



# Neural-symbolic Programming for Explainable Complex Question Answering

Lei HOU

Knowledge Engineering Group

Department of Computer Science and Technology


Tsinghua University

[houlei@tsinghua.edu.cn](mailto:houlel@tsinghua.edu.cn)

# Complex Question Answering

## □ Question Answering (QA)

- Typical application to demonstrate the system intelligence

 **Input Questions**  
*e.g., Who is the current principal of Tsinghua University?*



**Accurate Answers**  
*e.g., LI Luming*

## □ Complex Question Answering (CQA)

- Unable to directly retrieve the answer
- Various reasoning abilities are required, e.g., multi-hop, count, comparison, and logical operations

Paragraph A: **LostAlone** were a British rock band ... consisted of *Steven Battelle, Alan Williamson, and Mark Gibson*...

Paragraph B: **Guster** is an American alternative rock band ... Founding members *Adam Gardner, Ryan Miller, and Brian Rosenworcel* began...

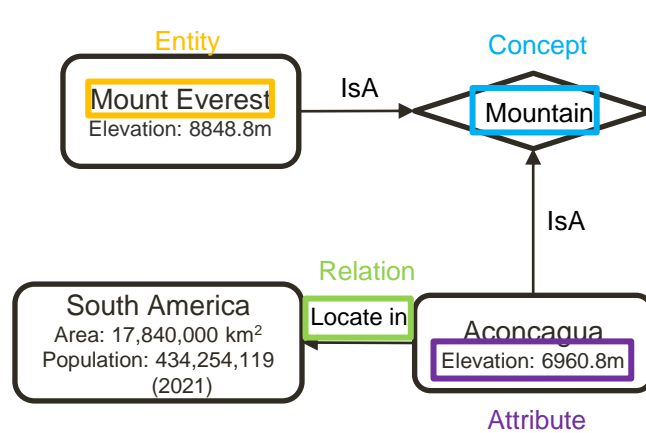
Q: Did **LostAlone** and **Guster** have the **same number of members**? (**yes**)



# Knowledge Sources for Complex QA

- CQA can utilize multiple knowledge sources

Q: Which is taller, Mount Everest or the highest peak in South America?



Knowledge Graph

The text snippet describes Aconcagua with the following annotations:

- Entity:** Aconcagua
- Concept:** mountain
- Relation:** in
- Attribute:** elevation

Text

**HLGreener**

Which is highest mountain in South America?

**ChatGLM**

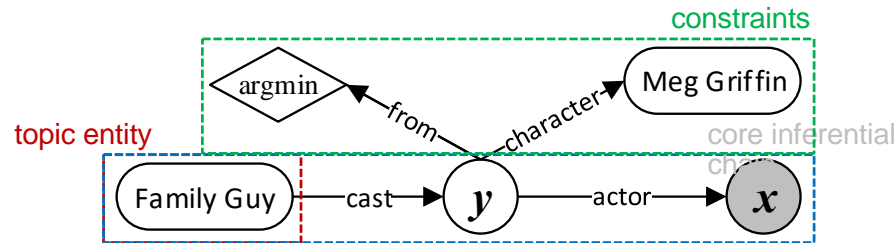
The highest mountain in South America is Aconcagua, located in Argentina. Aconcagua stands at approximately 22,841 feet (6,961 meters) above sea level, making it the highest peak in both the Southern and Western Hemispheres.

LLMs

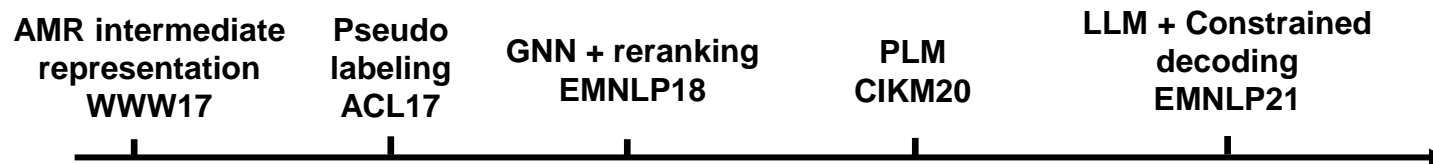
# Complex QA with KB

## □ Semantic Parsing

- Query graph generation (ACL15, EMNLP16, ACL17)

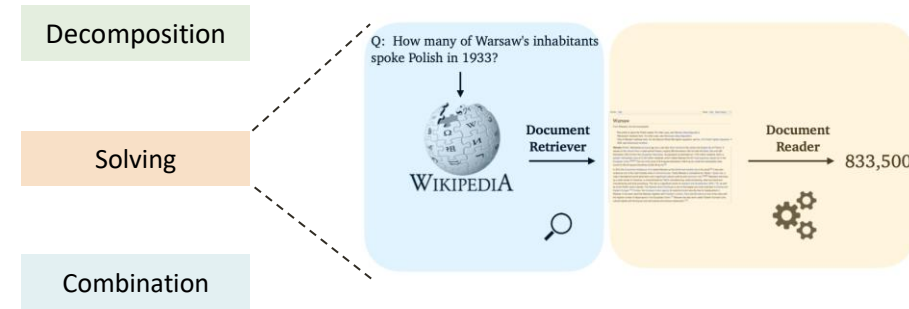
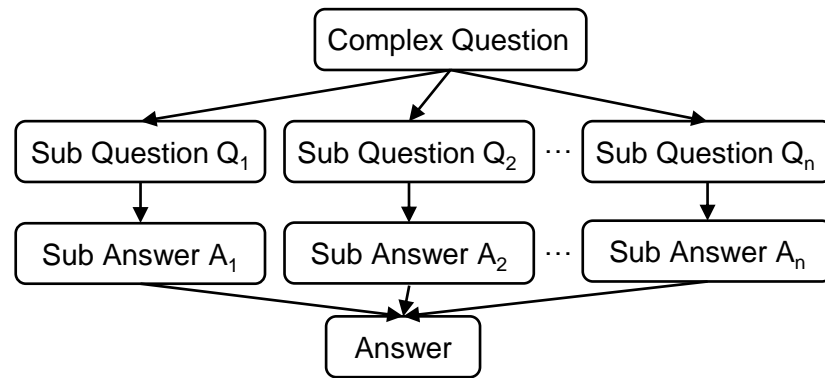


- Program induction (ACL17, IJCAL19, TACL20, EMNLP21)

