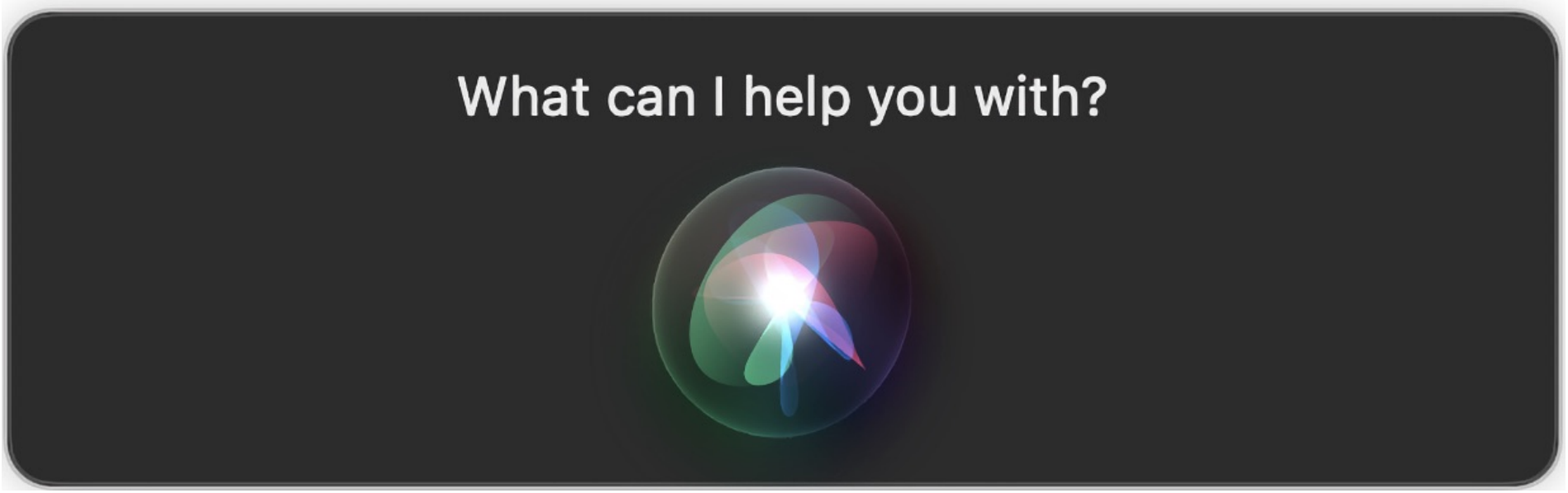
L98: Introduction to Computational Semantics

Lecture 16: Task-Specific Semantic Parsing

Weiwei Sun and Simone Teufel

Natural Language and Information Processing Research Group
Department of Computer Science and Technology
University of Cambridge

Lent 2021/22



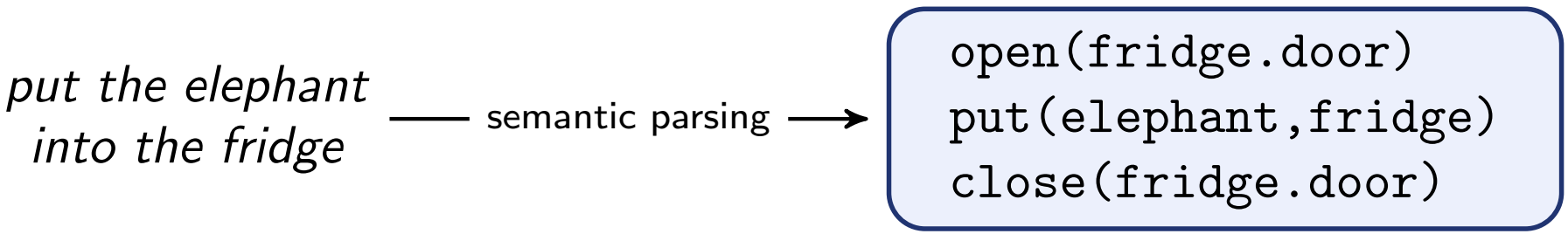
put the elephant into the fridge

Lecture 16: Task-Specific Semantic Parsing

1. Tasks
2. Datasets
3. Algorithms
4. Challenges

Tasks

How can siri put the elephant into the fridge?



How can siri put the elephant into the fridge?

put the elephant into the fridge —— semantic parsing ⟶

```
open(fridge.door)
put(elephant,fridge)
close(fridge.door)
```

Execute the code

Definitions

Wikipedia

Semantic parsing is the task of converting a natural language utterance to a logical form: a machine-understandable representation of its meaning.

nlpprogress.com

Semantic parsing is the task of translating natural language into a formal meaning representation on which a machine can act.

