Rapid Text-based Authoring of Defeasible Higher-Order Logic Formulas, via Textual Logic and Rulelog (Summary of Invited Talk)

Benjamin N. Grosof[*]

Abstract

We present textual logic (TL), a novel approach that enables rapid semi-automatic acquisition of rich logical knowledge from text. The resulting axioms are expressed as defeasible higher-order logic formulas in Rulelog, a novel extended form of declarative logic programs. A key element of TL is textual terminology, a phrasal style of knowledge in which words/word-senses are used directly as logical constants. Another key element of TL is a method for rapid interactive disambiguation as part of logic-based text interpretation. Existential quantifiers are frequently required, and we describe Rulelog's approach to making existential knowledge be defeasible. We describe results from a pilot experiment that represented the knowledge from several thousand English sentences in the domain of college-level cell biology, for purposes of question-answering.

1 Introduction and Requirements Analysis

1.1 Reducing the Cost of Authoring Rich Logical Knowledge

A key goal in the field of expressive knowledge representation and reasoning (KRR) is to reduce the cost of authoring rich logical knowledge.

Rulelog is an expressive knowledge representation logic that is an extended form of declarative logic programs (LP). Rulelog transforms into normal (unextended) LP. Previous work on Rulelog, implemented in XSB [12, 10], Flora-2 [3], SILK [9], and Cyc [2], has developed novel methods that help improve scale-able evolution and combination of such KB's, and thus the cost of overall knowledge acquisition (KA). These methods enable: defeasibility, based on argumentation theories (AT's) [11], i.e., *AT-defeasibility*; higher-order syntax, based on hilog [1] and other meta-knowledge enabled by rule id's, i.e., *hidlog*; bounded rationality *restraint*; [6]; interactive authoring, based on a rapid edit-test-inspect loop and incremental truth maintenance; knowledge debugging, based on a graphical integrated development environment with justification graphs and reasoning trace analysis; and knowledge interchange, based on strong semantics and knowledge translations. Rulelog's full set of major features was first implemented in SILK [9]. A W3C RIF dialect based on Rulelog is in draft [9], in cooperation also with RuleML [8].

In this work on Rulelog, we present another, more radical step that we have developed in order to further reduce such cost: a method that enables text-based authoring, based on a novel approach called *textual logic (TL)*. We also present a novel expressive feature of Rulelog: *omniformity*, which permits defeasible existentials, and is used to support TL.

[*]Benjamin Grosof & Associates, LLC, USA.

1.2 In Quest of a Dream

> "In dreams lie responsibilities" — Delmore Schwartz.

> "Classic knowledge-based AI [artificial intelligence] approaches to QA [question-answering] try to logically prove an answer is correct from a logical encoding of the question and all the domain knowledge required to answer it. Such approaches are stymied by two problems: the prohibitive time and manual effort required to acquire massive volumes of knowledge and formally encode it as logical formulas accessible to computer algorithms; and the difficulty of understanding natural language questions well enough to exploit such formal encodings if available. Techniques for dealing with huge amounts of natural language text, such as Information Retrieval, suffer from nearly the opposite problem in that they can always find documents or passages containing some keywords in common with the query but lack the precision, depth, and understanding necessary to deliver correct answers with accurate confidences." — IBM Watson FAQ.

What if was "cheap" to acquire massive volumes of knowledge formally encoded as logical formulas?

What if it was "easy" to understand natural language questions well enough to exploit such formal encodings?

A central dream for semantic technology is to make knowledge (K) and reasoning be deeper and cheaper — to overcome the famous "knowledge acquisition bottleneck" of AI. That would enable creation of widely-authored, very large knowledge bases (KB's) that automatically answer sophisticated questions (Q's) and proactively supply info, about science, business, and government, with (collectively) broad and deep comprehensiveness.

These KB's would *harness humanity's accumulated storehouse of knowledge*, and be *learned from communication and instruction*, as well as from observation. Such harnessing and learning is *far more powerful than learning from individual experience.* Yet machine learning has not primarily focused on it, to date.

