Scalable, Efficient and Precise Natural Language Processing in the Semantic Web

Thesis Defence

Shane Peelar

peelar@uwindsor.ca

School of Computer Science
University of Windsor
Windsor, Ontario, Canada

December 17 2020

Doctoral Committee

External Examiner: Dr. Diana Zaiu Inkpen

School of Electrical Engineering and Computer Science
University of Ottawa

External Reader: Dr. Richard Caron

Department of Mathematics and Statistics

Internal Reader: Dr. Jianguo Lu

School of Computer Science

Internal Reader: Dr. Pooya Moradian Zadeh

School of Computer Science

Advisor: Dr. Richard Frost

School of Computer Science

Outline

1. Introduction
2. Demonstration
3. Our Approach
4. Original Contributions
5. Conclusions
6. Future Work

Introduction: Motivation

- The Internet of Things (IoT) has become an established presence in everyday life
  - lightbulbs, televisions, refrigerators, etc …
- The Semantic Web offers an opportunity to unify interaction with these devices
  - A platform for storing curated domain-specific information
- However, there are a lack of expressive user-friendly ways of interacting with the Semantic Web
  - Existing query languages are too low level

Introduction: The Semantic Web

- Commonly referred to as “Web 3.0”
- First coined by Tim Berners-Lee in 2001 [3]
- A set of standards by the W3C for interacting with remote *triplestores* [2]
  - Triples typically described as *subject* - *predicate* - *object*, all Uniform Resource Identifiers (URIs)
  - Traditionally entity-based
  - Example: `<hall> <discovered> <phobos> .`
- Advantages:
  - No web scraping required, all machine readable
  - Simple representation facilitates processing (RDF) [5]
  - Hierarchy of information can be described in OWL [18]
- Requires *reification* for cross referencing

Introduction: The Problem

Problem Statement

There is a strong need for expressive user-friendly modes of interaction to the Semantic Web

Why is this problem important?

- Accessibility – people who have disabilities should be able to use the Semantic Web
- Ease of Interaction – users should not need to be experts in software
  - Should augment an expert's ability to look for information with provable results

Introduction: The Problem

- One approach: use a Natural Language Query Interface (NLQI)
- Users with certain disabilities particularly stand to benefit from a NL approach
  - Modalities: speech, text
- Voice Assistants are now commonplace, especially on smartphones
  - Find related information (Machine Learning)
  - Privacy concerns due to centralization
  - Can't be used in confidential settings

Introduction: The Problem

A NLQI approach should:

- Give users control of their data
  - Support local knowledge-bases
- Work locally using commodity low-power hardware
  - NL queries should not need a powerful server for evaluation
- Accommodate highly sophisticated queries
- Be tolerant of ambiguity
- Give the user confidence in the correctness of the result

Introduction: The Problem

- NL is inherently syntactically ambiguous:

  "discovered a moon that spins in 1877"

  - `discovered (a (moon that spins)) [in 1877]`
  - `discovered (a (moon that (spins [in 1877])))`
- Semantic ambiguity also exists: consider the word "depart"
- Users may not enter grammatically correct sentences
- Users may misspell words
- Users may not clearly state what they mean
- The situation does not look good for arbitrary user input!

Introduction: Constraining the Problem

- But for NLQIs, the situation isn't as bad as it seems
  - The set of possible inputs are constrained to Q&A-type sentences
  - A full understanding of the underlying NL is not needed
- NLQIs may be *wide* or *narrow* in scope
- Machine Learning is effective for *wide*-scoped interfaces
- But how can we handle highly specific queries, such as: `how many moons that orbit a red planet were discovered by nicholson`?
- The goal of this research is to produce a framework for NLQIs that can be used with expert systems and highly domain specific databases for precise *narrow* queries

Introduction: Compositional Semantics

- One approach is to use a NLQI based on a *Compositional Semantics* (CS)
- The idea: each word in the query has a corresponding mathematical *denotation* that allows the query to be treated as a mathematical expression
  - `who discovered a moon` $\Rightarrow$ `who (discovered (a moon))`
  - `hall discovered a vacuumous moon in 1877`
    $\Rightarrow$ `hall (discovered (a (vacuumous moon)) [in 1877])`
- First described by Richard Montague [6]
  - "*I reject the contention that an important theoretical difference exists between formal and natural languages.*" (Montague, 1970)
  - The denotations are formally described in the Typed Lambda Calculus

Introduction: Related work

Previous attempts at building expressive Natural Language Query Interfaces:

- ORAKEL (2004) [4]
- QuestIO (2008) [23]
- AutoSPARQL (2011) [17]
- SQUALL (2014) [7]

Introduction: Thesis Statement

Thesis Statement

A scalable, efficient, expressive and precise method for processing natural-language queries to the semantic web can be built using a compositional semantics.