Tutorial on neural semantic parsing

Semantic parsing, the study of translating natural language utterances into machine-executable programs, is a well-established research area and has applications in question answering, instruction following, voice assistants, and code generation.

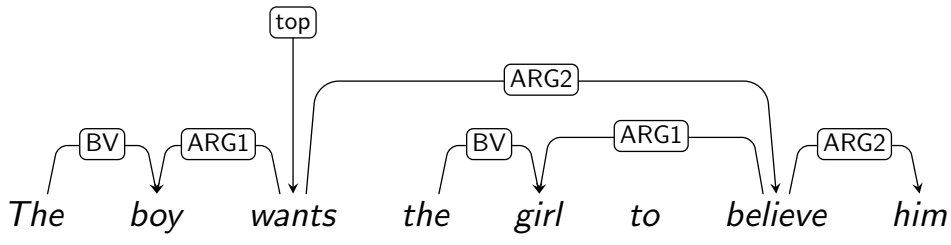
Linguistically-motivated, task-independent parsing

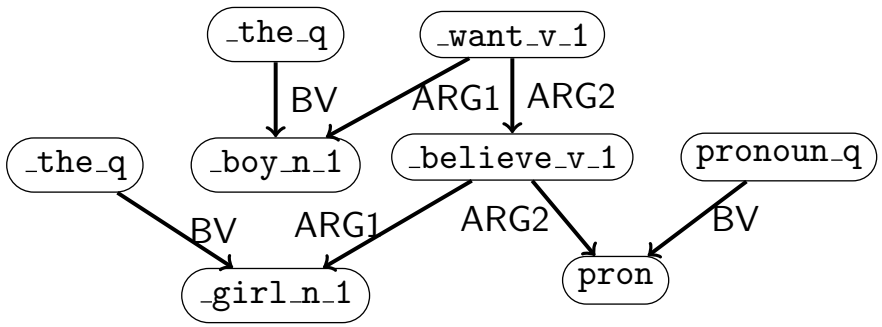
Text

The boys wants the girl to believe him

parsing

Meaning





$\lambda x.\lambda y.\text{boy'}(x) \wedge \text{girl'}(y) \wedge \ldots$

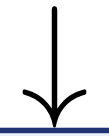
$\ldots\ldots$

! a machine-understandable representation of its meaning ✔

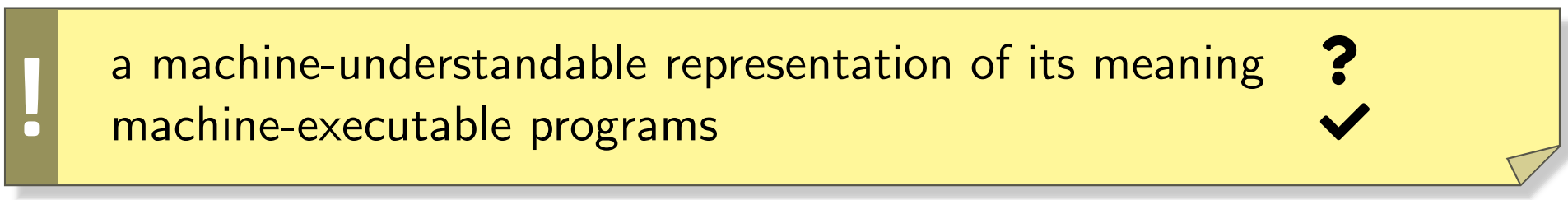
machine-executable programs ✘

Instructions

Go to the third junction and take a left



```
(do-seq (do-n-times 3
  (move-to forward-loc
    (do-until
      (junction current-loc
        (move-to forward-loc)))))
  (turn-right))
```



Code generation

Put the giraffe into the fridge

```
open(fridge.door)
take_out(elephant,fridge)
put(giraffe,fridge)
close(fridge.door)
```

! a machine-understandable representation of its meaning ✗
machine-executable programs ✓

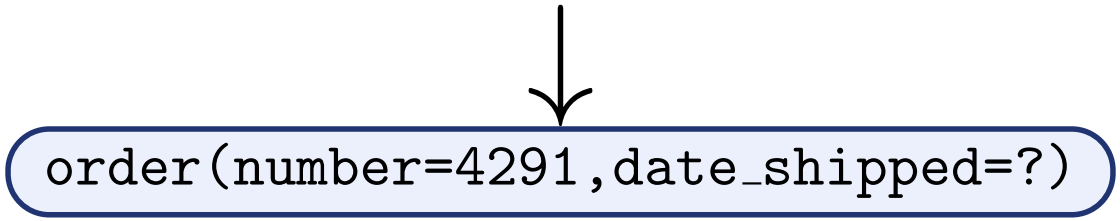
Conversational semantic parsing (Aghajanyan et al., 2020)

- *The lion king is having a birthday party, which animal doesn't go?*
- *The giraffe.*
- *Why?*
- *It's still stuck in the fridge.*

▷reasoning

Knowledge-Based Question Answering (KBQA)

Has my order number 4291 been shipped yet?