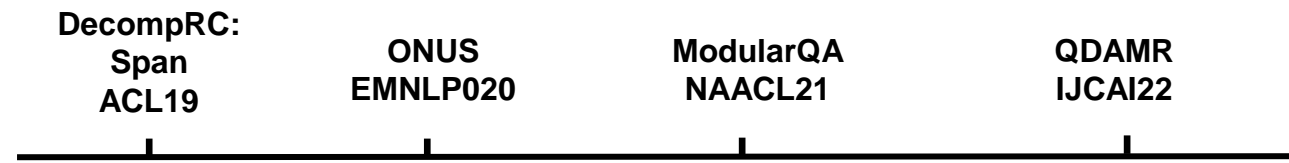


# Complex QA with Text

## Question Decomposition



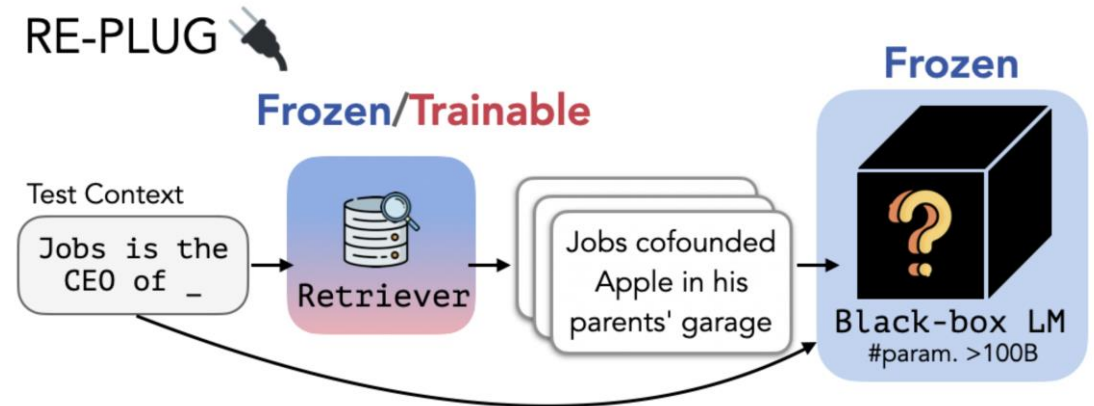
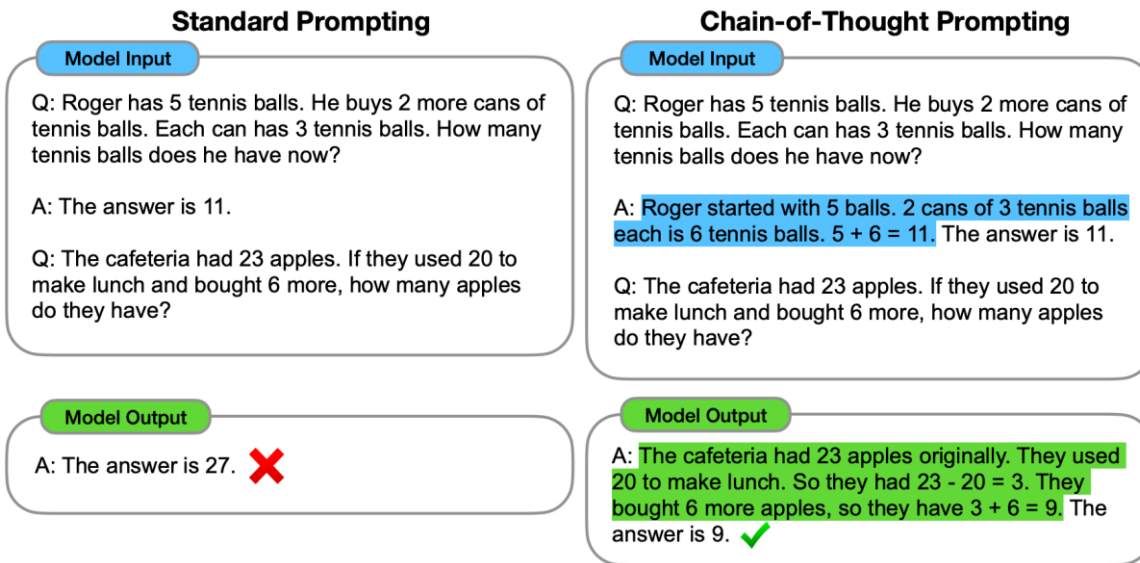
- Supervised models: seq2seq encoder-decoder model (TACL20)
- Weakly supervised models:



# Complex QA with LLM

- Chain-of-Thought Reasoning
  - CoT (NeurIPS22)
  - Zero-shot CoT (NeurIPS22)
  - Self-consistency (ICLR23)

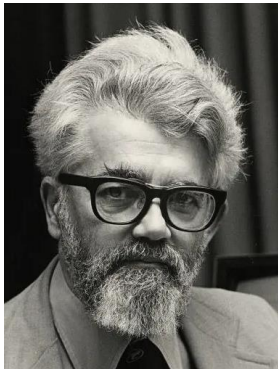
- Retrieval-Augmented Generation
  - Self-Ask (EMNLP23)
  - ReAct (ICLR23)
  - IRCot (ACL23)
  - MCR (EMNLP23)



# Deep Understanding and Reasoning



## □ John McCarthy: Decompose a Complex Task into Sub-problems



AN EXAMPLE FOR NATURAL  
LANGUAGE UNDERSTANDING AND THE  
AI PROBLEMS IT RAISES

**John McCarthy**

Computer Science Department  
Stanford University  
Stanford, CA 94305  
jmc@cs.stanford.edu  
<http://www-formal.stanford.edu/jmc/>

“A 61-year old furniture salesman was pushed down the shaft of a freight elevator yesterday in his downtown Brooklyn store by two robbers while a third attempted to crush him with the elevator car because they were dissatisfied with the \$1,200 they had forced him to give them.

The buffer springs at the bottom of the shaft prevented the car from crushing the salesman, John J. Hug, after he was pushed from the first floor to the basement. The car stopped about 12 inches above him as he flattened himself at the bottom of the pit.

### Basis

1. A **formalism** capable of expressing the assertions of the sentences, called **artificial natural language (ANL)**
2. A **data structure** for expressing the facts (apart from expressing the sentences). Certain programs can be applied to the data structures

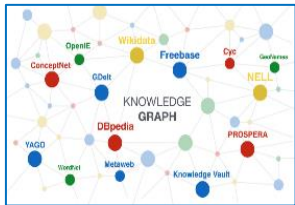
### Process

1. A **parser** that takes English into ANL
2. An **understander** that constructs the **facts** from a text
3. Expression of **general information** about the world
4. A **problem solver** that could answer the above questions based on the facts and general information

# Neural-symbolic Programming for CQA



## ④ Solver (Program Executor) (problem solver)



图谱问答



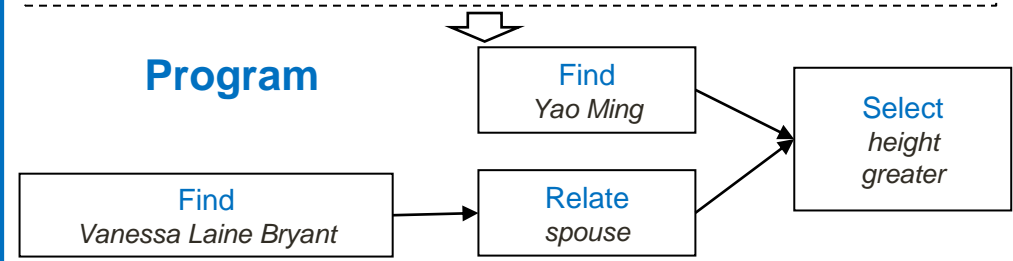
信息检索



大模型提示

## ③ Scheduler (Automatic Programming) (semantic parser)

Who is taller, Yao Ming or the spouse of Vanessa Laine Bryant?



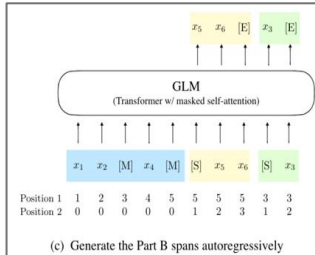
## ① Knowledge Manager (data structure & understander)



知识图谱

Input Paragraphs:  
The Sum of All Fears is a best-selling thriller novel by Tom Clancy... It was the fourth of Clancy's Jack Ryan books to be turned into a film...  
Dr. John Patrick Jack Ryan Sr., MCVO (Hon.), Ph.D. is a fictional character created by Tom Clancy who appears in many of his novels and their respective film adaptations...  
Net Force Explorers is a series of young adult novels created by Tom Clancy and Steve Pieczenik as a spin-off of the military fiction series...  
Question: What fiction character created by Tom Clancy was turned into a film in 2002?  
Answer: Jack Ryan

文本语料



大语言模型

## ② Manipulator (Programming Language) (artificial natural language)

```

SPARQL:    SELECT (COUNT(DISTINCT ?e) AS ?count) WHERE {
               ?e instance_of ?c . ?c name "film" .
               ?e director ?e_1 .
               ?e_1 name "Stanley Kubrick" }

Cypher:   MATCH (n1)-[:DIRECTOR]->(n2) WHERE
               n1:Film AND n2.name='Stanley Kubrick'
               RETURN COUNT(DISTINCT n1)

KoPL:     Find(Stanley Kubrick).
               Relate(director,backward).
               FilterConcept(film).Count()

Lambda-DCS: ( call @listValue (call .size (
                  ( call @filter ( call @getProperty
                    ( call @singleton en.film )
                    ( string ! type ) ) ( string director )
                    ( string = ) ( en.stanley_kubrick ) ) ) )