Achieving this dream could create huge amounts of social value. In the remainder of this presentation, we discuss technical requirements in the context of this dream.

1.3 Logical Expressiveness

Logical knowledge is desirable for several reasons. First, *accuracy*: it can provide high-precision of answers. Second, *transparency*: it can provide detailed justifications/explanations. Third, *sharability*: particularly when semantic and semantic-web-friendly, it facilitates reusability and merging of larger KB's, via knowledge interchange.

Expressive richness of logical knowledge is desirable because it enables more kinds of knowledge and reasoning, e.g., scientific, to be represented and automated. Richness is required to represent the logical substance of many text statements, which involves: negation, modifiers, quantifiers (for-all, exists), implication, and conjunction — all flexibly composed. Also, some text statements are about other statements. Thus text requires expressiveness equivalent to that of: first-order-logic (FOL) formula syntax, plus some meta expressiveness, especially higher-order syntax.

Another requirement for expressiveness is that the logic must be *defeasible*, so that it can gracefully handle exceptions and change. Defeasibility is needed: to represent the empirical character of knowledge; to aid the evolution and combination of KB's, i.e., to be *socially scalable*; and to represent causal processes and "what-if's" (hypotheticals, e.g., counterfactual). In other words, defeasibility is needed to represent *change in knowledge and change in the world.* Yet, despite this expressive richness, inferencing in the logic must be computationally scalable, and thus at least *tractable* (worst-case polynomial-time). SPARQL and SQL databases are tractable, for example.[1]

Defeasibility of knowledge that is existentially quantified, in particular, is required expressively to support TL. Existentials appear in many naturally arising text sentences that must be represented as defeasible, e.g., about biology. For example, in the relatively simple sentence "Each eukaryotic cell has a visible nucleus.", the "a" before "nucleus" is an existential. Yet this statement has exceptions. Red blood cells are eukaryotic, yet lack nuclei. Eukaryotic cells lack visible nuclei during anaphase, a step within the process of cell division.

1.4 Text-based Authoring

Text-based authoring is desirable for several reasons. Natural language (NL) — not logic — is the language of "business users", for KA and also for QA. NL is required for broad accessibility by the knowledgeable community of (potential) contributors, in science and many similar areas. In particular, NL is much more broadly accessible and familiar to subject matter experts (SMEs), as opposed to knowledge engineers (KEs) trained in logic. Examples of SME's include scientists, business process owners, executives, lawyers, doctors, educators, engineers, analysts, civil servants, merchants, soldiers, chefs, and members of many other occupations. NL is required also for ordinary end users, e.g., students, citizens, shoppers, salespersons, clerks, and patients — i.e., for the community of (potential) "consumers" of the knowledge in science and many similar areas. Even KE's usually find much easier to articulate and understand text than logic. Most of the world's knowledge is currently described in text, so working from text sources is crucial. Economic scalability of KA thus requires authoring to be text-based, rather than directly in strict logical syntax which requires KE skill.

Economic scalability requires not only that the authoring be accessible in this regard, but also that it take a relatively low amount of effort per sentence. This implies, first, that the encoding *text should not be onerously restricted.* Second, there must be methods for *rapid disambiguation with logical and semantic precision.*

Previous approaches to text-based KA of rich logical K have suffered from major difficulties. One category of approaches permit the input text to be fairly unrestricted NL. Next, we consider that category. Natural language processing (NLP) technology has not yet been able (in general, i.e., reliably) to "go all the way" in fully automatic text interpretation: much of the semantics is not captured. Substantial further disambiguation is needed, for most sentences. Also, the NLP field has been messy and unstandardized in regard to components and forms of info. It has been weak architecturally in regard to flexibly composable, reusable components. Thus fully automatic NLP has typically produced logical K that is inaccurate (quite noisy), and/or shallow/partial (thus inoperational for desirably effective QA). Fully automatic NLP has also tended to produce K that is opaque, i.e., difficult to understand; often, one sees K that is in terms of the parser's innards. For KE author to finish the text interpretation, they tend to need skills in not only logic, but also NLP as well, i.e., to be a "super" KE. In consequence overall, it has been quite costly to do text-based KA of rich logical K, and it has hardly been used practically, especially compared to the dream.

