Towards a Rewriting Framework for Textual Entailment

Vivek Nigam[1]

Centro de Informática
Universidade Federal da Paraíba
João Pessoa, Brazil

Valeria de Paiva[2]

NL and AI Research Lab
Nuance Communications
Sunnyvale, USA

Abstract

Recent years have seen a surge of interest in the problem of textual inference, that is, automatically determining whether a natural-language hypothesis can be inferred from a given premise. This surge has been followed by an interest from the logical community in finding suitable logics for textual inference. The Textual Inference Logic (TIL) is one of these logics. This paper details our first steps towards a rewrite framework for textual inference using TIL. We implement our framework in the computational tool Maude and demonstrate by example that it can be used for textual inference.

Keywords: Textual Inference, Textual Inference Logic, Maude, Rewrite Systems, automated rewriting.

1 Introduction

The aim of computational semantics is to find techniques for automatically constructing semantic representations for expressions of human language, representations that can be used to perform inference. The most basic criterion for success as far as a semantics of natural language sentences is concerned is that of *textual entailment*, which is of establishing when sentences follow from others, when they are consistent with each other, and when they contradict each other.

Traditional formal semantic analyses of human language typically presuppose formalisms with high-expressive power (for example, higher-order logic augmented

[1] Email: vivek.nigam@gmail.com
[2] Email: valeria.depaiva@gmail.com

with modalities) but in computational semantics some variant of first-order logic is generally preferred. First-order logic is able to deal (at least to a good approximation) with a wide range of interesting linguistic phenomena and has much-better computational properties. First-order logic offers an attractive compromise between the conflicting demands of expressiviness and inferential effectiveness. However, other choices, especially variants of description logics, which are even better-behaved computationally could also be considered.

Recent years have seen a surge of interest in the problem of textual inference, that is, automatically determining whether a natural-language hypothesis can be inferred from a given premise. A broad spectrum of approaches have been explored, ranging from shallow-but-robust to deep-but-brittle, especially in the recognizing textual entailment community [8]. Up to now, the most successful approaches have used fairly impoverished semantic representations, relying on measures of lexical or semantic overlap, pattern-based relation extraction, or approximate matching of predicate-argument structure.

One of us has worked with a system that calculates textual inferences as part of the implemented system Bridge, developed at Xerox PARC. The implemented system is described in a collection of papers, including for instance [4], while the logical system itself is described in [5, 10, 12]. The system Bridge is a monolithic system, where several hard problems in Natural Language Processing (NLP) are intertwined: thus one needs to get the right parse, the right entities recognized and typed appropriately, the right disambiguation, the right selectional restrictions, etc.

The logical system TIL (Textual Inference Logic) tries to abstract away from the difficulties of creating the logical representations of the sentences to concentrate on the logical difficulties of reasoning with (a particular kind of) representations. The system TIL is a neo-Davidsonian system [15], a small extension of a version of First-Order Logic (FOL) with **contexts** in the style of J. McCarthy [16]. TIL is one of the systems associated with Natural Logic and it distinguishes itself by the unorthodox treatment of quantification in terms of instantiation of concepts within (McCarthy-style) contexts. This mechanism of quantification, as well as the specific kinds of concepts and roles described, are informed by the modeling of negation and other intensional phenomena, ubiquitous in natural language.

The work described in this note continues the work on reasoning with TIL representations, using different tools for the reasoning and assuming that the representations provided by the NLP module are correct. This decision and some discussion on how to go about finding new NLP tools can be found in [11]. We use **rewriting** techniques to reason about our representations (as did the Bridge system beforehand), but we use a generic and well-known framework for rewriting, Maude, instead of one dedicated to linguistic representations [7].

The basic idea is that a representation R entails a representation R', ($R \vdash R'$) if the representation R can be rewritten to R', using the automated system. The representation R consists of the representation of one or more assertions, as seen in the examples below. As a special case we have $R \vdash \bot$, where the representation R entails falsehood, meaning that R is inconsistent, corresponding, for example, to to

the sentences e.g. *The crow slept* and *The crow did not sleep*.

This paper is organized as follows. Section 2 reviews the representation of sentences as TIL assertions, while Section 3 reviews TIL inference. Section 4 introduces our rewriting framework for textual entailment. It also describes how instances of our framework can be implemented in Maude so to prove the textual entailment of sentences. We carried out some (preliminary) experiments, which are also described in Section 4. Finally, we conclude by poitning out to future work in Section 5.

