Autopia and *The Truelist:* Language Combined in Two Computer-Generated Books

Nick Montfort

Autopia (Troll Thread, 2016) and *The Truelist* (Counterpath, 2017) are computer-generated literary books. I reported at ELO 2014 on two of my text-generating "novel machines" (Montfort 2014). The two projects discussed in this paper are about novel-size, but are different sorts of projects. *Autopia's* text consists of headline-style sentences made entirely of the singular and plural names of cars. This project manifests not only as a print-on-demand book from a post-digital publisher, but also as a web project and a gallery installation. *The Truelist*'s 140 pages of verse are available in offset printed book form and also as a complete studio recording; additionally, anyone is welcome to run the even simpler and shorter program to generate the exact text. The main component of each line of *The Truelist* is a solid compound (or kenning, or conceptual blend). Neither program is interactive in the usual sense, but they are both short, use no external data or libraries, and are explicitly licensed as free software, inviting people to explore or revise them as programmers. *Autopia* and *The Truelist* both produce straightforward combinations of a limited set of linguistic units. They differ, however, in several ways: certainly in register, but also in that one of them, *Autopia,* is meant to be an ultimately illegible flood of micronarratives, the other, *The Truelist,* I hope will welcome a complete reading by some dedicated and imaginative individuals.

Small and Stand-Alone Programs

The Python version of *Autopia* is 185 lines long, while *The Truelist* program, in Python 2, fits on one line printer page: 66 lines. This accommodates the program's license, docstring, acknowledgments, meaningful variable and function names, and a few comments. The JavaScript version of *Autopia* does incorporate speech synthesis, which involves more than 2MB of code, although the sound is seldom turned on during gallery exhibition. These programs are stand-alone: No external data, from local or online sources, is used.

Origins of Autopia

I did not set out to elevate nor to parody the words that compose the text of *Autopia,* all of which are the names of products. I simply wanted to see what those names have to say for themselves.

I was staying in the Los Angeles area for a while when I began writing down and categorizing the model names of cars, restricting myself to names that read as properly-spelled dictionary words. (I later included makes of cars, such as Ford and Dodge.) I found it compelling that automobiles were often named after animals, but that certain types of animals were prevalent: fast ones (cougar, lynx, rabbit); horses, of the sort that once served as transport but have been displaced by automobiles (bronco, mustang, pinto); strong ones (eagle, ram). Cars have been named for fantastic birds, too: the thunderbird, the firebird, the phoenix. Several cars are named after native peoples of the present-day United States: Dakota, Navajo, Cherokee. There are also different lines of work represented, some of which are from pioneer society (explorer, pathfinder, tracker) and some of which are more recent or particularly distinguished professions (aviator, diplomat).

After starting to record these familiar names, based on cars I saw around me, I began to sift them into categories that, while obvious, are seldom discussed explicitly. To begin with, I recorded some automobile names in a notebook, having no idea about how I (or a computer program) might eventually use them. I realized after a while that these names could serve to narrate brief but perhaps interesting stories. Many car names clearly function as both verbs and nouns (e.g., Focus, Caravan, Eclipse) and could combine with other names to make brief sentences that could, in some cases, declare incidents of interest or even read as newspaper headlines from particular points in U.S. history.

Perhaps I was encouraged to bring these names up against each other, and to keep them in their lanes, because I went running several mornings in Anaheim, right alongside Disneyland. Just inside the park was the linear bumper-car attraction "Autopia," one of the original rides.

As I developed the car names I was collecting into components of a text-generating system, I found that the generated sentences could speak in a wide range of ways. The outputs were able to suggest upper-class activities (NEW YORKER GOLFS), offer mathematical results (OPTIMA FIT MATRIX AXIOM), and even relate to contemporary issues such as immigration (AMIGOS FORD RIO).

Continuing my work on the project showed me that language could combine in some amusing ways, but also, at times, profound and resonant ones. This particularly pleased me, as I enjoy making work that has playful aspects as well as a serious point. There is something funny, at least in the sense of peculiar, about automobile names combining to make the sentence "EXPLORERS RAM DAKOTAS." One's reception of this sentence might change when being reminded of the Dakota War of 1862, after which hundreds of the Dakota people were tried for capital crimes. Without having an attorney and without any witnesses being called, more than 300 were found guilty and sentenced to be hanged. Some convictions were handed out after less than five minutes in front of the judge. President Abraham Lincoln (after whom a whole line of automobiles is named) commuted the sentences of most of these people, but 38 were nevertheless executed in the largest mass execution in United States history. There is an unpleasant historical truth behind "EXPLORERS RAM DAKOTAS."

