Extracting higher-order logic formulas from English sentences

Alexandre Rademaker Guilherme Lima

IBM Research

DELPH-IN, 2024

Example I

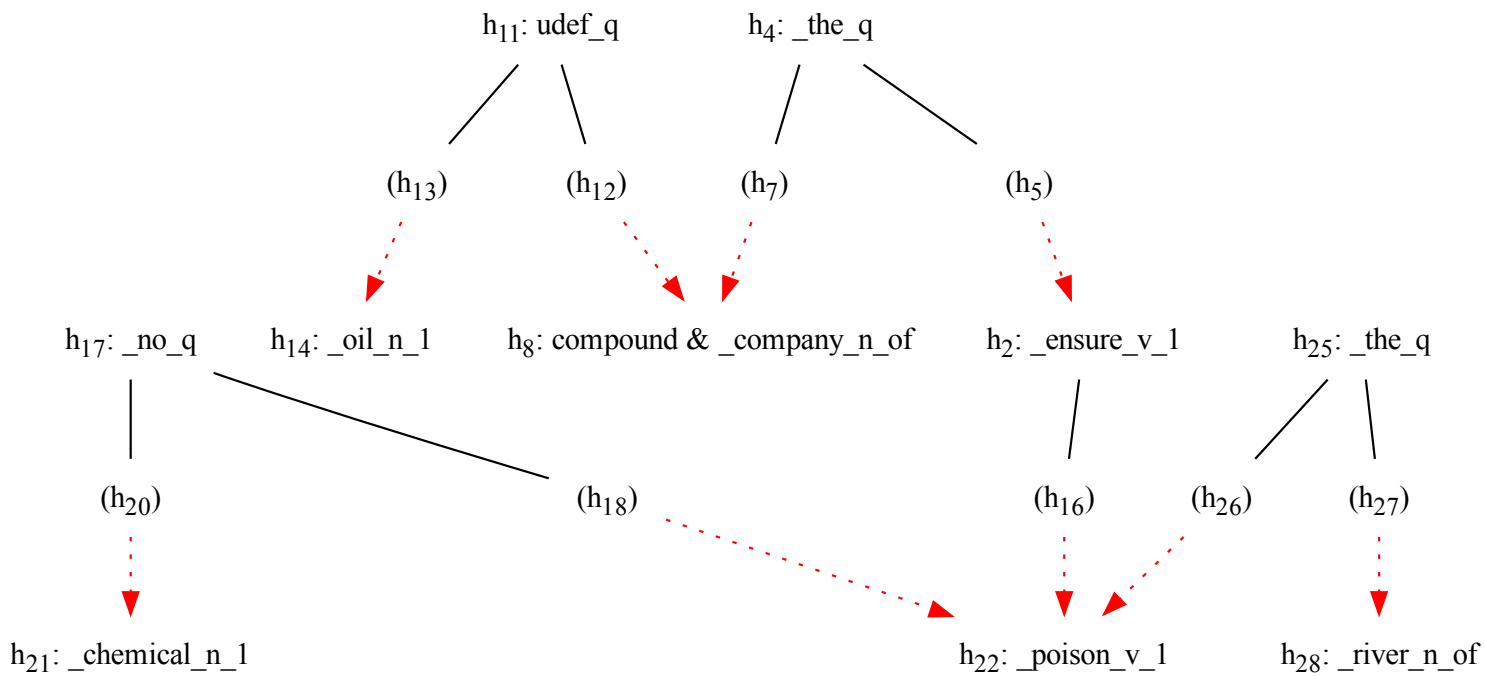
"The oil company ensured no chemicals poisoned the river."

$\langle\, h_1, e_3,$
h_4:_the_q$\langle 0:3\rangle$(ARG0 x_6 {PERS *3*, NUM *sg*}, RSTR h_7, BODY h_5),
h_8:compound$\langle 4:15\rangle$(ARG0 e_9 {SF *prop*, TENSE *untensed*, MOOD *indicative*, PROG -, PERF -}, ARG1 x_6, ARG2 x_{10}),
h_{11}:udef_q$\langle 4:7\rangle$(ARG0 x_{10}, RSTR h_{13}, BODY h_{12}),
h_{14}:_oil_n_1$\langle 4:7\rangle$(ARG0 x_{10}),
h_8:_company_n_of$\langle 8:15\rangle$(ARG0 x_6, ARG1 i_{15}),
h_2:_ensure_v_1$\langle 16:23\rangle$(ARG0 e_3 {SF *prop*, TENSE *past*, MOOD *indicative*, PROG -, PERF -}, ARG1 x_6, ARG2 h_{16}),
h_{17}:_no_q$\langle 24:26\rangle$(ARG0 x_{19} {PERS *3*, NUM *pl*, IND *+*}, RSTR h_{20}, BODY h_{18}),
h_{21}:_chemical_n_1$\langle 27:36\rangle$(ARG0 x_{19}),
h_{22}:_poison_v_1$\langle 37:45\rangle$(ARG0 e_{24} {SF *prop*, TENSE *past*, MOOD *indicative*, PROG -, PERF -}, ARG1 x_{19}, ARG2 x_{23}),
h_{25}:_the_q$\langle 46:49\rangle$(ARG0 x_{23} {PERS *3*, NUM *sg*, IND *+*}, RSTR h_{27}, BODY h_{26}),
h_{28}:_river_n_of$\langle 50:55\rangle$(ARG0 x_{23}, ARG1 i_{29})
$\{\, h_1 =_q h_2, h_7 =_q h_8, h_{13} =_q h_{14}, h_{16} =_q h_{22}, h_{20} =_q h_{21}, h_{27} =_q h_{28} \,\}\, \rangle$

