

# On Einstein's Razor: Telesis-Driven Introduction of Complexity into Apparently Sufficiently Non-Complex Linguistic Systems

Quinn Tyler Jackson<sup>1</sup>  
New Westminster, British Columbia  
quinn-j@shaw.ca

---

“Never express yourself more clearly than you are able to think.”—Niels Bohr

“It is wrong to say that a good language is important to good thought, merely; for it is the essence of it.”—Charles Sanders Peirce

## Abstract

*The notion that a linguistic system that is powerful enough to accept any acceptable language but insufficiently complex to meet specific goals or needs is explored. I nominate Chomsky's generative grammar formalism as the least complex formalism required to describe all language, but show how without the addition of further complexity, little can be said about the formalism itself. I then demonstrate how the  $\mathcal{O}(n)$  parsing of pseudoknots, a previously difficult to solve problem, becomes tractable by the more complex  $\mathcal{S}$ -Calculus, and finally close with a falsifiable hypothesis with implications in epistemological complexity.*

**Keywords:** parsing, scientific inquiry, telesis, necessary complexity

## 1 Introduction

I shall begin by disclaiming that this paper, although populated with formalisms, is intended to be formal or scholarly, but rather, is intended to stimulate discussion and thought; indeed, it has modest and informal beginnings. On a discussion list on Yahoo, Clifford Pickover asked the following question (*q.v.* [Pickover]):

A few of you seem to take the position that it is possible or likely that the mind (soul?) does not arise from the brain. I'm almost surprised by this, only because many of you who hold this theory tend to be skeptics and shy away from what I'd call “paranormal” subjects, like the existence of an afterlife or the existence of God.

1. If you do believe that the mind exists beyond the brain, and perhaps the brain is merely the organ by which the soul interfaces to the world, when do you think the mind comes into being and is destroyed?
2. If the brain mediates the sensory input, what exactly do you think the mind *is* if it never had this input? For example, if a person from birth was devoid of all senses, what do you think the mind would be thinking or experiencing? Would a thought have ever gone through such a person's head? If a mind never had a thought, sense, or experience, does it exist at all?

If you believe the mind is some non-material essence, not understandable in terms of physics, aren't you starting to introduce a concept that introduces an additional degree of complexity than what seems like a simpler physical argument?

**Table 1.1—Pickover's Appeal to Occam's Razor**

As it applies to scientific inquiry, Occam's Razor is simple: given two explanations for an observed phenomenon, prefer the simpler. For many problems, this principle holds well. Although the notion that there might be invisible balloons filled with lighter-than-air gas holding up birds in flight, the phe-

---

<sup>1</sup> Society fellow.

nomenon of bird aviation is sufficiently and entirely explained by other models that the introduction of the non-falsifiable notion of “invisible balloons” is not necessary, and ought to be discarded.

This paper focuses on a related concept that follows as a corollary from Occam’s Razor, which I shall call herein *Einstein’s Razor*<sup>2</sup>. The notion that a theory should be as simple as possible (but no simpler) is attributed to Einstein, and thus the name of this new razor.

Einstein’s Razor brings forward the following notion: *even when a simple explanation is theoretically sufficient, it is sometimes insufficient to reach desired goals*. That is, some *needs* or *goals* are not necessarily attainable by the simpler of two or more systems. As relates to this paper, formal linguistic models will be used as a concrete example of this principle.

## 2 Observations

I will now nominate the simplest linguistic model necessary<sup>3</sup> to account for all language and computation.

### 2.1 Chomsky’s Model: The Simplest of Sufficiently Powerful Formal Linguistic Models

With [Chomsky 1956] came the introduction of a formal system that demonstrated that generative grammars are of sufficient power that an infinite variety of linguistic utterances can follow from a finite set of rules. Specifically:

A *generative grammar* is a 4-tuple  $G = (V_N, V_T, S, P)$  where:

- $V_N$  is a finite set of *nonterminal symbols*,
- $V_T$  is a finite set of *terminal symbols*,
- $S \in V_N$  is the distinguished *start symbol* of the grammar, and
- $P$  is a finite set of productions of the form  $X \rightarrow Y$  where:
  - $X \in (V_N \cup V_T)^*$ ,  $V_N (V_N \cup V_N)^*$ , and
  - $X \in (V_N \cup V_T)^*$ .

#### Definition 1—Generative Grammar

The above model is of maximal simplicity to account for *all* legal utterances in *all* human languages, and, in fact, for all computable functions. In order to *prove* the formalism in Definition 1 meets this end, however, *further complexity* must be introduced to the formalism itself by the entities providing the proof. That is to say, if the *need for proof* is introduced, either the formalism itself must directly become more complex, or a meta-model must be introduced, and the introduction of the meta-model increases the complexity of the entire system simply by its introduction, since the proof cannot be seen outside the addition of the new model. The meta-model implies more complexity than is originally needed by the simple start condition.

