GraphQ IR: Unifying the Semantic Parsing of Graph Query Languages with One Intermediate Representation

Lunyiu Nie[1], **Shulin Cao**[1], **Jiaxin Shi**[2], **Jiuding Sun**[1],
Qi Tian[2], **Lei Hou**[1], **Juanzi Li**[1], **Jidong Zhai**[1]

[1] Department of Computer Science and Technology, Tsinghua University
[2] Huawei Cloud Computing Technologies Co., Ltd.
{nlx20, caosl19, sjd22}@mails.tsinghua.edu.cn, shijx12@gmail.com,
tian.qi1@huawei.com, {houlei,lijuanzi,zhaijidong}@tsinghua.edu.cn

Abstract

Subject to the huge semantic gap between natural and formal languages, neural semantic parsing is typically bottlenecked by its complexity of dealing with both input semantics and output syntax. Recent works have proposed several forms of supplementary supervision but none is generalized across multiple formal languages. This paper proposes a unified intermediate representation (IR) for graph query languages, named GraphQ IR. It has a natural-language-like expression that bridges the semantic gap and formally defined syntax that maintains the graph structure. Therefore, a neural semantic parser can more precisely convert user queries into GraphQ IR, which can be later losslessly compiled into various downstream graph query languages. Extensive experiments on several benchmarks including KQA Pro, Overnight, GrailQA and MetaQA-Cypher under standard i.i.d., out-of-distribution and low-resource settings validate GraphQ IR's superiority over the previous state-of-the-arts with a maximum 11% accuracy improvement.

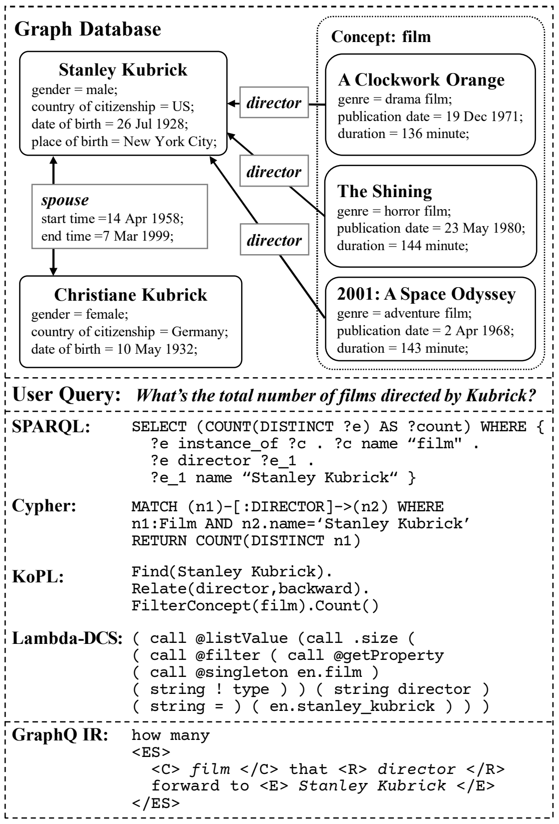
1 Introduction

By mapping natural language utterances to logical forms, the task of semantic parsing has been widely explored in various applications, including database query (Yu et al., 2018; Talmor and Berant, 2018) and general-purpose code generation (Yin and Neubig, 2017; Campagna et al., 2019; Nan et al., 2020). Although the methodology has evolved from earlier statistical approaches (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010) to present Seq2Seq paradigm (Zhong et al., 2017; Damonte and Monti, 2021), the semantic gap between natural language and logical forms still lies as the major challenge for semantic parsing.

As shown in Figure 1, in graph query languages (*e.g.*, SPARQL, Cypher, Lambda-DCS, and newly emerged KoPL, etc.), graph nodes, edges and their respective properties constitute the key semantics of the logical forms (Pérez et al., 2009), which are very different from the expression of natural language utterances. Such discrepancy significantly hinders the learning of neural semantic parsers and therefore increases the demand for labeled data (Yin et al., 2022). However, due to the laborious efforts and language-specific expertise required in annotation, such demand cannot always be satisfied and thus becomes the bottleneck (Li et al., 2020b; Herzig et al., 2021).

