Adopting Tractarian logic to improve semantic software

In constructing symbolic logic, Frege, Peano and Russell always had their eye on its application to mathematics alone, and they never gave any thought to the representation of real states of affairs. (Wittgenstein's conversation with Waismann, 1929)

Abstract

Semantic software products, whose purpose is to process information based on a formal understanding of the meaning of that information, are grounded in logic; overwhelmingly First Order Logic (FOL). In this sense, logic provides the "abstract" material technology by which semantic software is engineered. In the same way that tangible materials enable -and limit- what can be constructed, FOL enables a huge amount of modern software engineering, but also limits progress in domains where background knowledge needs to interact with new information. In our paper, we focus on one such domain, which plays a ubiquitous role in semantic software applications: translating information from a natural form (e.g., token sequences for natural languages, data bases or spreadsheets) into an explicitly logical form (e.g., FOL). And we outline the real world problems that occur in these kind of semantic software applications owing to specific characteristics of the FOL upon which they are based. We then describe how Wittgenstein in the Tractatus and his lectures from the early 1930s advocates logical approaches relevant to semantic software that differ in significant ways from what became absorbed into consensus FOL. In particular, we argue that Wittgenstein's views on logic provide practical guidance for the mapping of sentences into putative propositions, for differentiating classically behaved propositions from other wff, and for the role of logic in natural language processing. Finally, we describe two distinct semantic engineering efforts that are explicitly based on a Tractarian approach.

Introduction

Semantic software processes information based on some formal understanding of the meaning of that information. Semantic software typically provides interpretation, representation and/or reasoning capabilities. It includes (1) Data/knowledge-bases which represent and support querying over an internal canonical form[1] and (2) natural language processing 'NLP' which converts text into an internal canonical form that expresses facts, rules and definitions.[2]

Semantic software is explicitly grounded in logic, overwhelmingly First Order Logic 'FOL' (Barwise/Etchemendy 1999). For example, Relational databases, which are the predominant form of data management for organizations around the world, are implementations of Codd's Relational Model (Codd 1970), which itself was explicitly grounded in FOL. NLP also uses FOL as its predominant target representation (Collins 1999). In this sense, logic provides the 'abstract' material technology from which semantic software products are engineered.

Software applications where the semantics are pre-defined (e.g., banking and e-commerce applications), form the information backbone of our global economy. They are incredibly reliable. FOL has proven to be an incredibly successful underlying abstract material technology for these sorts of software applications.

However, for other domains where the semantics are hard to predefine, or where the amount of background knowledge required for the software to perform correctly is high, what we call "knowledge-intensive domains", FOL-based software exhibits real world problems. Sometimes the problem is that the software generates incorrect inferences based on the information entered into the system. This happens, for example, when incomplete or corrupted assertions are entered into a database and the database needs to infer facts about the whole of the data entered. Sometimes the problem is that the software is unable to accept information in a particular form (e.g., normal adult level writing) because it cannot interpret the information, even though it may have the internal representations required to understand it. Our goal is to show that these semantic software problems (or at least critical aspects of them) can be traced to specific characteristics of the FOL upon which they are constructed; and that by modifying the logic upon which we build semantic software, by improving our abstract material technology (i.e., our logic) we can engineer software that can operate more successfully in knowledge intensive domains.

The remainder of this essay is organized as follows. In section I, we describe how semantic software for reasoning over large scale database applications and for natural language processing exhibit problems that can be traced to their reliance on FOL. In section II, we describe how Wittgenstein in the Tractatus and his lectures from the early 1930s advocated logical approaches that, to our reading,

[1] For example, Oracle (2016), Teradata (2016), DB2 (2016), CYC (2016), and OWL (2016).
[2] For example, OpenNLP (2016), Stanford parser (2016), Berkley parser (2016).

provided an alternative approach for building semantic software. Finally, in section III, we show how research in semantic software is generating software that behaves (without any explicit attributions) more like what was advocated by Wittgenstein than by FOL. And we provide examples drawn largely from our own research and development into semantic software that is based on Wittgenstein's alternative approach (see also Shavel 1990), which shows promise, and results for resolving the real world information problems described in section I.

Section I. Problems with semantic software

Problem 1: The logical form required for representing and reasoning over missing and inapplicable (i.e., messy) data.

With their basis in FOL, RDBs assume that all wff are evaluable; that they denote propositions. Consider, for example[3], the World Bank (as an exemplar large organization) which maintains one of the world's largest databases of economic, environmental and social data (i.e., assertions) organized by country and time where a single assertion might look like

Wff#1: `82 = life expectancy(France, 2015).`

For the purposes of logic, these assertions form the world of data against which wff are evaluated. For example, the wff `60<= Life expectancy (France, 2015)` would evaluate to True against the world data.[4] In addition to evaluating against specific country-year level data, wff also need to be evaluated across aggregates of countries or years. In other words, it is often necessary across a collection of countries and/or years to know the ratio of those for which some wff, (e.g., life expectancy <= 60) is true versus false. The problem is that many of the country- and time-specific wff are not immediately evaluable; either because the data is missing (e.g., caloric intake data or a predicate constraint on caloric intake data such as "Is < 2000 calories per day" is missing for a sub-Saharan country for a particular year) or the predicate is not applicable to the argument (e.g., agricultural wages are not applicable to a city state like Hong Kong regardless of the year).

Consider processing a World Bank logical formula, Formula 1, such as:

[3] This example is drawn from the author's personal experience as an aggregation specialist at the World Bank.
[4] The wff could also appear in a more traditional form as "82(life expectancy, France, 2015)".

Formula 1:
```
IF Count (Countries whose per capita agricultural wages increased
this year over last year)
/ Count (Countries whose per capita agricultural wages did not increase this
year over last year) < 1
THEN Include per capita agricultural wages in Annual Board Review
ELSE Do Not include
```

The expression inside each of the two count expressions (e.g., countries whose per capita agricultural wages increased this year over last year) denotes a wff to be evaluated over a collection of country-times. Each of the two outer “count” expressions denote the count of inner wff evaluations that evaluated to true. If there are 250 countries, each country has, in theory, a value for its per capita agricultural wages for each year. Between any two years, the change in wages can be calculated. The numerator in the Formula 1 is the count of true wff; the denominator is the count of true negations of the wff in the numerator. Since the general form of the logical Formula 1 is

```
IF Count(P) / Count(!P) > 1

THEN DO X
```

the sum of the numerator count and the denominator count should equal the sum of country-times. So long as every country has a per capita agricultural wage for this year and last year, the formula can compute without a problem. But what is the logically correct inference to make if the data for per capita agricultural wages (i.e., predicate) for some country, e.g., Sierra Leone, is missing? Neither the wff

Wff#2:
```
Greater than( (per capita agricultural_wages(Sierra Leone, This
year)) , ( per capita agricultural_wages(Sierra Leone, Last year))
```

or the wff

Wff#3: `Less than or equal to ( (per capita agricultural_wages(Sierra Leone, This year)) , (per capita agricultural_wages(Sierra Leone, Last year))`

can evaluate if Sierra Leone's per capita agricultural wages are missing. Sierra Leone would no longer be counted either in the numerator or denominator of the aggregate formula. Yet we know that Sierra Leone has an agricultural sector and that that sector has an aggregate wage; so it would appear that the country should belong in one of the two groups. If the formula must calculate (e.g., because the World Bank directors are meeting and agricultural wages will either be or not be on the agenda – so delaying calculation till after Sierra Leone has reported on its agricultural wages is tantamount to putting Sierra Leone in the denominator), it would appear that some inferred value needs to be assigned to Sierra Leone so that it is counted –either in the numerator or the denominator.