```



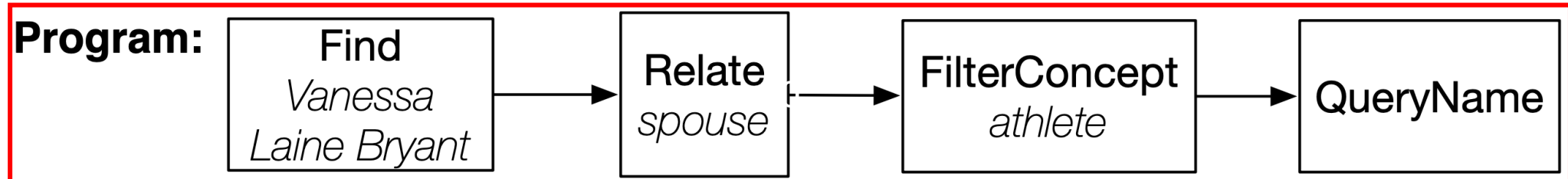


# Knowledge-oriented Programming Language

- Represent natural language questions as programs composed of basic functions, with knowledge elements as parameters

**Question:** Who is the athlete that is the spouse of Vanessa Laine Bryant?

**SPARQL:** `SELECT ?e WHERE { ?e <pred:instance_of> ?c . ?c <pred:name> "athlete" . ?e <spouse> ?e_1 . ?e_1 <pred:name> "Vanessa Laine Bryant" . }`



Principle

- Explicit, Transparent, and Modular Reasoning
- Easy to Understand and Human Interaction
- Apply to KG, Text and LLM



# KoPL Framework

## Knowledge Arguments

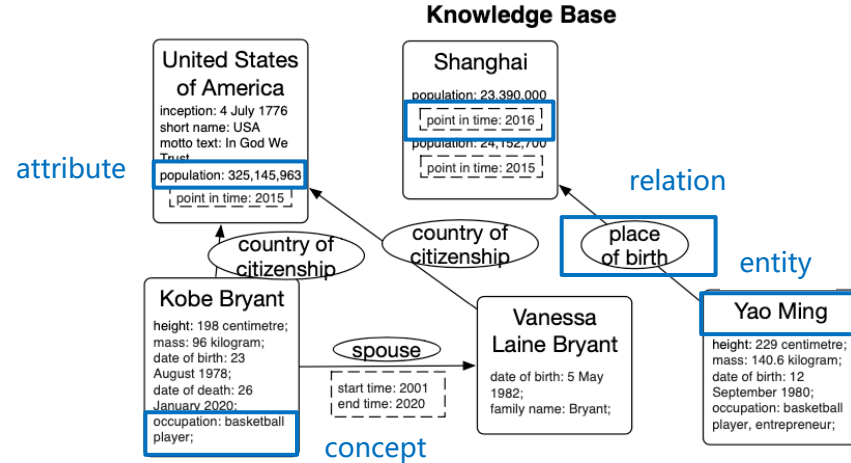
## Atomic Functions

- Operation functions

- Query functions

## Program

- Composition of basic functions, and the execution result is the answer

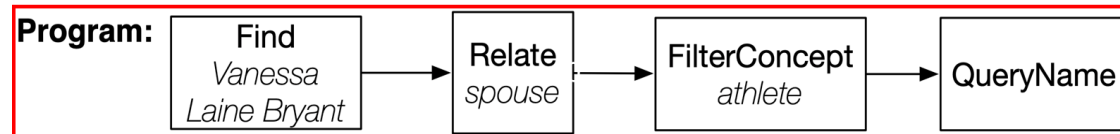


Function	Description	Example	Inputs → outputs
<i>Find</i>	Find all <b>entities</b> with the given name	<i>Find</i> (Yao Ming)	$() \times (Name) \rightarrow (Entities)$
<i>FilterConcept</i>	Find those belonging to the given <b>concept</b>	<i>FilterConcept</i> (athlete)	$(Entities) \times (Name) \rightarrow (Entities)$
<i>FilterStr</i>	Filter entities with an <b>attribute</b> condition	<i>FilterStr</i> (gender, male)	$(Entities) \times (Key, Value) \rightarrow (Entities, Facts)$
<i>Relate</i>	Find entities that have a specific <b>relation</b> with the given entity	<i>Relate</i> (capital, forward)	$(Entity) \times (Pred, Dir) \rightarrow (Entities, Facts)$
<i>And</i>	Get the <b>intersection</b> of two entity sets	---	$(Entities, Entities) \times () \rightarrow (Entities)$
<i>Or</i>	Get the <b>union</b> of two entity sets	---	$(Entities, Entities) \times () \rightarrow (Entities)$

Function	Description	Example	Inputs → outputs
<i>QueryName</i>	Query the <b>entity</b> name	<i>Find</i> (Yao Ming)	$(Entity) \times () \rightarrow (string)$
<i>QueryAttr</i>	Query the <b>attribute value</b> of the entity	<i>QueryAttr</i> (height)	$(Entity) \times (Key) \rightarrow (Value)$
<i>QueryRelation</i>	Query the <b>relation</b> between two entities	<i>QueryRelation</i> (Yao Ming, Shanghai)	$(Entity) \times (Entity) \rightarrow (Pred)$
<i>QueryRelationQualifier</i>	Query the <b>qualifier value</b> of the fact (Entity, Pred, Entity)	<i>QueryRelationQualifier</i> (drafted by, point in time)	$(Entity, Entity) \times (Pred, QKey) \rightarrow (Qvalue)$
<i>SelectBetween</i>	From two entities, find the one with a <b>greater or less</b> attribute value	<i>SelectBetween</i> (height, greater)	$(Entity, Entity) \times (Key, Op) \rightarrow (string)$
<i>SelectAmong</i>	From an entity set, find the one with the <b>largest of smallest</b> attribute value	<i>SelectAmong</i> (height, greatest)	$(Entities) \times (Key, Op) \rightarrow (string)$
<i>VerifyStr</i>	Query the output attribute value and the given value <b>whether are equal</b> as string	---	$(value) \times (value) \rightarrow (boolean)$
<i>Count</i>	Query the <b>number of entities</b>	---	$(Entities) \times () \rightarrow (number)$

**Question:** Who is the athlete that is the spouse of Vanessa Laine Bryant?

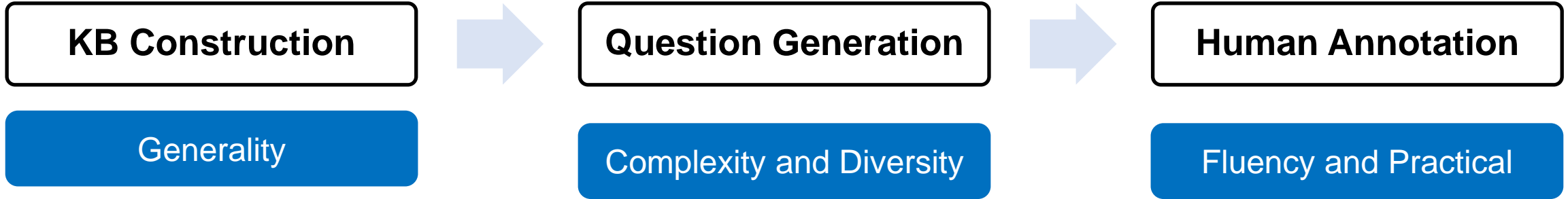
**SPARQL:** `SELECT ?e WHERE { ?e <pred:instance_of> ?c . ?c <pred:name> "athlete" . ?e <spouse> ?e_1 . ?e_1 <pred:name> "Vanessa Laine Bryant" . }`





