BRUTUS and the Narrational Case Against Church's Thesis

– extended abstract –

Selmer Bringsjord
The Minds & Machines Laboratory
Dept. of Phil., Psych. & Cognitive Science
Department of Computer Science
Rensselaer Polytechnic Institute
Troy NY 12180 USA
selmer@rpi.edu • www.rpi.edu/∼brings

Dave Ferrucci
T.J. Watson Research Center
IBM
Yorktown Heights, NY 10598 USA
ferrucci@us.ibm.com

Overview

There was a time, about a decade ago, when one of us (Bringsjord) thought that he would "crack" story generation this way: figure out what makes a story interesting from the standpoint of logic, proceed to formalize interestingness in some logical system, and then code this formalization ... and presto! — he would then have a program that generates interesting stories. This idea proved to be a painful dead end. (Many of you are probably not surprised.)

Whereas sometimes counter-examples to proposed definitions for concepts lead the way to improved definitions that promise to eventuate in *the correct* definition, nothing of the sort happened in dialectic at RPI about what conditions are necessary and sufficient for a story's being interesting. *In fact*, it began to occur to us that perhaps the set of all interesting stories — denoted $\mathcal{S}^I$ — was such that (i) there is no algorithm for deciding whether or not a story is in this set, and (ii) there is an algorithm for shooting down any algorithm proposed to decide $\mathcal{S}^I$. Some readers will immediately realize that properties (i) and (ii) characterize what are known in computability theory as **productive sets**. If the set of all interesting stories is a productive set, and if humans are able to decide, swiftly and routinely, which stories are interesting and which are not, then Church's famous and almost universally affirmed thesis — that what can be effectively computed is co-extensive with what can be algorithmically computed — must be false. We adumbrate and defend this reasoning herein.[1] If the reasoning is correct, the upshot would seem to be that those who would build literarily creative agents in the form of conventional computers and computer programs may at best find engineering "tricks" for giving these agents some mere approximation of what human authors tap when they craft interesting narrative. We conclude this paper with a discussion of one of such trick that we have followed in building BRUTUS.

[1]The full version of this reasoning can be found in our book on BRUTUS (Bringsjord & Ferrucci 1999).

Church's Thesis

At the heart of CT is the notion of an **algorithm**, characterized in traditional fashion by Mendelson as

> an effective and completely specified procedure for solving a whole class of problems. ... An algorithm does not require ingenuity; its application is prescribed in advance and does not depend upon any empirical or random factors. ((Mendelson 1986), p. 225)

An **effectively computable** function is then said to be the computing of a function by an idealized "worker" or "computist" following an algorithm.[2] (Without loss of generality, we can for present purposes view all functions as taking natural numbers into natural numbers; i.e., for some arbitrary f, $f : \mathbf{N} \rightarrow \mathbf{N}$). CT also involves a more formal notion, typically that of a so-called **Turing-computable** function (or, alternatively, and equivalently, that of a μ-**recursive** function, or, ...). Mendelson employs Turing's approach, and Turing machines will be familiar to most readers; we'll follow him: a function $f : \mathbf{N} \rightarrow \mathbf{N}$ is Turing-computable iff there exists a TM M which, starting with n on its tape (perhaps represented by n |s), leaves $f(n)$ on its tape after processing. (The details of the processing are harmlessly left aside for now.) Given this definition, CT amounts to

CT A function is effectively computable if and only if it's Turing-computable.[3]

The Arithmetic Hierarchy

(Cognoscenti be forewarned: Space constraints make a thorough presentation of AH impossible.

[2]In his inaugural writings (independent, by the way, of Turing's), Post (Post 1936) spoke of mindless "workers," humans whose sole job was to slavishly follow explicit, excruciatingly simple instructions. These are "computists" for Turing.

[3]This is often called the Church–*Turing* Thesis for obvious reasons.