How the Traffic Looks

I originally developed *Autopia* in Python for output in a terminal window. This version produces lines with at most one headline-like sequence in them. On the Web, where *Autopia* takes the form of a JavaScript program in a Web page, the text progresses left (on the top) or right (on the bottom), as if the flow of traffic.

The final Python version, the Web version (along with gallery installations using it, in which visitors are invited to linger over the lanes of text), and the print-on-demand book all maintain a similar plain and monospaced typographical presentation. The terminal window may seem like a view onto the past, but as McLuhan said, we look at the present through a rear-view mirror. Instead of dressing *Autopia* up to look like a standard, image- and video-heavy website of the mid-2010s, or even choosing to set it in either elaborate or slick-looking type, I have chosen to let the view from the terminal be reflected in all of *Autopia*'s manifestations. The website sports twelve "lanes" of traffic in each direction. While there is no freeway or highway that has twelve adjacent, main lanes in each direction, there are points on Interstate 10 where the total number lanes — including those on access roads, along with the main lanes and high-occupancy vehicle lanes — do indeed number twenty-four.

```
--------------------------------------------------------------------------------
------------NAVIGATORS VENTURE--------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------AZURE GRAND CHEROKEE FOCUSES-----------------
----------------------------PHANTOM AMIGO PROBES BRATS--------------------------
--------------------------------------------------------------------------------
-----------------------------------------------------CABALLERO CARAVANS---------
--------------------------------------------------------------REBEL CARAVANS----
-----------------------------------------RIVIERA AVALANCHES---------------------
-------------------------------SIENNA DAKOTA DODGES CIVIC SCIMITARS-------------
------------------------------PHANTOM JIMMY LASERS WRANGLERS--------------------
--------------------------------------------GRAND CHEROKEE VENTURES-------------
-------------------------------------------------------DIPLOMAT GOLFS-----------
--------------------------------------------------------------------------------
----CONTINENTAL VOYAGER LASERS BEETLE-------------------------------------------
------------------------------------------PREMIER COMANCHES RAM COUGARS---------
```

From the Troll Thread book Autopia *— the top of the first page of the main section. This presents the output of the Python version of the program in a monospace font, the same way this output appears in a terminal window. The 256-page book could only be read in its entirety by a performance artist or very dedicated graduate student.*

```
================================================================================
-NOBLE CHALLENGER ESCORTS PREFECT---------------------GRAND CHEROKEES RENDEZVOUS
------TROOPERS INTRIGUE DEFENDER--------------------------NEW YORKER RALLIES----
ETS---------------------VIPER DARTS------------------ENVOYS RALLY---------------
MPERIAL EXPLORERS----------------------------------WINDSTARS NOVA---------------
S INTRIGUE BRAT--------------CAVALIER MARQUIS ESTEEMS CHAMPS--------------------
----------------------------NEW YORKER DODGES SUPERB JAVELIN----------------SMAR
-------------------COMANCHE FOCUSES-------------------------HOMBRES RENDEZVOUS--
-----RANGERS RENDEZVOUS-----------SONATA EXPRESSES RAIDERS-----------PHANTOM DEF
-----------------------CONTINENTAL WRANGLER DODGES SUBURBAN EDGE----------------
---------------------COOPERS ESCORT NAVIGATOR------------------IMPERIAL RAIDER E
--PHOENIXES ESCAPE----------------------SUPERB BRONCO RAMS CHALLENGER-----------
SES-----------CABALLERO SPRINTS-----------LANCERS COUGAR ROADMASTERS------------
================================================================================
================================================================================
PHOENIXES ESCAPE----------------------CHARGER RAMS CIVIC SCIONS-----------------
------------------AZURE CONQUESTS RAMPAGE--------------------SWIFT VILLAGERS PRO
RUS ASPIRES---------------------------SMART RAIDERS ESCORT SUPERB DASHERS-------
OUNTAINEER---------------NEON PREFECTS DODGE LASER------------------------VILLAG
----JIMMYS ESTEEM TRACKERS---------------------------------------ROADMASTER GOLF
FOCUSES PHANTOM BERETTA----------------------------------RABBIT BOLTS-----------
NDEZVOUS----------------------------NEON UPLANDER ACCLAIMS INTREPID PROTÉGÉ-----
-------SMART NEW YORKER ASPIRES-----------------------------REGAL PACERS FOCUS P
-------WRANGLERS CARAVAN-------------------REGAL PROWLERS SPARK CIVIC ENCORE----
---VIPERS ASPIRE-----------COMANCHES STORM PHANTOM RIVIERA---------------IMPERIA
URE STARLETS PROBE INTREPID VILLAGER---------------NAVIGATORS DODGE ARROW-------
-STARLET ACCLAIMS HOMBRES---------------------CIVIC COOPER RAMS JIMMY-----------
================================================================================
....................................................................book..py..nm
```