# Automatic KoPL Programming

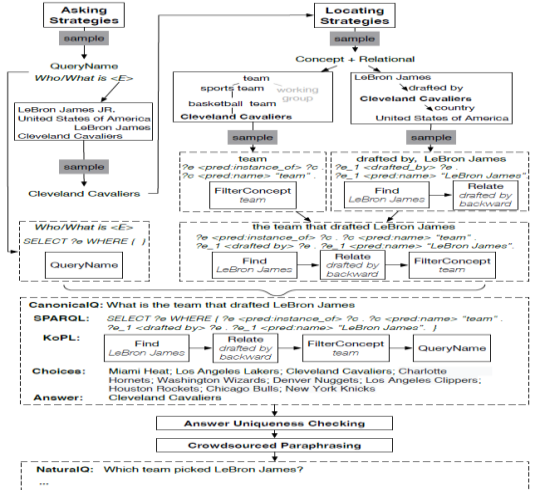
## □ KQA Pro: Questions with KoPL Programs



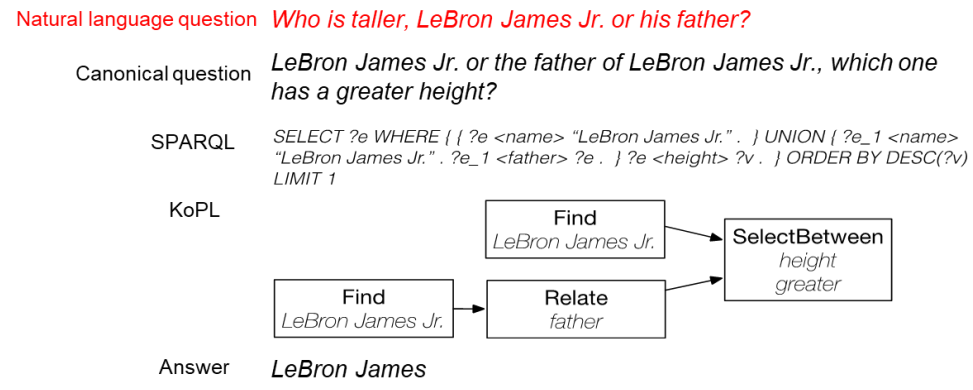
- Based on the widely-used FB15K-237
- Extract the equivalent part from Wikidata
- Adding 3,000 ambiguous entities
- Extracting various types of knowledge

# Con.	# Ent.	# Name	# Pred.	# Attr.
794	16,960	14,471	363	846
# Relational	# Literal	# Qualifier		
415,334	174,539	309,407		

Small and Dense Wikidata Subset



Sampling + Recursively Composing



Paraphrase and Evaluation



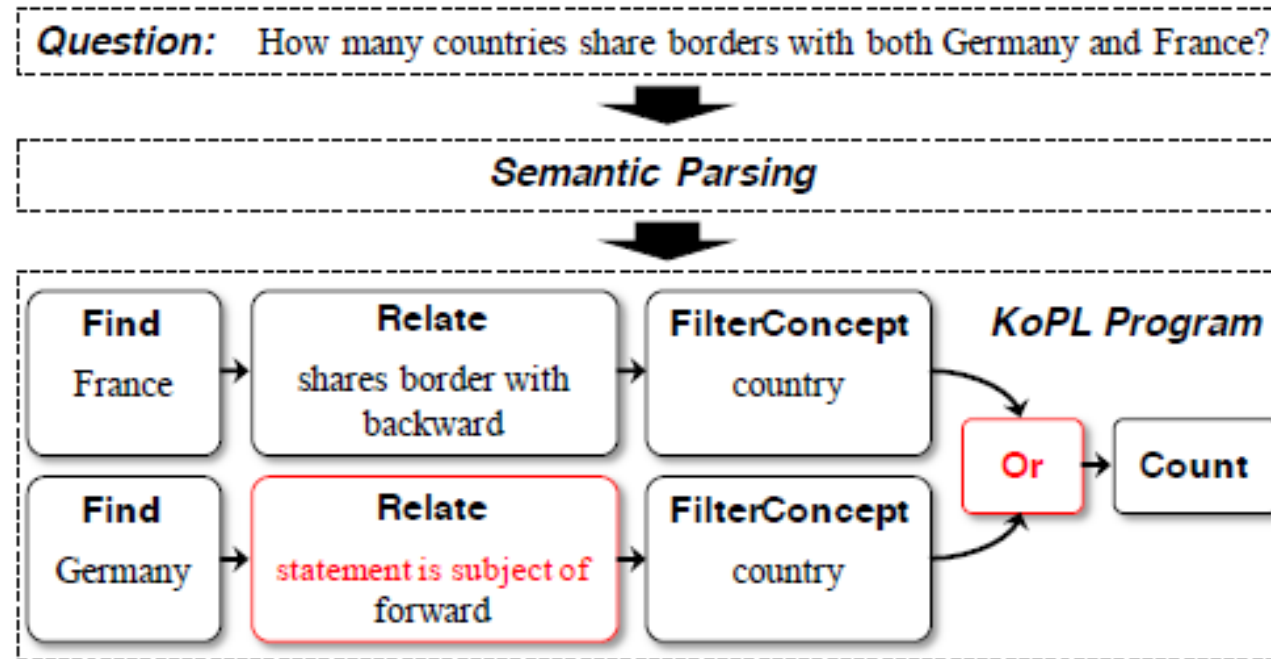
# Automatic KoPL Programing

- Total Questions: 117,970; Training/Validation/Test: 94,376/11,797/11,797
- KQA Pro + Seq2Seq Model → Automatic Programming
- KQA Pro: Evaluate Fine-grained Reasoning Abilities

Model	Overall	Multi-hop	Qualifier	Comparison	Logical	Count	Verify	Zero-shot
KVMemNet	16.61	16.50	18.47	1.17	14.99	27.31	54.70	0.06
SRN	-	12.33	-	-	-	-	-	-
EmbedKGQA	28.36	26.41	25.20	11.93	23.95	32.88	61.05	0.06
RGCN	35.07	34.00	27.61	30.03	35.85	41.91	65.88	0.00
RNN SPARQL	41.98	36.01	19.04	66.98	37.74	50.26	58.84	26.08
RNN KoPL	43.85	37.71	22.19	65.90	47.45	50.04	42.13	34.96
BART SPARQL	89.68	88.49	83.09	96.12	88.67	85.78	92.33	87.88
BART KoPL	90.55	89.46	84.76	95.51	89.30	86.68	93.30	89.59
BART KoPL <sup>CG</sup>	77.86	77.86	61.46	93.61	77.88	79.17	89.01	76.04
Human	97.50	97.24	95.65	100.00	98.18	83.33	95.24	100.00

# Interactive Programming

- Automatic Programming cannot guarantee to be accurate



- Could we debug the programs conveniently?



# Interactive Programming

- ❑ Drag and drop functions
- ❑ Check the intermediate results
- ❑ Correct type and arguments of functions

How many countries share borders with both Germany and France? Parse

17 → 4

select a function + Clear All Run on Wikidata Run Code

**Find**  
Entity Name: France  
Check intermediate result

**Relate**  
Relation: shares borders with  
Direction: forward  
Check intermediate result

**FilterConcept**  
Concept: country  
Check intermediate result

**Or**  
Check intermediate result

**Count**  
Check intermediate result

**Find**  
Entity Name: Germany  
Check intermediate result

**Relate**  
Relation: statement is subject of  
Direction: forward  
Check intermediate result

**FilterConcept**  
Concept: country  
Check intermediate result

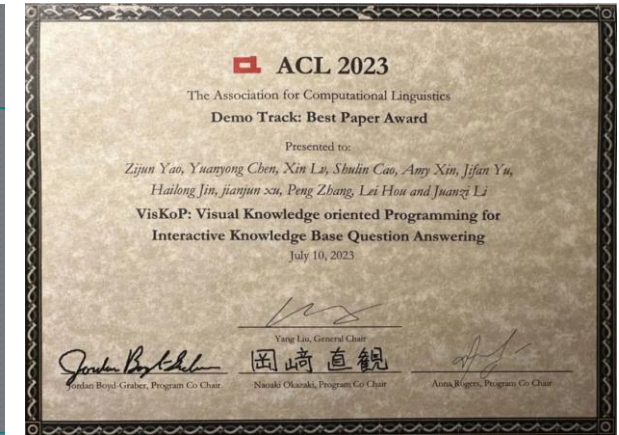
**And**  
shares borders with  
Check intermediate result