Need to address:

- Scalability: needs to be able to query very large triplestores
- Efficiency: needs to be able to run on low power hardware
- Expressiveness: handle superlatives, prepositions, comparatives, negation...
- Precision: verification of result

Why the thesis is non-obvious:

- Because of criticisms and examples of non-compositionality in language

Demonstration

A live demonstration of our approach is available at this URL:

`http://speechweb2.cs.uwindsor.ca/solarman/demo_sparql.html`

Some example queries that can be handled include:

- `what discovered no moon in 1877`
- `what discovered a non moon`
- `allen discovered no moon at no places`
- `what discovered the most moons using no telescopes`
- `phobos and deimos were not discovered by not hall`
- `a person does not exist`

Our Approach

- The scoping for the denotations is determined by parsing
  - Ambiguous queries will have multiple valid parses
  - Semantic ambiguity is permitted – a word may have multiple possible denotations
- Let users decide what they mean, rather than try to guess it
  - Expose the possible meanings/parses through the interface
- Use an *event-based* view of information [12]
  - Results are *auditable* since the result contains the events that justify their inclusion
- Constrain the problem to expert systems for domain-specific applications
  - Let the *wide*-scoped interfaces delegate to *narrow* systems where appropriate
- Query is an expression in lambda calculus which we then evaluate formally

Our Approach: The FDBR

- The *Function Defined by a Binary Relation*
- The idea: convert an arbitrary BR into a function by grouping items in the domain together with the sets of elements they map to in the codomain
- For example:

$$discover\_rel_{subject \rightarrow object} = \{(e_{hall}, e_{phobos}), (e_{hall}, e_{deimos}), \ldots\}$$

$$\text{FDBR}(discover\_rel_{subject \rightarrow object}) = \{(e_{hall}, \{e_{phobos}, e_{deimos}\}), \ldots\}$$

- First used by Frost et. al in 1989 [8] to provide a denotation for binary transitive verbs in MS

Original Contributions

In the process of researching the problem, Dr. Frost and I published papers addressing aspects of the thesis statement:

- "An Extensible Natural-Language Query Interface to an Event-Based Semantic Web Triplestore" [13]
- "A New Data Structure for Processing Natural Language Database Queries" [14]
- "A Compositional Semantics for a Wide-Coverage Natural-Language Query Interface to a Semantic Web Triplestore" [21]
- "A New Approach for Processing Natural-Language Queries to Semantic Web Triplestores" [20]
- "Accommodating Negation in an Efficient Event-Based Natural Language Query Interface to the Semantic Web" [22]

Original Contributions: Superlatives

- Our approach can accommodate superlative phrases such as “most” and “the most”, including within chained prepositional phrases
  - We take “most” to mean “greater than half”
  - We take “the most” to mean “more than anything else”
- For example:
  - “`who discovered most moons that orbit mars`”
  - “`who discovered the most moons that orbit jupiter`”
- Ferré (SQUALL) in 2014 stated no NLQIs can handle this [7]

Original Contributions: Query Evaluation

How are queries evaluated?

- The event semantics define a filter over the triplestore
  - They take in a triplestore as an argument and return answers with respect to those triples
- The "Getts" semantics:
  1. Determine exactly what information is required from the triplestore
  2. Optimize those into as few queries as possible
- These are joined into a 2-tuple, using *biapplicative bifunctor* application (operator $\ll\!*\!\gg$) to evaluate both semantics in parallel

`evaluate` takes the queries formed by the Getts semantics and executes them on the triplestore, passing the results into the event semantics in order to obtain an answer.