2 TIL Representations

It is traditional for logics of Knowledge Representation to be fragments of first-order logic (FOL). By contrast, it is traditional for logics for natural language semantics to be higher-order intensional logics. Our logic has concepts, which make it look like a "description logic", that is, a fragment of FOL, but it also has contexts, a possible-worlds-like construct that, we hope, is expressive enough for the needs of natural language.

The basic syntactic notions in TIL are very different from the ones in FOL. Following the model of Description Logics [3] we have **concepts**, subconcepts and **roles**. Some of our reasons for using a *concept denoting analysis* instead of an individual denoting analysys when mapping noun phrases to logic are discussed in [6]. The main reasons are being able to deal with non-existent entities (for example when mapping *"Negotiations prevented a strike"* we do not want to say that there exists negotiations N and there exists a strike S and $prevented(N, S)$, as the prevented strike does not really exist in the actual world) and accounting for downward monotonicity entailments.

But unlike description logic where formulas are build from concepts created by the modelling process via logical operators similar to the ones in FOL, in TIL we have **concepts** from a given ontology O that is a parameter of the system. For the ontology O you can think of Cyc, KM, SUMO or as we will do here WordNet.

Instead of variables we have *skolem constants* that carve out a subset of the concept that we are describing. For example, if our sentence is *The crow slept* we will have a skolem constant corresponding to the predicate for 'sleep' (`sleep-2`) one for the predicate corresponding to 'crow' (`crow-1`), a role connecting the two skolem constants, (`role(sb,sleep-4,crow-1)`) and a role to express the cardinality of the noun, singular (`sg`), as follows:

```
a crow slept.

Conceptual Structure:
      role(cardinality_restriction,crow-1,sg)
      role(sb,sleep-4,crow-1)
      subconcept(crow-1,[crow#n#1,crow#n#2,,brag#n#1])
      subconcept(sleep-4,[sleep#v#1,sleep#v#2])

Contextual Structure:
```

```
      instantiable(crow-1,t)
      instantiable(sleep-4,t)
      top_context(t)

Temporal Structure:
      trole(when,sleep-4,interval(before,Now))
```

`Conceptual Structure`, `Contextual Structure` and `Temporal Structure` are pretty-printing labels to help the reader. From now on we will suppress all the information about temporal structure, as are are not dealing with it, yet. Similarly we will only consider one primitive concept for each skolem. This corresponds to doing *manual disambiguation* and we would like to relax this constraint as soon as the basic inferences are in place. The important assertion is `role(sb,sleep-4,crow-1)` which corresponds to the information that the subject of the `sleep-4` event is the crow `crow-1`. The contextual structure is not very important in this simple example, where we only have one context, the top or true context `t`. But the contextual structure is what makes this logic an extension of FOL.

For example, for a sentence like *Ed knew that the crow slept.* we have two contexts, the true context and the context of what was known by Ed, the context named `ctx(sleep-8)`.

```
Ed knew that the crow slept.

Conceptual Structure:
      alias(Ed-0,[Ed])
      role(cardinality_restriction,Ed-0,sg)
      role(cardinality_restriction,crow-6,sg)
      role(prop,know-1,ctx(sleep-8))
      role(sb,know-1,Ed-0)
      role(sb,sleep-8,crow-6)
      subconcept(Ed-0,[male#n#2])
      subconcept(crow-6,[crow#n#1])
      subconcept(know-1,[know#v#1])
      subconcept(sleep-8,[sleep#v#1])

Contextual Structure:
      context(ctx(sleep-8))
      context(t)
      context_lifting_relation(veridical,t,ctx(sleep-8))
      context_relation(t,ctx(sleep-8),crel(prop,know-1))
      instantiable(Ed-0,t)
      instantiable(crow-6,ctx(sleep-8))
      instantiable(crow-6,t)
      instantiable(know-1,t)
      instantiable(sleep-8,ctx(sleep-8))
```

```
instantiable(sleep-8,t)
```

Corresponding to a single formula in FOL, in TIL we have a collection of assertions that, read conjunctively, correspond to the semantics of a (fragment of a) sentence in English.