The Web version of Autopia *has a very similar visual appearance, although the texts move right-to-left on the top and left-to-right on the bottom as sentences are also produced as generated speech.*

The Form and Function of Autopia

Autopia is a distributional literary work, with each of the headline-sized micro-narratives that it produces is a sample from a distribution. The manifestation of the project in a print-on-demand book is composed of many such samples. The 248 pages of output plus frontmatter and code show that fundamentally, *Autopia* is a program that can generate text without limit.

Autopia does not function as if sampling from a uniform distribution; it draws from one that is weighted. Thus, some sentences are more likely to be produced than others, just as some cars are more common than others.

The specific way in which language is generated in *Autopia* is via a semantic grammar. The program has rules for combining automobile names in ways that are sensitive to the meaning of these names. For instance, there is the grammar rule “ANIMATE Dart_s.” The “_s” ending functions so that if the ANIMATE slot is filled with something plural, DART will be produced; otherwise, DARTS is generated. The semantic part, though, is the use of ANIMATE, rather than a part-of-speech category such as NOUN, in the rule. This expresses that the program is considering a semantic, not just a syntactical, category: only animate (that is, living or independently moving) things can dart. Animate things can be natural or fantastical creatures, as the list of them reveals:

```
u'ANIMATE' : [u'Beetle_s', u'Bronco_s', u'Charger_s', u'Colt_s',
u'Conquest_s', u'Diablo_s', u'Eagle_s', u'Falcon_s', u'Firebird_s',
u'Fox_es', u'Gremlin_s', u'Hornet_s', u'Impala_s', u'Kitten_s',
u'Lynx_es', u'Mustang_s', u'Phoenix_es', u'Pinto_s', u'Rabbit_s',
u'Ram_s', u'Robin_s', u'Shadow_s', u'Silhouette_s', u'Skyhawk_s',
u'Skylark_s', u'Spider_s', u'Spirit_s', u'Sunbird_s', u'Taurus_es',
u'Thunderbird_s', u'Titan_s', u'Viper_s']
```

While the narratives produced by *Autopia* are animated by corporate naming and computational assembly, they are not beyond human comprehension. People can choose to understand the production of these narratives completely — as I, of course, do — and can use the system as the basis for their own work. While machine learning and artificial intelligence interventions can produce compelling results, some bedazzle and obfuscate. I intend to show in my projects, including *Autopia*, that interesting computational manipulation of language can be done with systems that are simple and comprehensible. The automobile naming process may be more mysterious and opaque, but I would suggest that systems such as *Autopia*, as well as more conventional types of research and writing, can help us understand this non-human naming process, and its engagement with culture, as well.