Original Contributions: Query Evaluation

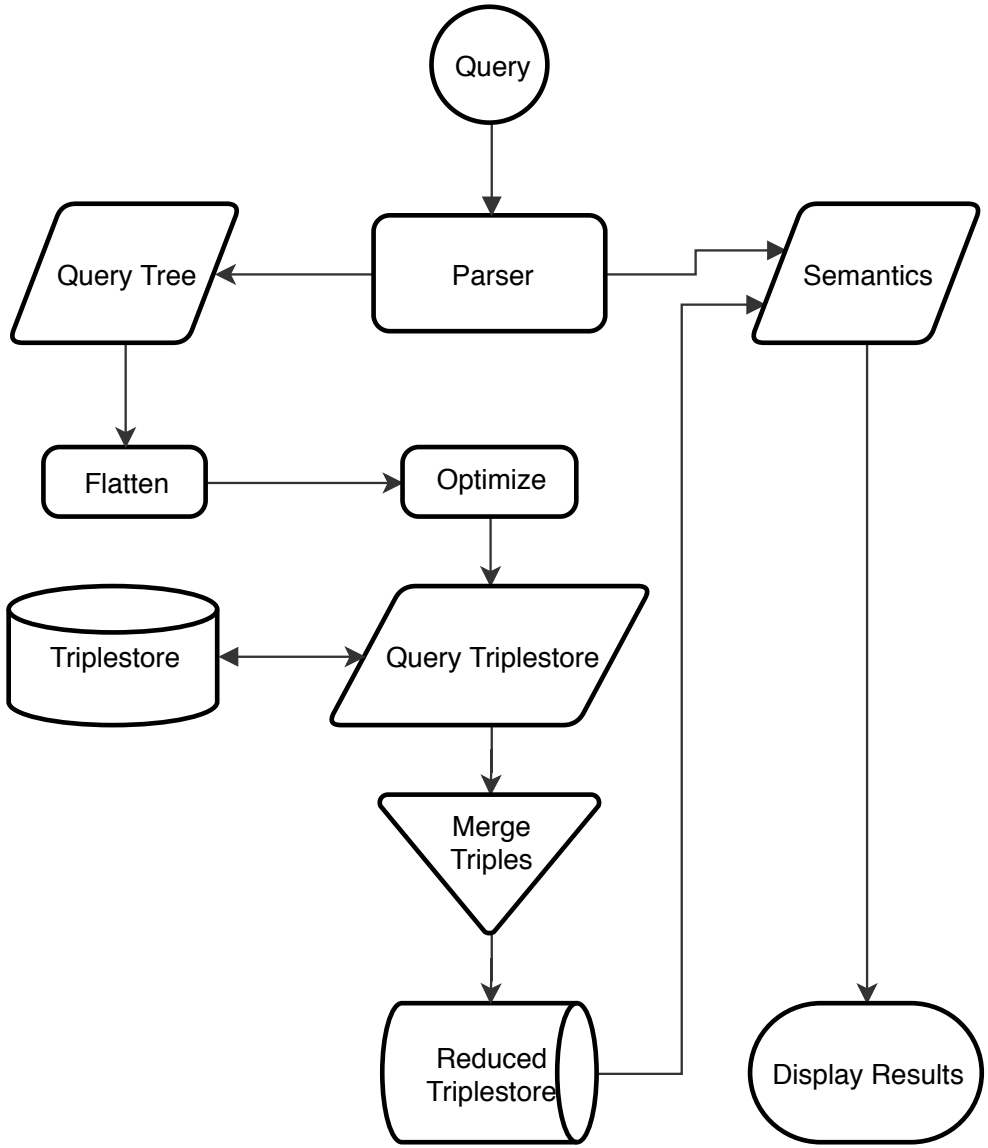


Figure 1: System Architecture

Original Contributions: Query Evaluation

- We've shown that it is possible to apply memoization to the query evaluation process itself
- The "Getts" semantics assigns a unique name to each subexpression of a query
- This can be used to re-use results from already evaluated sub-expressions, including in highly ambiguous queries
- Can also be used to compute and store FDBRs offline for faster retrieval

Original Contributions: The $n^2 - n$ Functions Defined by an n-ary relation

- We've shown how to accommodate n-ary transitive verbs in a compositional manner
  - The new idea: Convert n-ary relations to binary relations, then apply the existing FDBR machinery
- This makes it possible to query from subject to object, subject to implement, object to year, etc. $n(n-1) = n^2 - n$ choices.

Table 1: The "Discover" Relation

subject	object	date	implement	location
...	...	...	...	...
hall	phobos	1877	refractor_telescope_1	us_naval_observatory
...	...	...	...	...

Original Contributions: Applicability to Relational Databases

- We've shown how our approach can be adapted to relational databases in addition to Semantic Web triplestores
- There is a direct correspondence between event-based triplestores and relational databases:
  - Event identifier corresponds to the Primary Key
  - Event type corresponds to the relation or table
  - Event roles correspond to the columns of the relation
- This is owing to the triplestore querying primitives used by the Getts semantics allowing for independence from the triplestore query language itself

Original Contributions: Negation

- We've shown how to accommodate negation with our approach in cases where the Closed World Assumption holds
- The new idea: The notion of the complement of an FDBR
  - Track the cardinality of FDBRs and the cardinality of the set of all entities during a query
  - Extend the FDBR operations (union, intersect) to support complements
  - Requires only the cardinality, not the actual complement, to evaluate many queries
  - "drop-in" denotations for "not", "no", and "non"
  - Denotation for transitive verbs supporting superlatives, prepositional phrases, and negation

Conclusions

- We've tackled many features of English that are non-compositional including superlatives, chained prepositional phrases, comparatives, etc... (Expressiveness and Precision)
- We've improved our query evaluation computational complexity to polynomial time (Scalability)
- Works on low power consumer routers (Efficiency):
  - Prototype: `https://solarman.inbetweennames.net/`
- Works directly within the browser with WebAssembly*
  - Prototype: `https://speechweb2.cs.uwindsor.ca/solarman-wasm/`
  - Goal: make the web browser a first class citizen of the Semantic Web

Future Work

- Our approach is now ready to be directly benchmarked against other NLQIs
  - Question Answering over Linked Data (QALD)
  - TREC
  - YAGO QA benchmark
- Non-event based triplestores (Timbr.ai, OWL approaches)
  - We have obtained industry interest in our approach
  - Want to use DBPedia directly as a knowledge store
- Evaluation
  - Need to conduct a study involving users for evaluation
  - Ideally benchmarking using established metrics
  - Want to know what kinds of questions will be people be asking
  - Aimed for ISWC 2021

Thank you for attending!
Questions/comments?

References

[1] T. W. W. W. C. (W3C). *RDF 1.1 N-Triples*. https://www.w3.org/TR/n-triples/. [Online; accessed 11-November-2016]. 2014 (cit. on p. 40).