First, from the 13 readings.

Example II

A dominance graph



We have 42 different possible scope trees!

Example III

```
(_no_q ?x17
       (_chemical_n_1 ?x17)
       (_the_q ?x23
              (_river_n_of ?x23 ?i29)
              (_ensure_v_1 ?e3 ?x17
                           (_the_q ?x6
                                  (udef_q ?x10
                                         (_oil_n_1 ?x10)
                                         (and (compound ?e9 ?x6 ?x10)
                                              (_company_n_of ?x6 ?i15)))
                                  (_poison_v_1 ?e24 ?x6 ?x23)))))
```

```
(_the_q ?x23
        (_river_n_of ?x23 ?i29)
        (_no_q ?x17
               (_chemical_n_1 ?x17)
               (_the_q ?x6 (udef_q ?x10
                                  (_oil_n_1 ?x10)
                                  (and (compound ?e9 ?x6 ?x10)
                                       (_company_n_of ?x6 ?i15)))
                           (_ensure_v_1 ?e3 ?x17 (_poison_v_1 ?e24 ?x6 ?x23)))))
```

Example IV

"The oil company ensured no chemicals poisoned the river."

$$\exists\, x_{10}, \_oil\_n\_1\ x_{10} \wedge (\exists\, x_{23}, \_river\_n\_of\ x_{23} \wedge (\exists\, x_6, (\exists\, e_9, compound\ e_9\ x_6\ x_{10} \wedge \_company\_n\_of\ x_6) \wedge (\exists\, e_3, \_ensure\_v\_1\ e_3\ x_6\ (\forall x_{19}, \_chemical\_n\_1\ x_{19} \rightarrow \neg(\exists\, e_{24}, \_poison\_v\_1\ e_{24}\ x_{19}\ x_{23})))))$$

$$\exists\, x_{10}, \_oil\_n\_1\ x_{10} \wedge (\forall x_{19}, \_chemical\_n\_1\ x_{19} \rightarrow \neg(\exists\, x_{23}, \_river\_n\_of\ x_{23} \wedge (\exists\, x_6, (\exists\, e_9, compound\ e8\ x_6\ x_{10} \wedge \_company\_n\_of\ x_6) \wedge (\exists\, e_{24}\ e_3, \_ensure\_v\_1\ e_3\ x_6\ (\_poison\_v\_1\ e_{24}\ x_{19}\ x_{23})))))$$

Not first-order logic!

Reasoning with MRS I

In the context of DELPH-IN.

There is not much work on logical reasoning from MRS at DELPH-IN applications.

LKB (grammar development environment) preliminary implementation of MRS to FOL.

The Glue semantics for HPSG is another approach. Not developed further.

Reasoning with MRS II

"For the purposes of this paper, we will take the object language to be predicate calculus with generalized quantifiers [2].

(2) This should not be taken as suggesting that we think it is impossible or undesirable to give MRS a model-theoretic semantics directly. But given that our main interest is in the computational and compositional use of MRS, it is more appropriate to show how it relates to a relatively well-known language, such as predicate calculus, rather than to attempt a direct interpretation."

(Minimal Recursion Semantics: an Introduction, Ann Copestake, Dan Flickinger, Carl Pollard & Ivan A. Sag)

Reasoning with MRS III

Converting MRS output to a logical form (Ann Copestake's comments)

- FOL is not sufficient for arbitrary sentences
- MRS can be translated into many actual logics
- no existing logic is adequate for all the phenomena of NL
- individual logics can capture all the individual phenomena but these logics don't work out as single well-behaved logic.

Reasoning with MRS IV

"[...] Overall, then, an adequate model of natural language understanding must incorporate commonsense reasoning with linguistic and non-linguistic information. Furthermore, this reasoning must be **defeasible**, because the content of an utterance depends critically on contextual information that's hidden rather than observable. Thus, content is estimated under uncertainty, and one can , therefore change one's mind on gathering subsequent observable evidence about the context."