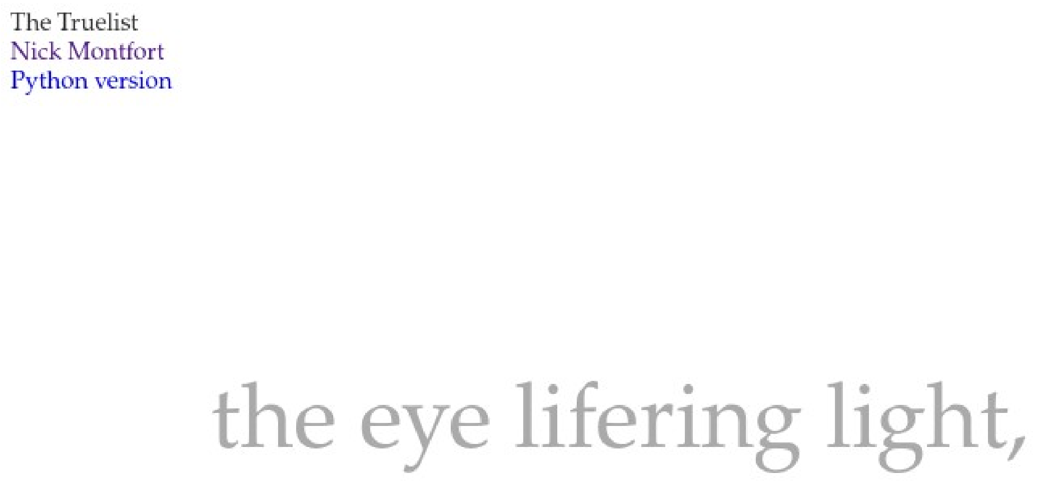
Composing The Truelist

I taught a class called "Small Poetry Machines" for a summer session of the School for Poetic Computation (SFPC), a session that took place July 27—August 9, 2015 at the New York City community gallery Babycastles, then in its second-floor location on the north side of West 14th Street in Manhattan. Instead of delving into online APIs or even some of the powerful but complicated language systems one can download and install, we explored small, self-contained text generators, with reference to more than two dozen examples at that point. A core example for the class was compound word generation, which I had done for instance in my short and snappy "Upstart," a system that combines the first part and the last part of actual technology company names to continually produce made-up technology company names: Redberry, Youbot, Kickhat, Facestarter, Blackbook, and so on. (This computational poem is at https://nickm.com/poems/upstart.html, and an updated version, "Re-Upstart," appears at https://nickm.com/poems/re-upstart.html.) The only constraint was that one of the source names would never be produced.

While we had abundant examples, and were working mainly Python, I decided to devise a program in JavaScript as well as Python that would produce compound words, and specifically solid compounds (with no space or hyphen between parts), in a straightforward and not particularly clever way. It would use many of the most common English words that *participate* in compounds—in other words, that form either the beginning or ending of compound words. Having developed a list of such words with a little offhand and exploratory programming, I created both JavaScript and Python text generators that kicked things off with "Now they saw the [word1+word2]," and continued with a litany of phrases, sometimes simply "the [word1+word2]," and sometimes a slightly elaborated version. I am not the only one whose classroom examples are the starting point for computer-generated literary books; Allison Parrish, a fellow teacher at this SFPC session, did some of the initial work toward her book *Articulations,* also published by Counterpath (2018), when teaching at Fordham University (Heflin 2020).

This earliest version of *The Truelist* already incorporated some of the more detailed rules I ended up using to combine words. For instance, what might be called tautonyms, or reduplicative words, such as “fishfish” or “backback,” were not allowed. Another selection would be made if the system selected the same beginning and end word. In the first version, if “men” was selected for the second part of the compound world, it would be expanded into “men and women,” and similarly, if “women” was chosen it would be expanded into “women and men.” This applied only to the plural form of these words, and only in this position. At the same time, “king” was included as one of the words and “queen,” which participates in far fewer compounds, was not. While all of this was not exactly the same in the final version of *The Truelist,* it is also a system that is “egalitarian” in at least one way while also acknowledging of patriarchy. When the first part of the compound ended with the same letter as began the next part, what would otherwise be a solid compound would be interrupted by a hyphen.

I would make some changes after this early draft—for instance, I developed separate lists of “pre” (first word) and “post” (second word) nouns. This first sketch, however, was done quickly.


The Truelist
Nick Montfort
Python version

the eye lifering light,

A screenshot from the unreleased first JavaScript version of The Truelist, *August 2015. This line is one of the most elaborated forms, although lines can be longer, because “and” is sometimes prefixed to them.*

While I wrote this early draft in August 2015, actually one JavaScript and one Python version, including the short, all-permissive free/libre/open-source software license that I always do, I was not completely pleased with the result and did not choose to put this version of *The Truelist* online. I found much of the language powerful, but the framework seemed to me to be that of a very uninteresting shaggy dog story of sorts, providing no reason to keep or stop reading. In the JavaScript version, a new phrase appeared in the middle of the browser window every three and one thirds of a second. This automatic updating worked in other cases, but didn't serve the imagination as I hoped it would in this one.