Concepts in TIL are similar to Description Logic concepts in that they correspond pre-theoretically to sets of objects that satisfy a certain property, like predicates in FOL. Unlike concepts in description logics, our concepts are not always unary predicates (we have concepts for `love,give` for example). We have two kinds of concepts, primitive concepts extracted from an idealized version of the chosen ontology and constructed-on-the-fly concepts, which are always sub-concepts of some primitive concept. These second kind of concepts are dynamic, created by the implemented NLP system when we feed it English sentences. These dynamic concepts are created and placed in the hierarchy/ontology in use, as best as we can, at run time. We assume that our concepts are as fine or as coarse as the sentences that we deal with require. We also assume that our ontology is not circular or inconsistent. (In practice it is hard to show that this is indeed the case, especially with more expressive ontologies.)

Concepts are related to other concepts via *roles* as seen above. Deciding which roles will be used with which concepts is a major problem in computational linguistics. We bypass this problem by assuming that roles are assigned in a consistent, coherent and maximally informative way by the NLP module that we assume appropriate for the task at hand.

One crucial feature of the system TIL is its use of *contexts* and its approach to modelling negation, implication and quantification as well as propositional attitudes and other intensional phenomena. There is a first initial context (written as `t`) that corresponds roughly to what the author of the sentence takes the world to be like. More precisely, in an interpretation of a sentence, the top context corresponds to what the author of the sentence is committed to, with respect to what the world she is describing is like. But since this circumlocution is awkward, we will usually talk about this top level context as the 'true context'. From a practical perspective, contexts in our logic were conceived as syntactic ways of dealing with intensional phenomena, including negation and non-existent entities. They support making existential statements about the existence and non-existence in specified possible worlds of entities that satisfy the intensional descriptions specified by our concepts.

Traditional *propositional attitudes* predicates relate contexts and concepts in our logic. Thus a concept like 'knowing' or 'believing' or 'saying' introduces a context that represents the proposition that is known, believed or said. Contexts are used to 'fence-off' concepts and corresponding roles. Contexts also allow us to localize reasoning: the existence of the knowing event and of Ed are supposed to happen in the true world, while the sleeping of the crow is only supposed to happen in the world of the things known by Ed. In some cases (e.g. in the case of the verb *know that*) we can percolate up the truth of assertions in inner contexts up to the outside context, by assuming that what is known, must be true. In many cases we cannot.

The happening or not of events is dealt with by the *instantiability/uninstantiability* predicate that relates concepts and contexts. Note that all the reasoning to deal with moving assertions out of inner contexts to the top one, to construct the meaning of the sentences, is at the moment dealt with by the NLP module. In this work we simply assume that the representations are correctly constructed. Eventually we want our system to calculate which assertions can be truthfully percolated, which ones cannot, but this should be in a second phase of this work.

While we may be prepared to make the simplifying assumption that if 'X is known' than 'X is true', we certainly do not want to make the assumption that if 'X is said' than 'X is true'. We say that the context introduced by a knowing event is *veridical* with respect to the initial context t, while the context introduced by a saying event is *averidical* with respect to the initial context. Negation introduces a context that is *anti-veridical* with respect to the original context. Having introduced notions of veridicality, antiveridicality and averidicality between contexts we have expanded the expressive power of our language of representations considerably. Thus we have a fairly general mechanism of contexts (these can clearly be iterated), which can represent some positive and some negative information. Similarly to McCarthy's logic we also have 'context lifting rules' that allow us to transfer veridicality statements between contexts, in a recursive way. A precise description of the algorithm explaining how these context lifting rules work for specific classes of verbs (marked in the lexicon) can be found in [18].

3 Textual Inference

The reason for introducing event concepts was the fact that they make some inferences that can be complicated in other semantical traditions very easy. For example it is obvious how to obtain *Ed arrived in the city* from the sentence *Ed arrived in the city by bus*. This inference corresponds simply to conjunction dropping in our logic. But of course there is much more to textual inference than simply dropping of conjuncts.

Inference in TIL is very rudimentary. We can 'drop clauses' like in most event semantics. From the sentence *Ed walked and Mary talked* we are able to infer both *Ed walked* and *Mary talked* by simply forgetting the respective clauses in the original representation.