What follows is in no way a detailed, accomplished introduction to AH.[4])

Suppose we have some **totally computable** predicate

$$S(P, u, n)$$

iff TM M, running program P on input u, halts in exactly n steps ($= M_P : u \rightarrow_n$ halt). (Throughout AH our machines, architecturally speaking, are always simply TMs.) Predicate S is totally computable in the sense that, given some triple (P, u, n), there is some program $P^\star$ which, running on some TM $M^\star$, can infallibly give us a verdict, **Y** ("yes") or **N** ("no"), for whether or not S is true of this triple. ($P^\star$ could simply instruct $M^\star$ to simulate M for n steps and see what happens.) This implies that $S \in \Sigma_0$, i.e., that S is a member of the starting point in AH, a point composed of totally computable predicates. But now consider the predicate H, defined by

$$H(P, i) \text{ iff } \exists n S(P, i, n).$$

Since the ability to determine, for a pair (P, i), whether or not H is true of it, is equivalent to solving the full halting problem, we know that H is not totally computable. Hence $H \notin \Sigma_0$. However, there is a program which, when asked whether or not some TM M run by P on u halts, will produce **Y** iff $M_P : u \rightarrow$ halt. For this reason H is declared **partially computable**, and hence in Σ_1. To generalize, informally, the syntactic representation of AH is:

Σ_n set of all predicates definable in terms of totally computable predicates using at most n quantifiers, the first of which is *existential*

Π_n set of all predicates definable in terms of totally computable predicates using at most n quantifiers, the first of which is *universal*

Δ_n $\Sigma_n \cap \Pi_n$

We have, based on this scheme, the Arithmetic *Hierarchy* because, where $\subset$ is proper subset,

$$\Sigma_0 \subset \Sigma_1 \subset \Sigma_2 \ldots$$
$$\Pi_0 \subset \Pi_1 \subset \Pi_2 \ldots$$
$$\text{for every } m > 0, \Sigma_m \neq \Pi_m$$
$$\Pi_m \subset \Sigma_{m+1}$$
$$\Sigma_m \subset \Pi_{m+1}$$

It's possible to devise a more procedural view of (at the least the lower end of) AH. Σ_0 and Σ_1 have already been viewed procedurally. How, then, could Π_1, the first genuinely uncomputable stop in AH, be viewed procedurally? Peter Kugel (Kugel 1986) has aptly called Π_1-procedures **non-halting procedures**; here's how they essentially work. (Notice that the term 'procedures' is being used, rather than 'programs'. This is crucial, for reasons discussed in a moment.) Let R be a totally computable predicate; then there is some program P which decides R. Now consider a corresponding predicate $G \in \Pi_1$, viz.,

$$G(x) \text{ iff } \forall y R(x, y).$$

Here's a non-halting procedure P^+ (not a program: we count P^+'s last output (if there is one) as its result) for solving G, in the sense that a **Y** is the result iff Gx:[5]

- Receive x as input
- Immediately print **Y**
- Compute, by repeatedly calling $P, R(x, 1), R(x, 2), \ldots$, looking for a **N**
- If **N** is found, erase **Y** and leave result undefined

Notice that it would be *impossible* to represent this procedure as a *program*. This can be verified by looking at any formal specification of 'program.' For example, in *Mathematical Logic*, by Ebbinghaus et al. (Ebbinghaus, Flum, & Thomas 1984), programs can have only one PRINT instruction. (Typically, procedures are formalized by *adding* to programs the notion of an "oracle.") But the point is that once you tinker with the formalization of 'program,' the world of uncomputability opens up, a paradise that is richer than the small, familiar subhierarchy on the computable side, which runs from finite automata, to push-down automata, to linear-bounded automata, to TMs. Space doesn't permit exhibiting AH in its full glory, but here's one more interesting fact about it, viz., that procedures corresponding to Σ_2, **trial-and-error procedures**, can rather easily solve the full halting problem — as follows. Let a TM — $M^{\star\star}$ — with n^M as input (Gödel number of arbitrary TM M) output **N** immediately, and then let it simulate M. If M halts, $M^{\star\star}$ erases **N**, outputs **Y**, and halts itself. If we count $M^{\star\star}$'s last output as its output, the full halting problem is solved!