Database

Order number	Date ordered	Date shipped
4290	2/2/13	2/2/13
4291	2/2/13	2/2/13
4292	2/2/13	

! a machine-understandable representation of its meaning ✗
machine-executable programs ✓

Datasets

KBQA in 1996

GeoQuery: 880 questions with annotated logical forms

- J Zelle and R Mooney. 1996. Learning to parse database queries using inductive logic programming.
- Manually created by experts.
- Knowledge bases at the time were all of small sizes.
- Limited domains with a small number of database predicates.

Example

- *How many cities are there in the US?*
- *How many major cities are in states bordering Texas?*
- *What is the smallest state by area?*

```
answer(A,count(B,(city(B),loc(B,C),const(C,countryid(usa))),A))
```

History of TREC-QA

- organised by NIST, from 1998-2007
- Trial year 1998 (TREC-8): reverse-engineered questions
  ▷ 43: What costume designer decided that Michael Jackson should only wear one glove?
  ▷ 109: Where does Buzz Aldrin want to build a permanent, manned space station?
  ▷ 119: What nobel laureate was expelled from the Phillipines before the conference on East Timor?
  ▷ 114: How much stronger is the new vitreous carbon metal invented by the Tokyo Institute of Technology compared with the material made from cellulose?
  ▷ 139: How many people did the United Nations commit to help restore order and distribute humanitarian relief in Somalia in September 1992‘?
- More realistic questions from 1999 onwards: harvesting from search engine's query logs
- In subsequent years, harder question conditions

Voorhees, Ellen M. "The Rrec-8 question answering track report." Trec. Vol. 99. 1999.

QA with big knowledge base

WebQuestions (Berant et al., 2013): 5810 question–answer pairs

- Google Suggest API: natural questions that begin with a wh-word and contain exactly one entity. Google was founded in 1998.
- Amazon Mechanical Turk task: workers answer the questions using only the Freebase page of the questions' entities. MTurk was launched publicly in 2005.
- No annotation on formal meaning representations.

Example

Where was Barack Obama born?

Natural Questions

- Questions consist of real anonymized, aggregated queries issued to the Google search engine.
- An annotator is presented with a (question, Wikipedia page) pair. The annotator returns a (long answer, short answer) pair.
- 307,373 training examples with single annotations;
- 7,830 examples with 5-way annotations for development; and
- 7,842 examples 5-way annotated for test.
- `https://ai.google.com/research/NaturalQuestions`

Natural Questions

Example

- **Question**: *can you make and receive calls in airplanemode*
- **Wikipedia Page**: Airplane mode
- **Long answer**: Airplane mode, aeroplane mode, flight mode, offline mode, or standalone mode is a setting available on many smartphones, portable computers, and other electronic devices that, when activated, suspends radio-frequency signal transmission by the device, thereby disabling Bluetooth, telephony, and Wi-Fi. GPS may or may not be disabled, because it does not involve transmitting radio waves.
- **Short answer**: BOOLEAN:NO

A question from Quora

What are the best questions on Quora?

Here are some answers

- *What can I learn right now in just 10 minutes that could be useful for the rest of my life?*
- *What are the most amazing photos you have ever taken?*
- *What are some dumb questions you have been asked?*
- *What character's death in a movie or a TV series upset you the most?*
- *What are some important things about finance everyone should know, regardless of their profession?*
- *Why are so many people content with just earning a salary and working 9–6 their entire adult life?*

`https://www.quora.com/What-are-the-best-questions-on-Quora-8`

Questions from MathOverflow

- MathOverflow is a working mathematicians' QA platform
- PhD Stathopolous (2022): harvest MathIR queries from MathOverflow
- Idea: we need Questions whose answer is (ideally) exactly one document (or part of the document).

Question: *I am looking for a reference which shows that the following statements are equivalent for a complex vector bundle E:*

- *E is a holomorphic vector bundle.*
- *There is a Dolbeault operator $\bar{\partial}_E$, i.e. a $\mathbb{C}$ operator $\bar{\partial} : \Omega^{0,0}(E) \to \Omega^{0,1}(E)$ which satisfies the Leibniz rule and $\bar{\partial}_E^2 = 0$.*

This is stated without proof in Huybrechts' Complex Geometry: An Introduction.

Answer:

A. Moroianu gives a detailed proof on pp. 72-74 of his Lectures on Kähler geometry (Theorem 9.2), available on the internet. (The preprint has it as Theorem 3.2.)...