We have one more option: declare the fact that all languages are accepted by generative grammars an *axiom*, and reduce the whole matter to a matter of faith. This option is highly unsatisfactory, since the very fact that [Chomsky 1956] introduced the formalism implies that the formalism did not exist in that form before that point, so how did such an axiom just *appear* so recently in human history<sup>4</sup>? Are we to believe that someone can come along with a formal model and simply declare that it *just works*?

<sup>2</sup> A quick search of the Internet via Google™ will show that the term “Einstein’s Razor” is certainly *not* due to me.

<sup>3</sup> Simpler Turing-powerful models exist, but they are certainly no less opaque as regards solving the questions posed in this paper.

<sup>4</sup> See §3 for the comment on [Panini]’s 5<sup>th</sup> Century B.C. introduction of formal context-free grammars.

Ultimately, in an attempt to avoid this axiomatic approach, we decide as scientific inquirers to allow the introduction of more complexity, either by introduction into the formalism, or by encasing the original system in the safe cocoon of a meta-model. Meta-models cannot be used to defer complexity; that would be akin to formalized prestidigitation. We *need* to add complexity in order to meet our *goal* of avoiding unnecessary tautology. This *need* is goal driven, and the goal in this case is simply stated: avoid tautology unless absolutely required to resort to it.

## 2.2 Complexity Satisfies Need, Need Implies Goals, Goals Imply Telesis

If all we have is a hammer, we need to know if the problems we wish to solve all can be solved by nails or by screws that will withstand being used as nails. If the problem we have is the provision of a mathematical proof that the formalism of Definition 1 is sufficient to account for all languages, we not only do not have the means to do so, we do not even have the means to prove we do not have the means!

In the case of the definition of formal languages, we can introduce the following restrictions upon the forms of four classes of language (*cf.* [Chomsky 1968]):

### Type 0 Grammars – Unrestricted

Type 0 grammars have no restrictions on their productions, other than that there must be at least one nonterminal on the left-hand side of all productions. The left-hand side can contain many symbols, as can the right. Thus, for each production  $\alpha \rightarrow \beta$ ,  $|\alpha| \geq 0$  and  $|\beta| \geq 1$ .

### Type 1 Grammars - Context-Sensitive

Type 1 grammars add the restriction that the number of symbols on the left-hand side of a production must always be equal to or less than the number of symbols on the right-hand side. At least one production must have a left-hand side with more than one symbol. Thus, for each production  $\alpha \rightarrow \beta$ ,  $|\alpha| \geq 1$  and  $|\beta| \geq 1$  and  $|\alpha| \leq |\beta|$ .

### Type 2 Grammars - Context-Free

Type 2 grammars can have left-hand sides of only one symbol, which must be nonterminal, and right hand sides of terminals or nonterminals. Thus, for each production  $\alpha \rightarrow \beta$ ,  $|\alpha| = 1$  and  $|\beta| \geq 1$ .

### Type 3 Grammars - Regular

Type 3 grammars add the restriction that each right-hand side must be either left or right linear.

A *left-linear* regular grammar has right-hand sides that are all either terminals, or a single nonterminal followed by a terminal. ( $A \rightarrow a$  or  $A \rightarrow Ba$ )

A *right-linear* regular grammar has right-hand sides that are either terminals, or single terminals followed by a single nonterminal. ( $A \rightarrow a$  or  $A \rightarrow aB$ )

A regular grammar may not be both right and left linear; it must be one or the other.

### Definition 2—The Successive Restrictions of Grammar Classes

Have we satisfied a need by doing this bit of dress up? If our goal is to show that some language  $a^n b^m$  is acceptable to a finite state automaton, we indeed have, since by adding complexity to the model, we now have in our possession more than just an axiomatic hammer; we have a finite state automaton, a pushdown automaton, a linear bounded automaton, and an infinite tape Turing Machine. The tool we need to take out of the box is a function of which language we need to better understand.

If what we have in a particular instance is a pushdown automaton, and we wish to know whether or not our tool available tool will be able to find some string of the form  $a^n b^n c^n$ , we can now make that determination absolutely in the negative: a linear bounded automaton is required for that particular task. Adding complexity has facilitated assessing our ability to attain specific linguistic goals.

Is the complexity introduced by Definition 2 sufficient to answer all of our questions about language? The answer to this, of course, is *no*. Although it would be nice if it were so, further complexity must be introduced as our goals move. When do goals move? *When existing complexity is deemed insufficient to satisfy perceived need.*

When Occam's Razor is insufficiently sharp to split hairs, Einstein's Razor is required.