To overcome these challenges, many works adopt complementary forms of supervision, such as the schema of database (Hwang et al., 2019), results of the execution (Clarke et al., 2010; Wang et al., 2018, 2021), and grammar-constrained decoding algorithms (Krishnamurthy et al., 2017; Shin et al., 2021; Baranowski and Hochgeschwender, 2021). Although effective, the additional resources that these methods rely on are not necessarily available in practice. By normalizing the expression (Berant and Liang, 2014; Su and Yan, 2017) or enriching the structure (Reddy et al., 2016; Cheng et al., 2017; Hu et al., 2018) of natural language utterances, another category of works proposes various intermediate representations like AMR (Kapanipathi et al., 2021) to ease the parsing of complex queries. However, the transition from their IRs to the downstream logical forms may incur extra losses in precision (Bornea et al., 2021). Besides, these representations are usually coupled to specific data or logical forms and thus cannot be easily transferred to other tasks or languages (Kamath and Das, 2019).

In industry, aside from SPARQL, many other graph query languages such as Cypher (Francis et al., 2018) and Gremlin (Rodriguez, 2015) are equally or even more commonly used in graph database interaction (Angles, 2012; Seifer et al., 2019). However, most graph query semantic parsing works only support SPARQL (Talmor and Berant, 2018; Dubey et al., 2019; Keysers et al., 2020)