Intermediate Result

```
copy
```

```
▼ {
  ▼ 0: {
    entity_label: "United Kingdom"
  },
  ▼ 1: {
    entity_label: "Kingdom of Westphalia"
  },
  ▼ 2: {
    entity_label: "Brazil"
  },
  ▼ 3: {
    entity_label: "Germany"
  },
  ▼ 4: {
    entity_label: "Andorra"
  }
}
```

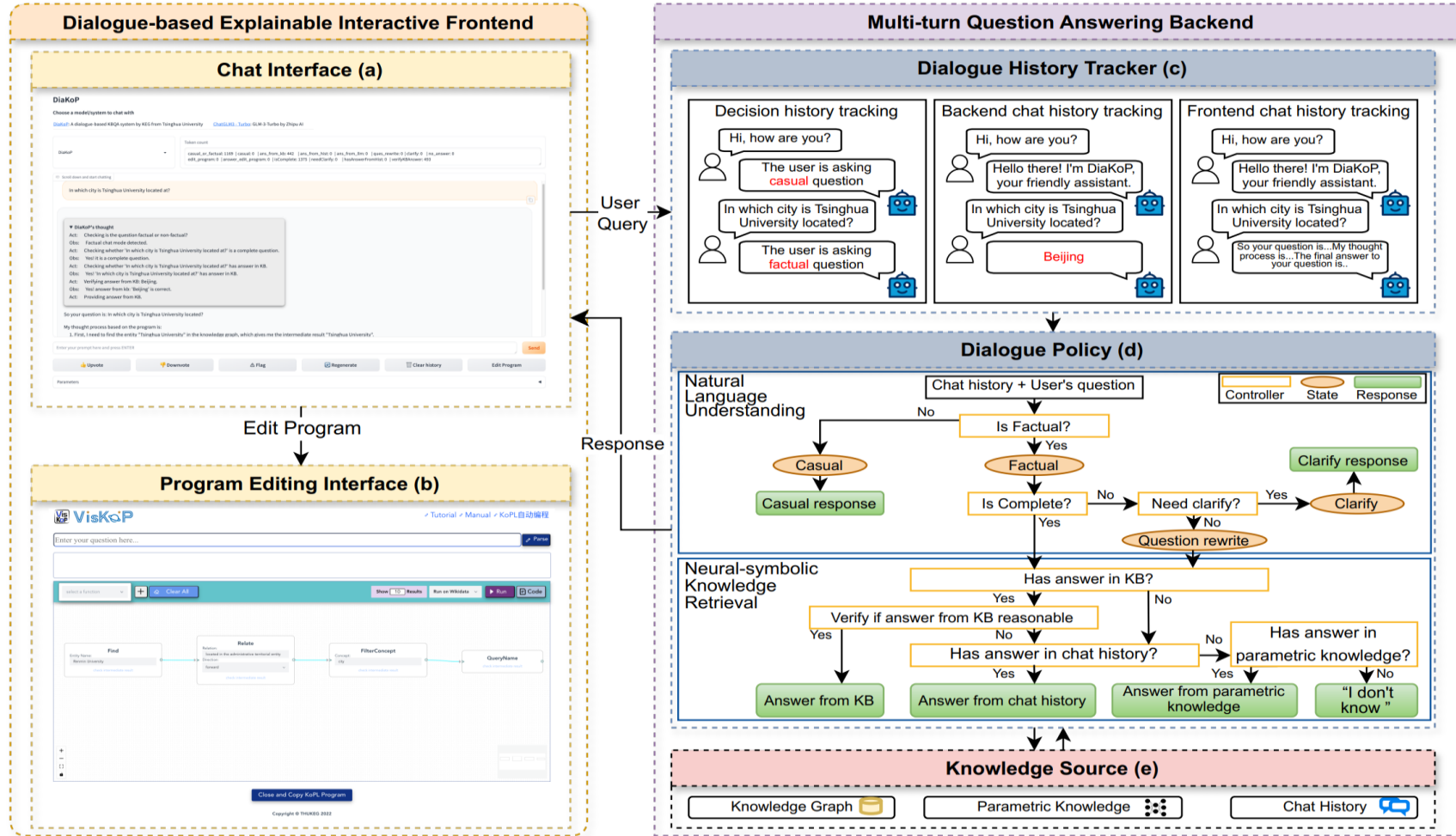
Cancel OK



ACL 2023 Demo Track  
Best Paper Award

# Conversational Programming

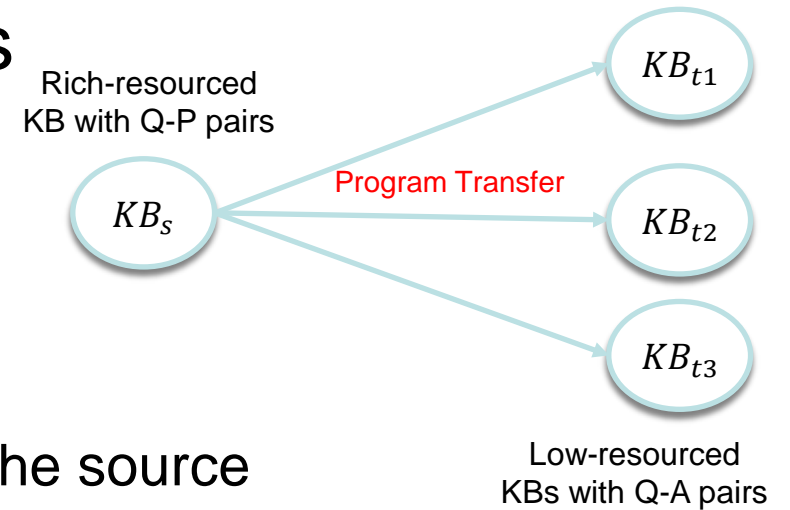
## Dialogue Policy + Dialogue History Tracking