On the other hand, what is the logically correct inference to make if the data/predicate for some country, say, Monaco, do not exist? Not because any predicates are missing, but rather because the country has no agricultural sector. Unlike the situation with missing data, when the predicates are not applicable, it would appear that the country should not be counted in either the numerator or denominator. For what could it mean to infer a value where it is known that none exists?

Consider a regional development manager who has the responsibility of increasing agricultural wages across countries in her region of responsibility. And imagine that the regional manager upon annual review will be promoted if the ratio of countries under her jurisdiction that have increasing agricultural wages to those that do not is greater than or equal to one. Else she will be fired. In other words the truth evaluation of

Wff#4: `Greater than(Count(countries where 'agricultural wages increased'), Count(countries where 'agricultural wages did not increase')),`

determines whether the manager is promoted or fired. If there are 20 countries under her jurisdiction and all but one has reported its agricultural wages. And the wff#2 evaluates to true for 9 countries and to False for 10 countries, the way the 20th country is handled can make the difference between the wff#4 being true or false. For example, if the individual wff for the 20th country is removed from the aggregate calculus, wff#4 will evaluate to False and the manager will be fired. But if the assertion underlying the wff for the 20th country is assigned an inferred value and that inferred value, based on past performance evaluates to True (for wff#2) then wff#4 will evaluate to True and the manager will receive a promotion.

Since errors can be produced in predicate formulas such as wff#4 that predicate over collections of individual wff either when the individual wff are incorrectly assumed to be or to not be evaluable,[5] or when software treats in a homogeneous manner all wff that fail to meet propositional constraints, evaluability of wff would appear to be a logical issue that has ramifications for the design of semantic software.

The central question that this empirical problem raises for logic is whether there is a logical distinction between a wff and a proposition and if so whether there is a logical need to differentially reason over wff that do not meet the constraints of being a proposition. For example, do wff#2 and #3 always constitute propositions? Even when the underlying data is missing or inapplicable? If so, can the problem of representing and reasoning over this messy data be solved by adopting some form of Quinian regimentation scheme or an appropriate multi-valued logic, perhaps a 4 valued one?[6] If not, what if any differential reasoning should occur over the wff that do not constitute propositions? The answers to these questions would determine how collections of wff are reasoned over. As they touch on the nature of a proposition, they would appear to constitute a part of logic's core technology.

Problem 2: the logical form required for representing the semantics of natural language

From the automated reservation systems at airlines to the i-phone's Siri, natural language processing 'NLP' is critical semantic software that is increasingly relied upon in our wired world. NLP works by converting sequences of surface language symbols into a canonical form, typically FOL, over which reasoning can take place (Domingos et al., 2006). The overwhelming majority of NLP software perform a lexical parse prior to extracting semantics prior to capturing logical assertions (Russell 2003). Lexical parses extract parts of speech such as nouns, verbs, adjectives etc.. These in turn need

[5] In the commercial world, (drawn from the author's personal experiences as an expert witness in such lawsuits) companies have sued software vendors because their software produced these kinds of wrong inferences.
[6] In the 1990s, there was considerable debate over whether Relational Databases should adopt a multi-valued logic in an effort to reason more consistently with messy data. Though Codd adopted a 4 valued logic in the Relational Model V2 (Codd 1990), prominent members of the database community (Date 2000 & McGoveran 1993/1994] were adamant that databases remain wedded to classical FOL. The debate was not resolved within the database community.

to be converted into semantic types (e.g., objects, processes, space, properties, time,) which then get assembled into FOL assertions (e.g., some object is asserted to have some properties) called semantic representations. The semantic representations are then reasoned with in conjunction with the entity's storehouse of knowledge.

The problem with this sequence of processing, however, is that lexical parsing is not semantically neutral (Croft 1991). For example, a noun followed by a verb (in the active form) often denotes an object realizing a process (e.g., "Grace walks"). And a noun followed by an adjective often denotes an object having some attribute ("The book is blue"). So lexical parsing seems to pick up much of the same information as high level semantic analysis. But lexical information is factored very differently (e.g., a lexical parse tree vs. a semantic graph). And the correspondence of lexical items to semantic ones is also not straightforward. For example, nouns (e.g., "happiness") can also denote feelings; and verbs (e.g., "is") can denote relations. There is thus an impedance mismatch between the structure of the information output from lexical analysis and the structure of the information required to reason with in conjunction with a knowledgebase. Not surprisingly, the impedance mismatch between lexical and semantic parsing has led more recently to the abandonment of full lexical parses in favor of light parsing (e.g., extracting dependency relationships) as an initial step prior to a semantic parse (Kwiatkowski et al., 2013), which is still followed by a logical representation.

The movement towards light lexical parses implies that not all lexical information is essential for the production of formal/logical representations.[7] But surely some lexical information must be necessary. This is because pure semantic parsing (i.e., the associating of words with semantic types) would not pick up the assertoric structure of a sequence of tokens (i.e., what is asserted of what – the grammatical subject and predicate), which is required to determine whether the sequence even constitutes a wff and if so, what is being asserted. For example, given the sentence "Jessie's brother used to live in the house next to the big red house", a purely semantic analysis would identify the semantic types found in the sentence viz. (Person-family relation-temporal negation-spatial relation-size-color-building). But the identified semantic types, alone, would not represent the assertoric relations that bind them into an assertion (e.g., that size and color are predicated of building; that a person realizes a living process in a space occupied by a building; that a building may exist in a spatial relationship to another building etc..).

[7] We fully acknowledge that some lexical surface features (e.g., the fact that in English adjectives may precede a noun whereas in French the noun comes first or that in English "A verb B" denotes an active form while "A was verb by B" denotes the opposite, passive form) are required to map between token sequences and an internal formal representation. The question we are addressing is the nature of the internal lexical grammar.