User Query: *What's the total number of films directed by Kubrick?*

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 ) ) )
```

GraphQ IR:
```
how many
<ES>
  <C> film </C> that <R> director </R>
  forward to <E> Stanley Kubrick </E>
</ES>
```

Figure 1: A a property graph extracted from Wikidata (Vrandecic and Krötzsch, 2014). We present a relevant user query with its corresponding logical forms in different query languages and in GraphQ IR.

while very few works target other graph query languages. Meanwhile, no existing tools or IR can support the data conversion among multiple graph query languages (Moreira and Ramalho, 2020; Agrawal et al., 2022). Such lack of interoperability has not only hindered the semantic parsing of low-resource languages but also limited the potential of querying heterogeneous databases (Mami et al., 2019; Angles et al., 2019).

In this paper, we propose a unified intermediate representation for graph query languages, namely GraphQ IR, to resolve these issues from a novel perspective. The designs of GraphQ IR weigh up the semantics of both natural and formal language by (a) producing the IR sequences with composition rules consistent with modern English (Tomlin, 2014) to close the semantic gap; and (b) maintaining the fundamental graph structures like nodes, edges, and properties, such that the IR can be automatically compiled into any downstream graph query languages without any loss.

Instead of directly mapping the user query to the logical form, we first parse natural language into GraphQ IR, then compile the IR into the target graph query languages (*e.g.*, SPARQL, Cypher, Lambda-DCS, KoPL, etc.). Therefore, language-specific grammar features that initially posed a huge obstacle to semantic parsing are now explicitly handled by the compiler. Additionally, with the GraphQ IR as a bridge, our implemented source-to-source compiler can support lossless translation among multiple graph query languages and thus unify the annotations of different languages for eliminating the data bottleneck.

To validate the effectiveness of GraphQ IR, we conducted extensive experiments on benchmarks KQA Pro, Overnight, GrailQA and MetaQA-Cypher. Results show that our approach can consistently outperform the previous works by a significant margin. Especially under the compositional and few-shot generalization settings, our approach with GraphQ IR can demonstrate a maximum 11% increase in accuracy over the baselines.

The main contributions of our work include:

- We propose GraphQ IR for unifying the semantic parsing of graph query languages and present the IR design principles that are critical for bridging the semantic gap;
- Experimental results show that our approach can consistently achieve state-of-the-art performance across multiple benchmarks under the standard i.i.d, out-of-distribution, and low-resource settings.
- Our implemented source-to-source compiler unlocks data interoperability by supporting the bi-directional translation among different graph query languages. The code and toolkit are publicly available at `https://github.com/Flitternie/GraphQ_IR`.

2 GraphQ IR

In this section, we formalize the grammar and the expressiveness of our GraphQ IR based on the definition of property graph and regular path query. Then we summarize the design principles of GraphQ IR for bridging the semantic gap between natural and formal language as well as unifying different graph query languages.

2.1 Definition

As the top of Figure 1 demonstrates, a graph database can be expressed as a collection of property graphs that include **Entity** (graph nodes, *e.g.*, Stanley Kubrick), **Attribute** (node properties, *e.g.*, date of birth), **Concept** (node label, *e.g.*, film), **Relationship** (graph edges, *e.g.*, spouse) and **Qualifier** (edge properties, *e.g.*, start time).

Therefore, to evaluate the expressiveness of GraphQ IR, we start by giving the definition of property graph: a directed labeled multigraph where each node or edge can contain a set of property-value pairs (Angles, 2018).

Definition 1 (***Property graph***)**.** A property graph G is a tuple (N, E, ρ, λ, σ) where:
(1) N is a finite set of *nodes*.
(2) E is a finite set of *edges* such that $N \cap E = \varnothing$.