We can do trivial inferences like identity and we can compose derivations:

$$\frac{}{s \to s} \qquad \frac{s \to t \qquad s \to r}{r \to t}$$

Lastly we can conclude that assertions are inconsistent together, for example *Ed slept* and *Ed did not sleep* will provide us with `instantiable(sleep-0, t)` and `uninstantiable(sleep-2, t)`, and when `sleep-0` and `sleep-2` are unified we obtain an inconsistency.

Monotonicity Reasoning we will begin with some intuitive semantic relations that NL expressions stand in, as the examples below show:

$$\frac{\text{Nina has a canary, canary} \sqsubseteq \text{bird}}{\text{Nina has a bird}} \qquad \frac{\text{Ed kissed Nina, kiss} \sqsubseteq \text{touch}}{\text{Ed touched Nina}}$$

$$\frac{\text{Every carp is a fish, carp} \sqsupseteq \text{koi}}{\text{Every koi is a fish}} \qquad \frac{\text{She didn't give him a bird, bird} \sqsupseteq \text{canary}}{\text{She didn't give him a canary}}$$

But note that the clauses we construct also satisfy the usual monotonicity patterns, both in positive and in negative form. Thus

$$\frac{\text{Ed arrived in the city by bus}}{\text{Ed arrived in the city}} \qquad \frac{\text{Ed did not arrive in the city}}{\text{Ed did not arrive in the city by bus}}$$

while *Ed did not arrive in the city by bus* does **not** entail that *Ed did not arrive in the city*. Similarly a limited amount of 'going up and down the taxonomy' can be accounted for this way, using monotonicity markers.

$$\frac{\text{Ed arrived in the city, Ed} \sqsubseteq \text{person}}{\text{A person arrived in the city}} \qquad \frac{\text{Ed arrived in Rome, Rome} \sqsubseteq \text{city}}{\text{Ed arrived in a city}}$$

Implicative commitments Some previous work has concentrated efforts into uncovering, marking and using the context structure of the logic to provide inferences associated with kinds of verbs with implicative behavior. This work is discussed in [18] and [9]. Here we simply give an example of each one of the classes of "implicative commitments" behavior described by Nairn, Condoravdi and Karttunen.

There are six such classes, depending on whether positive environments are taken to positive or negative ones. Thus for example the verb "manage" takes positive predicates (e.g "Ed managed to close the door" → "Ed closed the door") to positive predicates and negative ones ("Ed didn't manage to close the door" → "Ed didn't close the door"). By contrast the verb "forget (to)" inverts the polarities: "Ed forgot to close the door" → "Ed didn't close the door" and "Ed didn't forget to close the door" → "Ed closed the door".

More complicated are the verbs that only show their implicative behavior either in positive or negative situations. For example we have positive implicatives like the verb "force (to)" takes positive polarities and produces positive polarities (e.g "Ed forced Mary to paint" → "Mary painted"), but if "Ed didn't force Mary to paint" we cannot tell whether Mary painted or not. While "refuse (to)" only works to produce negative polarity (e.g. 'Mary refused to sing" → "Mary did not sing"). Accommodating this fine-grained analysis into the traditional logic description is a very interesting topic for further work. In this note we simply assume that our black box does this kind of reasoning that is necessary to construct the representations.

4 A Rewrite Framework for Textual Inference

In this note we consider a implementation, using the traditional rewriting system Maude to reason about the logical representations produced by the NLP module we are considering. We hand-correct the representations we are given by the NLP mod-

ule, as the goal here is not to obtain correct representations, but to work logically with correct representations.

4.1 Using Maude

The Maude system is an implementation of rewriting logic[17] developed at SRI International. It is similar in its general approach to Joseph Goguen's OBJ3 implementation of equational logic, but based on rewriting logic rather than order-sorted equational logic, and with a heavy emphasis on powerful meta-programming based on reflection, but we do not use these facillities.

Maude modules (rewrite theories) consist of a term-language plus sets of equations and rewrite-rules. Terms in a rewrite theory are constructed using operators (functions taking 0 or more arguments of some sort, which return a term of a specific sort). Operators taking 0 arguments are considered constants, and one constructs their term-language by these simple constructs.

Roughly speaking, a rewrite theory is a triple (Σ, E, R), with (Σ, E) an equational theory with Σ a signature of operations and sorts, and E a set of (possibly conditional) equations, and with R a set of (possibly conditional) rewrite rules. Equational semantics is obtained as the special case in which $R = \emptyset$, so we only have the semantic equations E.