### 2.3 Genetic Secondary Structures: Complex Goals with Simple Answers?

In the field of bioinformatics, pattern matching has encountered several interesting difficult to parse constructions (*cf.* [Searls], [Rivas & Eddy], and [Geigerich & Reeder]). RNA sequences have a small alphabet:  $\Sigma = \{A, U, C, G\}$  (adenine, uracil, guanine, and cytosine). The pairs (A, U) and (C, G) are *complements*, one of the underlying lexemes in RNA pattern matching. Although only RNA is discussed here, what is said of RNA also applies to DNA, despite DNA's different alphabet and complements.

The most basic syntax of RNA secondary structures is found in *loops*, such as:

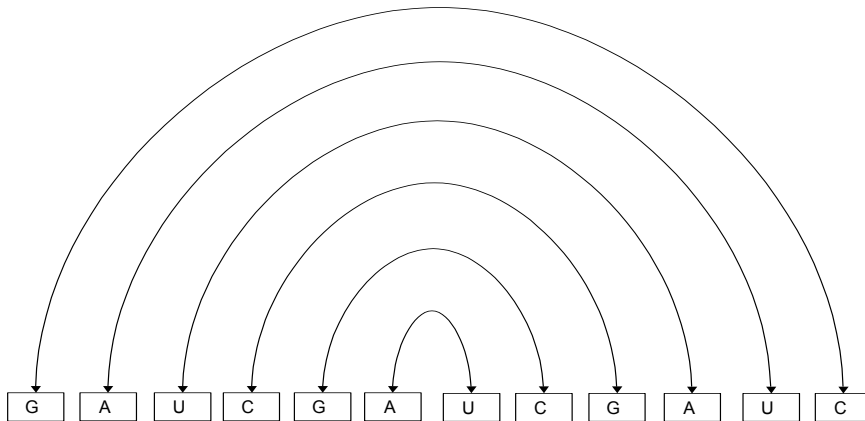


Figure 2.1—A Simple RNA Loop

Loops are easily seen to be a Type 2 language and merit no further discussion here, since their provable recognition does not require an introduction of complexity beyond that found in Definition 2.

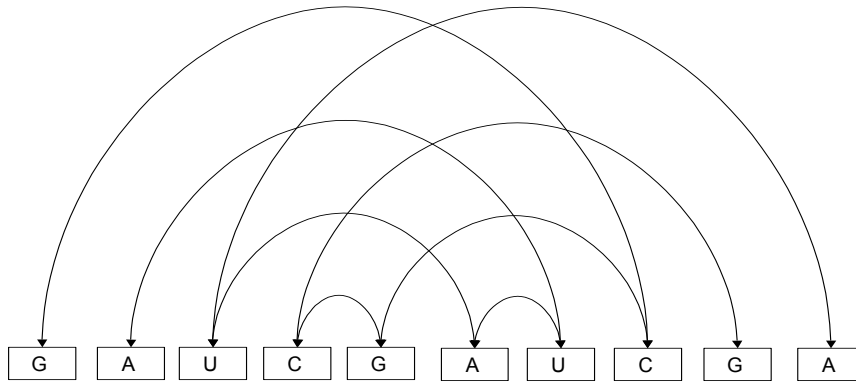
One of the more interesting secondary structures encountered in bioinformatic pattern matching is that of pseudoknots, defined here as:

An RNA pseudoknot is defined as the language  $\mathcal{L}_{\text{knot}} = \{\alpha_1 \Sigma_1 \alpha_2 \Sigma_2 \beta_1 \Sigma_3 \beta_2\}$  where:

- $\alpha_i = \{a, u, c, g\}^+$
- $\Sigma_i = \{a, u, c, g\}^*$  such that:
  - $\Sigma_1 \neq \alpha_1$ ,
  - $\alpha_2, \beta_1 \neq \Sigma_2$ , and
  - $\beta_1, \beta_2 \neq \Sigma_3$ .
- $\beta_i =$  complement of  $\alpha_i$ .

**Definition 3—RNA Pseudoknots**

At first, pseudoknots have the appearance of being simply a special case of cross-agreement, such as found in the language  $\mathcal{L} = \{a^n b^m c^n d^m \mid m, n > 0\}$ . Of special note, however, are the *filler* sequences (denoted  $\Sigma_i$  in Definition 3), which may or may not be present, and the fact that there may or may not be some overlap of the sequences. This configuration can be diagrammed thus:



**Figure 2.2—Pseudoknot Interleave**

This shows us that there is cross-agreement, but because of the potential overlap, and to the correlation relationship found in the complement branches, the Type 2 grammar that would accept the loop of Figure 2.1 is insufficient for the purposes of accepting pseudoknots. Also, in the language  $\mathcal{L} = \{a^n b^m c^n d^m \mid m, n > 0\}$ , it is the lengths of the substrings that are in cross-agreement, not the constituent characters of the substrings themselves.