To explain how I found a way past this, eventually completing the book over the next year, it's necessary to discuss two other things that a compound word is, why exhaustion can be more interesting than sampling, and, before we get to those issues, what it means to be true.

True as a Bicycle Wheel

The Truelist is obviously a list or catalog. But in what way was it supposed to be true? Certainly not in the sense of not being false or not being fictional. It is meant to be generative and to prompt imagination. The work is true in the way a wall, or a wheel, is true: It is aligned, it follows a principle. However useless we might find Marcel Duchamp's bicycle wheel, plugged into a stool, it is nevertheless true if it does not wobble when spun. *The Truelist* needed to be true in this way.

Solid Compounds, Kennings, and Conceptual Blends

The output of the program, as eventually formulated, consists of four-line stanzas, with each line based around a compound word, some quite unusual, some more conventional. These words can be understood linguistically as solid compounds, which are common in some languages, not only English but also German, Dutch, Norwegian, and Russian. They appear only in exceptional cases in others, such as French and Spanish, even though nouns can be juxtaposed in other ways in such languages.

Poetically, some compounds have a history as *kennings,* words that evoke metaphorical systems. They are a staple of Old Germanic verse, and are usually riddle-like rather than being outright descriptions. These are not always, but often, solid compounds. One of the most famous in English occurs in *Beowulf:* "bānhūs," that is, "bonehouse," meaning the body. While some scholars in the 20th Century distinguished kennings from metaphors, the contemporary understanding of metaphor as conceptual (underlying, not restricted to language) and of poetic metaphor as related to everyday metaphorical thought has made it easy to understand kennings as a specific type of compressed language that conveys metaphor (Holland 2005).

To understand that "bānhūs" has a some cognitive foundation and is not mere language play, consider conceptual blending, initially developed by Gilles Fauconnier and Mark Turner. A famous example is an intricate and well-structured boat race blend that pits a historical and present-day boat (Turner 1993). But a blend can be expressed even in a single word, as with the difference between a "boathouse" (where the blended space has the personified boat as a resident of the house) and a “houseboat” (in which there is a human rider/inhabitant of the vehicle/structure) (Goguen and Harrell 1993).

Understanding blends, or making new ones, involves active cognitive work, as conceptual blends have structure and combine concepts in partial ways. The development of novel ones has been explored in the digital poetry system GRIOT and other projects by its developer (Harrell 2013). Consider how some entailments of a conceptual blend seem clearer, or more optimal, than others, based on how we understand the structure of conceptual combination. One could threaten “I shall knock down your bonehouse and leave you to wander forever,” but it would be much less sensible to ask “does your bonehouse have a large south-facing window?” That a house sits on a particular site with a particular orientation, and that it has glazing, is not activated in this blend as I understand it, while this blend does use the idea that a house can be stronger or weaker, and that it provides shelter for a soul or spirit.

Here I’ve offered only a tiny peek into the history of kennings and the ideas behind conceptual blending. Nevertheless, with these in mind, consider the first line from the first quatrain of the 140-page generated text: “Now they saw the foothills,” ending, as all but the last line of each section does, with a comma. “Foothills” is a solid compound, a perfectly typical dictionary world. Although few would describe it as a kenning, it projects a conceptual metaphor. The hills are the “feet” of a mountain range, which must be considered to be a “body” for the metaphor to work. Like bodily things, both the hills and the mountains are physical and form a mass. This is a metaphor we would certainly skip right over in typical reading, but there it is.

The second line is very brief — “and the airking,” — and introduces a word that is unlikely to occur in any dictionary. What is an “airking”? Perhaps a fantasy king given dominion over the air, or an eagle or other flying apex predator, or perhaps a modern-day monarch who has no real power but only makes announcements “on air.” It could mean several things, but not *anything.* It does not indicate a sheet metal screw or a persimmon. Readers can make something of this word if they bring their own cultural and individual histories to the challenge of this second line. If they match their imaginations to the language that is offered here and throughout the book, something (yet not anything) will result.

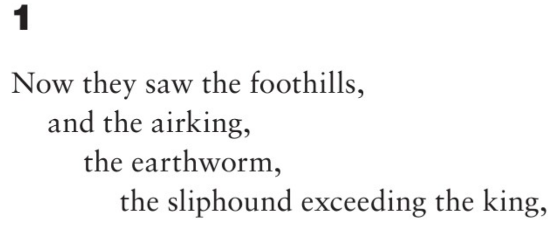
To form interesting solid compounds, the common words selected needed to resonate in combination, inviting metaphorical association and consideration.