The logical predicates of our natural languages representations are not too numerous, we have (sub)concepts, roles, contexts and a few relations between these. But the concepts that the representations use would be in a useful system in the order of 135 thousand, which is the number of synsets in WordNet[14], our putative ontology (Any other ontology worth using for real language understanding would probably be of the same order of magnitude.) This number of concepts is too large for most automated deduction systems, so instead of facing the challenge of scalability right at the outset of the project, we opted for considering a drastically reduced vocabulary, to check the feasibility of our ideas and the robustness of our tools.

Also while Maude is free software, and tutorials are available online, our black box that produces representations for sentences is not, so we are using for this experiment the public semantic representations available from the the Pargram Project of a short story with only 11 sentences. We now describe the domain of our experiment.

4.2 Rewrite Framework

We introduce the rewrite framework for textual inference using Maude notation. We believe that the notation is self-explanatory. For more information on this notation, we also point out to Maude's documentation [1]. Finally, the Maude implementation of this framework can be found at [2].

Basic Sorts

The basic sorts (or types) that we use in our implementation are `Relations`, `SBasic` and `UnifSet`. TIL relations, such as crow $\sqsubseteq$ bird, belong to the sort

`Relations`, while concept and contextual assertions, such as `instantiable(drink-0,t)` belong to the `SBasic` statement basic sort. Finally, the third basic sort, `UnifSet`, contains unification of skolem constants, such as `crow-6 := bird-1`. This last sort is necessary for textual inference, namely, for unifying skolem constants, as described below in the rewrite rules.

Configurations

We specify the textual inference queries using configurations. These are defined as follows using Maude notation

```
op { _ | _ | _ | _ } : SBasic UnifSet Relations SBasic -> Conf .
```

This command specifies that a configuration is a tuple with four components. The first component of sort `SBasic` specifies the premises of the textual inference problem in the form of conceptual and contextual assertions. We call these the premise assertions. The second component of sort `UnifSet` keeps track of the unification of skolem variables that have been performed. Initially it is empty, specified by the operator `none`. The third component of sort `Relations` specifies the conceptual relations, inherited from the parameter ontology. Finally, the fourth component of sort `SBasic` specifies the goal sentence that we would like to infer also in the form of conceptual and contextual assertions. We call these the goal assertions.

The following equation specifies that a configuration is inconsistent if its premise asserts that the same skolem constant is both instantiable, represent by the operator `ins`, and noninstantiable, represented by the operator `nIns`, in the top context.

```
eq { ins(sk1, top) nIns(sk1, top) pre | unifs | rels | goal } = inconsistent .
```

Thus any configuration with conflicting assertions will be rewritten to the constant `inconsistent`.

Sample of Rewrite Rules

The base case is when there are no further assertions to prove. In this case, then we return a token for `success` and the unifications used. This is specified by the following rewrite rule, where `none` denotes the empty set:

```
rl[REDUCTION-FINAL]: { pre | uni | rels | none  } =>  succ(uni) .
```

The following rule specifies that a goal assertion, `bas`, that is already in the set of premise assertions has been solved and therefore can be removed from the goal assertions.

```
rl[REDUCTION]:
{ bas pre | uni | rels | bas goal  } => { bas pre | uni | rels | goal } .
```

The following rule infers one case (of several other cases that are elided here) when two skolem constants, represented in Maude by using the `sk` operator, can be unified.

```
crl[UNIFICATION-INS]:
{ ins(sk1, ctx) subconcept(sk1, str1) pre | uni | rels
```

```
  | ins(sk2, ctx) subconcept(sk2, str1) goal  } =>
{ pre1 | ([ sk1 := sk2 ]) uni | rels
  | ins(sk2, ctx) subconcept(sk2, str1) goal }
if pre1 := ( ins(sk1, ctx) subconcept(sk1, str1) pre) [ sk1 -> sk2 ] .
```

This rule specifies that two skolems `sk1` and `sk2` can be unified only if they are associated to the same concept denoted by the string `str1`. If this is the case, then a new unification clause, `[ sk1 := sk2 ]` is created and moreover the substitution `[ sk1 -> sk2 ]`, which replaces `sk1` by `sk2`, is applied to the premise assertions.