[1] i.e., for querying, when the number of distinct logical variables per query is bounded; this is often described in terms of data complexity being tractable

A second category of approaches compromises by only allowing "controlled" NL, i.e., restricts the vocabulary, grammar, and/or phrases. This typically requires much upfront phraseological work to define exactly what's allowed. It also requires the author to become familiar with the particular phraseology and its restrictions. It still requires the author to have KE skills if the generated logic is rich. In consequence overall, it has been still quite costly and not used practically to nearly the extent envisioned in the dream.

2 Textual Logic

TL overall is a logic-based approach to both text interpretation and text generation, for both KA and question answering (QA). In TL: text is mapped to logic; logic is mapped to text; and these mappings themselves are based on logic.

The spirit of TL has a "Gettysburg" principle: "logic for the text, of the text, by the text".[2] "*For* the text" means for the sake of text, in that knowledge and questions are input in text form, and answers and explanations are output in text form. "*Of* the text" means that the content of the text is represented in logic, and that the mappings in and out of text are represented in logic. "*By* the text" means that once there is, sufficiently, logic for the text and of the text, then logical knowledge can be specified by text and viewed as text.

A novel aspect of TL is *textual terminology* — a phrasal style of knowledge. Words, and more generally word senses, are employed directly as logical constants. Each constant is a hilog functor (i.e., a function or predicate). A textual phrase corresponds (one-to-one) to a logical term; there is a natural style of composition.

Another novel aspect of TL is that it leverages defeasibility. "The thing about NL is that there's a gazillion special cases."[3].

During TL text interpretation, authors (1.) *articulate* sentences in text, then (2.) logically *disambiguate* those sentences, and (3.) *generate* logical axioms as output. These three steps are, in general, *interactive*, i.e., semi-automatic. Multiple authors may collaborate in these steps, for each sentence, including to divide the labor, edit, and review/comment/rate.

Next we describe particulars for our TL work to date, which we call *TL phase 1 (TL1)*, on text interpretation. The text is in English. The Linguist^{TM} tool from Automata, Inc. was employed, together with SILK. Linguist^{TM} leverages the ERG lexical ontology and associated PET parser, which are open source. ERG has a large vocabulary and broad coverage. Disambiguation (step (2.)) is highly interactive, via a novel GUI-based approach that enables users to disambiguate relatively rapidly. Disambiguation has several sub-steps in which an author specifies additional information, as needed, about: (a.) the parse; (b.) quantifier types and scopes; (c.) co-references; and (d.) word senses. Word sense, so far, is limited in need/use/implementation. Logic generation (step (3.)) is fully automatic, and outputs Rulelog axioms. One *main* axiom is generated for each disambiguated (text) sentence. In addition, *support* axioms are generated that represent auxiliary information, e.g., about paraphrases and types. The support axioms are used together with the main axioms for purposes of inferencing. Other annotation axioms are generated, also, as part of comprehensively capturing the

[2] "Gettysburg" here refers to Abraham Lincoln's Gettysburg address (1863) in which he said that the cause for which the Union soldiers died (in the great Civil War battle of Gettysburg) was that "government of the people, by the people, and for the people, shall not perish from the earth.

[3] Peter E. Clark, private communication

info specified during disambiguation. Articulation (step (1.)) is fully manual. The (text) *encoding* sentence it produces must meet two restrictions, but those restrictions are not onerous for purposes of KA. The first restriction on text is that the sentence be *stand-alone*, i.e., (nominal) co-reference is within a sentence, not between sentences. The second restriction is that the text be *straightforward*, i.e., it should minimize ellipsis (missing words), rhetoric, and metaphor. In the articulation step, sentences may be drawn from a *source* text and then reformulated. E.g., in our pilot TL KA experiment, sentences were drawn from a first-year college-level biology textbook chapter on membranes. Some textbook sentences were used verbatim, i.e., not changed during articulation. However, other textbook sentences were not stand-alone straightforward text, or were too long to be most productively disambiguated, thus were reformulated during articulation to clarify or break them up into multiple (encoding) sentences.