Exhaustive and Deterministic Combinations

There were a final two insights in composing *The Truelist.* The program needed to operate exhaustively and should be deterministic.

Exhaustion meant that rather than drawing a sample from a distribution of compound words, continuing the process for however long one wishes, every possible combination of first word and last word would be produced. Each solid compound, kenning, or blend occurs in exactly one line. This principle means it makes no sense to have even one more or line or to omit one. It explains why one should read all of it rather than a few lines. *The Truelist* says everything there is to say about the combination of these words into larger words. Exhaustion is a property of other artworks and poetry: Brion Gysin went through every permutation of words in some of his permutation poems, while John F. Simon, Jr., developed *Every Icon* (1997), a digital artwork which eventually presents every possible 32 pixel by 32 pixel monochrome image — except that the heat death of the universe will occur long before the program finishes running.

Determinism means that although there is code to perform a sort of "mixing" that is somewhat random-like, the same exact text results each time the program is run. That too is part of the program being true to a principle, like their being one correct arrangement of books (according to some organizational principle) in a library.



1

Now they saw the foothills,
    and the airking,
        the earthworm,
            the sliphound exceeding the king,

The first quatrain of The Truelist *as typeset in the Counterpath book. The program produces quatrains with similarly staggered lines.*

After understanding that *The Truelist* needed to be "true" in these two ways, most of my year was spent generating and reading many different mixings of the text — different orderings of the 2300 lines based around the same number of compound words. Begin with several obscure words and the text would fall flat; the same was true if the beginning of the poem consisted of many conventional words. There needed to be oscillation: the common "foothills" at first, then the strange "airking," then back to the conventional "earthworm," and then on to something like "sliphound" … The programming and meta-writing project became a shuffling and reading project, one that occupied most of the time between August 2015 and August 2016, when the manuscript was finally complete. I made only minor changes to which words were generated during that time.

Forks in These Roads

As programs, *Autopia* (whether in JavaScript or Python) and *The Truelist* are non-interactive, taking no input while they run. These systems are open to some types of interaction. One can study the code and can choose to interact with it by making changes; at least one critic has tweeted (@ugly_feelings, May 28, 2020) that he has done this with *The Truelist* (Klobucar 2019).

I don't know of "remixes" or "forks" of *Autopia* or *The Truelist* that have been released. There are many modifications of other simpler text generators of mine, such as "Taroko Gorge," with several modified versions collected at https://collection.eliterature.org/3/collection-taroko.html. That poetry generator, while often riffed upon, is a more conventional work computationally and poetically. The more complex systems of *Autopia* and *The Truelist* may never be as productive of remixes, but in addition to being generators of literary language, I mean them to invite interaction in the form of code reading and programming.

Acknowledgments

This paper incorporates some text about *Autopia* that I drafted for a catalog essay for the ICIDS 2018 Art Exhbition, Trinity College, Dublin, December 5–8, 2018. As of May 2020, the catalog has not yet been produced.

References

Goguen, Joseph A. and D. Fox Harrell. "Style as a Choice of Blending Principles." *Style and Meaning in Language, Art Music and Design,* 2004: 49–56.

Harrell, D. Fox. *Phantasmal Media: An Approach to Imagination, Computation, and Expression*. MIT Press, 2013.

Heflin, Judy. "AI-Generated Literature and the Vectorized Word." SM thesis, MIT, 2020.

Holland, Gary. "Kennings, Metaphors, and Semantic Formulae in Norse Dróttkvætt." *Arkiv för Nordisk Filologi* 120, 2005: 123–147.

Klobucar, Andrew. "Vagueness Machines: Linguistic Indeterminacies, Digital Objects, and the 'Qubit' Poem." In O'Sullivan, J., ed., *Electronic Literature Organization Conference & Media Arts Festival (ELO2019), Programme and Book of Abstracts,* University College Cork, Ireland. July 15–17, 2019: 89.

Montfort, Nick. "New Novel Machines: *Nanowatt* and *World Clock."* Trope Tank Technical Report TROPE-14-01, 2014.

Turner, Mark. *The Literary Mind: The Origins of Thought and Language*. Oxford University Press, 1996.