(3) $\rho : E \to (N \times N)$ is a total function. Specifically, $\rho(e) = (n_1, n_2)$ refers e is a directed edge from node n_1 to n_2.
(4) $\lambda : (N \cup E) \to L$ is a partial function where L is a set of labels. Specifically, if $\lambda(n) = l$ then l is the label of node n.
(5) $\sigma : (N \cup E) \times P \to V$ is a partial function with P a set of properties and V a set of values V. Specifically, if $\sigma(n, p) = v$ then the property p of node n has value v.

Subsequently, a graph path can be expressed as $\pi = (n_1, e_1, ..., e_{k-1}, n_k)$ where $k \geq 1$ with each e_i being the edge between n_i and n_{i+1}. The spelling of path, denoted as $\lambda(\pi)$, is the concatenation of edge labels $\lambda(e_1)...\lambda(e_{k-1})$ (Mendelzon and Wood, 1995; Baeza, 2013).

Definition 2 (***Regular path query***)**.** A regular path query has the general form $Q = x \xrightarrow{\alpha} y$ where x denotes the start point, α is a regular expression defined over $\lambda(\pi)$, and y denotes the endpoints of the query.

By incorporating ρ, λ, σ and their inverse function ρ^{-1}, λ^{-1}, σ^{-1}, such regular path query can be extended to support navigational queries towards any graph elements $\psi \in (N \cup E \cup L \cup P \cup V)$ (Wood, 2012; van Rest et al., 2016). We can now evaluate the expressiveness of a language.

Definition 3 (***Path query expressiveness***)**.** A path query q is expressible in a language $\mathcal{L}$, if there exists an expression $\varepsilon \in \mathcal{L}$ such that, for any subgraph $G' \subseteq G$, we have $\varepsilon(G') = q(G')$ (Fletcher et al., 2015).

We formalize GraphQ IR as a context-free grammar $(\mathcal{V}, \Sigma, \mathcal{S}, \mathcal{P})$ and present its non-terminals and productions in Appendix Table 7. Its $\mathcal{V}$ and $\mathcal{P}$ are respectively defined as the superset of the terminal set (n, e, l, p, v) and production set (ρ, λ, σ, ρ^{-1}, λ^{-1}, σ^{-1}) of regular graph query. Therefore, all path queries expressible in regular grammar are also expressible in the context-free grammar of GraphQ IR (Hopcroft et al., 2007). Furthermore, GraphQ IR also supports extended operations like *Union*, *Difference* and *Filter* to express complex graph query patterns (Angles et al., 2017).

Empirically, GraphQ IR can express all graph query patterns that appeared in benchmarks KQA PRO, OVERNIGHT, GRAILQA and METAQA-Cypher, with details elaborated in Section 4.1.

2.2 Principles

We summarize several principles in designing GraphQ IR in this way: present in a syntax close to natural language while preserving the structural semantics equivalent to formal languages.

2.2.1 Diminishing syntactical discrepancy

To facilitate the training of the neural semantic parser, the target IR sequence should share a similar syntax in correspondence to the input utterance.

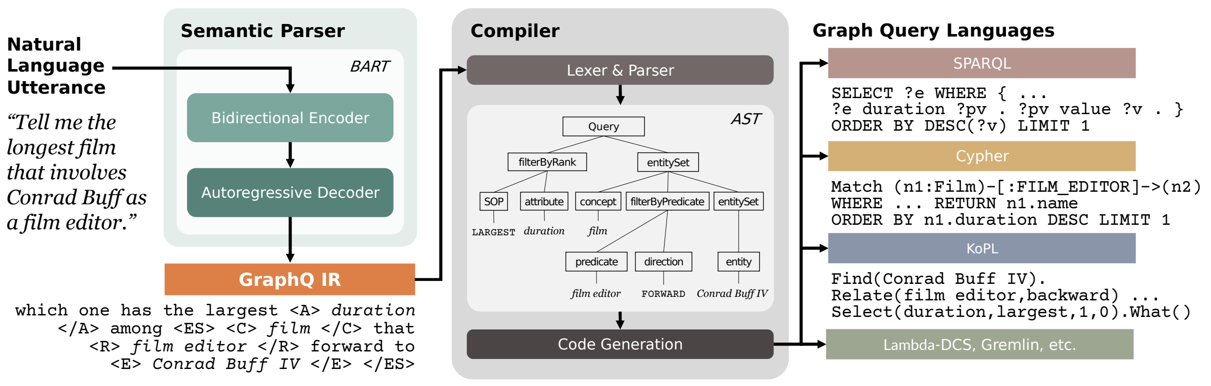
To achieve this, the IR structure should first match how users typically raise queries. Therefore, we simplify the triple-based structure in graph query languages into a more natural subject-verb-object syntactic construction (Tomlin, 2014). Take Figure 1's task setting as an example, the two triples `(?e instance_of ?c)` and `(?c name "film")` as the entity concept constraint in SPARQL are simplified to the sentence subject "`<C>` *film* `</C>`" in GraphQ IR. Multi-hop relationship and attribute queries are formulated as relative clauses similar to the English expression and thus can be comfortably generated by a language-model-based neural semantic parser.

Secondly, IR should also leave out the variables (*e.g.*, `?e`, `?c` in SPARQL) and operators (*e.g.*, `SELECT`, `WHERE`, `RETURN`, etc.) in logical forms that cannot be easily aligned to natural language utterances. Alternatively, human-readable operators are adopted in GraphQ IR, as illustrated in Appendix Table 7.

2.2.2 Eliminating semantic ambiguity

In formal languages, multiple parallel implementations can achieve the same functionalities. However, such redundancy and ambiguity in semantics may pose challenges to the neural semantic parser.

For example, in Lambda-DCS, there co-exist three implementations for constraining one's concept (*e.g.*, Kobe is a *player*), respectively through:



```
which one has the largest <A> duration
</A> among <ES> <C> film </C> that
<R> film editor </R> forward to
<E> Conrad Buff IV </E> </ES>
```

```
SELECT ?e WHERE { ...
?e duration ?pv . ?pv value ?v . }
ORDER BY DESC(?v) LIMIT 1
```

```
Match (n1:Film)-[:FILM_EDITOR]->(n2)
WHERE ... RETURN n1.name
ORDER BY n1.duration DESC LIMIT 1
```

```
Find(Conrad Buff IV).
Relate(film editor,backward) ...
Select(duration,largest,1,0).What()
```

Figure 2: Overall implementation of our proposed framework. The user queries are first converted to GraphQ IR sequences by a semantic parser and subsequently transpiled into the target graph query languages by a compiler.

- ***EventNP***: `(call @getProperty (en.player.kobe_bryant) (call @reverse (string player)));`
- ***TypeNP***: `(call @getProperty (call @singleton ( en.player )) (string !type));` and
- ***DomainNP***: `(call @getProperty ( en.player.kobe_bryant ... (call @domain (string player))) (string player)).`

When designing an IR, such redundant and ambiguous semantics should be clarified into more definitive and orthogonal representations (Campagna et al., 2019). Thus in GraphQ IR, we unify all such unnecessary distinctions and prune redundant structures in logical forms to distill the core semantics. In the previous example, GraphQ IR only requires a simple noun modifier "`<C>` *player* `</C>`" as the concept constraint. This not only makes the language clearer for users and semantic parsers to comprehend, but also facilitates the next-step compilation from the IR to the downstream formal language.

2.2.3 Maintaining graph structural semantics

In addition to the aforementioned designs to improve alignment with natural language, the syntax of IR also needs to maintain the key structures of graph queries for subsequent lossless compilation.

Specifically, IR should keep track of the data types of graph structural elements. We design GraphQ IR to be strong-typing by explicitly stating the type of terminal nodes with respective special tokens, *e.g.*, `<E>` for `Entity`, `<R>` for `Relation`, `<A>` for `Attribute`, etc. Values of different types are also differentiated in GraphQ IR with our pre-defined or user custom indicators, *e.g.*, `string`, `number`, `date`, `time`, etc.

Furthermore, IR should also preserve the hierarchical dependencies that are critical for multi-hop queries. We introduce `<ES>` as a scoping token in GraphQ IR to explicitly indicate the underlying dependencies among the clauses produced by an `EntitySet`, as shown in Appendix Table 7. Such scoping tokens in GraphQ IR can facilitate the compiler to recover the hierarchical structure and finally convert the IR sequences into one of the graph query languages deterministically.

3 Implementation

We depict the full picture of our proposed framework in Figure 2. The neural semantic parser first maps the input natural language utterance into GraphQ IR. Thereafter, the GraphQ IR sequence is fed into the compiler and parsed into an abstract syntax tree for downstream graph query language code generation.

3.1 Neural Semantic Parser

To verify the above principles in practice, we formulate the conversion from natural language to our GraphQ IR as a Seq2Seq task and adopt an encoder-decoder framework for implementing the neural semantic parser.

As shown in the left part of Figure 2, the encoder module of the semantic parser first maps the input natural language utterance X to a high dimensional feature space with non-linear transformations for capturing the semantics of the input tokens. The decoder module subsequently then interprets the hidden representations and generates the IR sequence by factorizing the probability distribution:

$$p(IR) = \prod_{i=1}^{n} P(y_i|X, y_1, ..., y_{i-1}), \qquad (1)$$

where y_i is the i-th token of IR sequence with in total n tokens. Specifically, we implement this encoder-decoder network with BART (Lewis et al., 2020), a pretrained language model that is proficient in comprehending the diverse user utterances and generating the GraphQ IR sequences that are structured in natural-language-like expressions.

Please note that the implementation in this part is orthogonal to our GraphQ IR and can be substituted by other semantic parsing models.

3.2 Compiler

The implementation of GraphQ IR's compiler comprises a front-end module that generates an abstract syntax tree from the IR sequence and a back-end module that transforms the tree structure into the target graph query language.