[2] T. W. W. W. C. (W3C). *RDF 1.1 Semantics*. https://www.w3.org/TR/rdf11-mt/. [Online; accessed 06-September-2016]. 2014 (cit. on p. 5).

[3] T. Berners-Lee, J. Hendler, O. Lassila, et al. "The Semantic Web". In: *Scientific american* 284.5 (2001), pp. 28–37 (cit. on p. 5).

[4] P. Cimiano, P. Haase, J. Heizmann, and M. Mantel. *Orakel: A portable natural language interface to knowledge bases*. Tech. rep. Technical report, Institute AIFB, University of Karlsruhe, 2007 (cit. on p. 12).

References (con't)

[5] W. W. W. Consortium. *Resource Description Framework (RDF) Model and Syntax Specification.* http://www.w3.org/TR/PR-rdf-syntax/ (cit. on p. 5).

[6] D. Dowty, R. Wall, and S. Peters. *Introduction to Montague Semantics*. Dordrecht, Boston, Lancaster, Tokyo: D. Reidel Publishing Company, 1981 (cit. on pp. 11, 51).

[7] S. Ferré. "SQUALL: The expressiveness of SPARQL 1.1 made available as a controlled natural language". In: *Data & Knowledge Engineering* 94 (2014), pp. 163–188 (cit. on pp. 12, 18).

[8] R. Frost and J. Launchbury. "Constructing natural language interpreters in a lazy functional language". In: *The Computer Journal* 32.2 (1989), pp. 108–121 (cit. on pp. 16, 42, 44).

References (con't)

[9] R. A. Frost. "Realization of natural language interfaces using lazy functional programming". In: *ACM Computing Surveys (CSUR)* 38.4 (2006), p. 11 (cit. on p. 53).

[10] R. A. Frost, J. Donais, E. Matthews, and R. Stewart. "A denotational semantics for natural langauge query interfaces to semantic web triplestores". In: *Submitted for publication* (2014) (cit. on p. 44).

[11] R. A. Frost, J. Donais, E. Matthews, and W. Agboola. "A Demonstration of a Natural Language Query Interface to an Event-Based Semantic Web Triplestore". In: *ESWC*. Springer LNCS Volume 8798. 2014, pp. 343–348 (cit. on p. 45).

[12] R. A. Frost, J. Donais, E. Matthews, W. Agboola, and R. Stewart. "A Denotational Semantics for Natural Language Query Interfaces to Semantic Web Triplestores". In: *poster paper in proc. of ESWC 2104*. 2014 (cit. on p. 15).

References (con't)

[13] R. A. Frost and S. M. Peelar. "An Extensible Natural-Language Query Interface to an Event-Based Semantic Web Triplestore". In: *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018*. Ed. by K.-S. Choi, L. E. Anke, T. Declerck, D. Gromann, J.-D. Kim, A.-C. N. Ngomo, M. Saleem, and R. Usbeck. Vol. 2241. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 1–16. URL: http://ceur-ws.org/Vol-2241/paper-01.pdf (cit. on p. 17).

References (con't)

[14] R. A. Frost and S. M. Peelar. "A New Data Structure for Processing Natural Language Database Queries". In: *Proceedings of the 15th International Conference on Web Information Systems and Technologies, WEBIST 2019, Vienna, Austria, September 18-20, 2019*. 2019, pp. 80–87. URL: https://doi.org/10.5220/0008124300800087 (cit. on p. 17).

[15] R. Hafiz, R. Frost, S. Peelar, P. Callaghan, and E. Matthews. *The XSAIGA Package*. https://hackage.haskell.org/package/XSaiga, 2020 (cit. on p. 46).

[16] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. "YAGO2: A spatially and Temporally Enhanced Knowledge Base from Wikipedia". In: *Artificial intell.* 194 (2013), pp. 28–61 (cit. on p. 43).

References (con't)

[17] J. Lehmann and L. Bühmann. "AutoSPARQL: Let users query your knowledge base". In: *The Semantic Web: Research and appl.* Springer, 2011, pp. 63–79 (cit. on p. 12).

[18] D. L. McGuinness, F. Van Harmelen, et al. "OWL web ontology language overview". In: *W3C recommendation* 10.10 (2004), p. 2004 (cit. on p. 5).

[19] S. Peelar. "Accommodating prepositional phrases in a highly modular natural language query interface to semantic web triplestores using a novel event-based denotational semantics for English and a set of functional parser combinators". MA thesis. University of Windsor (Canada), 2016 (cit. on pp. 37, 39, 44).

References (con't)

[20] S. M. Peelar and R. A. Frost. "A New Approach for Processing Natural-Language Queries to Semantic Web Triplestores". In: *Web Information Systems and Technologies - 15th International Conference, WEBIST 2019, Vienna, Austria, September 18-20, 2019, Revised Selected Papers*. Ed. by A. Bozzon, F. J. D. Mayo, and J. Filipe. Vol. 399. Lecture Notes in Business Information Processing. Springer, 2019, pp. 168–194. URL: https://doi.org/10.1007/978-3-030-61750-9%5C_8 (cit. on pp. 17, 38, 44).