The Basic Idea Behind the Narrational Refutation of Church's Thesis

Productive sets have two properties:

P1 They are classically undecidable (= no program can decide such sets).

P2 There is a computable function f from the set of all programs to any such set, a function which, when

[4] For a comprehensive discussion of uncomputability, including the Arithmetic Hierarchy, a good text is (Davis, Sigal, & Weyuker 1994). Bringsjord provides a self-contained introduction to the realm of "super"-computation in (Bringsjord 1998b).

[5] The reader should satisfy himself or herself that the following procedure *does* decide G.

given a candidate program P, yields an element of the set for which P will fail.

Put informally, a set A is productive iff it's not only classically undecidable, but also if any program proposed to decide A can be counterexampled with some element of A. Clearly, if a set A' has these properties, then $A' \notin \Sigma_0$ and $A' \notin \Sigma_1$. If A' falls somewhere in AH, and is effectively decidable, then CT falls. But what could possibly fit the bill? We have become convinced that the set $\mathcal{S}^I$ of all interesting stories provides a perfect fit.

This no doubt catches you a bit off guard. Interesting stories? Well, let us remind you, first, that the view that there are productive sets near at hand is far from unprecedented. Douglas Hofstadter (Hofstadter 1982), for example, holds that the set $\mathcal{A}$ of all As is a productive set. In order to satisfy P1, $\mathcal{A}$ must forever resist attempts to write a program for deciding this set; in order to satisfy P2, there must at minimum always be a way to "stump" a program intended to decide $\mathcal{A}$. That $\mathcal{A}$ satisfies both these conditions isn't all that implausible — especially when one faces up to the unpredictable variability seen in this set. For example, take a look at Figure 1.

Figure 1: Various Letter As *(Taken from (let 1981). A similar array, but this time of all lowercase* As*, can be found on page 413 of (Hofstadter 1995).)*

It's easy enough to build systems capable of generating *un*interesting stories. For example, the world's first artificial story generator, TALE-SPIN (Meehan 1981), did a good job of that. Here, for example, is one of TALE-SPIN's best stories:

"Hunger"

Once upon a time John Bear lived in a cave. John knew that John was in his cave. There was a beehive in a maple tree. Tom Bee knew that the beehive was in the maple tree. Tom was in his beehive. Tom knew that Tom was in his beehive. There was some honey in Tom's beehive. Tom knew that the honey was in Tom's beehive. Tom had the honey. Tom knew that Tom had the honey. There was a nest in a cherry tree. Arthur Bird knew that the nest was in the cherry tree. Arthur was in his nest. Arthur knew that John was in his cave. ...

How are things to be improved? How is one to go about building an agent capable of creating truly interesting stories? It has been the sustained attempt to answer this question, in conjunction with the concept of productivity discussed above, that has persuaded me that CT is indeed false. Let us explain.

First, to ease exposition, let $\mathcal{S}^I$ denote the set of all interesting stories. Now, recall that productive sets must have two properties, P1 and P2; let's take them in turn, in the context of $\mathcal{S}^I$. First, $\mathcal{S}^I$ must be classically undecidable; i.e., there is no program (= TM) which answers the question, for an arbitrary story in $\mathcal{S}$, whether or not it's interesting. Second, there must be some computable function f from the set of all programs to $\mathcal{S}^I$ which, when given as input a program P that purportedly decides $\mathcal{S}^I$, yields an element of $\mathcal{S}^I$ for which P fails. It seems to us that $\mathcal{S}^I$ does have both of these properties — because, in a nutshell, our research group seems to invariably and continuously turn up these two properties "in action." Every time someone suggests an algorithm-sketch for deciding $\mathcal{S}^I$, it's easily shot down by a counterexample consisting of a certain story which is clearly interesting despite the absence in it of those conditions P regards to be necessary for interestingness. (It has been suggested that interesting stories must have inter-character conflict, but monodramas can involve only one character. It has been suggested that interesting stories must embody age-old plot structures, but some interesting stories are interesting precisely because they violate such structures, and so on.)