The compiler front-end is responsible for performing the lexical and syntax analysis on the IR sequence. The lexer first splits the sequence into lexical tokens, which are subsequently structured into a parse tree with *LL(*)* parsing strategy (Parr, 2013) according to the pre-defined grammar in Section 2.1. As such, GraphQ IR sequence can be automatically constructed into an abstract syntax tree (AST) that contains syntactic dependencies and hierarchical structures.

The compiler back-end will then traverse the abstract syntax tree and restructure the nodes and dependencies into one of the downstream graph query languages. We formalize the code generation as a tree mapping process, where the subtrees carrying equivalent information are aligned according to pre-defined transformation rules. To illustrate, we present 2 examples of generating SPARQL and Lambda-DCS queries respectively in Appendix Figure 5 and Figure 4.

Similarly, we also implement the compiler that supports conversion from graph query languages to GraphQ IR. Thus, with the IR as a middleware, our toolkit can also achieve the transpilation between any two graph query languages supported.

4 Experiments

In this section, we evaluate GraphQ IR on several benchmarks under different task settings.

4.1 Datasets

For evaluation, we test on benchmarks KQA PRO, OVERNIGHT, GRAILQA and METAQA-Cypher that altogether cover graph query languages SPARQL, KoPL, Lambda-DCS, and Cypher.

In all experiments, the GraphQ IR sequences are automatically converted from the original logical forms of the respective datasets by the bidirectional compiler without extra re-annotation.

KQA Pro KQA PRO (Cao et al., 2022a) is a large-scale dataset for complex question answering over Wikidata knowledge base (Vrandecic and Krötzsch, 2014). It is the largest KBQA corpus that contains 117,790 natural language questions along with the corresponding SPARQL and KoPL logical forms, covering complex graph queries involving multi-hop inference, logical union and intersection, etc. In our experiment, it is divided into 94,376 train, 11,797 validation, and 11,797 test cases.

Overnight OVERNIGHT (Wang et al., 2015) is a semantic parsing dataset with 13,682 examples across 8 sub-domains extracted from Freebase (Bollacker et al., 2008). Each domain has natural language questions and pairwise Lambda-DCS queries executable on SEMPRE (Berant et al., 2013). It exhibits diverse linguistic phenomena and semantic structures across domains, e.g., temporal knowledge in CALENDAR domain and spatial knowledge in BLOCKS domain. We use the same train/val/test splits as in the previous work (Wang et al., 2015).

GrailQA GRAILQA (Gu et al., 2021) is a knowledge base question answering dataset with 64k questions grounded on Freebase (Bollacker et al., 2008) that evaluate generalizability at three levels, *i.e.*, i.i.d, compositional generalization and zero-shot. To focus on the sole task of semantic parsing, we replace the entity IDs (*e.g.*, `m.06mn7`) with their respective names (*e.g.*, `Stanley Kubrick`) in GRAILQA's logical forms, thus eliminating the need for an explicit entity linking module as in previous works (Chen et al., 2021; Ye et al., 2022). Since GRAILQA's test set is not publicly available for such transformation, we report the validation set results for our evaluation, which have been studied to show consistent trends with the test set (Gu and Su, 2022).

MetaQA-Cypher METAQA (Zhang et al., 2018) contains more than 400k multi-hop QA pairs over WikiMovies knowledge base (Miller et al.,

	Multi-hop	Qualifier	Comparison	Logical	Count	Verify	Zero-shot	Overall
Baselines								
RGCN (Schlichtkrull et al., 2018)	34.00	27.61	30.03	35.85	41.91	65.88	-	35.07
BART+SPARQL (Cao et al., 2022a)	88.49	83.09	96.12	88.67	85.78	92.33	87.88	89.68
BART+KoPL (Cao et al., 2022a)	89.46	84.76	95.51	89.30	86.68	93.30	89.59	90.55
CFQ IR (Herzig et al., 2021)	87.51	81.32	95.70	90.33	86.23	92.20	87.12	88.96
Our Approach								
GraphQ IR	**90.38**	**84.90**	**97.15**	**92.64**	**89.39**	**94.20**	**94.20**	**91.70**

Table 1: Test accuracies on KQA PRO dataset. Data are categorized into MULTI-HOP queries with multi-hop inference, QUALIFIER knowledge queries, COMPARISON between several entities, LOGICAL union or intersection, COUNT queries for the quantity of entities, VERIFY queries with a boolean answer, and ZERO-SHOT queries whose answer is not seen in the training set.

	Bas.	Blo.	Cal.	Hou.	Pub.	Rec.	Res.	Soc.	Overall
Baselines									
SPO (Wang et al., 2015)	46.3	41.9	74.4	54.0	59.0	70.8	75.9	48.2	58.8
CrossDomain* (Su and Yan, 2017)	88.2	62.2	**82.1**	78.8	80.1	86.1	83.7	83.1	80.6
Seq2Action (Chen et al., 2018a)	88.2	61.4	81.5	74.1	80.7	82.9	80.7	82.1	79.0
DUAL (Cao et al., 2019)	84.9	61.2	78.6	67.2	78.3	80.6	78.9	81.3	76.4
2-stage DUAL* (Cao et al., 2020)	87.2	**65.7**	80.4	75.7	80.1	86.1	82.8	82.7	80.1
Our Approach									