For example, a source sentence (with id ss72) is "Some transport proteins, called channel proteins, function by having a hydrophilic channel that certain molecules or atomic ions use as a tunnel through the membrane (see Figure 7.10a, left).". The KE articulates a foreground encoding sentence (with id es2298) based on ss72: "Channel proteins have a hydrophilic channel.". Another KE disambiguates es2298, producing a Rulelog axiom with formula

```
forall(?x5)^(channel(protein)(?x5) ==>
               exist(?x8)^(have(?x5,?x8) and hydrophilic(channel)(?x8));
```

(here, shown in SILK's human-consumption syntax). This axiom is defeasible and includes meta-facts about its prioritization tag, author, creation-time, etc. (not shown above).

TL1 also includes two text-oriented extensions within: the KRR/KA system for Rulelog itself (both of these were implemented in SILK). The first is to employ simple text generation that is specified by KB's (i.e., rules). The generated text is displayed in the UI for KA and related tasks, notably for viewing justifications [5]. The second is (a subset of) *textic*, a novel technique for fine-grain mixing of text-style syntax with logic-style syntax. Textic is an extension of the concrete (human-consumption) syntax of Rulelog's KR language, that improves readability and writeability. In textic, a word is typically treated as a functor having arity 1, and a space is typically interpreted as a left parenthesis. The textic expressive syntactic feature is defined by a deterministic (reduction) transformation, similar in spirit to several other Rulelog features such as hilog, defeasibility, head conjunction, and body disjunction.

3 Omniform Rules

Rulelog is a logical extension of declarative logic programs (LP) that has a unique set of logical features including hidlog, AT-defeasibility, omniformity, and restraint. Rulelog transforms into normal LP.

Next, we describe omniformity, which enables defeasible knowledge to be existential.

An omniform rule (*omni* for short) permits a rule head and body each to be any formula in first-order logic (FOL) syntax. I.e., each can be a formula composed freely from the usual first-order logic (FOL) connectives (disjunction as well as conjunction and (strong) negation) and quantifiers (existential as well as universal). In addition, the head and body formulas each may employ hilog, and thus be higher-order (HOL), rather than first-order, in their syntax. (Note that hilog does impose a few non-onerous restrictions as compared to full higher-order syntax, e.g., the head formula may not be simply a single variable.) Furthermore, the body may employ negation-as-failure (*naf*) but (as usual in LP) only outside the scope of strong negation (*neg*). For convenience, the usual FOL implication and equivalence connectives are also permitted.

The semantics of the omniformity feature is defined via a transformation that reduces any omniform rule to a set of one or more rules that are *conjunctive.* A conjunctive rule is one whose head is a single literal, and whose body is a conjunction of literals. (A literal is an atom preceded possibly by *neg* and/or naf; as usual in LP, naf must not appear within the scope of *neg*.) This omni transformation *OT* generalizes three transformations previously employed in SILK: (head) omnidirectionality [4], Lloyd-Topor [7], and head conjunction splitting. Splitting here means a rule $(H1 \; and \; ... \; and \; Hn) \; :- B;$ is transformed into a set of n rules: $\{Hi \; :- B \; ; \mid i = 1, ..., n\}$.

In *OT*, *neg* is first driven to be innermost. Then the head is put into *tight normal form (TNF)*, by pushing: *exist* inward past *or*; $forall$ inward past *and*; $forall$ inward past *or* when the disjunct does not mention the $forall$'s quantified variable; and *exist* inward past *and* when the conjunct does not mention that *exist*'s quantified variable. Then, recursing top-down on the expression tree: existentials are skolemized; disjunctions are directionalized, cf. omni-directionality, but generalized to non-literal expressions; and conjunctions are split. *OT* transforms the body in a manner similar to Lloyd-Topor. TNF addresses a subtlety that directionalizing should be done "before" skolemizing. TNF differs, in general, from Skolem normal form (used in FOL resolution theorem proving).