# Program Transfer

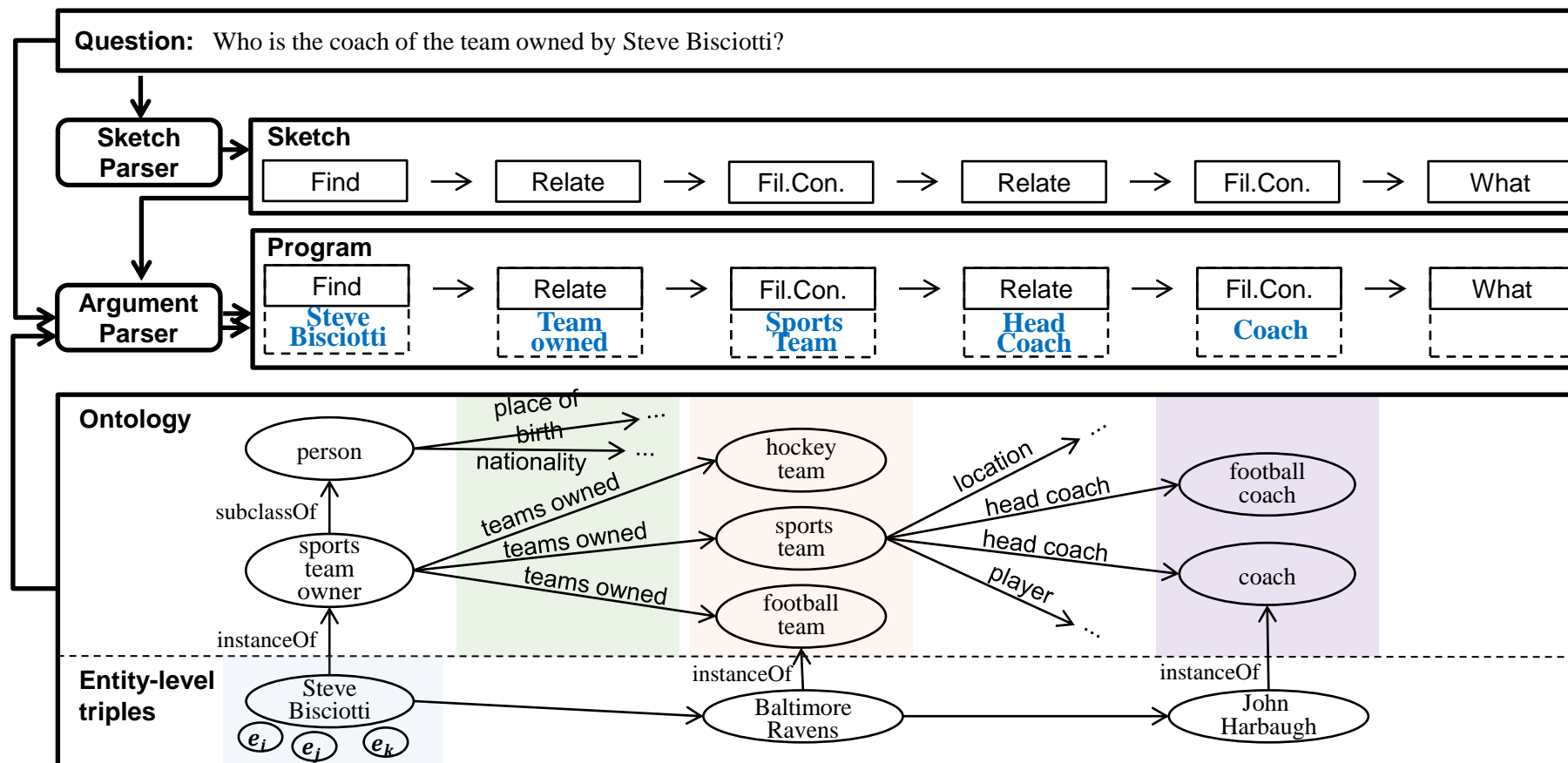
- ❑ Automatic programming relies on parallel question-program pairs for the given KB, which are usually lacking
- ❑ Main idea: transfer annotations across KBs
  - What to Transfer
    - Questions and KBs are both heterogeneous
  - Unseen KB Elements
    - Most elements in the target KB are not covered in the source
  - Huge Search Space
    - KBs often contain millions of entities, concepts and relations





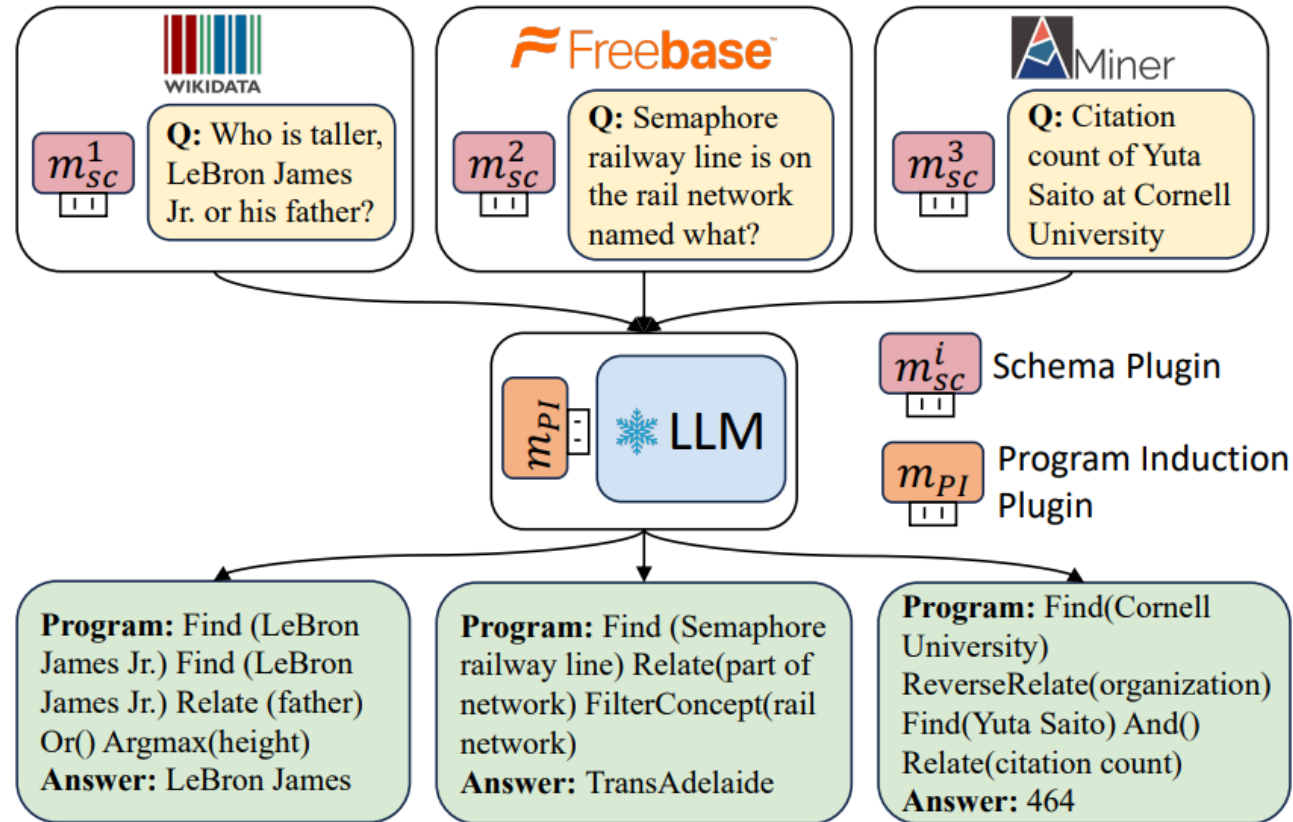
# Program Transfer

## Two-stage Parsing with Ontology-guided Pruning

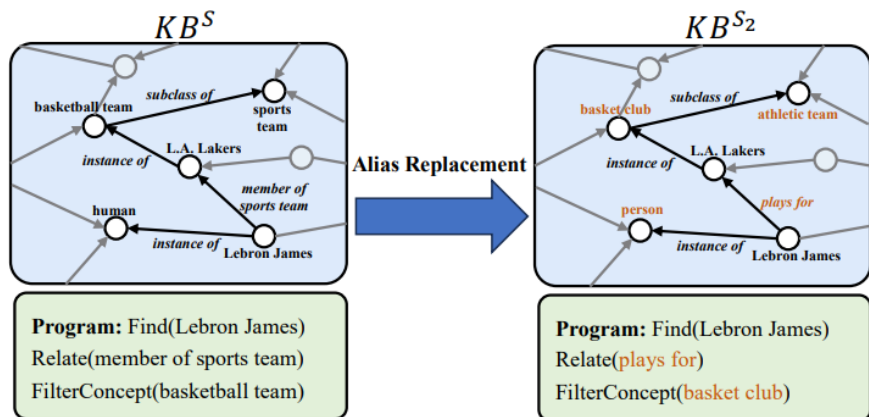


# KB-Plugin for LLM Programming

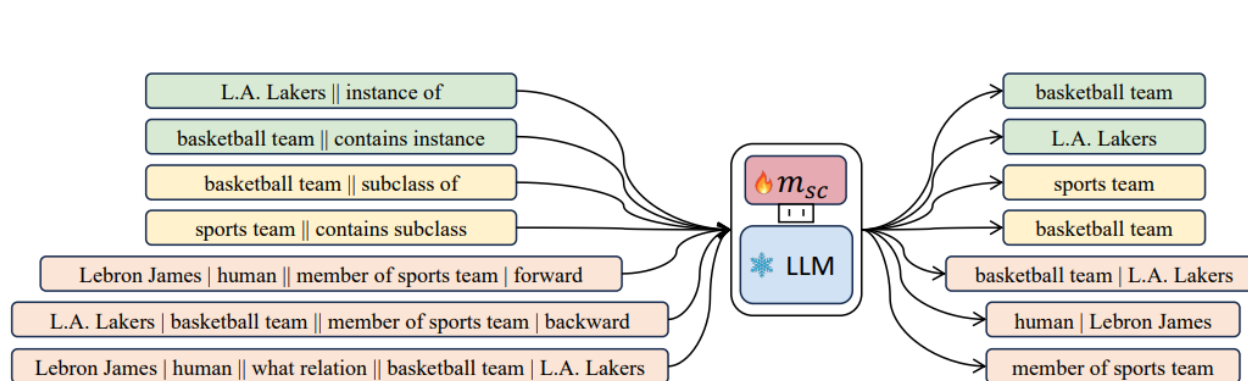
- LLM: question structure 🤖 + KB awareness 🧐
- Main idea: general PI plugin + various KB schema plugins