The situation we have arrived at can be crystallized in deductive form as follows.

$\mathbf{Arg}_3$

	(9)	If $\mathcal{S}^I \in \Sigma_1$ (or $\mathcal{S}^I \in \Sigma_0$), then there exists a procedure P which adapts programs for deciding members of $\mathcal{S}^I$ so as to yield programs for enumerating members of $\mathcal{S}^I$.	
	(10)	There's no procedure P which adapts programs for deciding members of $\mathcal{S}^I$ so as to yield programs for enumerating members of $\mathcal{S}^I$.	
$\therefore$	(11)	$\mathcal{S}^I \notin \Sigma_1$ (or $\mathcal{S}^I \notin \Sigma_0$).	10, 11
	(12)	$\mathcal{S}^I \in$ AH.	
$\therefore$	(13)	$\mathcal{S}^I \in \Pi_1$ (or above in the AH).	disj syll
	(14)	$\mathcal{S}^I$ is effectively decidable.	
$\therefore$	(15)	CT is false.	*reductio*

Clearly, Arg_3 is formally valid. Premise (9) is not only true, but necessarily true, since it's part of the canon of elementary computability theory. What about premise (10)? Well, this is the core idea, the one expressed earlier by Kugel, but transferred now to a different domain: People who can *decide* $\mathcal{S}^I$, that is, people who can decide whether something is an interesting story, can't necessarily *generate* interesting stories. Students in *Autopoeisis*[6] have been a case in point: With little knowledge of, and skill for, creating interesting stories, they can nonetheless recognize such narrative. That is, students who are, by their own admission, egregious creative writers, are nonetheless discriminating critics. They can decide which stories are interesting (which is why they know that the story generators AI has produced so far are nothing to write home about), but *producing* the set of all such stories (including, as it does, such works as not only *King Lear*, but *War and Peace*) is quite another matter. These would be, necessarily, the *same* matter if the set of all interesting stories, $\mathcal{S}^I$, was in either Σ_0 or Σ_1, the algorithmic portion of AH.

But what's the rationale behind the claim that $\mathcal{S}^I$ is effectively decidable? The rationale is simply the brute fact that a normal, well-adjusted human computist can effectively decide S-I. Try it yourself: First, start with the sort of story commonly discussed in AI, for example:

"Shopping"

> Jack was shopping at the supermarket. He picked up some milk from the shelf. He paid for it and left.[7]

Well? Your judgement? Uninteresting,we wager. Now go back to "Hunger," and come up with a judgement for it, if you haven't done so already. Also uninteresting, right? Now render a verdict on the following story.

"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, 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 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's book-lined office. Dave noticed that Hart's'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.

This story is in fact "authored" by BRUTUS. Note that we have placed the term 'author' in scare quotes. Why? The reason is plain and simple, and refers to Lady Lovelace's famous objection to Alan Turing's famous claim that computers would by the turn of the century be thinking things: BRUTUS didn't *originate* this story. He is capable of generating it because two humans spent years figuring

[6] The project that was the precursor to our work on BRUTUS. See, e.g., (Bringsjord 1992a).

[7] From page 592 of (Charniak & McDermott 1985). The story is studied in the context of attempts to resolve pronouns: How do we know who the first occurrence of 'He' refers to in this story? And how do we render the process of resolving the pronoun to Jack as a computational one?

out how to formalize a generative capacity sufficient to produce this and other stories, and they then are able to implement part of this formalization so as to have a computer produce such prose. The engineering method followed here is known as **reverse engineering**.

As to the story itself, you did find it at least somewhat interesting, right?

Now at this point some readers may be thinking: "Now wait a minute. Isn't your position inconsistent? On the one hand you cheerfully opine that 'interesting story' cannot be captured. But on the other you provide an interesting story! – a story that must, if I understand your project, capitalize upon some careful account of interestingness in narrative."

The answer is that we have no such account. Rather, we have engineering tricks.