Looking into the actual MREC document, we can see that the solution is quite short:

THEOREM 3.2. *A complex vector bundle E is holomorphic if and only if it has a holomorphic structure $\bar{\partial}$.*

Anatomy of an MO thread

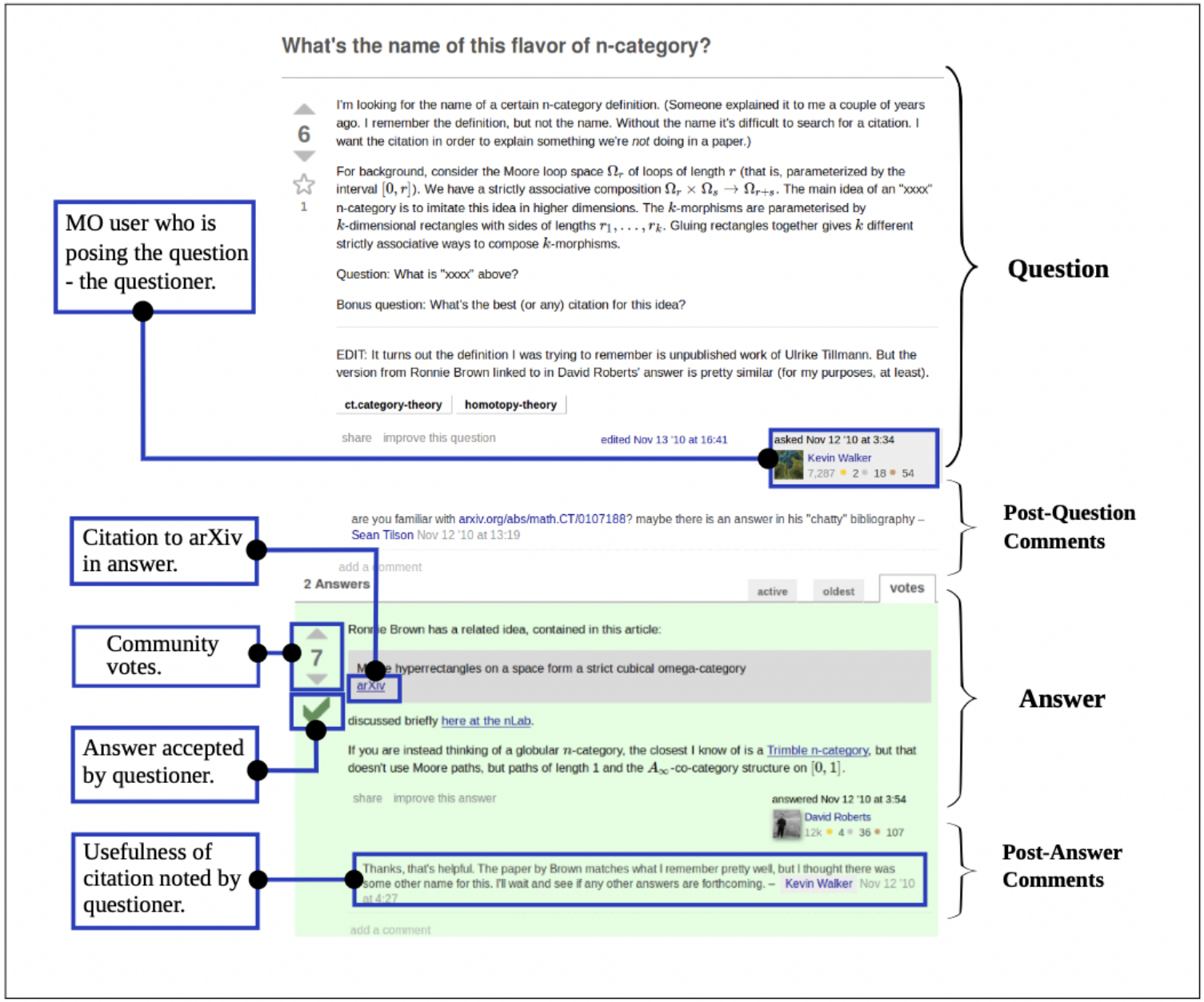


Figure 3.2: Anatomy of a MathOverflow thread (`http://mathoverflow.net/`

A chatty question

Prelude	There is a question someone (I'm hazy as to who) told me years ago. I found it fascinating for a time, but then I forgot about it, and I'm out of touch with any subsequent developments. It's a challenging question if I've gotten it right. Here it is: Suppose you have some kind of machine with two buttons, evidently designed by people with poor instinct for UI. The machine has many states in which the buttons do different things. Here are the assumptions: • There is no periodic quotient of the state space: no way to label states by an n-cycle so that both buttons advance the label by $1 mod n$. • It is not reversible: there are situations when two states merge into one. • It's ergodic: you can get from any state to any other state by some sequence of buttons. Now suppose its dinky little LCD is faded or broken, so you can't actually tell what the state it's in. (Formally, this is a finite state automaton, or an action of the free 2-generator semigroup on a finite set, and asks whether some element acts as a constant map).
SQ-1	Can anyone better identify the problem or fill in the history, and say whether it's still unsolved?
SQ-2	Is there necessarily a universal reset code, a sequence that will get you to a known state no matter where you start?