To date, linguistic models have not allowed researchers to definitively answer the following question: *Can some subset of the class of languages called genetic pseudoknots (at least as they have been defined here) be recognized in  $\mathcal{O}(n)$  time?*

If I just say to you that the answer is *yes*, you will justifiably ask me how I *know* that. If your need to know how I know that is sufficiently strong, you will not accept the response, “I just *know* that.” You will likely demand a proof, if you are any kind of mathematician or scientist at all. If I cannot

Submitted to Progress in Complexity, Information, and Design

formulate a proof based upon existing foundations and the existing level of complexity available to our discourse, you have every right to disregard my declaration.

To date, with formalisms of the level of complexity discussed so far, no one has been able to show the answer I am now suggesting. The hammers available have not been up to the task of this particular nail. The past discernment of scientists and mathematicians permitted further complexity than shown in Definition 1 by allowing the consecutive layers of restrictions already discussed. To answer the question as asked here, I must further introduce *more* complexity, since *more complexity* is just what is needed to give you a *simple* answer to that question. That additional complexity comes in the form of the  $\S$ -Calculus.

## 2.4 The $\S$ -Calculus

First introduced in [Jackson 2000], and later expanded upon in various papers including [Jackson 2002], the  $\S$ -Calculus allows us to arrive at the answer to our question. First, a formal definition of just what the  $\S$ -Calculus is must be presented:

A  $\S$ -grammar is a 7-tuple  $\S = (V_N, V_T, C, S, Q, P, \Phi)$  where:

- $V_N$  is a finite set of *nonterminal variables*,
- $V_T$  is a finite set of *terminal variables*,
- $C$  is a potentially infinite set of sets (called *classes*) of sequences of the form  $\gamma$  where:
  - $\gamma \in (V_N \cup V_T \cup C \cup Q \cup \Phi)^*$ .
- $S \in V_N$  is the distinguished *start symbol* (or *axiom*) of the  $\S$ -grammar,
- $Q \subseteq (V_N \cup C)$  is a potentially infinite set of *predicates* of the forms:
  - $\pi.\Psi(\gamma)^{\sim 0}$  (*free predicate form*) where:
    - $\pi \in (P \cup \Phi_\beta)$ , where  $\Phi_\beta$  is a *name-index* of any member of  $\Phi$  (below),
    - $\theta \in (V_N \cup C)$ , where the option  $\sim$  operator means “not” in the usual sense,
    - $\gamma$  is an  $\gamma$ -*expression* of the form  $(V_N \cup V_T \cup C \cup Q)^*$ .
  - $\{\alpha\}^0$  (*bound predicate form*) where:
    - $\alpha$  is an  $\alpha$ -*expression* (above), and
    - $\theta \in (V_N \cup C)$ .
- $P$  is a set of *productions* of the form  $X \rightarrow Y$  where:
  - $X \in (V_N \cup C)$ , and
  - $Y \in (V_N \cup V_T \cup C \cup Q \cup \Phi)^*$ .
- $\Phi$  is a finite set of  $\phi$ -*expressions* of the form  $\langle \alpha \rangle_\iota$  where:
  - $\alpha$  is an  $\alpha$ -*expression* (above), and
  - $\iota$  is an *index-expression* of the forms  $\bar{\beta}!$  where:
    - $\bar{\phantom{x}}$  is the *depopulation operator* and means “remove from the  $\S$ -grammar,” (the absence of this operator meaning “add to the  $\S$ -grammar”),
    - $!$  is the *locality operator* and means “local to this member,” (the absence of this operator meaning “global to the  $\S$ -grammar”), and
    - $\beta$  is a *name-index* in one of the forms:
      - $\pi.\mathcal{A}$ , where  $\pi \in (P \cup \Phi_\beta)$ , and  $\mathcal{A}$  is a *name*,
      - $(\pi.\mathcal{A})$  which indicates that the last member placed into  $\pi.\mathcal{A}$  is first *dereferenced* and the resulting value is to be treated as defined for  $\pi.\mathcal{A}$  above. (If  $\pi.\mathcal{A}$  is empty,  $(\pi.\mathcal{A})$  dereferences to the literal “ $(\pi.\mathcal{A})$ ” and not  $\lambda$ .)

### Definition 4— $\S$ -Grammars

The scope of the current paper is not to prove to you the proof method itself<sup>5</sup>, but to demonstrate the notion that introduction of complexity will allow me to use empirical methods to prove something that I otherwise would find almost impossible to prove.