We have also developed a new family of argumentation theories, called ATCO, that improves the behavior of (AT-)defeasibility in combination with omniformity, particularly with existentials, as compared to previous AT's.

4 Experiment: Case Study

We conducted a TL1 KA experiment during January-March 2013, that resulted in a case study in the rapid acquisition of rich logical knowledge from one chapter (on cell membranes) of a popular college-level biology textbook, with implications for biomedical education and research. A distributed team of collaborators — knowledge engineers (KE's) — started from effectively unconstrained natural language text and disambiguated various aspects of English sentences, semi-automatically translating text into defeasible higher-order logic formulas expressed in Rulelog, an extended form of declarative logic programs and a draft W3C RIF dialect, implemented in SILK. The distributed team's workflow authored and curated the knowledge base from the text into several thousand Rulelog axioms targeting question answering by a Digital Aristotle as part of Vulcan Inc.s Project Halo.

In this TL1 KA experiment, about 2,500 English encoding sentences were axiomatized. These included hundreds of questions.

A number of questions, some of them sophisticated, answered successfully using Rulelog inferencing (in SILK) on the axioms. However, due to resource limitations of the study, only relatively limited tests of question-answering (QA) were conducted. The focus of the experiment was on KA productivity, primarily, and KA coverage, secondarily.

Encoding sentence length averaged 10 words and ranged up to 25 words. One main defeasible axiom in Rulelog (SILK syntax) resulted from each sentence. On average, each such main axiom transformed into over 5 rules in normal (unextended) LP.

It took less than 10 minutes (of KE labor) on average per sentence to: author, disambiguate, formalize, review, and revise a sentence.

One should expect in future that more intensive QA testing, with attendant debugging of knowledge, would tend to increase the amount of KE labor effort on average per sentence.

On the other hand, one should expect in future that KA tooling and process improvements would tend to decrease the amount of KE labor effort on average per sentence.

Some book source sentences were also encoding sentences, i.e., were disambiguated verbatim from the book. More frequently, one book source sentence was articulated into two or three encoding sentences before disambiguation, in order to make them clearer and easier to disambiguate.

Collaboration resulted in an average of over 2 authors/editors/reviewers per sentence. Collaborative review and revision of the sentences, their disambiguation, and formalization, approximately doubled the average time per sentence.

The resulting axioms were typically more sophisticated than what skilled KE's typically produce when directly authoring into logical syntax.

The number of candidate parses (generated from the lexical ontology (ERG), essentially) per sentence averaged over 30, and commonly ranged into the hundreds. Disambiguation of the parse alone typically required a fraction of a minute. Typically the correct parse was not the parse ranked best by statistical natural language processing.

Expressive coverage was very good, due to Rulelog's expressiveness: all sentences encountered were representable. Terminological coverage was also very good, due to the textual terminology aspect of the TL approach: little hand-crafted logical ontology was required. Several hundred mostly domain-specific lexical entries were added to the ERG. There were some small (less than a few percent) shortfalls from implementation issues, in terminological coverage and in reasoning (e.g., about numerics) related to expressive coverage.

5 Discussion

In the experiment, the KE labor cost for TL1 KA was very roughly USD \$3–4/word (actual word, not simply 5 characters). That implies a cost of very roughly USD \$500–1500/page (at roughly 175–350 words/page). That is in the same ballpark as the cost of the labor to author the *text itself*, for many formal text documents, e.g., college science textbooks and some kinds of business documents. "Same ballpark" here means same order of magnitude.

The approach of Textual Logic plus Rulelog has major advantages for KA.

- Interactive disambiguation: relatively rapidly produces rich K with logical and semantic precision, starting from effectively unconstrained text.
- Textual terminology: greatly reduces the need for KE labor to specify logical ontology explicitly and to become familiar with it, since logical ontology instead emerges naturally and automatically from the texts phrasings. Textual terminology, and textic, also provide a bridge to work in text mining and "textual entailment".
- Rulelog as rich target logic: can handle exceptions and change and, moreover, is computationally tractable[4], due to its defeasibility and restraint features, respectively.