We can refine the conditions for unifying two skolem constants by using TIL ontological relations. Consider the following rewrite rule:

```
crl[UNIFICATION-INS-TBOX]:
{ ins(sk1, ctx) subconcept(sk1, str1) pre | uni
  | (str1 << str2) rels | ins(sk2, ctx) subconcept(sk2, str2) goal  } =>
{ pre1 | ([ sk1 := sk2 ]) uni | (str1 << str2) rels
  | ins(sk2, ctx) subconcept(sk2, str2) goal  }
if pre1 := ( ins(sk1, ctx) subconcept(sk1, str1) pre) [ sk1 -> sk2 ] .
```

It is similar to the previous rewrite rule. However, this rule allows for the unification of the skolem constants `sk1` and `sk2`, although the first is a subconcept of `str1` and the second of `str2`, which may be different. However, from the assertion `str1 << str2`, which is represented by `str1` $\sqsubseteq$ `str2` in the logic, we have that the concept str2 is more general than the concept `str1` and therfore the substitution `[ sk1 := sk2 ]` is sound.

We can also infer new relations that may be used to infer more unifications. The following rewrite rule is a forward-chaining rule on TIL relations.

```
crl[FORWARD-CHAIN]:
  { pre | uni | rel1 rel2 rels | goal } =>
  { pre | uni | rel1 rel2 rel3 rels | goal }
if rel3 := infer(rel1, rel2)
/\ (not mem(rel3, rels)) /\ (not (rel3 == rel1)) /\ (not (rel3 == rel2)) .
```

It specifies that if there are two relations `rel1, rel2` in the configuration, and these can infer a new relation `rel3`, that is not in the set of relations, and is not the same as the original two relations, then `rel3` should be added to the set of relations of the configuration. This is specified by the condition where `mem` is the membership relation.

One can specify elaborate inference systems for TIL relations. We implemented the constructive fragment of the inference system described in [13].

Proving Textual Entailment

Having described the infrastructure required, we are ready to prove in Maude that from a set of TIL relations $\mathcal{R}$, some premise sentence(s) represented by a set of assertions $\mathcal{P}$ entails some goal sentence(s) represented by a set of assertions $\mathcal{C}$. For this, we construct the corresponding configuration `the =` $\{\mathcal{P} \mid \mathit{none} \mid \mathcal{R} \mid \mathcal{C}\}$

Theorem	1	2	3	4	5	6	7
Number of States	10	5	429	23	33	96	22

Table 1
Summary of the results of experiments carried out. In all experiments, a single solution was found.

and use the following search command in Maude:

```
search the =>* succ(u:UnifSet) .
```

which will start searching using the rewrite rules defined to see whether it can prove the entailment. Maude uses a breadth-first search procedure.

Notice that all the rewrite rules that we have shown are clearly sound. Thus if a solution is found by Maude's the search engine (and assuming that Maude's search engine is correct) then it is indeed the case that the premise sentence(s) entail the conclusion sentence.

4.3 Proposed "theorems" and experimental results

In our experimental results, we proved the following textual entailment relations:

```
1. {a crow was thirsty.} |- {a thirsty crow}
2. {a thirsty crow} |- {a crow}
3. {ed arrived and the crow flew away.} |- {the crow flew away}
4. {ed knew that the crow slept |- the crow slept}
5. {ed did not forget to force the crow to fly} |- {the crow flew}
6  {the crow came out in search of water} |- {the crow came out}
7. {a crow was thirsty} |- {a bird was thirsty}
```

Table 1 summarizes the number of states that Maude needed to traverse in order to prove the corresponding theorem. In all experiments, only a few number of states (less that thousand states) were traversed and Maude was able to prove each theorem almost instanteneously. The only theorem that required more states was theorem 3 as it had more unification problems.

5 Conclusions

This joint collaboration is just starting. We have introduced a general rewriting framework that uses TIL assertions and inference system for textual entailment of sentences. We demonstrated by example that our framework can be implemented in Maude and used it to prove in an automated fashion whether a sentence follows from another.

There are many directions that we are considering pursuing with this work. First there is the software engineering issue of using a big lexical database such as WordNet as input for automated deductions. In our proof-of-concept framework we just extracted a few nouns and verbs required for the examples we wished to prove. The traditional wisdom is to use procedural attachments to extract from the big

database the required data for a given deduction. But it is reasonable to suppose that with NLP's recent success the automated reasoning community might consider some version of 'shallow theorem proving', that is theorem proving for common sense applications. We need to see if the automated theorem proving folks have provided new ways of adding big databases to theorem provers or what they do recommend.