### 2.4.1 New Insight into Pseudoknots via New Complexity Introduced in the §-Calculus

“By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems....”—Alfred North Whitehead

The following §-grammar accepts a constrained set of pseudoknots:

S	→ k
k	→ ( $\langle\langle\text{loop}\rangle_{\text{lhs}}\rangle_{\text{x}!} \text{tail} \Psi(x)^{\text{make\_reverse}} \Psi(\text{rev})^{\text{second\_loop}} \Psi(x)^{\text{-double\_loop}} \mid (\Phi(k.\text{rhs})_{-k.\text{rhs}} \Phi(k.\text{lhs})_{-k.\text{lhs}})$ )
double_loop	→ k.lhs $\omega$ knot_rhs guac
knot_rhs	→ k.rhs
make_reverse	→ $\Phi(\lambda)_{\text{x}!} (\langle\text{[gauc]}\rangle_{\text{c}!} \Phi(\text{c x})_{\text{x}!}^+ \Phi(x)_{\text{rev}}^{\uparrow}$
second_loop	→ $\Omega \langle\text{loop}\rangle_{\text{x}!} \text{tail} \Psi(x)^{\text{make\_reverse}} \Phi(\text{rev})_{\text{k.rhs}}^{\text{k.rhs}}$
tail	→ $\text{[gauc]}\{0-32\}$
loop	→ $\langle\text{loop\_lhs}\rangle_{\text{x}!} \Psi(x)^{\text{make\_rhs}} \langle\text{[1-32]}\rangle_{\text{v}!} \text{rhs} \Psi(y)^{\text{guac}}$
loop_lhs	→ $\text{[gauc]}\{6\}$
make_rhs	→ $\Phi(\lambda)_{\text{x}!} (\langle\text{[g]}\Phi(\text{[c] x})_{\text{x}!} \mid (\text{[a]}\Phi(\text{[u] x})_{\text{x}!}) \mid (\text{[c]}\Phi(\text{[g] x})_{\text{x}!}) \mid (\text{[u]}\Phi(\text{[a] x})_{\text{x}!})\rangle^+ \Phi(x)_{\text{rhs}}^{\uparrow}$
guac	→ $\text{[guac]}^+$

**Table 2.2—Pseudoknot §-Grammar**

The pseudoknot §-grammar above<sup>6</sup> achieves its end by decomposing the problem into several tractable sub-problems, in delineated steps:

- accept a *loop* (loop  $\Lambda_1$ ) by:
  - accepting a sequence up to 6 characters in length and constructing the sequence’s complement
  - skipping over everything encountered that is not that constructed complement
- accept a *tail* of up to 32 characters in length (tail  $\tau_1$ )
- reverse both the loop and tail, into the trie  $\rho$ , by means of a local trie and a very simple production
- scan  $\rho$  for another loop followed by tail (loop  $\Lambda_2$ , tail  $\tau_2$ )
- verify that  $|\Lambda_1| + |\Lambda_2| < |\rho|$

**Table 2.3—A Constrained Pseudoknot-accepting Algorithm**

<sup>5</sup> See [Jackson 2002] for this, in part, which shows §-grammars to be Turing-powerful iff the C, Q, and  $\Phi$  sets of the 7-tuple are all non-empty.

<sup>6</sup> First presented in [Jackson 2005], but in A-BNF form, *q.v.* Appendix A here.

In the case of pseudoknots, loops are not defined as tightly as in Figure 2.1, but rather as:

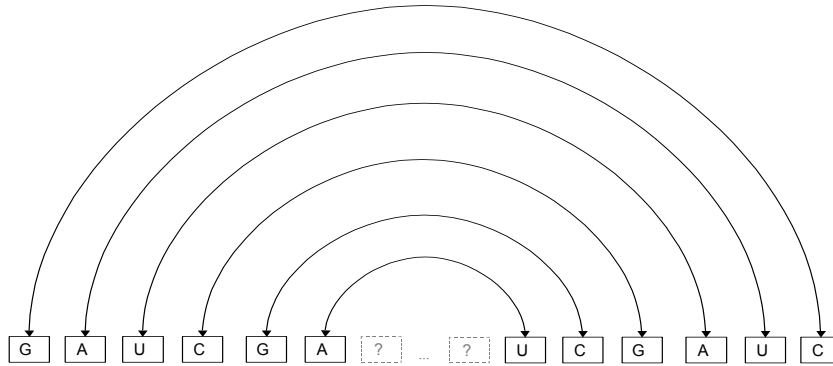


Figure 2.3—A Loosened Loop

Submitted to Progress in Complexity, Information, and Design

In other words, loops in this case must begin and end with sequences at least  $n$  characters long, with  $n$  being 6 for empirical performance tests. This looser specification allows for the *filler* of Definition 3. The *tail* is simply everything that follows the last character of a loop (or, for these tests, up to 32 symbols). Put succinctly: if a string has a loop followed by a tail when parsed *in both directions* (and is not merely two loops in close vicinity of one another that otherwise do not overlap, the constraint enforced by  $|\Lambda_1| + |\Lambda_2| < |\rho|$ ), it is a pseudoknot.