The ascription of assertoric relations to a sequence of tokens would appear to be an area of overlap between lexical and logical analysis. After all, is that not what stating "f(a)" implies? Could it be that logical grammar should play (or be acknowledged that it does already play) a role in the process of extracting wff from sequences of tokens? Could it be that logical analysis (i.e., the relating of tokens according to a logical grammar) should not follow semantic analysis but rather should precede or co-occur with it in some way? If so, this would suggest a significant departure from traditional NLP architectures. Would it improve performance?[8] The answers to these questions could redefine the role that logic plays in the construction of meaning (as a semantic/logical representation). And consequently, the logical form for representing ordinary language in a way that can be formally reasoned over. The answers to these questions could also help refine the boundaries between logical, lexical and semantic analysis (and what might constitute their formal disciplines or domains or enquiry).

Section II: Logic revisited; Wittgenstein reconsidered

In this section, we highlight the major differences between Wittgenstein and FOL as regards the two central questions that arose in section I, namely (1) the difference between a wff and a proposition as it relates to the logical form required to represent missing and inapplicable data and (2) the relationship between logical, lexical and semantic information for the purpose of natural language processing. In the process, we introduce a logical model (qua core technology) based on our reading of Wittgenstein's approach that addresses and postulates answers to both of these central questions.

1. Messy data: the difference between a wff and a proposition

Whereas FOL and its non-classical offshoots treat all kinds of declarative expressions (regardless whether called "wff", "sentences" or "propositions") as of a single kind with the question being which truth-values are supported by these declarative expressions,[9] Tractarian logic distinguishes wff/sentences from propositions (5.4733)[10] and suggests a process for differentiating wff that are

[8] As judged in various benchmarks such as TAC (2015) or Semeval (2015).

[9] Classical True/False or some non-classical variant: True/False/Indeterminate; True/False/True AND False; True/False/True OR False etc…

[10] Whereas "Frege asserts that a legitimately constructed sentence must have a sense", "legitimately constructed sentences are [rather] *possible* propositions." (5.4733)

propositions from those that are not. Legitimate propositions in this scheme maintain their bivalence (4.023).

The complex fact, illustrated in Figure 1, of a girl running can be used to illustrate Wittgenstein's views.

Figure 1

In (5.5423), Wittgenstein states that two individuals might see different propositions or logical *arrangements* in the same fact/complex.[11] Since the complex fact is constituted by two objects: "Grace" and "run", we can form two distinct wff with the same "f(a)" form: "Runs(Grace)" and "Grace(Runs)".[12] Consider now the following Table 1:

	The specific fact (verbally represented)	Is wff#1 `Runs(Grace)` a proposition?	`Runs(Grace)`	Is wff#2 `Grace(Runs)` a proposition?	`Grace(Runs)`
1	**Grace runs**	Yes	True	Yes	True
2	**Grace** walks	Yes	False	No	Unevaluable argument
3	Daniel **runs**	No	Unevaluable argument	Yes	False
4	Daniel walks	No	Unevaluable argument	No	Unevaluable argument

Table 1 specifies (1) the two alternate wff (labeled as such because we do not wish to straightforwardly assume that they will evaluate); (2) a series of facts beginning with the fact pictured

[11] The complexity of the fact, what Wittgenstein calls "logical multiplicity" (4.04) governs the number of distinct propositions that can be generated for a single fact/complex.
[12] The presence of a name ("Grace") where a predicate generally occurs, and the presence of a predicate ("run") where a name generally occurs in "Grace(Runs)" will be dealt with soon.

above – in row 1 of the table; (3) a collection of related facts in rows 2-4; (4) in the cells that form the intersection of a fact and a wff, the evaluation of the wff when applied to the fact; (5) in the cell to the left of each wff evaluation whether the WFF constitutes, for that fact, a proposition. Fact elements that match wff elements (in either wff) are highlighted in **boldface red**.

Studying Table 1, notice that the facts specified in rows 2-4 diverge from the fact in row 1 that made both wff (qua propositions) true. Note also that the two wff differ in terms of the conditions by which their status changes from True to False or from True to unevaluable. Thus in row 2, the fact varies by one element from the original fact: Grace is walking instead of running. For wff#1, this makes the wff (qua proposition) false. This is because the argument, namely "Grace", still matches the fact. But the factual action is *walking* not *running* so the wff which asserts that the action is *running* is false. In contrast for wff#2, its argument no longer matches an element of the fact. So the predicate "Grace" never gets to evaluate and the wff as a whole is considered unevaluable, because the argument fails to evaluate. This is because from wff#2, we cannot find an instance of *running* from where we can evaluate the predicate that identifies who is running. In row 3 the original fact is altered by a different element namely the identity of the person doing the running. So here wff#1 becomes unevaluable because its argument, "Grace", cannot match any element of the fact; whereas wff#2, which had been unevaluable relative to fact 2, is now simply a false proposition. In row 4, the original fact is completely altered. And neither of the two wff are evaluable.

Since the same facts that make some wff false make others unevaluable and vice versa, we conclude that the full meaning of a wff can only be captured by a sequenced process of evaluation that (1) differentiates between wff that do and do not behave as classical propositions, and (2) maintains bivalency for classically behaved propositions. We informally define the sequenced evaluation of a wff as follows. A sequenced evaluation is a process that take as input a wff and yields either a classical proposition which is guaranteed to evaluate to True XOR False or a non-classical proposition.

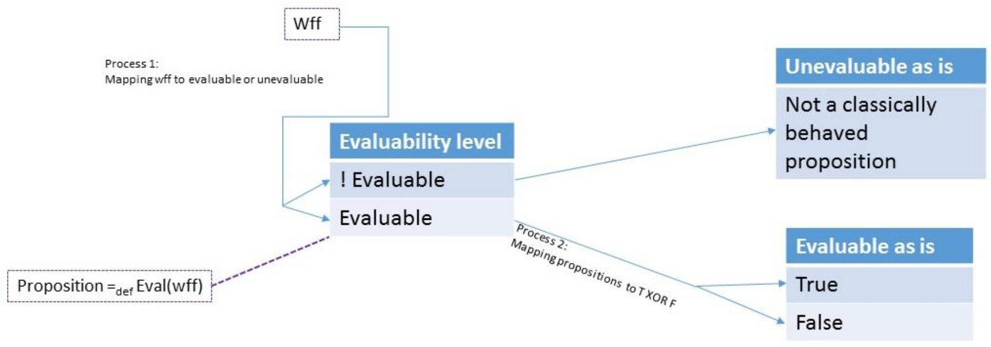

Wff
Process 1:
Mapping wff to evaluable or unevaluable
Evaluability level
! Evaluable
Evaluable
Proposition $=_{def}$ Eval(wff)
Unevaluable as is
Not a classically behaved proposition
Process 2:
Mapping propositions to T XOR F
Evaluable as is
True
False

Figure 2 A sequenced evaluation

The process, labelled #1 in figure 2 above, begins with a sequence of tokens constituting a wff. It then tries to map some of the wff tokens to argument variables/dimensions and to values of those variables/dimensions. If successful on both accounts, (i.e., the argument successfully evaluates), it then tries to map the remainder of the wff tokens to a predicate variable/dimension and determine whether the predicate variable/dimension applies to the argument. (Wittgenstein 1980, p. 7) If this succeeds, the wff is checked for the presence of a predicate value. If the process is successful (i.e., it is determined that a predicate value exists), the wff is determined to be an evaluable proposition and process #2, the evaluation of the proposition resulting in T XOR F may begin. If any of the tests fail, the wff -as is- does not constitute an evaluable proposition.