[4]when radial restraint is utilized

- Rulelog supports knowledge interchange (translation and integration) with: both LP and FOL; all the major semantic tech/web standards (RDF(S), SPARQL, OWL, RIF, CL, SBVR); Prolog, SQL, and production rules. (Although for many of these, with restrictions.)

The approach appears to be significant progress on the famous "KA bottleneck" of AI. It provides "better, faster, cheaper" logical knowledge. That logical knowledge is usable on a variety of KRR platforms.

It's early days still in developing and pursuing this approach, so lots of future work remains to do. One direction is tooling, e.g., to leverage inductive learning to aid disambiguation. Another direction is more KA experiments, e.g.: to push on QA; and to scale up.

A third direction is to try out the approach in various applications. In terms of system architecture, this usage context will often call for specialized UI and service interfaces from apps to TL. Rulelog KRR can make use of databases and other service resources in the apps-relevant environment.

6 Summary

Rich logical knowledge is desirable for its accuracy, transparency, coverage depth, and reusability. Economically scalable KA and QA of rich logical knowledge require methods for: (1.) rapid disambiguation in text-based authoring; and (2.) defeasibility plus tractability in the logical KRR. Textual Logic plus Rulelog is a step forward in both regards. Future directions include more on tooling, more ambitious experiments, and exploring applications.

Acknowledgements

This work was partly supported by Vulcan, Inc., as part of the Halo Advanced Research (HalAR) project, which the author led and which centered around the development of Rulelog and SILK. This presentation describes work done by the author jointly with Paul V. Haley (Automata, Inc., USA), Carl Andersen (Raytheon BBN Technologies, USA), Brett Benyo (Raytheon BBN Technologies, USA), Michael Kifer (Stony Brook University, USA), and Paul Fodor (Stony Brook University, USA). Thanks to the entire HalAR team for helpful discussions and supporting implementation. Thanks also to the overall Project Halo team at Vulcan, Inc.

References

[1] W. Chen, M. Kifer, and D.S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.

[2] Cyc. Cyc. http://www.cyc.com (project begun in approx. 1984), 2013.

[3] Flora-2. Flora-2. http://flora.sourceforge.net (project begun in approx. 2000), 2013.

[4] Benjamin Grosof, Carl Andersen, Mike Dean, and Michael Kifer. Omni-directional Hyper Logic Programs in SILK and RIF. In *Proc. RuleML-2010, the 4th Intl. Web Rule Symp. (Demonstration and Poster)*, 2010.

[5] Benjamin Grosof, Mark Burstein, Mike Dean, Carl Andersen, Brett Benyo, William Ferguson, Daniela Inclezan, and Richard Shapiro. A SILK Graphical UI for Defeasible Reasoning, with a Biology Causal Process Example. In *Proc. RuleML-2010, the 4th Intl. Web Rule Symp. (Demonstration and Poster)*, 2010.

[6] Benjamin Grosof and Terrance Swift. Radial Restraint: A Semantically Clean Approach to Bounded Rationality for Logic Programs. In *Proc. AAAI-13, the 27th AAAI Conf. on Artificial Intelligence*, July 2013.

[7] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, Germany, 1984.

[8] RuleML. Rule Markup and Modeling Initiative. http://www.ruleml.org (project begun in approx. 2000), 2013.

[9] SILK. SILK: Semantic Inferencing on Large Knowledge. http://silk.semwebcentral.org (project begun in 2008), 2013.

[10] Terrance Swift and David Scott Warren. XSB: Extending Prolog with Tabled Logic Programming. *TPLP*, 12:157–187, January 2012.

[11] H. Wan, B. Grosof, M. Kifer, P. Fodor, and S. Liang. Logic Programming with Defaults and Argumentation Theories. In *Int'l Conference on Logic Programming*, July 2009.

[12] XSB. XSB. http://xsb.sourceforge.net (project begun in approx. 1993), 2013.