Engineering Interesting Narrative

Though we believe, on the strength of the foregoing analysis and argument, that interestingness is uncomputable, we are of course once again faced with our engineering challenge: How is BRUTUS to *appear* to have mastery over interestingness on par with human authors? The bulk of the answer to this question is provided in our book (Bringsjord & Ferrucci 1999). For example, therein we try to show that interestingness is promoted if the narrative generated by BRUTUS triggers readerly imaging; and we present a list of heuristics for how to produce the desired reader response. The same can be said for the issue of point of view, which we also discuss in the book: in order for BRUTUS to write interesting stories, he must carry the reader into a landscape of consciousness, and there seem to be certain determinate ways to enable this.

A number of thinkers who have come before us have grappled directly with the problem of capturing the essence of interesting narrative; first and foremost among them probably comes Roger Schank. Schank (Schank 1979) believes that what makes a story interesting can be captured in computable schemes, but in the paper in question he doesn't provide the scheme itself — this, despite the fact that on page 278 of that paper Schank promises: "What I shall attempt to do now is to *define* interestingness ..." (emphasis ours). In the next section of Schank's paper, which is supposed to contain the definition in question, all we find are *ad hoc* strategies relating to interestingness. To motivate these strategies, Schank does present certain stories, most prominently the following two:

Schank Story 1 — Interesting

John was walking down the street eating an ice cream cone. He saw a man walk into the bushes and begin to undress. Soon a crowd had gathered and the police came to investigate. While they were there a giant explosion occurred two blocks away. People came running in their direction screaming that there had been a terrible accident. Many were bleeding and one man had lost an arm. Meanwhile a fire broke out in the park. People said there as a conspiracy afoot because a bomb had been sighted nearby only yesterday. When an epidemic broke out the following week, everyone knew aliens had landed.

Schank Story 2 — Uninteresting

John was walking down the street eating an ice cream cone. He saw a man walk into the park and begin to read. Soon some pigeons had gathered and a boy came to feed them. While they were there a truck drove by a few blocks away. People who came walking towards the park said that it was a diesel truck. Many were hot and one man was tired. Meanwhile the park got really crowded. People said there was a new park being built nearby because a construction crew had been sighted only yesterday. When construction began the following week, everyone knew that the mayor had kep his promise.

What he induces from these two stories is that death and danger are "absolutely interesting," while two "operators," UNEXPECTED EVENTS and PERSONAL RELATEDNESS, can be applied to topics to make them more interesting. Schank points out that this implies that when the subject of death is an argument to PERSONAL RELATEDNESS, the result is interesting. Likewise, a story about unexpected death would be interesting. Schank lists other topics that are "absolute interests," e.g.,

- Power
- Sex
- Money
- Destruction
- Chaos
- Romance
- Disease

This list prompts Schank to say: "So what is interesting according to the above? If someone you knew died from having sex for a lot of money, that would be very interesting by my rules." ((Schank 1979), p. 290)

All this kind of stuff is utterly "seat of the pants." Schank simply presents a few stories, and is then off to the races, isolating certain properties exemplified in these stories. Nowhere here is there anything like a definition; nowhere is there even a *hint* of the sort of formal accounts which we desire; there isn't even inductive evidence for the list of "interests" Schank promotes. There is no denying, of course, that a story about, say, death caused by having sex for money, is at least potentially interesting, but this is just one trick out of innumerable similar ones. Where is it to end? How many tricks are there? To seek tricks for generating interesting narrative in

general fits well with our own modus operandi, and so we are quite at home with what Schank cooks up, and with *how* he does so (the bottom line is that he just picks things out of thin air to get the job done, or least *partially* done!), nothing in his work casts the slightest doubt upon our contention that the set $\mathcal{S}^{\mathcal{I}}$ is uncomputable.[8]

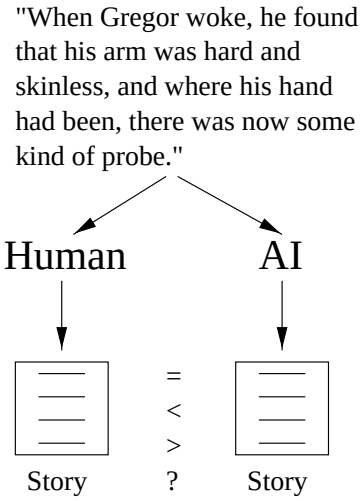

Figure 2: The Short Short Story Game, or S^3G for Short.