[21] S. M. Peelar and R. A. Frost. "A Compositional Semantics for a Wide-Coverage Natural-Language Query Interface to a Semantic Web Triplestore". In: *IEEE 14th International Conference on Semantic Computing, ICSC 2020, San Diego, CA, USA, February 3-5, 2020*. IEEE, 2020, pp. 257–262. URL: https://doi.org/10.1109/ICSC.2020.00054 (cit. on pp. 17, 44).

References (con't)

[22] S. M. Peelar and R. A. Frost. "Accommodating Negation in an Efficient Event-based Natural Language Query Interface to the Semantic Web". In: *Proceedings of the 16th International Conference on Web Information Systems and Technologies, WEBIST 2020, Budapest, Hungary, November 3-5, 2020*. Ed. by M. Marchiori, F. J. D. Mayo, and J. Filipe. SCITEPRESS, 2020, pp. 83–92. URL: https://doi.org/10.5220/0010148500830092 (cit. on p. 17).

[23] V. Tablan, D. Damljanovic, and K. Bontcheva. "A natural language query interface to structured information". In: *European Semantic Web Conference*. Springer. 2008, pp. 361–375 (cit. on p. 12).

Appendix: Example query

- Example: the moons that were discovered by Nicholson
- moons: `lysithea, ananke, carme, sinope`
- implements: `hooker_telescope, refractor_telescope_2`
- `ananke, carme, lysithea-hooker_telescope`
- `sinope-refractor_telescope_2` (also used to discover `almathea`)

Appendix: Contributions to the SpeechWeb project

- Joined the SpeechWeb project in 2014
  - Other students also working on it at that time: the parser, "Gangster" semantics
  - Wrote the demonstration program used by Frost at ESWC
- Showed that chained prepositional phrases can be handled in a CS (Masters Thesis) [19]
- Showed how the word "by" can be treated like a preposition
- Showed how to accommodate n-ary transitive verbs in an event semantics
- Showed how the FDBR can be used to answer other "non-compositional" queries (superlatives, comparatives)

Appendix: Contributions to the SpeechWeb project

- Showed it is possible to memoize the semantics to improve the asymptotic complexity of evaluation significantly [20]
- Showed that it is possible to accommodate negation efficiently
- Improved X-SAIGA parser so that it can be used in the browser
- Showed that the approach works on low power hardware (consumer routers)
- Voice-enabled interface supporting speech input and output
- Tackled every non-compositional feature that has been raised by others

Appendix: Master's thesis contributions

- It was realized by Peelar and Frost that the FDBR could be used to answer many kinds of queries in NL
- In 2016, Peelar showed that it is possible to unify the treatment of complex English constructs (UEV-FLMS) using the FDBR [19]
    - A query can be thought of as a filter over the underlying database
- In particular, it could be used to accommodate chained Prepositional Phrases, which was a common critique of MS approaches

Appendix: Event-based triplestores

- The *subject* of a triple is the URI of an *event* in which something took place
- Data is already *reified*, extra information directly accessible
- Can add new columns and rows to a relation easily

Example (N-Triples format [1], full URI omitted):

```
<event1045> <subject>   <hall> .
<event1045> <object>    <phobos> .
<event1045> <type>      <discover_ev> .
<event1045> <year>      <1877> .
<event1045> <location>  <us_naval_observatory> .
<event1045> <implement> <refractor_telescope_1> .
```

Appendix: Querying the Semantic Web

- Many ways to query Semantic Web triplestores
  - SPARQL, Triple Pattern Fragments (Linked Data), SQL, etc …
- However, these query languages are not suitable for user-interfaces
- Example SPARQL query:

```
PREFIX sol: <http://solarman.richard.myweb.cs.uwindsor.ca#>
SELECT ?ev ?ent WHERE {
          ?ev sol:type sol:membership .
          ?ev sol:subject ?ent .
          ?ev sol:object sol:moon .
}
```

Appendix: Advantages of Compositional Semantics

CS has a few advantages when dealing with structured queries:

- Extensible: new denotations able to be added without modifying existing ones
- Syntactic categories can be type checked: "`hall`" and "`a person`" have the same type and can be substituted
- Expressive: small set of rules define an infinite language
- Can be used directly with an Executable Attribute Grammar [8] to "execute" NL as if it were a programming language
- Derivations directly in the language itself
- The result of the query is as correct as the underlying database

Appendix: Why CS isn't used more

- So, why aren't CS used more? Several reasons:
  - Complex linguistic constructs have been traditionally hard to handle
  - The CS as described by Montague is computationally intractable
  - Montague's denotation for transitive verbs is very complicated
  - Not robust with respect to user input
  - Need to translate the NL query to some kind of structured query language [16]
    - Underlying query language needs to be complex enough to support NL