What counts as an answer here?

- They can be direct and total
- They can be synthesised from two citations
- They can be more general or specific than what the question is after
- But there must be a truth-conditional trail from question to answer (manually done by dataset creator).

Analysing the question

SEQUENCE ID	THREAD ID	DESCRIPTION	ATTACHMENT	PAGE X OF Y
56	133673	Clarification of question and answer in step-by-step manner.	2	1

1. The Question

- $TV(M)$ is an invariant of three-manifolds M.
- $TV(M) \equiv |WRT(M)|^2$ according to Kevin Walker.
- $WRT(M)$ is the Witten-Reshetikhin-Turaev invariant of M.
- Both $TV(M)$ and $WRT(M)$ depend on a choice of quantum group $U_q(g)$ and a root of unity q.
- $WRT(M)$ is defined via "an integral over all g-connections on M"
- $TV(M)$ is essentially a discretization of the path integral

Converges if we use a quantum group

non-convergent analogue is the "Ponzano-Regge" model for a classical group.

$\equiv$

- A "state sum" over all assignments of representation of $U_q(g)$ to each of the edges of a fixed triangulation of M.
- Infoseeker expects that a discretization of the path integral would be a "state sum" over all assignments of "elements" of $U_q(g)$ to each of the edges of M. (a discretized connection of M)

Analysing the answer

2. The Answer

M: a manifold
Δ: a triangulation of M
Δ^*: the dual to Δ.
g_{e^*}: group elements
f^*: dual faces.

Ponzano-Regge model is obtained from the continuum partition function

$$Z_M = \int D_\omega D_e \exp\left[\frac{i}{16\pi G}\int_M \mathrm{tr}\,(e \wedge F(\omega))\right]$$

by

① Considering a triangulation Δ (and its dual Δ^*) of M, and replacing the set of configuration variables by discrete analogs (in the spirit of lattice gauge theory).

② The connection field is replaced by group elements g_{e^*} associated to the edges e^* of Δ^* AND representing the holonomy of the connection field along these edges.

③ The frame field is replaced by Lie Algebra elements X_e associated to the edges e of Δ and representing the integration of e along these edges.

④ The curvature 2-form is now represented as group elements G_e living on the edges e (or dual faces f^*) AND obtained as the ordered product of the group elements g_{e^*} for dual edges $e^* \subset f^*$, upon the choice of a starting dual vertex on the dual face.

⑤ The discretized partition function becomes

$$Z_{PR}(\Delta) = \left(\prod_{e^*}\int_{SU(2)} dg_{e^*}\right)\left(\prod_e \int_{su(2)} dX_e\right)\exp\left[i\sum_e \mathrm{tr}(X_e G_e)\right]$$

⑥ One can then integrate over the X_e variables

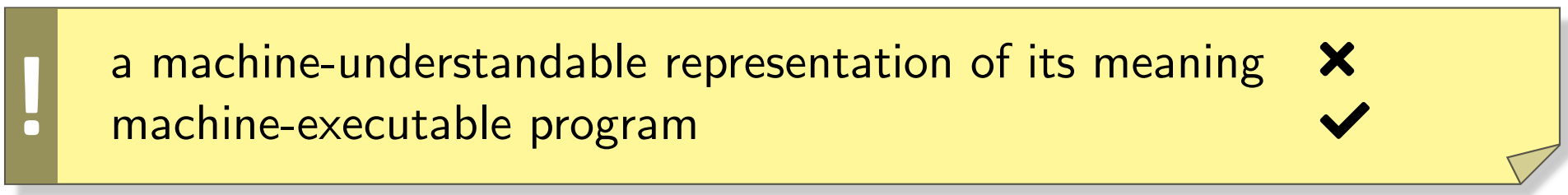
$$Z_{PR}(\Delta) = \left(\prod_{e^*}\int_{SU(2)} dg_{e^*}\right)\left(\prod_e \delta(G_e)\right)$$

Code generation: CoNaLa

A dataset crawled from Stack Overflow, automatically filtered, then curated by annotators.