(Linguistic Fundamentals for Natural Language Processing II, Emily M. Bender and Alex Lascarides)

Applications

Text Entailment (for validate the translation)

Short-term applications such as avoiding LLM hallucinations. Suppose a QA system based on LLM to transform NL questions into SPARQL queries. How to check if the produced SPARQL is ‘related to’ the question?

Agatha Puzzle (with Dan) (see PUZ001 and newsletter)

ULKB I

ULKB is a Python implementation of simple type theory (STT or HOL).

ULKB provides KB Federating the major public knowledge graphs (DBPedia, Wikidata etc)

Extends FOL — the logic of Isabelle HOL.

- types, boolean, functions ($\rightarrow$)
- quantification over arbitrary types (including functions and predicates and individuals)
- high-order functions and predicates

ULKB II

The language is high-order logic

- Expressions
  Terms, formulas, types
- Annotations
  Expression metadata: *KG links, weights, bounds, ...*
- Theories
  Axioms, theorems, definitions
- HOL kernel
  Type-checking, computation, deductive rules
- Import / export plugins
  Parsing, serialization, conversion

```
                                    # Python API
type  ::=  tvar                     TypeVariable(id)
       |   base                     BaseType(id)
       |   bool                     BoolType()
       |   type → type              FunctionType(type, type)

term  ::=  const                    Constant(id, type)
       |   var                      Variable(id, type)
       |   term term                Application(term, term)
       |   λ var ⇒ term             Abstraction(var, term)

form  ::=  term = term              Equal(term, term)
       |   ⊤                        Truth()
       |   ⊥                        Falsity()
       |   ¬ form                   Not(form)
       |   form ∧ form              And(form, form)
       |   form ∨ form              Or(form, form)
       |   form → form              Implies(form, form)
       |   form ↔ form              Iff(form, form)
       |   ∃ var, term              Exists(var, form)
       |   ∀ var, term              Forall(var, form)
```

MRS Logic I

MRS-Logic is a Python Library that uses PyDelphin, Ace, Utool, UKB and ULKB to convert sentences into STT formulas.

ULKB can call provers, e.g., E, Vampire, Z3, LNN, etc.

ULKB can convert formulas to SPARQL queries and results of queries to axioms, interacting with an endpoint.

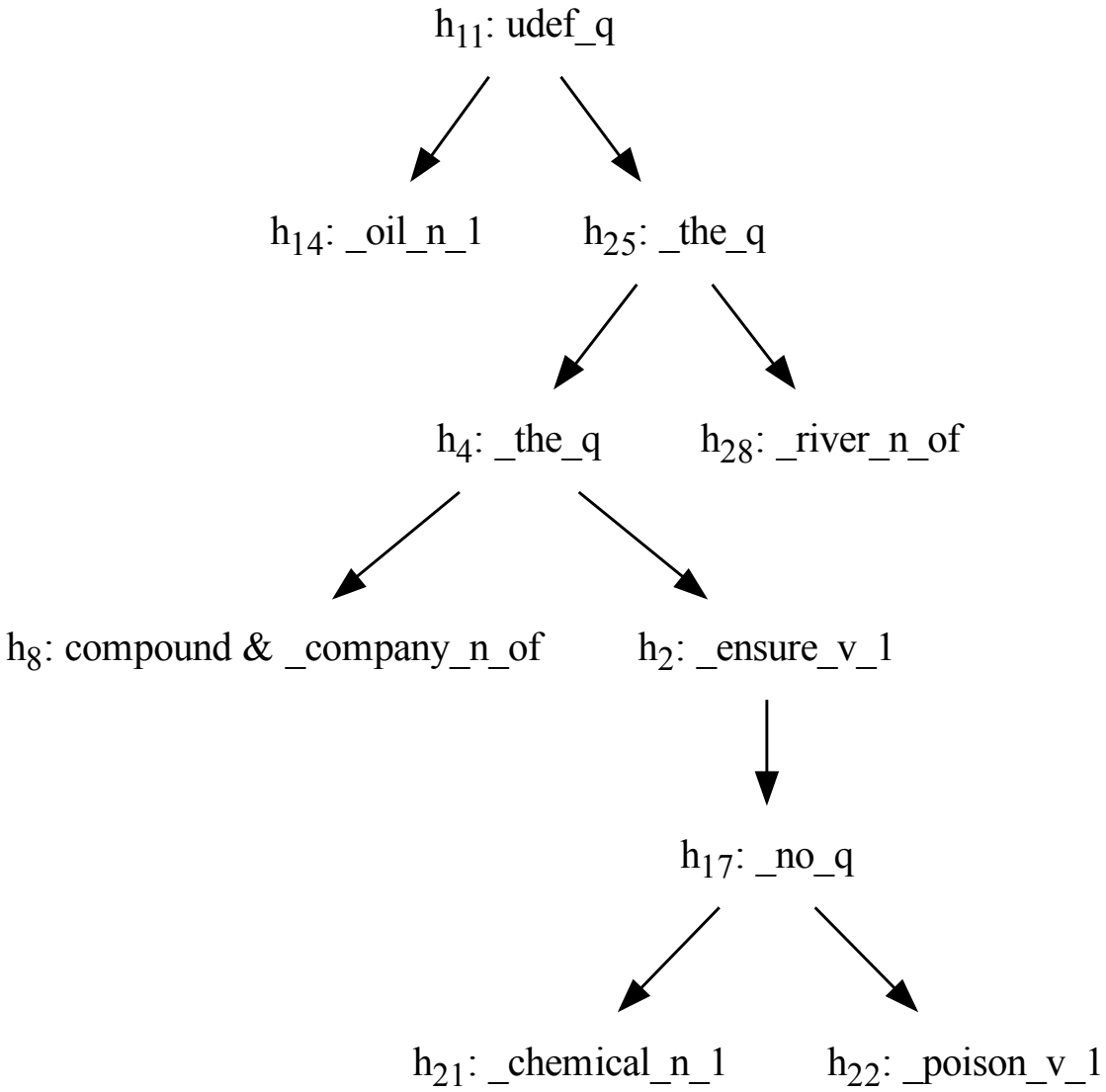
ULKB can serialize to RDF/OWL, TPTP, SMTLIB2, etc.