Appendix: Why CS isn't used more

- There have been major advancements to address these shortcomings:
  - In 1989, Frost et. al described FLMS, a tractable version of Montague's semantics, along with a denotation for 2-ary transitive verbs [8]
  - In 2014, it was shown that direct translation to an underlying query language is not required (EV-FLMS) [10]
  - In 2016, Peelar showed that chained prepositional phrases can be handled in a CS (Masters Thesis) [19]
  - In 2020, we showed that *N*-ary transitive verbs can be handled compositionally [21]
  - In 2020, we showed it is possible to memoize the semantics to improve the asymptotic complexity significantly [20]
  - In 2020, we now show it is possible to accommodate negation efficiently as well

Appendix: Query Evaluation

- We have shown that it is not necessary to directly translate the query to an underlying query language [11]
  - The denotation itself contains a small set of queries for the information they need
  - Batch these smaller queries together
- Three simple “query” functions used, can be implemented in any query language
  - Implementable directly in SPARQL, Triple Pattern Fragments (LDF), and even SQL

Appendix: Implementation

- Our implementation is written entirely in Haskell
- It is able to be used even on low power devices, such as consumer routers
  - `https://solarman.inbetweennames.net`
- The FDBRs are able to be generated offline or on the fly
- The semantics are completely memoized for efficient computation
- The code is completely open source, available on Hackage [15]
- Now can run entirely within the web browser:
  - `https://speechweb2.cs.uwindsor.ca/solarman-wasm/`

Appendix: Lambda Calculus

- Universal model of computation
- Consider the function: add $= (\lambda x.(\lambda y.x + y))$
- Example application:

add $3\ 4 = (\lambda x.(\lambda y.x + y))\ 3\ 4$

$\Longrightarrow (\lambda y.3 + y)\ 4$ This is an example of beta reduction

$\Longrightarrow 3 + 4$

add_three $=$ add $3 = \lambda y.3 + y$ Partial application of add returns a funct

add_three $6 \Longrightarrow$ add $3\ 6 \Longrightarrow 9$

Appendix: Montague Semantics

- Montague was the first person to propose that two phrases of the same class were of the same type
- Also first to propose that a proper noun denotes a function, not an entity
- Described semantics in the Typed Lambda Calculus
- Common Nouns are denoted by sets of entities
- Term-phrases take a set of entities and return a boolean value
  - Proper Nouns
  - Determiner Phrases (e.g, "`a moon`")

Appendix: Montague Semantics

Note: $\|w\|$ is the denotation of w

$$\|spin\| = spins\_pred$$
$$\|phobos\| = \lambda p.p\ e_{\text{phobos}}$$

$$\begin{aligned}
&\|phobos\ spins\| \\
\Longrightarrow &\|phobos\|\ \|spins\| \\
\Longrightarrow &\lambda p.p\ e_{phobos}\ \|spins\| \\
\Longrightarrow &\lambda p.p\ e_{phobos}\ spins_pred \\
\Longrightarrow &spins_pred\ e_{phobos} \\
\Longrightarrow &True
\end{aligned}$$

Appendix: Montague Semantics (con't)

$$\|every\| = \lambda p.\lambda q.\ \forall x\ (p\ x \Rightarrow q\ x)$$

$$\begin{aligned}
&\|every\ moon\ spins\| \\
\Longrightarrow &(\|every\|\ \|moon\|)\ \|spins\| \qquad \text{(from syntactic parsing)} \\
\Longrightarrow &(\lambda p.\lambda q.\ \forall x(p\ x \Rightarrow q\ x)\ moon_pred)\ spins_pred \\
\Longrightarrow &(\lambda q.\ \forall x\ (moon_pred\ x \Rightarrow q\ x))\ spins_pred \\
\Longrightarrow &\forall x\ moon_pred\ x \Rightarrow spins_pred\ x \\
\Longrightarrow &True \qquad \text{(every moon in the universe of discourse spins } (costly!))
\end{aligned}$$

Note: $\|phobos\|$ has same type as $\|every\ moon\|$

Appendix: Montague's treatment of transitive verbs

Transitive verbs are left uninterpreted until the rest of the phrase has been interpreted and reduced as far as possible.
The expression is then rewritten to another lambda expression using a syntactic sigma σ rule. See page 216 of *Dowty, Wall and Peters 1981*[6] .

- Complicated and difficult to implement
- No explicit denotation for transitive verbs
- Therefore no denotation, for example, for "discovered phobos"

Appendix: SQUALL

- A Constrained Natural Language (CNL) for the Semantic Web
- Translates NL queries directly to SPARQL using a CS
- Limited support for prepositions, superlatives
- Coverage of English is limited to non-chained constructs
- “Which author has affiliation...”, “Which author is affiliated with...”