GraphQ IR	88.2	64.7	78.6	72.0	77.6	83.3	84.9	81.6	79.5
GraphQ IR*	**88.2**	65.4	81.6	**81.5**	**82.6**	**92.9**	**89.8**	**84.1**	**82.1**

Table 2: Test accuracies on OVERNIGHT dataset. Methods with asterisk (*) involve cross-domain training.

2016). Many studies have previously worked on its SPARQL annotation (Huang et al., 2021). Instead, we reconstruct METAQA into Cypher as a few-shot learning benchmark to evaluate the interoperability achieved by GraphQ IR. To the best of our knowledge, this is also the first Cypher dataset in the community of semantic parsing.

4.2 Metric

We adopt *execution accuracy* as our metric based on whether the generated logical form queries can return correct answers. For queries with multiple legal answers, we require the execution results to exactly match *all* ground-truth answers.

4.3 Results

I.I.D. Generalization As Table 1 illustrates, on KQA PRO, our proposed approach with GraphQ IR consistently outperforms the previous approaches on all query categories. In particular, GraphQ IR exhibits good generalization under the complex MULTI-HOP, QUALIFIER and ZERO-SHOT settings with even larger margins over the baselines. We attribute this to its natural-language-like representations that effectively close the semantic gap and its formally-defined syntax that can be losslessly converted into downstream languages.

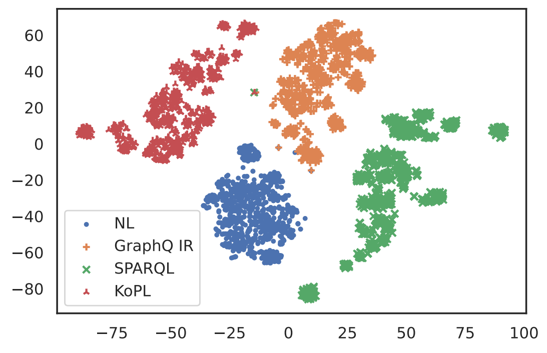
As for OVERNIGHT, our methods also significantly surpass the baselines as shown in Table 2.

	I.I.D.	CG	Zero-shot	Overall
Models with entity linking				
Gu et al. (2021)	58.6	40.9	51.8	51.0
Ye et al. (2022)	**86.7**	**61.7**	**68.8**	**71.4**
Models without entity linking				
BART	81.1	31.6	3.6	28.1
CFQ IR	86.8	46.6	5.3	34.0
GraphQ IR	**87.4**	**49.5**	**9.6**	**36.9**

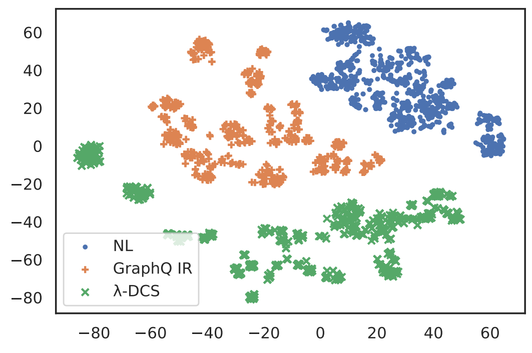
Table 3: Validation results on GRAILQA's i.i.d, compositional generalization and zero-shot data splits. The results of two groups of methods (*i.e.*, with/without entity linking) are not fully comparable.

Previous works usually train separate parsers for each of the eight domains due to their distinct vocabularies and grammars (Wang et al., 2015; Chen et al., 2018a). With an extra layer of GraphQ IR for unification, domain-specific data are now consolidated into one universal representation, and the training of one domain can thereby benefit from the others. Consequently, GraphQ IR* that gets trained on the aggregate data of all eight domains demonstrates the best results.

OOD Generalization Current neural semantic parsers often fail in generalizing to out-of-distribution (OOD) data (Pasupat and Liang, 2015; Keysers et al., 2020; Furrer et al., 2020). Therefore, we experiment on GRAILQA, a dataset that



(a) Embedding visualization on KQA PRO.



(b) Embedding visualization on OVERNIGHT.

Figure 3: t-SNE visualization of the sequence embeddings of the natural language utterance, GraphQ IR and downstream graph query languages that are randomly sampled from the validation set of KQA PRO and OVERNIGHT.

	1-shot	**3-shot**	**5-shot**
BART	73.93	91.99	94.37
GraphQ IR	72.05	93.73	95.16
GraphQ IR*	**84.91**	**95.31**	**96.13**

Table 4: Few-shot learning results on METAQA-Cypher dataset. GraphQ IR* model has formerly trained on KQA PRO dataset prior to the few-shot fine-tuning.

		NL ⇔ IR
KQA PRO	NL ⇔ SPARQL	-25.28%
	NL ⇔ KoPL	-16.57%
OVERNIGHT	NL ⇔ Lambda-DCS	-15.80%

Table 5: Semantic distance between natural language utterances and GraphQ IR (*i.e.*, NL ⇔ IR) relatively compared to the distance between natural language utterances and specified logical forms.

specifically stresses non-i.i.d. generalization.

We present the results in Table 3. Among the models without explicit entity linking modules, compared with the BART baseline that directly maps to the logical forms and the CFQ IR (Herzig et al., 2021) that particularly aims at SPARQL compositional generalization, GraphQ IR achieves the best overall performance and performs remarkably well also in compositional generalization and zero-shot data splits. This can be credited to our IR designs that clarify the redundant semantics and maintain the key hierarchical structure where its components can be flexibly combined or decomposed according to the pre-defined production rules.

Low-resource Generalization To verify whether GraphQ IR can aid the semantic parsing of low-resource languages, we reconstruct the METAQA dataset into Cypher, a graph query language commonly used in the industry but rarely studied in previous semantic parsing works (Seifer et al., 2019). To simulate the low-resource scenario, we adjust the data split to ensure that only 1, 3, and 5 samples of each question type appear in the training set under the 1-, 3-, and 5-shot settings.