Example

- *zip lists in python*
  `zip([1, 2], [3, 4])`
- *swap values in a tuple/list inside a list in python?*
  `map(lambda t: (t[1], t[0]), mylist)`
- *how to set global const variables in python*
  `GRAVITY = 9.8`



Code generation: NL2Bash

- Created by 10 Upwork freelancers who are familiar with shell scripting
- Collecting textcommand pairs from web pages such as question-answering forums, tutorials, tech blogs, and course materials.

Example

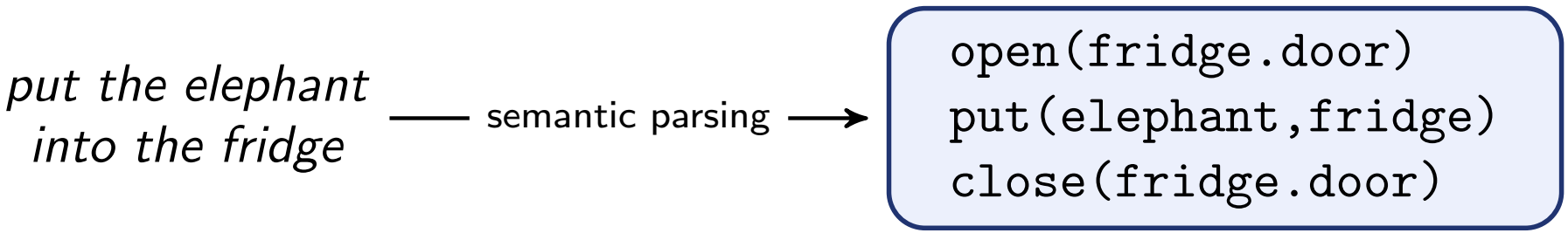
- *Display all lines containing PROBES in the current kernel's compile-time config file.*
  `grep PROBES /boot/config-$(uname -r)`
- *Add "A new line" on top of each *.py files under current directory*
  `perl -pi -e 'BEGIN { print "A new line" }' $(find . -name '*.py')`

a machine-understandable representation of its meaning ✗
machine-executable program ✓

Challenges

Grounding

A parser must know there is a magic function `put`



Grounding

A parser must know there is a magic function `put`

put the elephant into the fridge —— semantic parsing ⟶

```
open(fridge.door)
put(elephant,fridge)
close(fridge.door)
```

Properties to describe `Person` in Google Knowledge Graph

- `additionalName`: An additional name for a Person, can be used for a middle name.
- `alumniOf`: why not `studiedAt`? or for someone `cantab`?
- `birthDate`: why not `birthday`?
- `relatedTo`: The most generic familial relation.
- ...

`https://schema.org/Person`

Grounding

A parser must know there is a magic function `put`

put the elephant into the fridge —— semantic parsing ⟶

```
open(fridge.door)
put(elephant,fridge)
close(fridge.door)
```