MRS Logic II

- scope resolution (Utool)
- conversion to STT (recursive top-down on the scope tree)
  - generalized quantifiers
  - NL connectives are kept as non-logical predicates
  - introduction of the 'e' variables (narrow scope)
  - uninstantiated arguments, the 'i', 'u', 'p' variables
  - special predicates (e.g. `neg` ($\neg$), `_if_x_then` ($\rightarrow$))
  - high-order (`subord`, `nominalization`, coordinations etc)
  - `card` and `named` (strings and numbers in the CARG)
- word-sense disambiguation via UKB (annotations)

MRS is obtained from Text using Ace (Answer Constraint Engine) and English Resource Grammar (HPSG Grammar for English)

MRS Logic III



MRS Logic IV

$$\mathbf{T}[h_{11}] = \mathbf{T}[\text{udef\_q ARG0 } x_{10} \text{ RSTR } h_{14} \text{ BODY } h_{25}] = (\exists\, x_{10}, \mathbf{T}[h_{14}] \wedge \mathbf{T}[h_{25}])$$

$$\mathbf{T}[h_{14}] = \mathbf{T}[\text{\_oil\_n\_1 ARG0 } x_{10}] = \_oil\_n\_1(x_{10})$$

$$\mathbf{T}[h_{28}] = \mathbf{T}[\text{\_river\_n\_of ARG0 } x_{23} \text{ ARG1 } i_{29}] = \_river\_n\_of(x_{23})$$

$$\mathbf{T}[h_{21}] = \mathbf{T}[\text{\_chemical\_n\_1 ARG0 } x_{19}] = \_chemical(x_{19})$$

$$\mathbf{T}[h_{22}] = \mathbf{T}[\text{\_poison\_v\_1 ARG0 } e_{24} \text{ ARG1 } x_{19} \text{ ARG2 } x_{23}] = \exists\, e_{24}, \_poison\_v\_1(e_{24}, x_{19}, x_{23})$$

$$\mathbf{T}[h_{25}] = \mathbf{T}[\text{\_the\_q ARG0 } x_{23} \text{ RSTR } h_4 \text{ BODY } h_{28}] = (\exists\, x_{23}, \mathbf{T}[h_4] \wedge \mathbf{T}[h_{28}])$$

$$\mathbf{T}[h_4] = \mathbf{T}[\text{\_the\_q ARG0 } x_6 \text{ RSTR } h_8 \text{ BODY } h_2] = (\exists\, x_6, \mathbf{T}[h_8] \wedge \mathbf{T}[h_2])$$

$$\begin{aligned}\mathbf{T}[h_8] &= \mathbf{T}[\text{compound ARG0 } e_9 \text{ ARG1 } x_6 \text{ ARG2 } x_{10}, \text{\_company\_n\_of ARG0 } x_6 \text{ ARG1 } i_{15}] \\ &= (\mathbf{T}[\text{compound ARG0 } e_9 \text{ ARG1 } x_6 \text{ ARG2 } x_{10}] \wedge \mathbf{T}[\text{\_company\_n\_of ARG0 } x_6 \text{ ARG1 } i_{15}]) \\ &= (\exists\, e_9, compound(e_9, x_6, x_{10}) \wedge \_company\_n\_of(x_6))\end{aligned}$$