Then there is the issue of the reasoning to produce representations, at the moment, performed by the NLP module and manually disambiguated. We would like to be able to experiment with different NLP modules, especially open source ones. The hope is that the representations we are using are vanilla enough to make adaptations not too costly. After all most of the industry uses WordNet, and common grammatical features as basic vocabulary.

Thirdly there is the issue of sense disambiguation, which is a big problem in NLP, with several systems proposed to solve it. For our proof-of-concept implementation we could afford to do it manually, but for any reasonable use, we will need to decide on a separate module for disambiguation.

From a more foundational point of view, we are also interested on the complexity of the provability problem of TIL. We are also investigating what are the limitations of the logic, i.e., which class of sentences can be proved by TIL inference. We expect that such study will lead to sensible definitions for completeness of the logic.

Finally there is the issue that WordNet was not conceived as a semantic ontology, let alone one geared for inferences. So much of the information we would like to get from the lexicon is not there, yet. Together with Gerard de Melo, the second author is working towards an Implicative Lexicon, ImpLex [9] to complement WordNet, but this is only starting too.

References

[1] Maude homepage http://maude.cs.uiuc.edu. 2014.

[2] http://www.nigam.info/implementations/nlp.zip. 2014.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[4] Daniel G. Bobrow, Bob Cheslow, Cleo Condoravdi, Lauri Karttunen, Tracy H. King, Rowan Nairn, Valeria de Paiva, Charlotte Price, and Annie Zaenen. PARC's bridge and question answering system. In *Proceedings of Grammar Engineering Across Frameworks*, pages 26–45, 2007.

[5] D.G. Bobrow, V. de Paiva, C. Condoravdi, R. Crouch, L. Karttunen, T.H. King, R. Nairn, and A. Zaenen. A basic logic for textual inference. *In Procs. of the AAAI Workshop on Inference for Textual Question Answering, Pittsburgh PA*, page 27, 2005.

[6] D. Crouch J. Everett R. Stolle D. Bobrow de Paiva V. M. van den Berg Con-

doravdi, C. Preventing existence. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 17–19, Ogunquit, Maine, October 2001.

[7] R. Crouch and T. H. King. Semantics vis f-structure rewriting. In *Proc. of the LFG06 Conference, Universität Konstanz. Miriam Butt and Tracy Holloway King (Editors)*, 2005.

[8] I. Dagan, D. Roth, M. Sammons, and F. Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool, 7 2013.

[9] Gerard de Melo and Valeria de Paiva. Sense-specific implicative commitments. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2014)*, LNCS. Springer, 2014.

[10] V. De Paiva, DG Bobrow, C. Condoravdi, R. Crouch, L. Karttunen, TH King, R. Nairn, and A. Zaenen. Textual inference logic: Take two. *Proceedings of the Workshop on Contexts and Ontologies, Representation and Reasoning*, page 27, 2007.

[11] Valeria de Paiva. Bridges from language to logic: Concepts, contexts and ontologies. *Electronic Notes in Theoretical Computer Science*, 269(0):83 – 94, 2011. Proceedings of the Fifth Logical and Semantic Frameworks, with Applications Workshop (LSFA 2010).

[12] Valeria de Paiva. Contexts for quantification. In *Proceedings of CommonSense 2013*, Cyprus, 2013.

[13] Alex Djalali. Synthetic logic. *Linguistic Issues in Language Technology*, 9(2), 2013.

[14] C. Fellbaum. *WordNet: An electronic lexical database*. The MIT press, 1998.

[15] Peter N. Lasersohn. Event-based semantics. *Encyclopedia of Language and Linguistics, 2nd edition, ed. by K. Brown*, 4(1):316–320, 2006.

[16] J. McCarthy. Notes on formalizing context. In *Proc. of the 13th Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 555–560, 1993.

[17] José Meseguer and Grigore Roşu. The rewriting logic semantics project. *Theoretical Computer Science*, 373(3):213–237, 2007.

[18] Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. Computing relative polarity for textual inference. *Inference in Computational Semantics (ICoS-5)*, pages 20–21, 2006.