Appendix: Blackburn and Bos approach to Transitive Verbs

$$\|discover\| = \lambda z.z(\lambda xy.discover\_pred(y, x))$$

Works for 2-place transitive verbs (see *Frost ACM Surveys*[9])

Appendix: Blackburn and Bos approach to Transitive Verbs (con't)

MS: Direct Evaluation of NL Queries w.r.t. Datastore

"Who (stole (a car) [in (1918 or 1920), in (a (borough (of New_York)))])?"

$$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$$

$$\lambda\ldots(\lambda\ldots(\lambda\ldots\lambda\ldots)[\lambda\ldots(\lambda\ldots\lambda\ldots\lambda\ldots),\lambda\ldots(\lambda\ldots(\lambda\ldots(\lambda\ldots\lambda\ldots)))])$$

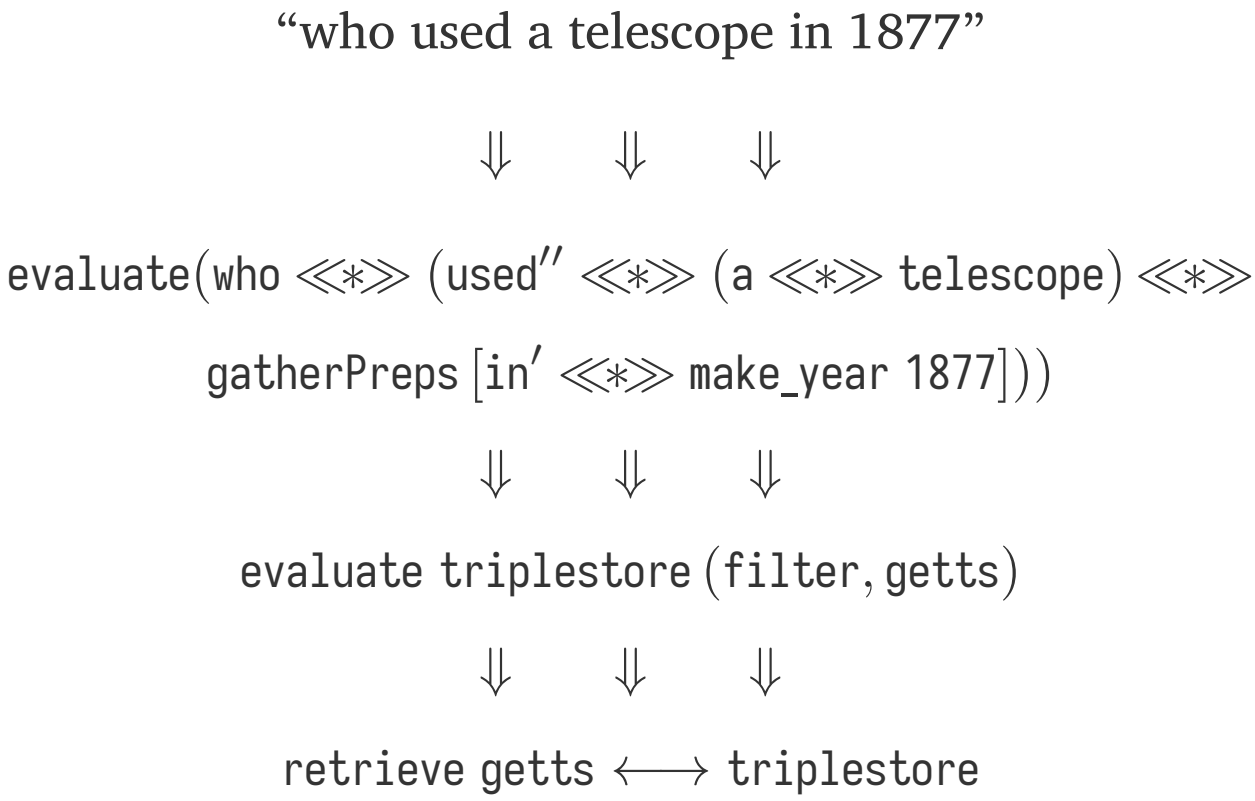
$$\Uparrow \quad \Uparrow \quad \Uparrow \quad \Uparrow$$