$$\mathbf{T}[h_2] = \mathbf{T}[\text{\_ensure\_v\_1 ARG0 } e_3 \text{ ARG1 } x_6 \text{ ARG2 } h_{17}] = (\exists\, e_3, \_ensure\_v\_1(e_3, x_6, \mathbf{T}[h_{17}]))$$

$$\mathbf{T}[h_{17}] = \mathbf{T}[\text{\_no\_q ARG0 } x_{19} \text{ RSTR } h_{21} \text{ BODY } h_{22}] = (\forall\, x_{19}, \mathbf{T}[h_{21}] \rightarrow \neg\mathbf{T}[h_{22}])$$

generalized quantifiers to STT I

We are producing STT, but we are still restricted to FOL provers. More later.

ERG uses generalized quantifiers, and we convert them to FOL quantifiers.

For now, we have a list of ERG quantifiers and their mappings.

generalized quantifiers to STT II

Existential quantifiers

_a_q, _another_q, _any_q, _both_q, _each_q, _some_q, _some_q_indiv, _that_q_dem, _this_q_dem, _which_q, free_relative_q, def_explicit_q, def_implicit_q, idiom_q_i, number_q, pronoun_q, proper_q, udef_q, **_the_q**

translated to

$$\exists\, x,\ RSTR \wedge BODY$$

generalized quantifiers to STT III

Universal quantifiers: `_all_q`, `_every_q`, `every_q`

translated to

$$\forall x,\ RSTR\ \rightarrow BODY$$

Unique quantifiers: **_the_q** and `which_q`

translated to

$$\exists! x,\ RSTR\ \wedge BODY$$

generalized quantifiers to STT IV

"the book is white"

We translate as $\exists\ x,\ book\ x$. The $\exists!$ is translated to FOL as

$$\exists\ x,\ book\ x\ \wedge \forall\ y,\ book\ y \rightarrow x = y$$

How to prove it? We usually don't have more than the utterance. We would need more axioms or hypotheses.

generalized quantifiers to STT V

Negation `_no_q`

translated to

$$\forall x,\ RSTR \rightarrow \neg BODY$$

predicates I

```
h:neg[ARG0: e15 ARG1: h16]
```

translated to

$$T(h) = \neg T(h16)$$

predicates II

```
h3:named[ARG0 x, CARG "value"]
```

translated to

$$value = x$$

predicates III

```
[ card LBL: h12 ARG0: e14 ARG1: x8 CARG: "N" ]
```

translated to

*card N e*14 *x*8

predicates IV

```
h1:_if_x_then[ARG0: e2 ARG1: h4 ARG2: h5]
h14:_then_a_1[ARG1: h16]
h4 qeq h14
```

translated to

$$T(h5) \rightarrow T(h4)$$

We are not using yet the “fingerprints”, we should! But translation would not be top-down on the scope tree anymore.

predicates V

Many predicates need a better translation.

- card
- compound
- nominalization
- `_and_c` and `_or_c`
- only (e.g. “they are the **only** people in the house.”)
- etc

‘e’ variables I

We need to introduce them in the narrowest possible scope because of negation.

‘e’ variables II

“A man is not walking”

```
[ TOP: h0
  INDEX: e2
  RELS: <
   [ _a_q<0:1>          LBL: h4 ARG0: x3 RSTR: h5 BODY: h6 ]
   [ _man_n_1<2:5>      LBL: h7 ARG0: x3 ]
   [ neg<9:12>          LBL: h1 ARG0: e8 ARG1: h9 ]
   [ _walk_v_1<13:20> LBL: h10 ARG0: e2 ARG1: x3 ] >
 HCONS: < h0 qeq h1 h5 qeq h7 h9 qeq h10 > ]
```

- $\neg(\exists\ x3,\ \_man\_n\_1\ x3 \wedge (\exists\ e2,\ \_walk\_v\_1\ e2\ x3))$
- $\exists\ x3,\ \_man\_n\_1\ x3 \wedge \neg(\exists e2,\ \_walk\_v\_1\ e2\ x3)$

‘e’ variables III

“No man is walking”

```
[ TOP: h0
  INDEX: e2
  RELS: <
   [ _no_q<0:2>        LBL: h4 ARG0: x3 RSTR: h5 BODY: h6 ]
   [ _man_n_1<3:6>     LBL: h7 ARG0: x3 ]
   [ _walk_v_1<10:17> LBL: h1 ARG0: e2 ARG1: x3 ] >
  HCONS: < h0 qeq h1 h5 qeq h7 > ]
```

