of successful(Betrayal) in the next box). The thematic instantiation process would ensure that the stories generated will include only events such that the betrayer will succeed in thwarting the betrayed. The instantiation of this type of betrayal might entail associated variations in plot, story structure, and language. A similar approach can be taken to produce a variety of specializations; for example, cases where the betrayer's goal is considered altruistic by a general audience but taken as an offense by the betrayed. This is a case that may apply to a parent betraying a child's trust to ultimately save the child from a tragic end. (You may recall that we discussed such a case in Chapters 4 and 5.) Many variations may be achieved by adding or relaxing constraints in the thematic relation.

```
relation successful(Betrayal)
   if Betrayer is the betrayer of the Betrayal
   and Betrayees.Evil.Goal is the goal of the Betrayer
   and Condition1 is the success of Betrayer.Evil.Goal
   and Condition2
   and Betrayed is the betrayed of Betrayal
   and G is the goal of the Betrayed
   and Condition3 is the success of G
   and not(Condition2)
   and BName is the name of the Betrayed
   and WName is the name of the Betrayer
   and 1.
```

6.7 Interestingness: From the Theme Down

Computer programs are not human; they are not affected by the emotional elements that are part and parcel of human drama. In fact, it is hard to even imagine how a computer could discover, from a sea of swimming 0s and 1s, what might be compelling to a living, feeling human being. (For an excellent treatment of what we
6.8. SAMPLE STORIES

...can realistically expect modern computers to understand, see [247].

However, engineering a computer to compose stories based on a pre-defined representation of what we already know to be interesting is a very different challenge. And we do already know, at least in broad strokes, what is interesting to human readers; successful authors, after all, routinely exploit this knowledge. Among the things that readers find interesting, as we noted in Chapter 5, are particular topics like sex and money and death (as the well-known cognitive scientist Roger Schank has explained: [205]), and also classic themes like betrayal, ruthless ambition, and unrequited love. The central thrust of BRUTUS’s approach to story generation is based on the assumed limitation of computers to genuinely grasp such things as interestingness and to begin with a well-established, interesting literary theme like betrayal, and then work down through plot, story structure, and finally through the details of language. Once a theme has been sufficiently formalized, it can be encoded for BRUTUS1 as a FLEX frame. This frame represents the static elements of the theme, and a FLEX relation represents the relationships between domain elements; the instantiation of these elements, structured and cast in English, yields a story. The theme and domain elements are elaborated and extended through time by simulation in the process of plot development. Here, significant variability can be realized by adjusting the actions of character and behavioral rules associated with them. Finally, the language generation process crafts prose through a disciplined expansion of a series of grammars which can be massaged to reveal different story paragraph and sentence structures. Our LAGs enable different literary objectives to drive the choices of words and phrases for generating a variety of sentences with specific literary effect.

6.8 Sample Stories

Here, together in sequence, are the four stories seen at previous points in the book. As you read them now, try to call upon what you have read in this book so that you can “demystify” the fact that they can be generated by a “mere” machine.

“Betrayal in Self-Deception” (conscious)

Dave Striver loved the university. He loved its ivy-covered clocktowers, its ancient and sturdy brick, and its sun-splashed...
verdant greens and eager youth. He also loved the fact that the university is free of the stark unforgiving trials of the business world — only this isn’t a fact: academia has its own tests, and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, one must pass an oral examination on one’s dissertation. This was a test Professor Edward Hart enjoyed giving.

Dave wanted desperately to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless inscriptions which, together, would certify that he had passed his defense. One of the signatures had to come from Professor Hart, and Hart had often said — to others and to himself — that he was honored to help Dave secure his well-earned dream.

Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Dave that it was absolutely first-rate, and that he would gladly sign it at the defense. They even shook hands in Hart’s book-lined office. Dave noticed that Hart’s eyes were bright and trustful, and his bearing paternal.

At the defense, Dave thought that he eloquently summarized Chapter 3 of his dissertation. There were two questions, one from Professor Rodman and one from Dr. Teer; Dave answered both, apparently to everyone’s satisfaction. There were no further objections.

Professor Rodman signed. He slid the tome to Teer; she too signed, and then slid it in front of Hart. Hart didn’t move.

“Ed?” Rodman said.

Hart still sat motionless. Dave felt slightly dizzy.

“Edward, are you going to sign?”

Later, Hart sat alone in his office, in his big leather chair, saddened by Dave’s failure. He tried to think of ways he could help Dave achieve his dream.

“Simple Betrayal” (no self-deception; conscious)

Dave Striver loved the university. He loved its ivy-covered clocktowers, its ancient and sturdy brick, and its sun-splashed verdant greens and eager youth. He also loved the fact that the university is free of the stark unforgiving trials of the business world — only this isn’t a fact: academia has its own tests,
and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, to become a doctor, one must pass an oral examination on one's dissertation.

Dave wanted desperately to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless inscriptions which, together, would certify that he had passed his defense. One of the signatures had to come from Professor Hart.

Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Striver that it was absolutely first-rate, and that he would gladly sign it at the defense. They even shook hands in Hart's book-lined office. Dave noticed that Hart's eyes were bright and trustful, and his bearing paternal.

At the defense, Dave thought that he eloquently summarized Chapter 3 of his dissertation. There were two questions, one from Professor Rodman and one from Dr. Teer; Dave answered both, apparently to everyone's satisfaction. There were no further objections.

Professor Rodman signed. He slid the tome to Teer; she too signed, and then slid it in front of Hart. Hart didn't move.

"Ed?" Rodman said.

Hart still sat motionless. Dave felt slightly dizzy.

"Edward, are you going to sign?"

Later, Hart sat alone in his office, in his big leather chair, underneath his framed PhD diploma.

"Betrayal in Self-Deception" (non-conscious)

Dave Striver loved the university — its ivy-covered clocktowers, its ancient and sturdy brick, and its sun-splashed verdant greens and eager youth. The university, contrary to popular opinion, is far from free of the stark unforgiving trials of the business world: academia has its own tests, and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, to become a doctor, one must pass an oral examination on one's dissertation. This was a test Professor Edward Hart enjoyed giving.

Dave wanted to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless
inscriptions which, together, would certify that he had passed
his defense. One of the signatures had to come from Professor
Hart, and Hart had often said — to others and to himself —
that he was honored to help Dave secure his well-earned
dream.

Well before the defense, Striver gave Hart a penultimate copy
of his thesis. Hart read it and told Dave that it was abso-
lutely first-rate, and that he would gladly sign it at the defense.
They even shook hands in Hart's book-lined office. Dave
noticed that Hart's eyes were bright and trustful, and his bearing
paternal.

At the defense, Dave eloquently summarized Chapter 3 of his
dissertation. There were two questions, one from Professor
Rodman and one from Dr. Teer; Dave answered both, appar-
etently to everyone's satisfaction. There were no further ob-
jections.

Professor Rodman signed. He slid the tome to Teer; she too
signed, and then slid it in front of Hart. Hart didn't move.

"Ed?" Rodman said.

Hart still sat motionless. Dave looked at him.

"Edward, are you going to sign?"

Later, Hart sat alone in his office, in his big leather chair. He
tried to think of ways he could help Dave achieve his goal.

"Self-Betrayal" (no self-deception; conscious version)

Dave Striver loved the university — at least most of the time.
Every now and then, without warning, a wave of ... well, it
was true: a wave of hatred rose up and flowed like molten
blood through every cell in his body. This hatred would be di-
rected at the ghostly gatekeepers. But most of the time Striver
loved — the ivy-covered clocktowers, the ancient and sturdy
brick, and the sun-splashed verdant greens and eager youth
who learned alongside him. He also loved the fact that the
university is free of the stark unforgiving trials of the business
world — only this isn't a fact: academia has its own tests,
and some are as merciless as any in the marketplace. A prime
example is the dissertation defense: to earn the PhD, to be-
come a doctor, one must pass an oral examination on one's
dissertation.

Dave wanted desperately to be a doctor. He had been work-
ing toward this end through six years of graduate school. In
the end, he needed the signatures of three people on the first page of his dissertation, the priceless inscriptions which, together, would certify that he had passed his defense. One of the signatures had to come from Professor Hart.

Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Striver that it was absolutely first-rate, and that he would gladly sign it at the defense. They shook hands in Hart’s book-lined office. Hart’s eyes were bright and trustful, and his bearing paternal.

“See you at 3 p.m. on the tenth, then, Dave!” Hart said.

At the defense, Dave eloquently summarized Chapter 3 of his dissertation. His plan had been to do the same for Chapter 4, and then wrap things up, but now he wasn’t sure. The pallid faces before him seemed suddenly nauseating. What was he doing?

One of these pallid automata had an arm raised.

“What?” Striver snapped.

Striver watched ghosts look at each other. A pause.

Then Professor Teer spoke: “I’m puzzled as to why you prefer not to use the well-known alpha-beta minimax algorithm for your search?”

Why had he thought so earnestly about inane questions like this in the past? Striver said nothing. His nausea grew. Contempt, fiery and uncontrollable, rose up.

“Dave?” Professor Hart prodded, softly.

God, they were pitiful. Pitiful, pallid, and puny.

“Dave, did you hear the question?”

Later, Striver sat alone in his appartment. What in God’s name had he done?

6.9 \textbf{BRUTUS$_1$ on the Web}

Information about BRUTUS, BRUTUS$_1$, and this book can be obtained on-line via the following URL.

\url{http://www.rpi.edu/dept/ppcs/BRUTUS/brutus.html}

This site includes most of the figures and illustrations in this book, as well as the Preface, and, courtesy of Microsoft Agent$^TM$, BRUTUS$_1$
can be called upon to narrate some of the stories seen earlier. Eventually the web site will allow direct interaction with more sophisticated incarnations of Brutus.