Properties to describe `Person` in Google Knowledge Graph

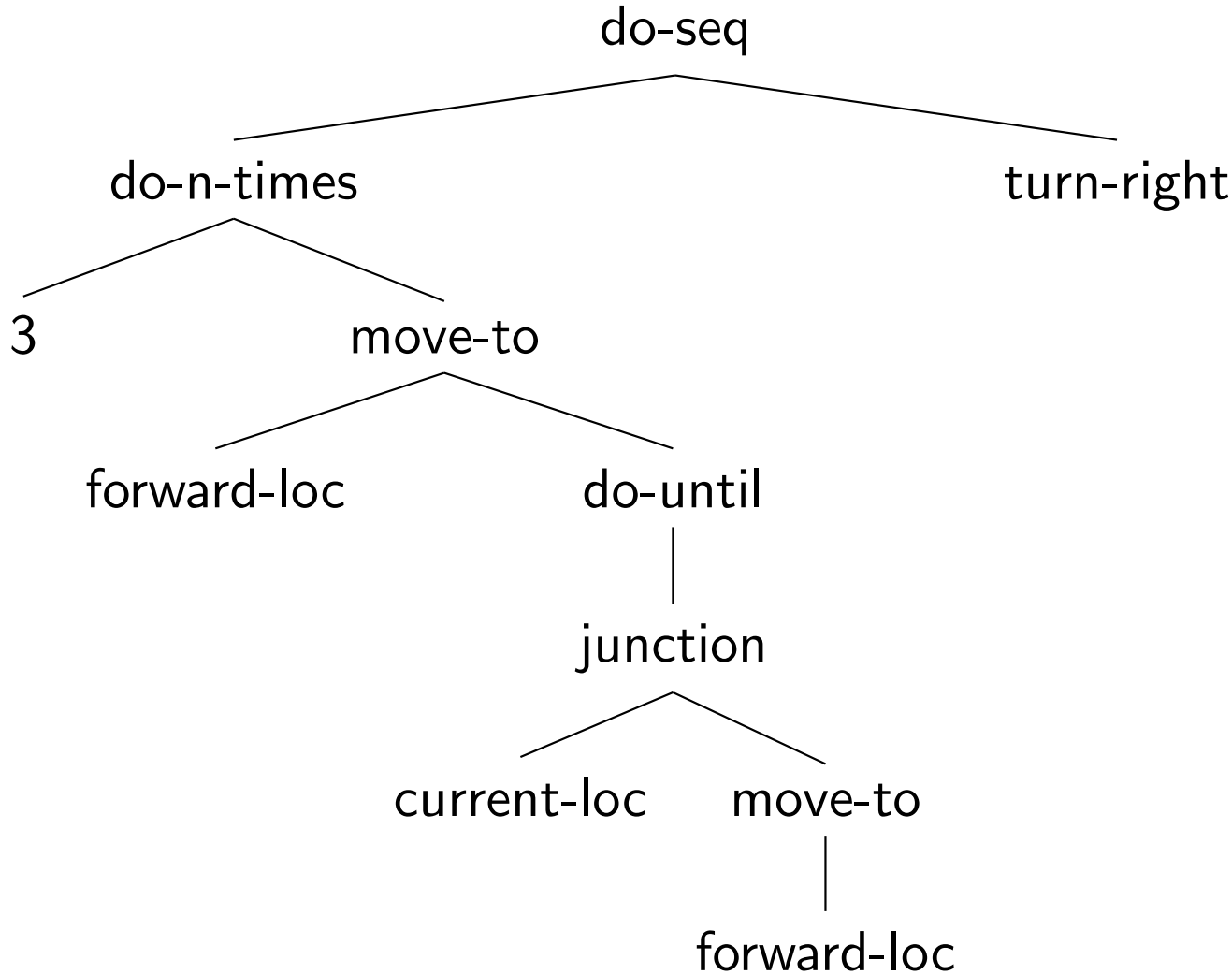
- `additionalName`: An additional name for a Person, can be used for a middle name.
- `alumniOf`: why not `studiedAt`? or for someone `cantab`?
- `birthDate`: why not `birthday`?
- `relatedTo`: The most generic familial relation.
- ...

`https://schema.org/Person`

Don't forget the size: By May 2020, Google Knowledge Graph had grown to 500 billion facts on 5 billion entities.

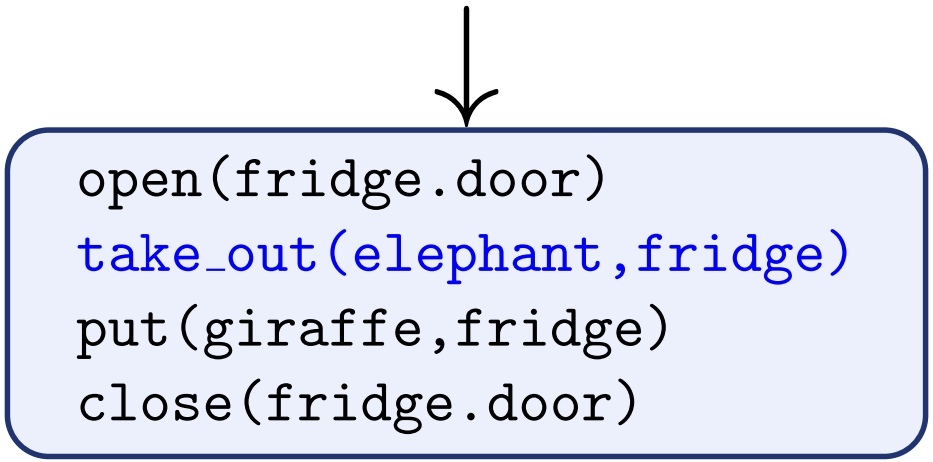
Meaning$_1$–meaning$_2$ mismatch

Go to the third junction and take a left



Context

Put the giraffe into the fridge



- To produce `take_out`, a parser must know there is an elephant in the fridge.
- How? From the history?
- From a visual sensor?

Beyond form transformation

Display all lines containing PROBES in the current kernel's compile-time config file.

```
grep PROBES /boot/config-$(uname -r)
```

A parser must know

- The path to that particular file.
- The functionality of `grep`
- ...