We can now fully specify the truth-conditions for a wff of the form "f(a)":

"f(a)" is true *iff* at the end of its sequenced evaluation the relationship between the argument and predicate matches the relationship between their corresponding objects;[13] **otherwise "f(a)" is false or unevaluable.**

It is important to distinguish false and unevaluable;[14] otherwise we would not be able to maintain a logical distinction between a fact that negates an assertion (e.g., the facts in rows 2 and 3) and a fact that has no impact on the truth or falsity of the assertion (the fact in row 4). Stated alternatively, if we are given a proposition "Grace runs" or "runs(Grace)", it is clear how the fact of Grace walking would negate the assertion: "No, Grace isn't running. She is walking". But it is not clear why an independent fact such as David walking should negate the assertion "runs(Grace)": "No Grace isn't running. David is walking". If the fact that David is walking is allowed to negate "Grace is running", why not the fact that the moon is made of blue cheese or 7 + 2 = 9? This is exactly what would happen if one does not distinguish false from unevaluable. The door is then open to allow any possible fact to negate an assertion.

One could argue that Wittgenstein is doing no more than suggesting a three valued logic, e.g., Lukasiewicz (1970) & Kleene (1952): a wff is either true or false or unevaluable. But this would violate Wittgenstein's strong commitment to Bivalence (4.023). For him, true and false depend on the prior establishment of P being a genuine proposition (i.e. evaluable).

[13] In semantic technologies, objects amount to values stored in a memory device.

[14] Accordingly, Wittgenstein claims that a sentence that mentions a **complex** that does not exist is false, as opposed to nonsensical (3.24); and that a sentence that mentions an **object** that does not exist is nonsensical (6.53).

2. Natural language representation: the relationship between logical, lexical and semantic analysis

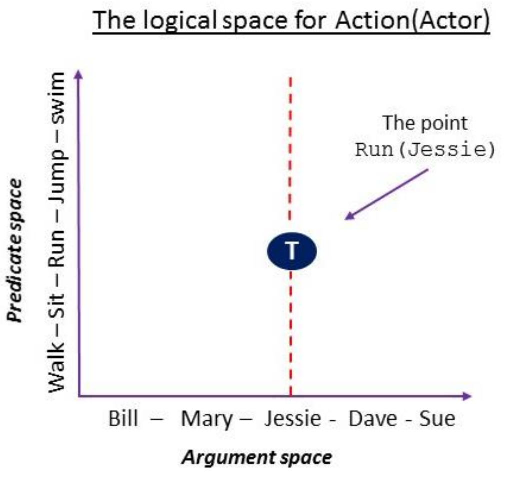
In (3.33) Wittgenstein writes that "in logical syntax the meaning of a sign should never play a role". This entails that for Wittgenstein, logical syntax (or grammar) is orthogonal to semantics, in the sense that syntactic categories can be arbitrarily correlated with meanings.

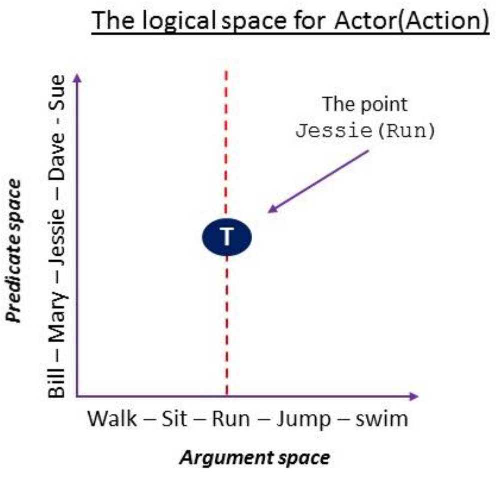
As we saw, a single fact of a girl running could be the veridical source for two distinct propositions: "Run(Grace)" and "Grace(Run)". The two propositions are each comprised of an actor (Grace) and an action (running). Though it may not be standard to treat the action as the argument and the actor as the predicate, no logical rules are violated by doing so. In fact, each proposition represents an answer to a legitimately distinct question. "Runs(Grace)" answers the question "What is Grace doing?". "Grace(Runs)" answers the question "Who is running?" The fact that *Grace* and *runs* can each figure in the argument or predicate role of a proposition, means that neither arguments nor predicates refer in isolation to anything specific in the world (e.g., arguments need not denote some general form of object).[15] Only when the arguments and predicates have been associated with semantic variables (e.g., specific actors and actions) do the ensuing propositions refer. Thus, there are no semantic implications to the fact that "Grace" is treated as an argument in a proposition. For in another she may be a predicate. The propositional roles of argument and predicate are thus orthogonal to whatever semantic types are (extra-logically) postulated as comprising the world.[16]

For Wittgenstein there is a process of substitution that resembles, for example, the projection between a musical score and a realized symphony (4.014). If that substitution is successful, the possible proposition becomes an actual proposition that "determines a place in logical space" (2.202, 3.4), which he likened to geometrical space (3.411). Except that instead of purely numeric variables, propositional variables can also be defined by stipulating values (3.316), in the sense that red, brown and blue could be the stipulated possible values of a color variable. Using the same example of Grace running as described above, the dual figures below illustrate, per our reading, what Wittgenstein means when he refers to propositions as points in logical space whose possible values (e.g., for actors and actions) are stipulated.

[15] See sections 3.314 – 3.317: "And the *only* thing essential to the stipulation is *that it is merely a description of symbols and states nothing about what is signified*." (3.317)

[16]"Language disguises thought. So much so, that from the outward form of the clothing it is impossible to infer the form of the thought beneath it, because the outward form of the clothing is not designed to reveal the form of the body, but for entirely different purposes." (4.002)


The logical space for Action(Actor)
Predicate space
Walk – Sit – Run – Jump – swim
The point
Run(Jessie)
T
Bill – Mary – Jessie - Dave - Sue
Argument space


The logical space for Actor(Action)
Predicate space
Bill – Mary – Jessie – Dave - Sue
The point
Jessie(Run)
T
Walk – Sit – Run – Jump – swim
Argument space

Though both figures appear to show the same logical point, the potential values for each of the two points -what Wittgenstein termed "the grammar" in his notebooks (1980)- is different. In the first figure, the argument space consists of persons and the predicate space consists of actions. The grammatical space within which "run(Grace)" is a point consists of all the points along the red dotted line. Namely, assertions of other actions that Grace might be performing. In contrast with the second figure, the argument space consists of actions and the predicate space consists of persons. And the grammatical space within which "Grace(runs)" is a point consists of all the points along the red dotted line; namely the assertions of other persons who might be running. So, we believe that Wittgenstein's view of propositions as points in logical space highlights not only the orthogonal relationship between logical "f(a)" structuring and semantic variables/dimensions but also the treatment of logical variables as computable elements. To use modern software terminology, Wittgenstein saw logic as a strongly typed system for representing and reasoning with information regardless of origin (e.g., natural language, or more structured forms).