$\forall x3, \_man\_n\_1\ x3 \rightarrow \neg(\exists\ e2,\ \_walk\_v\_1\ e2\ x3)$

‘i’ variables I

“A kid is sitting wearing gea”

```
[ TOP: h0
  INDEX: e2
  RELS: <
    [ _a_q<0:1> LBL: h4 ARG0: x3 RSTR: h5 BODY: h6 ]
    [ _kid_n_1<2:5> LBL: h7 ARG0: x3 ]
    [ _sit_v_1<9:16> LBL: h8 ARG0: e2 ARG1: x3 ]
    [ subord<17:29> LBL: h1 ARG0: e9 ARG1: h10 ARG2: h11 ]
    [ _wear_v_1<17:24> LBL: h12 ARG0: e13 ARG1: i14 ARG2: x15 ]
    [ udef_q<25:29> LBL: h16 ARG0: x15 RSTR: h17 BODY: h18 ]
    [ _gear_n_1<25:29> LBL: h19 ARG0: x15 ] >
  HCONS: < h0 qeq h1 h5 qeq h7 h10 qeq h8 h11 qeq h12 h17 qeq h19 > ]
```

. . . $\exists x15$, *_gear_n_1* $i14$ $x15$

How to quantify $i14$?

The SICK dataset I

How do we test the library? Robustness and correctness?

SICK dataset of text entailment. The SICK dataset includes 9,840 sentence pairs taken from images and video captions.

If sentences A and B are classified as an entailment, we should be able to prove $\Delta \vdash \mathbf{T}(A) \rightarrow \mathbf{T}(B)$ where Δ is a background theory.

If they are classified as a contradiction, we should be able to prove $\Delta \vdash \neg(\mathbf{T}(A) \wedge \mathbf{T}(B))$; otherwise, we consider them as neutral.

The SICK dataset II

A small theory of 24 axioms was added incrementally, experimenting with the system's adaptability to incremental addition of background knowledge.

The axioms cover simple lexical semantics gaps such as $\forall x, \mathit{man}\ x \rightarrow \mathit{person}\ x$ and $\forall x, \mathit{empty}\ x \rightarrow \neg \mathit{full}\ x$ that can be easily derived from resources like Wordnet. WSD in the future.

We also have some axioms related to the ERG abstract predicates, such as $\forall\ e\ x\ y, \mathit{compound}\ e\ x\ y \rightarrow \mathit{for}\ e\ x\ y$. ("cow milk" is an underspecification of "milk for cow")

The SICK dataset III

A A woman is slicing a carrot

B A carrot is being sliced by a woman

ENTAILMENT

A Several people are in front of a building which is covered by colors

B People are walking outside a building that has many murals on it

ENTAILMENT?

The SICK dataset IV

A Two dogs are wrestling and hugging

B There is no dog wrestling and hugging

CONTRADICTION

A A woman is not frying some food

B A woman is deep frying food

CONTRADICTION?

The SICK dataset V

A Three kids are sitting in the leaves

B Three kids are jumping in the leaves

NEUTRAL

The SICK dataset VI

Of the 6,077 unique sentences, 3,435 sentences have at least 5 readings. 2,055 sentences had less than five readings, 564 between 5 and 9 readings, and only 23 sentences were not parsed by ERG, some ungrammatical.

#	label
1424	CONTRADICTION
2822	ENTAILMENT
5596	NEUTRAL

Results I

SICK is very unbalanced regarding the entailment test, and the corpus contains a lot of repeated sentences. We created a subset of SICK, called SB-SICK (small and balanced SICK), with 330 pairs for each label and no sentence repetition.

label	true	false	%
CONTRADICTION	117	213	35
ENTAILMENT	132	198	40
NEUTRAL	330	-	100

Results II

A more detailed error error analysis is needed!

- search space
- missing ontological axioms?
- missing logical axioms?
- expressivity
- missing meaning postulates?

RTE I

Other hard cases for RTE. Thanks to Johan Bos.

- A man is smiling.
- A man is not smiling.

Inconsistent?

- Mary is smiling.
- Mary is not smiling.

What about two people named Mary?

RTE II

- Mary was smiling.
- Mary was not smiling.

Two different moments?

- Mary is smiling.
- She is not smiling.

What is the referent for 'she'?

RTE III

We need a precise task definition.

But remember, we didn't start trying to solve RTE, we want to test the MRS Logic.

What can we explore from the experiments so far? Considering the LLMs?

ambiguity and state explosion

"A man in a white shirt and sunglasses and a man in a black shirt and sunglasses are sitting at a table with four beer bottles"

6628 readings. The first reading (MRS) has 2.166.400 possible quantifiers scopes solutions.

We take only the first 2x2 (readings and solutions) for each sentence. 4x4 possible analysis for each pair of sentences.