Beyond form transformation

*Add "A new line" on top of each *.py files under current directory*

```
perl -pi -e 'BEGIN { print "A new line" }' $(find . -name '*.py')
```

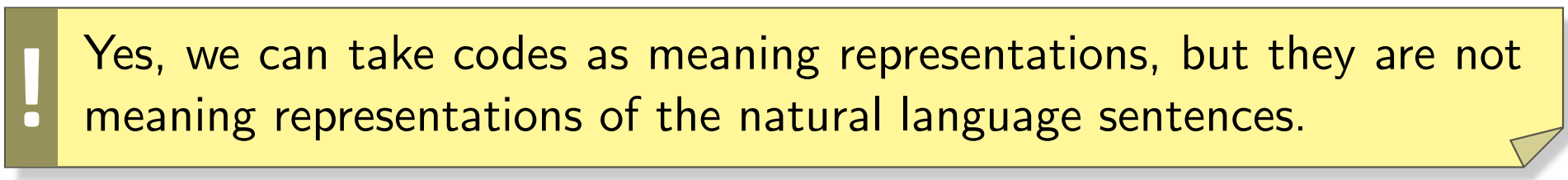
- A parser should know that before adding some information to a file, if must know where the file is.
- A parser should know that `find` can do that.
- If a parser knows `Perl`, it is great. If not, a parser can still try `Ruby`: `ruby -e ...`

Beyond form transformation

*Add "A new line" on top of each *.py files under current directory*

```
perl -pi -e 'BEGIN { print "A new line" }' $(find . -name '*.py')
```

- A parser should know that before adding some information to a file, if must know where the file is.
- A parser should know that `find` can do that.
- If a parser knows `Perl`, it is great. If not, a parser can still try `Ruby`: `ruby -e ...`



Supervision

Google Natural Questions

- Training: 307,373
- Development: 7,830
- Test: 7,842

Q1 What can we annotate? It is difficult to design a meaning representation for a specific task.

Q2 How can we annotate so many sentences? And no one says 307,373 examples are enough.

Meaning representation as latent variable

Algorithms

Programming language is also language

Many machine translation models are adopted.

Seq2seq semantic parsing

- The input is a sequence: *Add "A new line" on top of each *.py files under current directory*
- The output is also a sequence `perl -pi -e 'BEGIN { print "A new line" }' $(find . -name '*.py')`
- If the output is not a sequence, say AMR graph, one can choose linearise it.
- PENMAN notation: `(e / eat-01 :ARG0 (d / dog) :ARG1 (b / bone))`

Constraint decoding

- The output is a sequence `perl -pi -e 'BEGIN { print "A new line" }' $(find . -name '*.py')`

End-to-end systems

- Many modern (= NN-based) NLP applications don't assemble meaning piece by piece
- Instead, they are "end to end" systems (black boxes)
- Connection from one end (observation) to the other end (interpretation) is only evaluated at the very end.
- Non-black box systems are different in that they can be evaluated at internal points too, and therefore debuggable.

End-to-end systems: consequences

- If the data sets are build in a wrong way, the system will learn the wrong things or nothing.
- (And nobody notices, because they are black boxes.)
- In order to build good data sets, good knowledge about semantics and pragmatics is needed.
- Knowledge about semantic phenomena will also help you to invent better features for representation.

- *Add "A new line" on top of each *.py files under current directory*
  `perl -pi -e 'BEGIN { print "A new line" }' $(find . -name '*.py')`
- *Display process information twice, waiting one second between each, filtering out the header line.*
  `top -b -d2 -s1 | sed -e '1,/USERNAME/d' | sed -e '1,$d'`

Reading

- Pre-lecture: P Dasigi, S Iyer, A Suhr, M Gardner and L Zettlemoyer. ACL 2018 tutorial on neural semantic parsing. `https://github.com/allenai/acl2018-semantic-parsing-tutorial/`
- Post-L98: C Lee, J Gottschlich and D Roth. Toward Code Generation: A Survey and Lessons from Semantic Parsing. `https://arxiv.org/pdf/2105.03317.pdf`

About the test

- I offer a personal 10-minute one-on-one discussion for each of you
- You can choose when to have the meeting
- Possible timings: either
  - this afternoon (from 1pm)
  - on Thursday morning (from 10am)
- Please send me an email to set up your meeting slot
- You might benefit more from this meeting if you have already gathered some information
- You will also benefit more if you ask better questions

References I

Aghajanyan, A., Maillard, J., Shrivastava, A., Diedrick, K., Haeger, M., Li, H., ... Gupta, S. (2020, November). Conversational semantic parsing. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* (pp. 5026–5035). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.emnlp-main.408 doi: 10.18653/v1/2020.emnlp-main.408

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013, October). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1533–1544). Seattle, Washington, USA: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D13-1160

References II

Wong, Y. W., & Mooney, R. (2006, June). Learning for semantic parsing with statistical machine translation. In *Proceedings of the human language technology conference of the NAACL, main conference* (pp. 439–446). New York City, USA: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/N06-1056`

Wong, Y. W., & Mooney, R. (2007, June). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th annual meeting of the association of computational linguistics* (pp. 960–967). Prague, Czech Republic: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/P07-1121`