One of Our Engineering Tricks

One of our tricks for ensuring (at least a degree of) interestingness is this: Formalize immemorial themes present in belletristic fiction, and have BRUTUS generate stories in accordance with this formalization. We don't believe that BRUTUS *understands* these themes. (We don't believe computers will ever understand anything, because computers will forever have inner lives about as rich as a rock's. See, e.g., (Bringsjord 1992b), (Bringsjord 1999), (Bringsjord 1998a).) But we do believe that if we as engineers can build a system that generates fiction in accordance with these themes, these systems will have a much better chance of *seeming* to be literarily creative. Another way to put this is to say that systems "in command" of immemorial literary themes will have a much better chance of passing the "Short Short Story Game," or S^3G for short (Bringsjord 1998a). The idea here is simple; it is summed up in Figure 2. A human and a computer compete against each other. Both receive one relatively simple sentence, say: "Barnes kept the image to himself, kept the horror locked away as best he could." (For a much better one, see the "loaded" sentence shown in Figure 2.[9]) Both mind and machine must now fashion a short short story (about 500 words) designed to be truly interesting; the more literary virtue, the better. Our goal, then, is to build an artificial author able to compete with first-rate human authors in S^3G, much as Deep Blue went head to head with Kasparov.

An informal version of the definition of betrayal that underlies the story from BRUTUS seen above appears in Figure 3. Our presentation/demo at the NI symposium will focus on this definition and the "trick" that motivates it. We will also discuss the BRUTUS architecture (see Figure 4). However, keep in mind that this extended abstract and any presentation/demo is an excruciatingly compressed version of ideas, formalisms, arguments, sample code, etc. presented in our book (Bringsjord & Ferrucci 1999).

References

Bringsjord, S., and Ferrucci, D. 1999. *AI and Literary Creativity: Inside the Mind of Brutus, A Storytelling Machine*. Mahwah, NJ: Lawrence Erlbaum.

Bringsjord, S. 1992a. Cinewrite: an algorithm-sketch for writing novels cinematically, and two mysteries therein. In *Computers and Writing: State of the Art*. Dordrecht, The Netherlands: Kluwer.

Bringsjord, S. 1992b. *What Robots Can and Can't Be*. Dordrecht, The Netherlands: Kluwer.

[8] Schank isn't the only thinker to tackle the question of what literary interestingness consists in. For example, Wilensky (Wilensky 1983), Kintsch (Kintsch 1980), and Britton (Britton 1983), among others, have all considered the issue. None of the work offered by this trio includes an account of interestingness that succeeds in keeping things within the realm of the computable. For example, Kintsch holds that a story is interesting provided that (a) it evokes assumptions in the minds of the audience, then (b) denies some or all of these assumptions, which in turn causes (c) "some extra-mental operations that engage the cognitive machinery" ((Kintsch 1980), p. 596). Such an account certainly seems to fit standard detective stories well (in fact, Kintsch proudly quotes Hercule Poirot: "It gives one furiously to think"), but why couldn't someone (easily) write an interesting story that violates the (a)-(c) sequence? We leave it to the reader to devise such a story.