We need to remove logical equivalent formulas for each sentence. This is hard. Translate to prenex normal form, but it can increase exponentially the size of the formula.

The scope solver (Utool) could do that, but yet to be studied.

SPARQL Validation I

A vast amount of data is available in KGs, such as Wikidata or PubChem.

Both resources have a web interface and SPARQL endpoints for queries. However, a chat-like interface would be helpful for chemists.

QA is a well-studied subject in NLP. Some experiments with LLM are encouraging, but hallucinations are unacceptable for technical domains.

SPARQL Validation II

Some examples of questions we want to be able to answer:

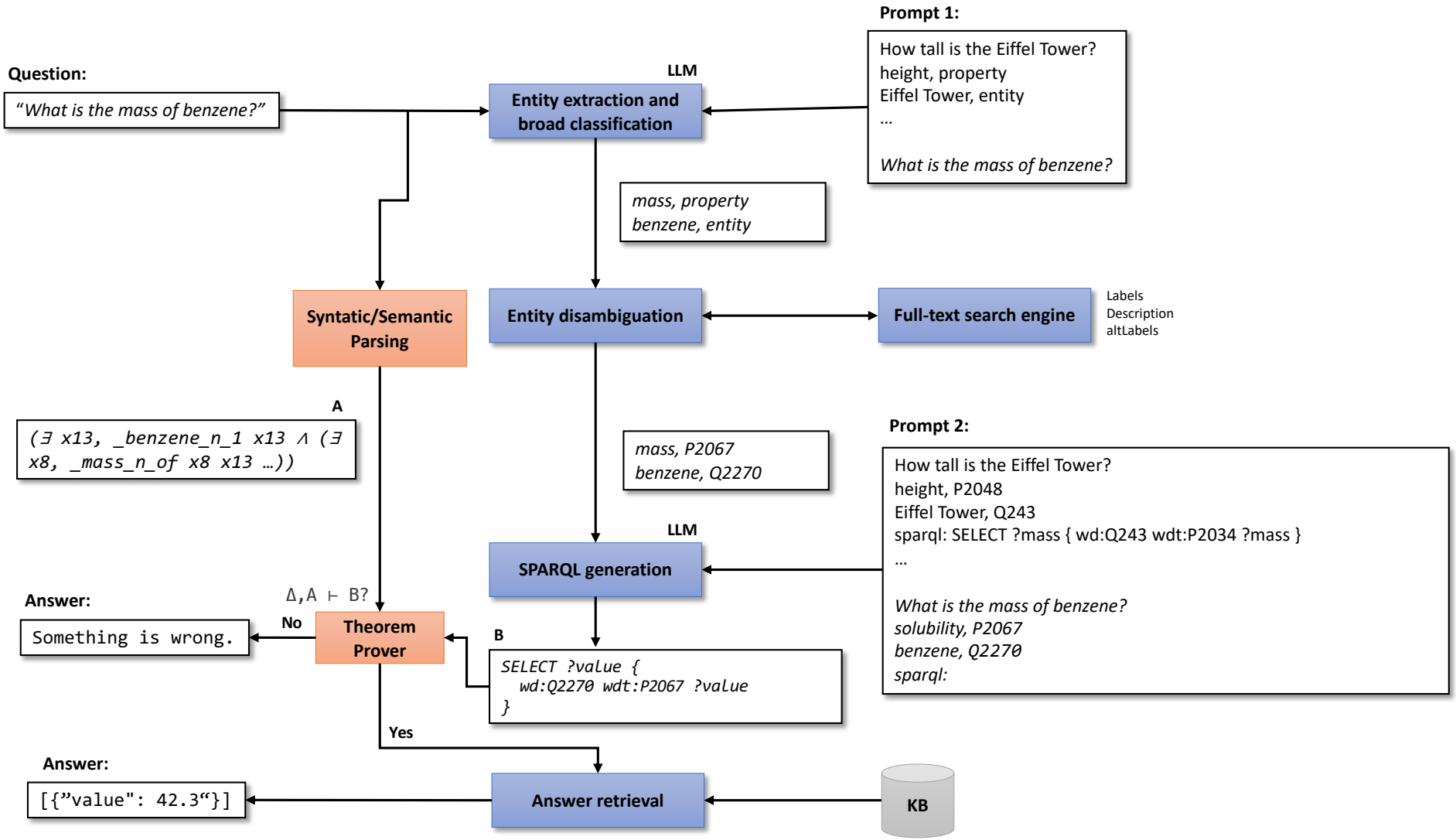
(1)
a. What is the mass of benzene?
b. Give me the benzene's toxicity.
c. What chemical compounds have less than 0.07 g/kg of solubility?
d. What is the electric dipole moment of the allyl alcohol?
e. What is the mass of the compound with InChIKey UHOVQNZJYSORNB-UHFFFAOYSA-N?