For these tests, the pseudoknot shown in Figure 2.4 below was placed at the end of a string of characters in the alphabet that did not make up a knot, but which contained 2 “near knots” for every 256 characters of prefix filler (near knots satisfying all but the final predicate, which verifies that the knot candidate is not, in fact, simply a *double\_loop*, thereby consuming some computational time):

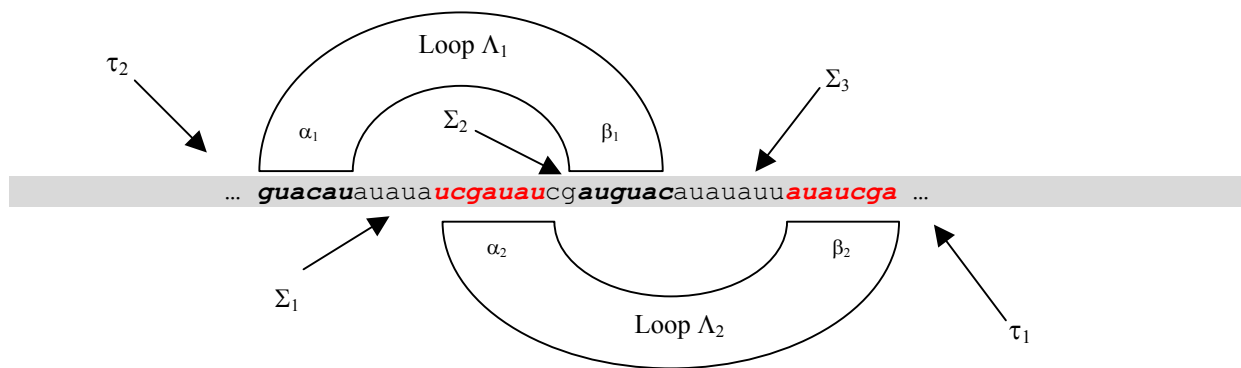


Figure 2.4—Test Pseudoknot Configuration

In Figure 2.4, the tail portion of loop  $\Lambda_1$  is  $\Sigma_3$ , and the tail portion of  $\Lambda_2$  is  $\Sigma_1$ . Filler was also placed at the end of the string. The length of the prefix strings were doubled for each timing run, and the total length of time required to find the pseudoknot in the middle of the entire string (by means of the §-grammar shown in Fig. 7.1) was measured.

Timings were done using the Grammar Forge™ parser generator tool<sup>7</sup>, using the A-BNF grammar shown in Appendix A, on a 2.61 GHz processor Intel machine with 1 GB of RAM, running under

<sup>7</sup> The Grammar Forge™ suite of tools, formerly the property of the present author, is now owned by Thothic Technology Partners, LLC.

Windows XP. The thread of execution used for testing was given real-time priority, in an attempt to reduce the effects of task switching during timing runs. The timing results were as follows:

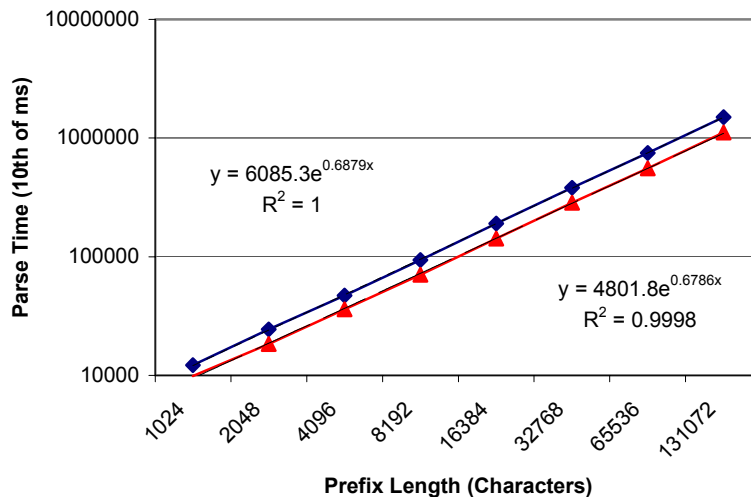


Figure 2.5—Parsing Performance of  $\mathcal{L}_{\text{knot}} = \{\alpha_1 \Sigma_1 \alpha_2 \Sigma_2 \beta_1 \Sigma_3 \beta_2\}$  (Logarithmic Scale)

These results show  $\mathcal{O}(n)$  performance<sup>8</sup>. (The lowermost line is a reference parse of identically long strings without the “near knots,” to be used a reference to determine the impact of introducing false near matches. This performance is also  $\mathcal{O}(n)$ , as expected.)

## 2.5 A Reminder, some Questions, a Hypothesis, a Falsifiable Predication, and a Consequence

### 2.5.1 The Reminder

To date, extant formalisms were not sufficiently complex to predict the existence of mechanisms able to recognize genetic pseudoknots in linear time. Only by adding complexity to the models available was it possible to prove they could exist at all.