[9] The actual opening is as follows:

> As Gregor Samsa awoke one morning from uneasy dreams he found himself transformed in his bed into a gigantic insect. He was lying on his hard, as it were armor-plated, back and when he lifted his head a little he could see a dome-like brown belly divided into stiff arched segments on top of which the bed quilt could hardly keep in position and was about to slide off completely. His numerous legs, which were pitifully thin compared to the rest of his bulk, waved helplessly before his eyes. ((Kafka 1948), p. 67)

Figure 3: Informal Version of Definitin of Betrayal

Def_B 8 Agent s_r betrays agent s_d at t_b iff there exists some state of affairs p and $\exists t_i, t_k, t_j$ $(t_i \leq t_k \leq t_j \leq t_b)$ such that

1 s_d at t_i wants p to occur;

2 s_r believes that s_d wants p to occur;

3′ $(3 \wedge 6') \vee$

6″ s_d wants at t_k that there is no action a which s_r performs in the belief that thereby p will not occur;

4″ there is some action a such that:

4″a s_r performs a at t_b in the belief that thereby p will *not* occur; and

4″b it's not the case that there exists a state of affairs q such that q is believed by s_r to be good for s_d and s_r performs a in the belief that q will not occur;

5′ s_r believes at t_j that s_d believes that there is some action a which s_r will perform in the belief that thereby p *will* occur.

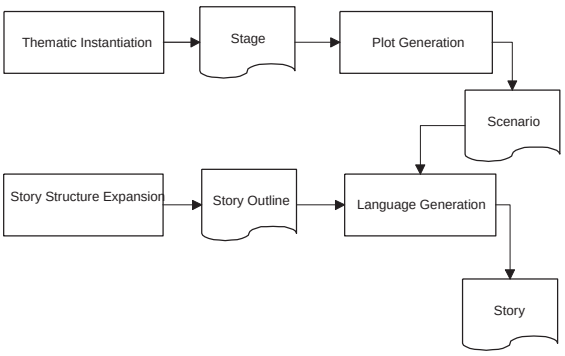

Figure 4: Brutus: A Composite Architecture

Bringsjord, S. 1998a. Chess is too easy. *Technology Review* 101.2:23–28.

Bringsjord, S. 1998b. Philosophy and 'super'-computation. In *The Digital Phoenix*. Oxford, UK: Basil Blackwell. 231–252.

Bringsjord, S. 1999. The zombie attack on the computational conception of mind. *Philosophy and Phenomenological Research* LIX:41–69.

Britton, B. 1983. What makes stories interesting. *Behavioral and Brain Sciences* 6:596–597.

Charniak, E., and McDermott, D. 1985. *Introduction to Artificial Intelligence*. Reading, PA: Addison-Wesley.

Davis, M.; Sigal, R.; and Weyuker, E. 1994. *Computability, Complexity, and Languages*. San Diego, CA: Academic Press.

Ebbinghaus, H. D.; Flum, J.; and Thomas, W. 1984. *Mathematical Logic*. New York, NY: Springer-Verlag.

Hofstadter, D. 1982. Metafont, metamathematics, and metaphysics. *Visible Language* 14(4):309–338.

Hofstadter, D. 1995. *Fluid Concepts and Creative Analogies*. New York, NY: Basic Books.

Kafka, F. 1948. The metamorphosis. In *The Penal Colony*. New York, NY: Schocken Books.

Kintsch, W. 1980. Learning from text, levels of comprehension, or: why anyone would read a story anyway. *Poetics* 9:87–98.

Kugel, P. 1986. Thinking may be more than computing. *Cognition* 18:128–149.

1981. *Graphic Art Materials Reference Manual*. New York, NY: Letraset.

Meehan, J. 1981. Tale-spin. In Schank, R., and Reisbeck, C., eds., *Inside Computer Understanding: Five Programs Plus Miniatures*. Englewood Cliffs, NJ: Lawrence Erlbaum. 197–226.

Mendelson, E. 1986. Second thoughts about church's thesis and mathematical proofs. *Journal of Philosophy* 87.5:225–233.

Post, E. 1936. Finite combinatory processes – formulation 1. *Journal of Philosophy* 1.3:103–105.

Schank, R. 1979. Interestingness: Controlling inferences. *Artificial Intelligence* 12:273–297.

Wilensky, R. 1983. Story grammars versus story points. *Behavioral and Brain Sciences* 6:529–591.