# KB-Plugin for LLM Programming

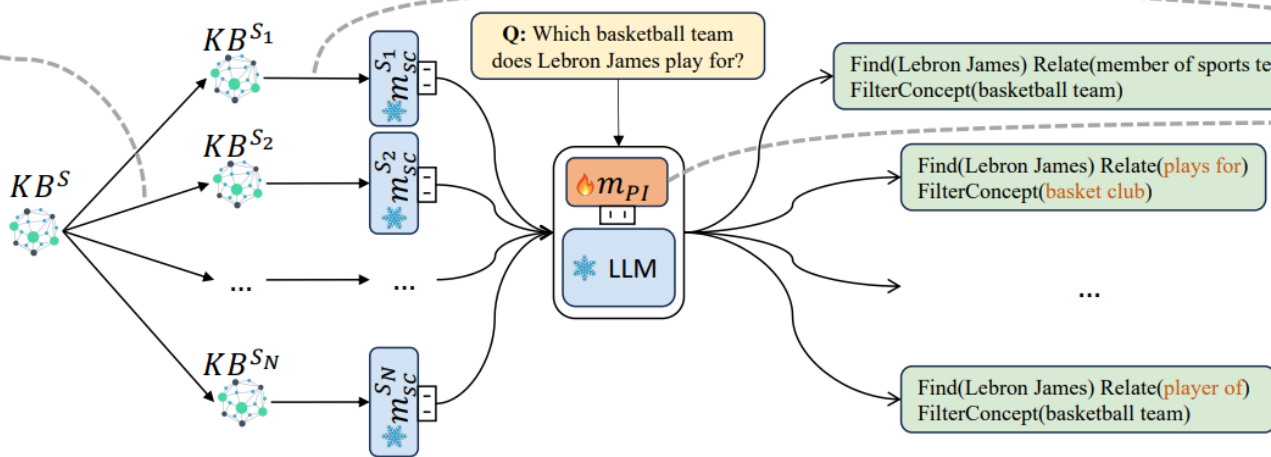


(a) KB Generation and Data Augmentation

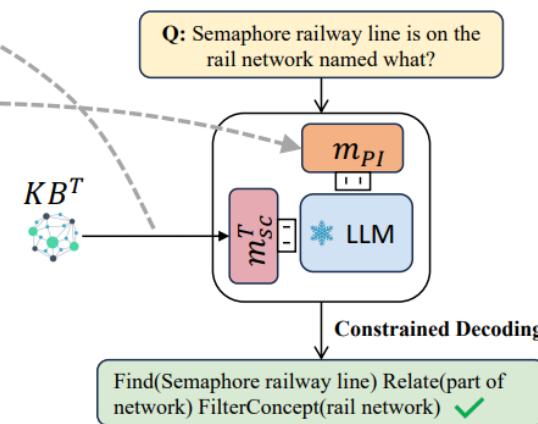


(b) Learning of Schema Plugin via Schema-relevant Triple Completion

(c) Learning of PI Plugin

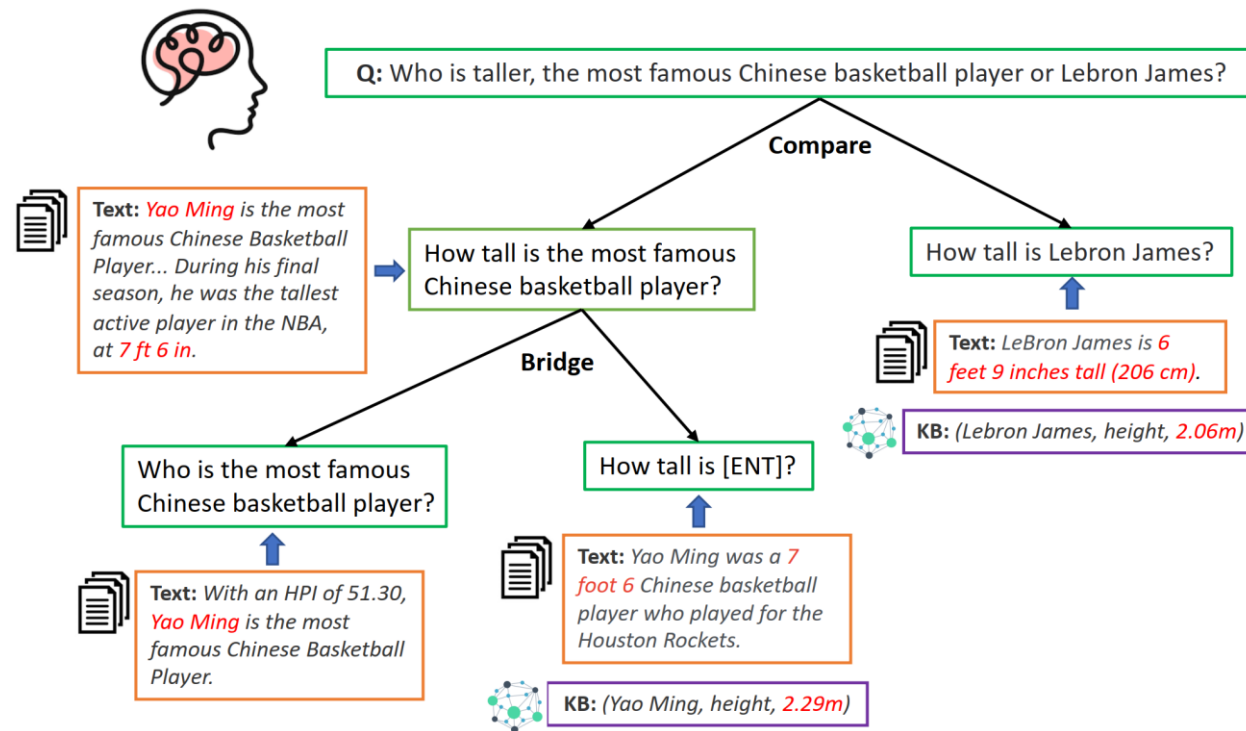


(d) Plugin Transfer



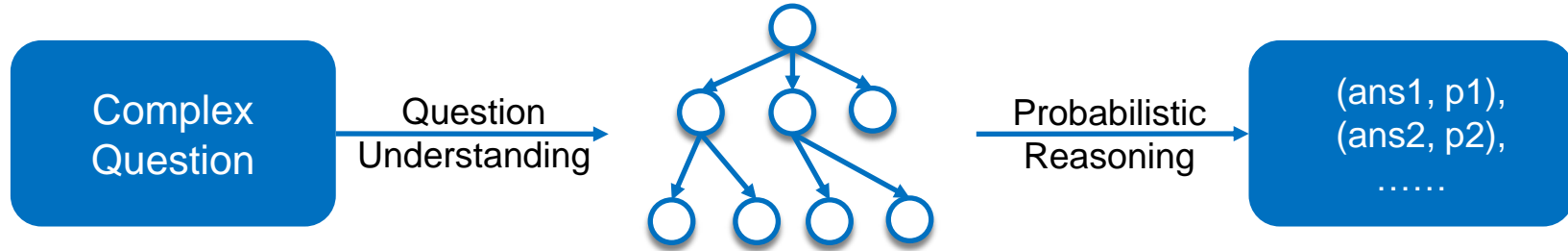
# Reason with Multiple Sources

- Knowledge scattered across multiple sources
  - Text, Knowledge Graphs, Large Language Models
- Main idea: Hierarchical Question Decomposition Tree