It cannot be forgotten, however, that all Turing-powerful formalisms are equivalent. Adding complexity to the simplest formalism (Chomsky’s) has not resulted in a formalism that can *theoretically* accept more than Chomsky’s formalism. That pseudoknots can be modeled by a  $\S$ -grammar means that pseudoknots can be modeled by Chomsky’s system. There exists some Chomsky form grammar that is the equivalent to the  $\S$ -grammar shown in Table 2.2.

All Turing-powerful formalisms are equal in their accepting power, but all algorithms are not equal in their computational complexity even when they provide the same results. This can be shown as follows: looking up factorials in an infinitely large array of integers is complex in space, but constant in time, whereas computing a factorial by iterative multiplication is less complex in space, but more so in time. Both result in the same thing. Table lookup demands very little in the way of proof of correctness compared to iterative multiplication.

<sup>8</sup> Given exponential regression analysis of the data set in the form  $y = ce^{bx}$  with a  $R^2 > 0.98$ , it is inferred that for  $\mathcal{O}(n^d)$ ,  $d \approx b/\ln(2)$ . The constant  $c$  part is discarded in the usual fashion.

### 2.5.2 The Questions

Given that some Chomsky form grammar must exist that accepts pseudoknots, what is that grammar? Given some present or future algorithm effected upon a Chomsky form grammar, without emulation of the  $\S$ -grammar engine (or some other engine, for that matter)<sup>9</sup>, is the Chomsky form algorithm able to accept the same inputs in  $\mathcal{O}(n)$  time?

### 2.5.3 The Hypothesis

If provably no algorithm can be constructed such that Chomsky form grammars can accept genetic pseudoknots in  $\mathcal{O}(n)$  time without first converting Chomsky form grammars into some more complex formalism, two formally equally powerful models do not yield equally powerful results. If this is so, *injected complexity can yield a formalism more suited to a goal or need even when a formalism is sufficient in a general sense to explain a phenomenon.*

### 2.5.4 The Falsifiable Prediction

I predict that some biological process or phenomenon (that can be correlated to a difficult language problem) will some day be found such that the process observed accepts some genetic language in elapsed time more quickly than the provably fastest algorithm applied to the simplest Turing-powerful grammar formalism (Chomsky's formalism), but in keeping with the parse time of some more complex language formalism (such as the  $\S$ -Calculus).

### 2.5.5 The Consequence if the Hypothesis Holds

If this prediction is found to hold, the simplest explanation sufficient to describe all language will be shown to not be the optimal explanation, even though the *optimal and preferred* explanation will be *more complex* and yet will have no more accepting power than the simpler one. Occam's Razor would be shown to be in serious need of sharpening.

### 2.5.6 The Consequence if I am Wrong in my Hypothesis

Nothing ventured, nothing gained; if I am wrong: no harm, no foul.

## 3 Conclusions

I have presented a well-known problem in bioinformatics and linguistics and have shown that only by the addition of complexity into a formalism are certain properties of language provable without resorting to axioms. I have also presented a falsifiable challenge to future researchers in linguistics with implications in epistemological complexity analysis. I shall close by stating this: Chomsky did not know, it is assumed, that his simple formalism would one day be put to the task of accepting pseudoknots, and yet his formalism is known to accept them. One day, finding pseudoknots in RNA or DNA sequences may be the lynchpin of the fate of a planet fighting against a nasty retrovirus; we cannot know. This to say that required complexity changes with need, and declarations of sufficient complexity based upon presupposed need may not be up to the task of the real world.

---

<sup>9</sup> Emulation would be cheating by introducing the complexity through the back door!

It should also not be overlooked that, as pointed out by [Okhotin], context-free grammars as a language formalism were first introduced in the 5<sup>th</sup> Century B.C. by [Panini], before geometric mathematical models were introduced by Thales and Pythagoras. After twenty-five centuries, linguistic models have become increasingly complex, even though languages such as English have become simpler in their constructions (despite their larger vocabularies).

### 3.1 A Far-Fetched Pseudoknot *Gedankenexperiment*

Declaring that something is not needed to explain a phenomenon because our current frame of telic reference does not require it is akin to divination, for it is highly unlikely that today's needs and goals and tomorrow's needs and goals can be anticipated by present models. When Panini formulated context-free grammars to assist in describing the grammar of Sanskrit, in no way did he anticipate computer language compilers. When Chomsky formulated generative grammars, he did not anticipate pseudoknots. Declaring the necessity of an element of a model presupposes comprehension of future utility.

In this *Gedankenexperiment*, let us imagine some retrovirus has appeared from depths of an Antarctic exploratory drill, and all human life on earth is in danger of being exterminated by the virus unless our understanding of pseudoknots is greatly enhanced. A time traveler only able to go backwards into the past has been tasked with convincing Chomsky to introduce sufficient complexity into his work that by the time grammar theory meets bioinformatics, grammar theory would already have sufficient tools to be able to address the issue threatening humankind.