Section III: Improving semantic software by using Tractarian 'logic/technology'

As relates to the development of semantic software, we summarize our understanding of Wittgenstein's key insights as follows:

1. The only way to empirically distinguish wff that are bivalent propositions from those wff that are not is to execute a process that tries to construct an evaluable proposition from the wff. If the process

succeeds, the wff is a classically behaved proposition with a guaranteed bi-valent truth-value. If the process does not succeed, the wff is not a classically behaved proposition.

2. Treat the parts of a proposition i.e., the ‘a’ role and the ‘f’ role as independent of any semantic content: *a*’s are not especially associated with objects; nor are *f*’s especially associated with properties. The logical roles are such that any semantic variables/dimensions can be inserted into any logical role.

Although there are numerous areas where we would characterize the direction of semantic software development as Tractarian,[17] the focus of this section is on our own research and development efforts in semantic software because they are grounded by design in Tractarian logic and therefore we can comment in more detail about the relation between the underlying logic as technology and the engineering of semantic software.

Semantic software aimed at solving problem 1: The logical form required for representing and reasoning over missing and inapplicable (i.e., messy) data.

In an effort to provide for flexible computational reasoning over large amounts of potentially messy data, and in response to the problems identified in section I part 1, we developed a multidimensional database (called FreeThink)[18] which introduced a formula language wherein wff that were determined during the process of formula computation to be unevaluable could be repaired within, or eliminated from the calculus. Our overriding goal was to enable the development of aggregate formula that represented reality as truthfully as possible.

While our calculus was consistent with our reading of the Tractatus’ process of sequenced evaluations (Author), we needed to extend that process to differentially reason over what were logically speaking two different kinds of unevaluability as identified in section I, according to the following two axioms:

[17] Including SQL (Date 2000) which allows any semantic variables/dimensions to play the role of key (argument) or non key (predicate) and Combinatory Categorical Grammars (Steedman 2000) that map words to logical roles before specifying their detailed semantics.
[18] FreeThink was developed by Power Thinking Tools. The company was sold to Computer Corporation of America in 1995.

(1) If a wff 'N' is unevaluable because the predicate data is missing but is guaranteed to exist, a value must be *substituted in* for the missing value for use in all formulas that reason over N.

(2) If a wff 'N' is unevaluable because the predicate data does not apply to the argument and is thereby guaranteed to not exist, N must be *dropped out* from any calculus that reasons over N.

Figure 3, below, extends figure 2's sequenced evaluation process to include reasoning over different kinds of unevaluable wff per the two axioms above.

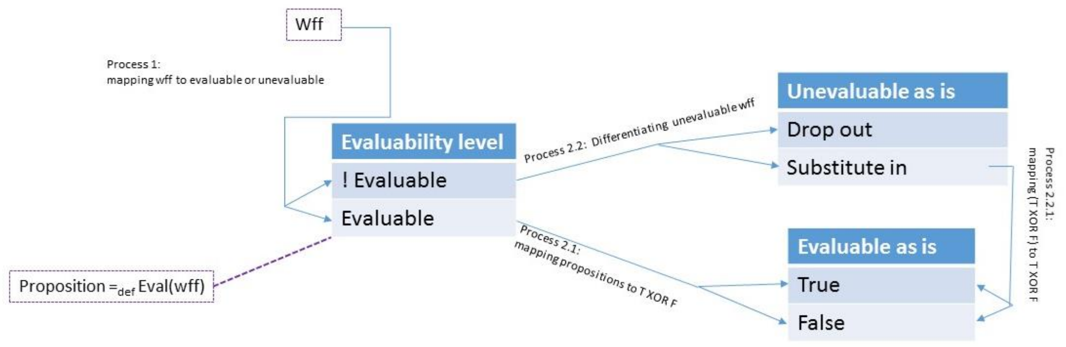

Wff
Process 1:
mapping wff to evaluable or unevaluable
Evaluability level
! Evaluable
Evaluable
Proposition =def Eval(wff)
Process 2.2: Differentiating unevaluable wff
Unevaluable as is
Drop out
Substitute in
Process 2.2.1:
mapping (T XOR F) to T XOR F
Process 2.1:
mapping propositions to T XOR F
Evaluable as is
True
False

Figure 3 Sequenced evaluations covering unevaluable wff

If process 1 results in an unevaluable wff, process 2.2 then evaluates whether the wff's predicate does or does not apply to the argument. Though in pure logic, applicability can be assumed, when working with any kind of real world information, process 2.2 must consult some knowledge store (or analyze the world) in order to determine whether the predicate is or is not applicable to the argument. If the predicate is determined to be applicable, it is treated as missing but otherwise existent. And according to axiom 1, a valid value must be substituted in. The substitution process is labelled 2.2.1 in the figure. Note that from the perspective of a process, machine or human, reasoning over a collection of propositions, with process 2.2.1 in use, the collection of propositions will be a mixture of evaluable wff and unevaluable wff for which substitutions were made. FreeThink used the best available predictive model to infer values where values were needed. If the predicate is determined to be inapplicable then according to axiom 2, the entire wff is dropped from the calculus.

It is important to note that the determination of a wff's evaluability occurred during the execution of the formula (i.e., at run time). Thus for example, a wff such as

Wff#5: `1800 <= per capital caloric intake (Country 'C'. Year, 'Y')` could be evaluated for any combination of country-year. And it could evaluate as a proposition for some, be missing and so in need of substitution for others, and be inapplicable for others still. FreeThink used case statements embedded within the formula syntax to specify the substitution function to evaluate in the case of a repairable wff.

Wff#6: `1800 <= per capital caloric intake (Country ‘C’. Year, ‘Y’)|Substitution function`

For example an analyst might have specified that if the wff #6 cannot evaluate due to missing data that it substitute a value based on the previous year’s caloric intake for the same country, or the nearest year for which data is available, or the nearest year for which data is available modified by the average rate of change for similar countries etc..

The Tractarian-based sequenced evaluation logic in FreeThink became the standard for how the World Bank aggregates and reasons over messy national level data: social, environmental and economic. It demonstrates that it is possible to represent and reason over this kind of information without sacrificing bivalency for evaluable wff i.e., propositions.

Semantic software aimed at solving problem 2: the logical form required for representing the semantics of natural language

Where semantic software for reasoning over messy data needed to extend Wittgenstein’s sequenced evaluation logic to differentiate between different kinds of unevaluable wff, semantic software for natural language processing needs to extend in the opposite direction - to the process of deciding whether a sequence of tokens is or is not a wff. Consider Figure 4 below which extends the sequenced evaluation diagram in Figure 3. Note the addition of process 0. Sequentially, process 0 is the first process that must be executed. All NLP software provides some kind of process 0 to determine whether a sequence of words/symbols[19] is or is not a wff. But how? As described in section I, most NLP software begins the understanding process with lexical dimensions and lexical grammar.