TRIPLESTORE

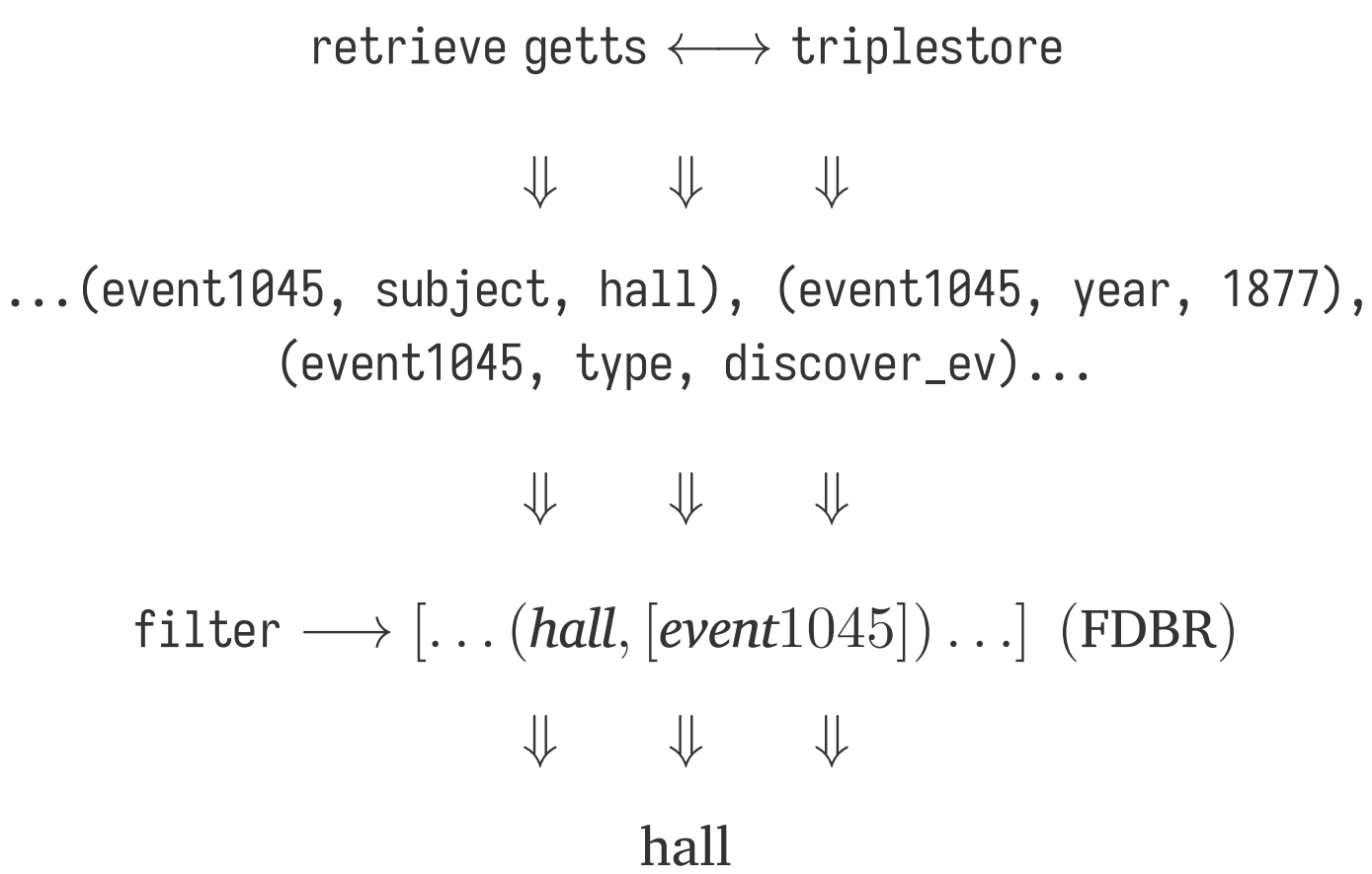
Where $\lambda\ldots$ are functions (which are the denotations of English words)

Some functions are defined $\Uparrow$ in terms of triplestore retrieval operations

Appendix: Flow of NL query process



Flow of NL query process (con't)



Appendix: Negation

- RDF is based on the Open World Assumption, which can be summarized as:
  "*The absence of evidence cannot be construed as being evidence of absence.*"
- This makes negation tricky to work with:
  - Obtaining the cardinality of a set may not be possible
  - Or it may be so large that it is not practical to work with
- But the Closed World Assumption may still hold for domain specific applications
- **How can we support use cases where the CWA holds?**

Appendix: Bi-functors

- In Category Theory, a Functor is a map between categories
- A bi-functor is a functor from a product category to another category (like a cartesian product e.g, $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R}$)
- In Functional Programming, a Functor is something that can be "mapped" over
  - `fmap :: (Functor p) => (a -> b) -> p a -> p b`
  - e.g, the elements of a list, or the nodes of a tree, or a stream of input
- A Bi-Functor takes on a similar meaning:
  - `bimap :: (Bifunctor p) => (a -> b) -> (c -> d) -> p a c -> p b d`
  - e.g, to map over a pair of values `(a,b)`, where `p = (,)`

Appendix: Applicative Functors

In Functional Programming, an Applicative Functor wraps function application

- `(<*>) :: Applicative p => p (a -> b) -> p a -> p b`
- `pure :: Applicative p => a -> p a`

E.g., useful for error handling: `maybe_function <*> maybe_value` could evaluate to either another value or a special error value `Nothing`

- Note that it must satisfy the Applicative Laws

Appendix: Bi-Applicative Bi-Functors

A Bi-applicative Bi-functor is the same concept applied to a Bi-Functor:

- `(<<*>>) :: Biapplicative p ⇒ p (a → b) (c → d) → p a c → p b d`
- `bipure :: Biapplicative p ⇒ a → b → p a b`

E.g., one could write: `(+2, +3) <<*>> (10, 20)` which evaluates to `(12, 23)`

- Note that it must satisfy the Bi-Applicative Laws

Appendix: Monads

Instances of Monads in Functional Programming follow these laws:

- Left-identity: `return a >>= f` $\equiv$ `f a`
- Right-identity: `m >>= return` $\equiv$ `m`
- Associativity: `(m >>= f) >>= g` $\equiv$ `m >>= (λx -> f x >>= g)`

Intuitively: a Monad "wraps" a computation within another one

- "For a monad `M`, a value of type `M a` represents having access to a value of type a within the context of the monad."
  – C. A. McCann

Many uses, including memoization, managing pure/impure computations, handling errors...