Chomsky would then be in a quandary as a scientist adhering to the notions behind Occam's Razor. Should he introduce complexity unnecessary to language theory to satisfy a goal put forth to him by the time traveler? After all, generative grammar theory as it stood in the late 50's was sufficiently complex to satisfy the needs of various branches of science for some years to come.

Perhaps Chomsky already *did* introduce the necessary complexity to solve the hypothetical plague of which I speak here, or maybe even Panini. I cannot help but wonder if the more complex models of today always existed in theory-space, contained within the apparently less complex models of yesterday. After all, the "injected" complexity does not refute the existing model; it simply assigns it greater *provable utility* against current goals. Is human telenovela injecting complexity, or are new human goals uncovering pre-existing complexity? Perhaps these questions are best left to the realm of poet philosophers such as Omar Khayyam.

And strange to tell, among that Earthen Lot  
Some could articulate, while others not:  
And suddenly one more impatient cried—  
Who is the Potter, pray, and who the Pot?

*The Rubaiyat of Omar Khayyam, Rubai #60 (Fitz Gerald, First Edition)*

## 4 Acknowledgements

I would like to thank César Bravo for bringing genetic pseudoknots to my attention.

## 5 References

- [Chomsky 1968] Noam Chomsky, “Formal Properties of Grammars,” in *Handbook of Mathematical Psychology*, Vol. 2, Wiley, New York, pp. 323-418, 1968.
- [Chomsky 1956] Noam Chomsky, “Three Models for the Description of Language,” *IRE Transactions on Information Theory*, Vol. 2 No. 2, pp. 113-123, 1956.
- [Geigerich & Reeder] Robert Geigerich & Jens Reeder, “From RNA Folding to Thermodynamic Matching, Including Pseudoknots,” *Report 2003-03*, Universität Bielefeld, Germany, 2003.
- [Jackson 2005] Quinn Tyler Jackson, “The  $\S$ -Calculus: Manageable Context-Sensitive Parsing,” (slides presented by César Bravo), Faculty of Engineering Sciences, Computer Science Division, Pontificia Universidad Católica del Perú, Lima, Peru, pp. 1-14, 20 Apr. 2005.
- [Jackson 2002] Quinn Tyler Jackson, “Some Theoretical and Practical Results in Context-Sensitive and Adaptive Parsing,” *Progress in Complexity, Information, and Design*, Vol. 1 No. 4, Dec. 2002.
- [Jackson 2000] Quinn Tyler Jackson, “Adaptive Predicates in Natural Language Parsing,” *Perfection*, Vol. 1 No. 4, Apr. 2000.
- [Okhotin] Alexander Okhotin, *Boolean Grammars: Expressive Power and Algorithms*, Ph.D. dissertation, School of Computing, Queens University, Kingston, Ontario, Aug. 2004.
- [Panini] Panini, *Ashtadhyayi: A Translation of Panini’s Authoritative Treatise on Sanskrit Grammar* (Srasa Chanda, ed. trans.), The Allahbad Press, 1891.
- [Pickover] Clifford Pickover, “The Mind is a Consequence of the Brain,” discussion list message #86261, posted at <http://groups.yahoo.com/CliffordPickover>, 30 Apr. 2005.
- [Rivas & Eddy] Elena Rivas & Sean R. Eddy, “The Language of DNA: A Formal Grammar that Includes Pseudoknots,” *Bioinformatics*, Vol. 16, No. 4, pp. 334-340, 2000.
- [Searls] David B. Searls, “The Language of Genes,” *Nature*, Vol. 420, Nov. 2002.

## APPENDIX A: A-BNF GRAMMAR USED TO OBTAIN P-KNOT RESULTS

```

grammar PseudoKnot
{
  S ::= knot;

  knot ::= ((($x($lhs<-(loop) tail)<x=make reverse><rev=second loop>)
            #psi(x!=double loop)) | (#phi(knot.rhs>-knot.rhs) #phi(knot.lhs>-knot.lhs));

  double_loop ::= knot.lhs ... knot_rhs guac;
  knot_rhs ::= knot.rhs;

  make_reverse ::= #phi(x<="" ($c('[gauc]') #phi(x<=c x))<+> #phi(rev|=x);

  second_loop ::= #scan $x(loop) tail<x=make_reverse> #phi(knot.rhs<-rev);

  tail ::= '[gauc]{0-32}';

  loop ::= $x(loop lhs)<x=make_rhs> $y('{1-32}') rhs<y=guac>;

  loop_lhs ::= '[gauc]{6}';

  make_rhs ::= #phi(x<=""
    ( ('g' #phi(x<="c" x) | ('a' #phi(x<="u" x) |
      ('c' #phi(x<="g" x) | ('u' #phi(x<="a" x)
    )<+> #phi(rhs|=x);

  guac ::= '[guac]+';
};

```