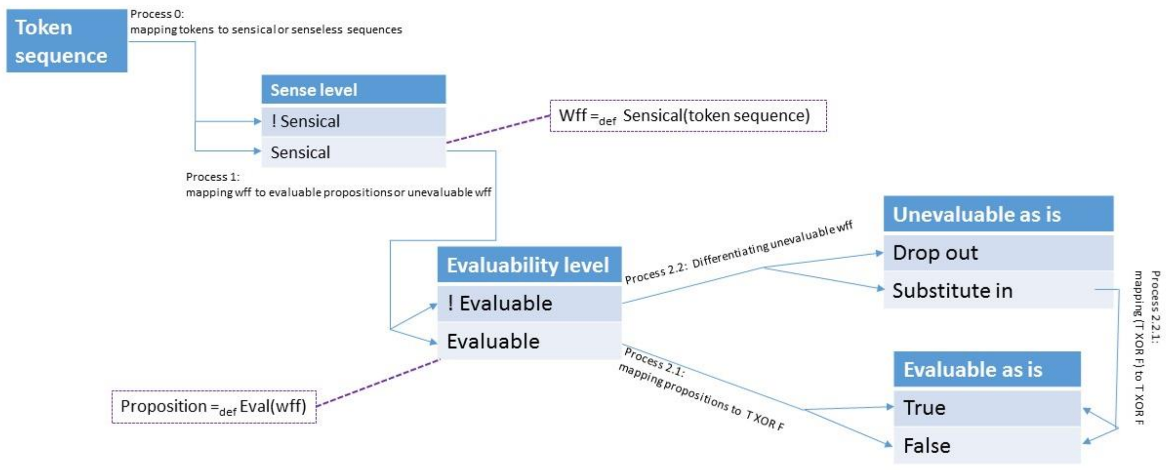

Token sequence
Process 0: mapping tokens to sensical or senseless sequences
Sense level
! Sensical
Sensical
Wff =$_{def}$ Sensical(token sequence)
Process 1: mapping wff to evaluable propositions or unevaluable wff
Evaluability level
! Evaluable
Evaluable
Proposition =$_{def}$ Eval(wff)
Process 2.2: Differentiating unevaluable wff
Unevaluable as is
Drop out
Substitute in
Process 2.2.1: mapping (T XOR F) to T XOR F
Process 2.1: mapping propositions to T XOR F
Evaluable as is
True
False

Figure 4 Sequenced evaluations covering the determination of sensical/senseless sequences of tokens

In contrast, per our reading of Wittgenstein when he asserts that a 'thought can never be of anything illogical" (3.03) or "we could never *say* what an illogical world looks like" (3.031), we believe he was treating abstract or logical grammar, i.e., the grammar that regulates the formation of expressions having the form "f(a)", as the minimal constraints on well formedness; the constraints that apply regardless of the meanings ascribed to the logical variables.

A logical basis for language

We now describe four relevant aspects of the ontology-driven natural language understanding and reasoning system we are building called Multi-Interpreter or 'MI' per our reading of Tractarian logic.

The first relevant Tractarian aspect of MI is its logical grammar. At the base of its language system is a collection of abstract dimensions (e.g., Boolean, Categorical, Rank, Integer, Rational, parent/child hierarchy, graph) for which comparison and manipulation operators are defined,[20] and from which

[19] Strictly speaking, the process begins with something called tokenization; which is the bucketing of sequences of characters into units called tokens. If subsequent language processing fails, the character sequence may be re-tokenized.
[20] A full treatment of the constraints on a well formed dimension are beyond the scope of this paper; see [Author 2012].

additional functions may be constructed. Implicit within such a collection of abstract dimensions is the specification of the minimal roles or parts of any dimension. The following roles are included:

- Dimension value (e.g., the values, 2, 3 and 4 of the Integer Dimension)
- Dimension operator (e.g., the operators + and – for the Integer Dimension)
- Dimension identifier (e.g., the identifiers 'Boolean' and 'Integer').

This is important for because we hypothesize that the process of interpreting (or generating) symbols in a surface language consists of mapping external words or symbols to one or more internal dimension roles.

A computable basis for sense

During the process of interpretation, words get mapped to internal dimension roles. Expanding on Wittgenstein's insights into the role of logical grammar in sense making, we hypothesized that the computable basis for sense is the ability of the dimensional representation of the sequence of tokens to be executed as a processor command. For example, consider the following dimension role terminology definitions:

- The capital letters N, M refer to the Dimensions n and m;
- N.v and M.v are the vth values of N and M respectively;
- N.v, N.w are the vth and wth values of N;
- ρ denotes a relation between 2 or more values of a given Dimension;
- φ denotes a function whose inputs are argument dimensions and outputs are predicate dimensions;
- Dimension symbols in bold parentheses **( )** denote arguments (what MI calls locators); dimension symbols to the left of the parentheses denote predicates (what MI calls contents).

MI recognizes 3 different combinations of logical dimension roles capable of being assigned a sensical interpretation as an assertion.[21] For clarity, we highlighted the argument of each expression in boldface.

[21] Thus, for example a pure sequence of operators "φ φ φ" or a pure sequence of dimension names "NMO" would be senseless. Other expression forms such as questions and commands are handled, but their discussion is beyond the scope of this paper.

Sensical combinations AS IS	**Interpretation and examples**
N.v, **(M.v)**	Asserting N.v for the **(argument M.v)** • Red **(car)** • Running **(girl)** • 420 meters tall **(empire state building)**
N.v, ρ **(N.x)** N.v, **(ρ, N.x)**	Asserting N.v as having the ρ relation to N.w • Bill is child of **(Mary)** • Bill **(is child of Mary)** • 6 is smaller than **(9)** • Jessie is a friend of **(Ron)**
N.v = φ**(M.v)**	Asserting that executing the function φ on the **argument M.v** produces the predicate N.v • 9 = + **(3, 6)** • Scott is name of the **(author of Waverly)**

These assertive (or declarative) expressions define wff which can be further evaluated to determine whether or not they are propositions. Quantifiers and other operators can be applied to these wff (Author 2012). And, as will be shown below, they can also be treated as general "f(a)" templates into which specifically dimensioned semantic expressions are classified.