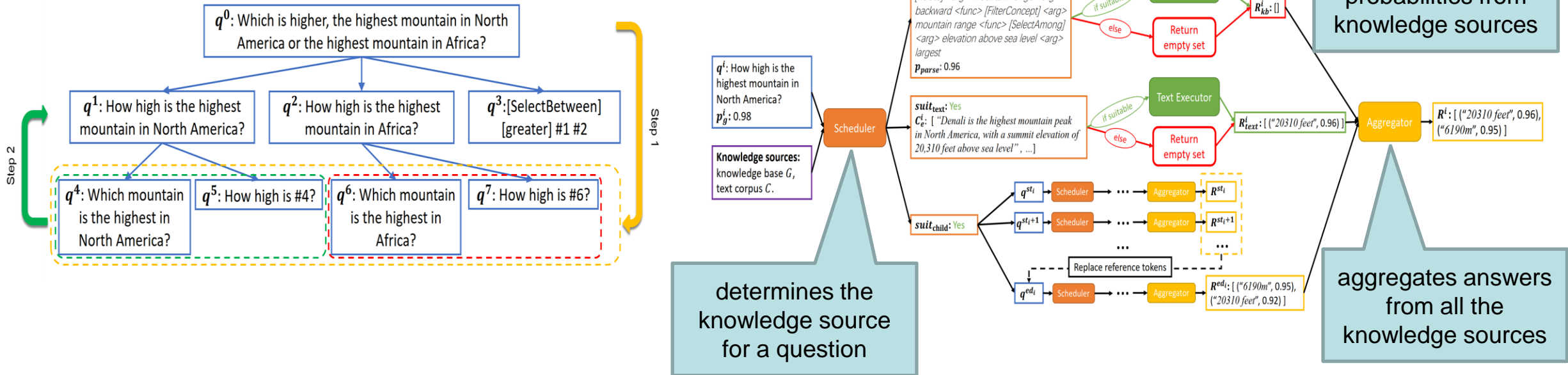


# Reason with Multiple Sources

## □ Framework



Hierarchical Question Decomposition Tree (HQDT)





# Reason with Large Language Model

- Chain-of-Thought: factually incorrect reasoning
- RAG CoT Reasoning: negative retrieval + limited sight
- Main idea: Probabilistic Tree-of-Thought

Who founded Harvard College?

**Harvard University**

Harvard was formed in 1636 by vote of the Great and General Court of the Massachusetts Bay Colony. It was initially called "New College" or "the college at New Towne". In 1638, the college became home for North America's first known printing press, carried by the ship John of London. In 1639, the college was renamed Harvard College after deceased clergyman **John Harvard**, who was an alumnus of the University of Cambridge. He had left the school £779 and his library of some 400 books. The charter creating the Harvard Corporation was granted in 1650.

**Closed-book QA**

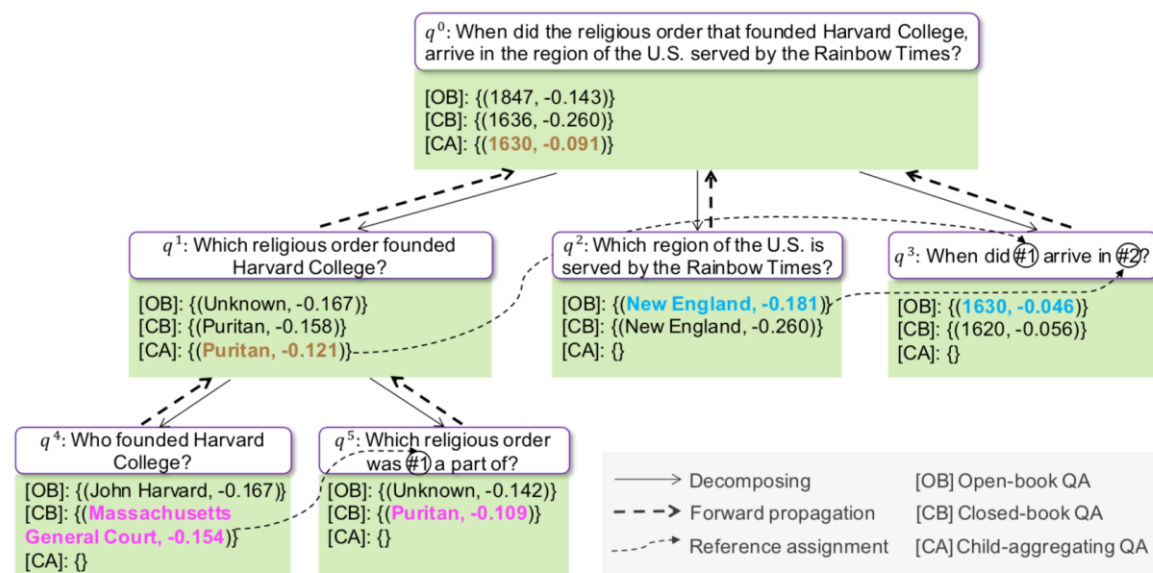
Q: Who founded Harvard College?  
A: Harvard College was founded by Massachusetts General Court. So the answer is: **Massachusetts General Court.**

✓

**Open-book QA**

#1 Title: Harvard University  
Text: Harvard University was formed in 1636 ...  
#2 Title: Edward Cochrane McLean  
Text: ...  
Q: Who founded Harvard College?  
A: Harvard College was founded by John Harvard. So the answer is: **John Harvard.**

✗



**Input**

Please generate a hierarchical question decomposition tree (HQDT) with json format for a given question. In this tree, the root node is the original complex question, and each non-root node is a sub-question of its parent. The leaf nodes are atomic questions that cannot be further decomposed.

<Examples>

Q: When did the religious order that founded Harvard College, arrive in the region of the U.S. served by the Rainbow Times?  
A:

**Output**

```

{
  "q0": "When did the religious order that founded Harvard College, arrive in the region of the U.S. served by the Rainbow Times?",
  "q1": "Which religious order founded Harvard College?",
  "q2": "Which region of the U.S. is served by the Rainbow Times?",
  "q3": "When did #1 arrive in #2?",
  "q4": "Who founded Harvard College?",
  "q5": "Which religious order was #1 a part of?"
}
  
```

## 1. Question Decomposition with In-context Learning

**Input**

Given a question and a context, answer the question and explain why.

<Examples>

Context:  
Which religious order founded Harvard College? Puritan.  
Which region of the U.S. is served by the Rainbow Times? New England.  
When did Puritan arrive in New England? 1630.  
Question:  
When did the religious order that founded Harvard College, arrive in the region of the U.S. served by the Rainbow Times?  
Answer:

**Output**

The religious order that founded Harvard College, the Puritans, arrived in the region of the U.S. served by the Rainbow Times, New England, in 1630. So the answer is: 1630.

## 2. Confidence Estimation based on Explanation Logit



# Conclusions

## □ Program Language

- KoPL: Knowledge-oriented Programming Language

## □ Automatic Programming

- KQA Pro: A Complex KBQA Dataset with KoPL Programs
- VisKoP + DiaKoP: Interactive/Conversational Programming

## □ Program Executor

- Program Transfer + KB Plugin: Programming across Different KBs
- RoHT: Reasoning over Multiple Knowledge Sources
- ProbTree: Reasoning with Large Language Model

# Thanks for Your Attention!



Thank all excellent collaborators (some are listed below)



Shulin CAO



Jiaxin SHI



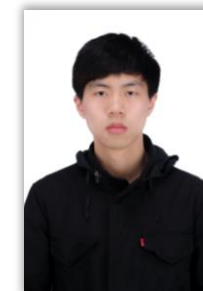
Jiajie ZHANG



Zijun YAO



Xin LV



Jinxin LIU