SPARQL Validation III

Despite being answerable by SPARQL queries involving only one or a few triple patterns, we already have some challenges:

- factoid questions.
- syntactic structure can vary greatly.
- properties are hardly mentioned by their labels (e.g., toxicity).
- property values are usually measured in complex units (e.g., LD50 Rat oral 3530 mg/kg).
- chemicals can be identified in various ways.

The Context


Question:
"What is the mass of benzene?"
LLM
Entity extraction and broad classification
Prompt 1:
How tall is the Eiffel Tower?
height, property
Eiffel Tower, entity
...
What is the mass of benzene?
mass, property
benzene, entity
Syntatic/Semantic Parsing
Entity disambiguation
Full-text search engine
Labels
Description
altLabels
A
(∃ x13, _benzene_n_1 x13 ∧ (∃ x8, _mass_n_of x8 x13 …))
mass, P2067
benzene, Q2270
Prompt 2:
How tall is the Eiffel Tower?
height, P2048
Eiffel Tower, Q243
sparql: SELECT ?mass { wd:Q243 wdt:P2034 ?mass }
...
What is the mass of benzene?
solubility, P2067
benzene, Q2270
sparql:
LLM
SPARQL generation
Answer:
Δ,A ⊢ B?
No
Something is wrong.
Theorem Prover
B
SELECT ?value {
wd:Q2270 wdt:P2067 ?value
}
Yes
Answer:
[{"value": 42.3"}]
Answer retrieval
KB

The formal validation

parsing, semantic representation, logical reasoning

```
1  SELECT ?v WHERE {
2    wd:Q2270 wdt:P2067 ?v .
3  }
```

$$\exists\, v,\ \text{«wdt:P2067»}\ \text{«wdt:Q2270»}\ v \tag{1}$$

$$\exists\, x_{13},\ \textit{\_benzene\_n\_1}\ x_{13}\ \wedge\ (\exists\, x_8,\ \textit{\_mass\_n\_of}\ x_8\ x_{13} \wedge\ (\exists s\ x_3,\ \textit{thing}\ x_3\ \wedge\ (\exists\, e_2,\ \textit{\_be\_v\_id}\ e_2\ x_3\ x_8))) \tag{2}$$

So we should be able to prove:

$$\Delta, \alpha_2 \models \alpha_1$$

next steps

From Python to Lean, code in the initial stage.

Proper evaluation of ERG parse rank model. How?

Use WordNet, from predicates to senses, maybe using the glosstag corpus.

How do we add axioms by demand? Works from Johan Bos. LLM?

Tips for dealing with ambiguity? Preferences from LLM?

dependent types I

“Corrosion prevented continuous contact.”

(Entailment, intensionality and text understanding. Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, Daniel G. Bobrow)

Anaphora examples. Proof-theoretic semantics. Previous work on Fracas test suite (here and Formal Semantics in MTT)

dependent types II

"Any farmer who owns a donkey beats it"

$$\prod_{z:\sum_{x:Farmer}(\sum_{y:Donkey} Owns(x,y))} Beats(p(z), p(q(z)))$$

The $z = (x, b)$ where x is a farmer and b is a proof-object for "exists a donkey that m owns it".

vs the FOL "For all farmers and for all donkeys, if the farmer owns the donkey then he beats it."

$$\forall x \forall y (Farmer(x) \wedge Donkey(y) \wedge Own(x, y) \rightarrow Beat(x, y))$$

dependent types III

I want to explore Dependent Type Semantics.

“formal grammar begins with what is well understood formally, and then tries to see how this formal structure is manifested in natural language, instead of starting with natural language in all it unlimitedness and trying to force it into some given formalism.”

(Type-theoretic Grammar, Aarne Ranta - preface describing the conversation with Per Martin-Löf)

https://www.grammaticalframework.org/

dependent types IV

FOL tends to lean heavily on the supply of a reasonable domain. But when quantification occurs over a variety of domains? We need to imagine some vast pool of individuals to pull out various people, times, and events.

"Everyone has at some time seen some event that shocked them"

Small wonder CS has looked to the discipline of types. Just as we want a "person" in response to "Who?", and a "place" in response to "Where?", programs need to compute with terms of the right type.

(The n-Category Café, Feb 11, 2020)

dependent types V

We want to build on top of modern tools like Lean Theorem Prover and Functional Language.

. . . , but we started with Python! Two libraries ULKB and MRS-Logic.

What about the LLM? Help with reasoning? Help on the language interpretation? Help on the reasoning steps?

Demo about the Agatha Puzzle! (e.g. VS Code)

Thank you!

arademaker@gmail.com