Treating *f* and *a* as information slots for semantic dimensions

The second relevant Tractarian aspect of MI is its separation of logical grammar and semantics. MI provides two distinct subsystems for processing natural language: an "f(a)" slot manipulator and a semantic dimension slot filler. Figure 5, a diagram from MI's technical manual, illustrates how a sequence of tokens is processed according to separate slot manipulating (i.e., logical) and slot filling (i.e., semantic) criteria. In the slot manipulating component of the figure, E_n refers to an arbitrary expression, L_{nn} refers to the nth argument slot in E_n, C_{nn} refers to the nth predicate slot in E_n. The Trs inside the boxes associated with L_{nn} and C_{nn} represent the dimension roles (called type roles in MI) associated with symbols that are assigned to either argument or predicate slots. The slot filling role component of the figure (whose detailed description is beyond the scope of this paper), illustrates how surface lexical features (called physical representations or 'phys reps' in MI) combine with dimension role classifiers for the purpose of associating symbols with slot filling dimension roles.

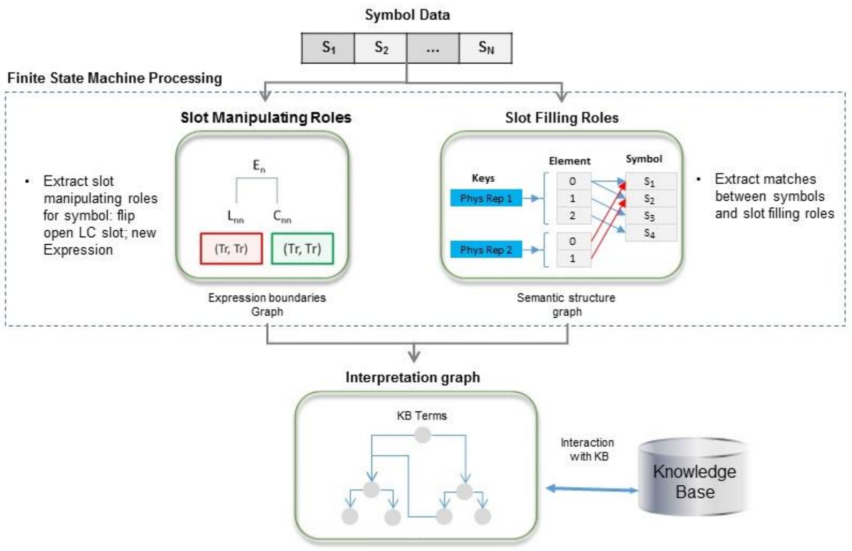


Figure 5: Overview of Multi-Interpreter's NLP process

MI treats the logical form "f(a)", and all of its compositions (e.g., expressions built by logically connecting multiple "f(a)" expressions, or expressions that are components of an "f" or an "a") as open slots into which semantic information gets placed. Any semantic information can be placed in any slot. A single word contains (or is associated with) one or more dimension roles. So, it is possible for a single word to carry both "f(a)" slot information and specific semantic information. For example, in the sentence "The book is blue", MI would begin by opening an *a* slot and place the semantic dimension corresponding to 'the book' in the *a* slot. The word 'is' would carry both a slot manipulation role, namely to close the *a* slot and open an *f* slot (effectively saying 'the information that follows the symbol *is* belongs in the predicate slot'); and a slot filling role, namely the placing of a time value of *now* in the open *f* slot. In addition to argument and predicate slot manipulation, other logical markers that MI extracts from text include markers for nested expressions (e.g., seeing the word "that" may produce a command to open a new "f(a)" expression nested within the currently open f or a slot), 2nd order expressions (e.g., seeing the word 'says' as in "Jessie says that snow is falling" will open a new "f(a)" expression nested within the "f" of the outer expression), and expression relations (e.g., seeing the word "because" may open two "f(a)" expressions connected by a logical operator such as IF-THEN).

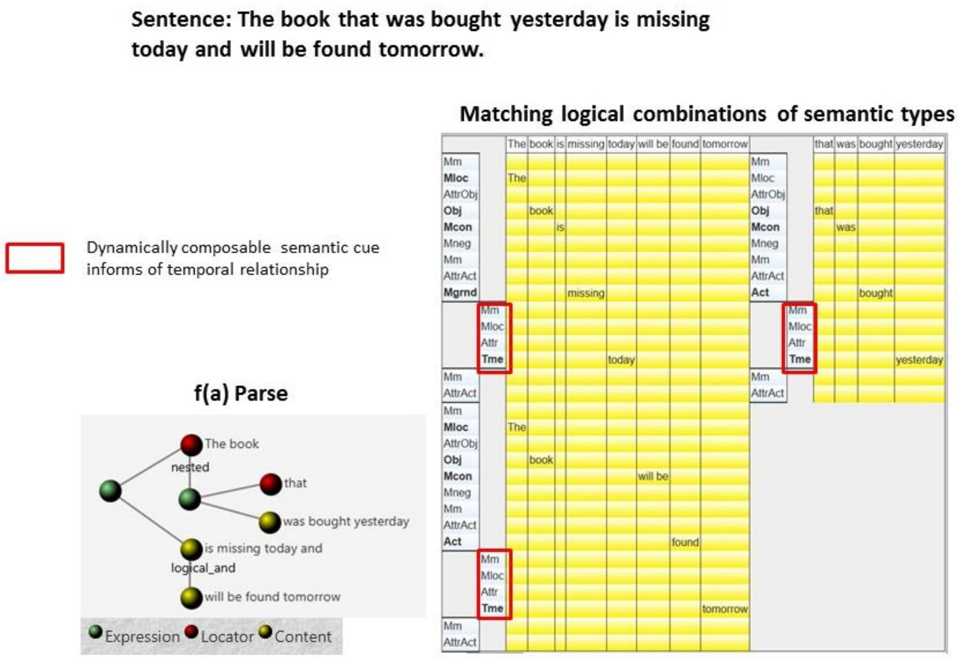
MI's semantic dimensions

The third relevant Tractarian aspect of MI is that the composition of semantic dimensions (e.g., Time, Space, Objects, Processes, Attributes) into semantic understandings relies solely on the logical grammar. Understanding is the arrangement of semantic dimensions according to logical grammar. The following are example compositions of semantic dimensions according to logical grammar:

Attribute(Object),
Attribute(Object, time),
Process(Object, Role),
Attribute(Process, Time) AND
Object(Process, Role, time)

The fourth relevant Tractarian aspect is MI's use of semantic dimensions plus lexical surface features (word order) to extract wff from sequences of words *without passing through a lexical representation*. MI accomplishes this by using its semantic dimensions – QUA ontology, to produce all logically possible combinations of its root semantic dimensions. Effectively, this is MI's combined logical-semantic grammar. Being a grammar, it can generate an unbounded number of compositions. The relevant question is how many base combinations are required. To date MI requires only 30 basic combinations (including the six shown above). So when MI analyzes a sequence of words, instead of first assigning lexical categories and then trying to map those to a semantic representation, as FOL-based systems typically do, MI's first step is to map logically grammatical combinations of semantic dimensions (e.g., Object-Process) directly to word sequences.

Figure 6 is a composite of two screenshots of MI's developer interface. It portrays the analysis of a simple sentence (shown at the top of the figure) containing nested clauses, logical, and temporal relations. The image on the left titled **f(a) parse** shows the "f(a)" slots created and the words that were assigned to each slot. (Note that MI uses the terms "locator" to refer to arguments "a", and the term "content" to refer to predicates "f"). The image on the right titled **matching logical combinations of semantic types** (i.e., dimensions) shows the matches that occurred between the words in the sentence shown in a modified order across the top row, and logical arrangements of semantic dimensions shown across two columns.


Sentence: The book that was bought yesterday is missing today and will be found tomorrow.

Matching logical combinations of semantic types

		The	book	is	missing	today	will be	found	tomorrow
Mm									
Mloc		The							
AttrObj									
Obj			book						
Mcon				is					
Mneg									
Mm									
AttrAct									
Mgrnd					missing				
	Mm								
	Mloc								
	Attr								
	Tme					today			
Mm									
AttrAct									
Mm									
Mloc		The							
AttrObj									
Obj			book						
Mcon							will be		
Mneg									
Mm									
AttrAct									
Act								found	
	Mm								
	Mloc								
	Attr								
	Tme								tomorrow
Mm									
AttrAct									

		that	was	bought	yesterday
Mm					
Mloc					
AttrObj					
Obj		that			
Mcon			was		
Mneg					
Mm					
AttrAct					
Act				bought	
	Mm				
	Mloc				
	Attr				
	Tme				yesterday
Mm					
AttrAct					

Figure 6: a composite screen shot of two of MI's developer interfaces

Though beyond the scope of this paper to fully describe MI, the items in the two columns with labels Mm, Mloc, AttrObj, Obj etc.. represent MI's semantic dimensions (e.g., Obj denotes object, Act denotes action) and MI's logical grammar (i.e. slot filling dimensions). Mloc represents an argument slot opening marker; Mcon represents a predicate slot opening marker; Mneg represents a negation marker etc.. The logical and semantic markers are grouped into instances of MI's basic 30 combinations. The sentence as a whole is represented as a composite of two basic combinations: attribute(object) for "the book is missing"; and action(object) for "was bought yesterday" and for "will be found tomorrow". The red boxes highlight MI's flexible capturing of temporal information.

MI is still a work in progress. But it has undergone preliminary testing against data sets representative of one of the sponsoring client's problem domain (a collection of several hundred field reports). The problem space is concerned with the interpretation of human generated activity reports, and machine generated databases of planned actions and the determination of when reported activities may impact the safety of planned actions. One of MI's strength's is its semantic interpretation of prepositions where it treats them as relations (e.g., of space, time, ownership, composition or control) between simple propositions.

The following is representative of a human activity report. We highlighted prepositions to illustrate their importance for building semantic representations.

Recent Reporting indicates ***that*** *Freedonia is expediting the deployment* ***of*** *medical personnel* ***to*** *the danger zone.* ***In*** *line* ***with*** *this reporting, imagery* ***from*** *May14 shows one medical convoy* ***with*** *two busses, one communications truck and two support vehicles* ***IVO*** *4304N 5231W.*

The reports were manually analyzed by a combined team of technical and domain experts to create a gold standard repository of semantic representations. Based on comparison with the gold standard representations, MI successfully interpreted over 90% of these reports and produced 10 alerts, 8 of which were correct. Though much work still needs to be done, we believe that the tests performed so far provide confidence that the core Tractarian approach will not need to change as we add more breadth to the ontology and greater language coverage.

Conclusion

We believe we have provided compelling evidence that semantic software products whose purpose is to process information based on a formal understanding of the meaning of that information (e.g., Relational databases, Object databases, Knowledge-bases, Natural language parsers and reasoners) are grounded in logic; overwhelmingly FOL. Second, we have shown the limitations of semantic software due to their grounding in FOL. Third, we argued that Tractarian logic (and more specifically, sequenced evaluations) can be used to overcome these limitations. Finally, we showed how Tractarian logic can be used to improve semantic software.

References

Barwise J., Etchemendy J. (1999). *Language, Proof and Logic*. New York: Seven Bridges Press

Codd, T. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13.

Codd, T. (1990). *The Relational Model for Database Management. Version 2*. Addison-Wesley Publishing Company

Collins, M. (1999). *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999

Croft, W. (1991). *Syntactic Categories and Grammatical Relations: The Cognitive Organization of Information*. The University of Chicago Press

Date, C. J. (2000). *An Introduction to Database Systems*. 8th edition Boston: Addison-Wesley

Date, C. J., Darwen, H. (2000). *Foundations for Future Database Systems*. Boston: Addison Wesley

Domingos, Kok, Poon et al., (2006). Unifying Logical and Statistical AI. In *AAAI*, 6, 2-7

Kleene, S. C. (1952). *Introduction to Metamathematics*. Amsterdam: North-Holland Pub.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. *Proceeding of EMNLP*. Seattle, Washington

Lukasiewicz, J. (1970). Philosophical remarks on many-valued systems of propositional logic (1920/1930). in *Selected Works*. L. Borkowski, ed.; O. Wojtasiewicz, trans. Amsterdam: North-Holland Pub.

McGoveran, D. (Dec.1993 - Mar. 1994). Nothing from Nothing. In *Database Programming & Design*.

Russell S., Norvig P. (2003). *Artificial Intelligence. A modern Approach*. Pearson Education New Jersey

Shavel S., Thomsen E. (1990). A Tractarian Approach to Information Modeling. Wittgenstein-Towards a Re-evaluation. Wien: Verlag Holder-Pichler-Tempsky

Steedman M. (2000). *The Syntactic Process*. Cambridge MA: MIT Press

[Author] (2002) …

[Author] (2012) …

[Author] (2015) …

Wittgenstein, L. (1921/1974). *Tractatus Logico Philosophicus*. London: Routledge & Kegan Paul

Wittgenstein, L. (1980). *Wittgenstein's lectures Cambridge 1930-1932*. Edited by Desmond Lee. Totowa, New Jersey: Rowman and Littlefield

Semantic software

Berkley Parser (2016): http://nlp.cs.berkeley.edu/software.shtml

CYC (2016): http://www.cycorp.us/

DB2 (2016): http://www.ibm.com/analytics/us/en/technology/db2/

OpenNLP (2016): http://opennlp.sourceforge.net/projects.html

Oracle (2016): http://www.oracle.com/technetwork/database/database-technologies/rdb/documentation/index.html & http://www.oracle.com/technetwork/database/database-technologies/rdb/documentation/index.html

Semeval (2015): http://alt.qcri.org/semeval2015/

Stanford Parser (2016): http://nlp.stanford.edu/software/lex-parser.shtml

Style Syntax (2012): https://www.w3.org/TR/owl2-syntax/

TAC (2015): http://www.nist.gov/tac/2015/KBP/data.html

Teradata (2016): http://www.info.teradata.com/Datawarehouse/eBrowseBy.cfm?page=TeradataReleaseDefinitions