The results in Table 4 indicate that our methods can remain robust under a low-resource setting with strong few-shot generalization. Specifically, the GraphQ IR* model that has in advance trained on KQA PRO (a dataset annotated in SPARQL and KoPL) demonstrates the most outstanding performance on METAQA-Cypher, especially under the most challenging 1-shot setting. Previous works in semantic parsing usually target a specified type of logical form and neglect the data interoperability across languages. With the GraphQ IR as a bridge, low-resource query languages can now leverage data from other languages. A universal semantic parser that can end-to-end support different languages also becomes possible.

5 Discussion

To further explore the reasons behind the superior performance of our methods, we compute and visualize the semantic distance between the natural language utterances and their corresponding logical forms or GraphQ IR.

Specifically, to simulate how a neural semantic parser processes the sequences in the above experiments, we use a pretrained BART-base model without fine-tuning to obtain the contextualized embeddings (Li et al., 2020a). For each sequence, we

Error Type	#	# OSC	Example
Inaccurate data annotation	28	-	**User utterance:** Out of newscasts that last 110 minutes, which is the shortest?
			Gold SPARQL: `SELECT ?e WHERE { ?e instance_of ?c . ?c name "newscast" . ?e duration ?pv_1 . ?pv_1 unit "minute" . ?pv_1 value ?v_1 . FILTER ( ?v_1 != "110"^^xsd:double ) . ?e duration ?pv . ?pv value ?v } ORDER BY ?v LIMIT 1`
			Generated IR: `which one has the smallest <A> duration </A> among <ES> <C> newscast </C> whose <A> duration </A> is number <V> 110 minute </V> </ES>`
			Compiled SPARQL: `SELECT ?e WHERE { ?e instance_of ?c . ?c name "newscast" . ?e duration ?pv_1 . ?pv_1 unit "minute" . ?pv_1 value "110"^^xsd:double . ?e duration ?pv . ?pv value ?v } ORDER BY ?v LIMIT 1`
Ambiguous query expression	27	27	**User utterance:** How is the kid's movie The Spiderwick Chronicles related to John Sayles?
			Gold SPARQL: `SELECT DISTINCT ?p WHERE { ?e_1 name "The Spiderwick Chronicles" . ?e_1 genre ?e_3 . ?e_3 name "children's film" . ?e_2 name "John Sayles" . ?e_1 ?p ?e_2 }`
			Generated IR: `what is the relation from <ES> <E> The Spiderwick Chronicles </E> (<ES> ones that <R> genre </R> backward to <E> kid film </E> </ES>) </ES> to <E> John Sayles </E>`
			Compiled SPARQL: `SELECT DISTINCT ?p WHERE { ?e_1 name "The Spiderwick Chronicles" . ?e_1 genre ?e_3 . ?e_3 name "kid film" . ?e_2 name "John Sayles" . ?e_1 ?p ?e_2 }`
Unspecified graph structure	13	9	**User utterance:** When did Tashkent become the capital of Uzbekistan?
			Gold SPARQL: `SELECT DISTINCT ?qpv WHERE { ?e_1 name "Tashkent" . ?e_2 name "Uzbekistan" . ?e_1 capital_of ?e_2 . [ fact_h ?e_1 ; fact_r capital_of ; fact_t> ?e_2 ] start_time ?qpv }`
			Generated IR: `what is the qualifier <Q> start time </Q> of <E> Uzbekistan </E> that <R> capital </R> to <E> Tashkent </E>`
			Compiled SPARQL: `SELECT DISTINCT ?qpv WHERE { ?e_1 name "Uzbekistan" . ?e_2 name "Tashkent" . ?e_1 capital ?e_2 . [ fact_h ?e_1 ; fact_r capital ; fact_t ?e_2 ] start_time ?qpv }`
Nonequivalent semantics	32	25	**User utterance:** When did Joseph L. Mankiewicz graduate from Columbia University?
			Gold SPARQL: `SELECT DISTINCT ?qpv WHERE { ?e_1 name "Joseph L. Mankiewicz" . ?e_2 name "Columbia University" . ?e_1 educated_at ?e_2 . [ fact_h ?e_1 ; fact_r educated_at ; fact_t ?e_2 ] end_time ?qpv }`
			Generated IR: `what is the qualifier <Q> start time </Q> of <E> Joseph L. Mankiewicz </E> that <R> educated at </R> to <E> Columbia University </E>`
			Compiled SPARQL: `SELECT DISTINCT ?qpv WHERE { ?e_1 name "Joseph L. Mankiewicz" . ?e_2 name "Columbia University" . ?e_1 educated_at ?e_2 . [ fact_h ?e_1 ; fact_r educated_at ; fact_t ?e_2 ] start_time ?qpv }`

Table 6: The analysis of 4 error types based on the failure cases as occurred in benchmark KQA PRO's test data. "# OSC" refers to the number of errors that can be fixed with one step correction on the IR's structure.

take the average of the encoder outputs across all word tokens to obtain a 768-dimensional vector as its sentence embedding (Ni et al., 2022). Thereafter, we measure the semantic distance between two sequences by computing the Euclidean distance (L2-norm) of their embeddings (Chandrasekaran and Mago, 2021).

We randomly sampled 1000 queries respectively from KQA PRO and OVERNIGHT's validation set. We compare the semantic distance between natural language utterances and the GraphQ IRs (*i.e.*, NL ⇔ IR), as well as the distance between natural language utterances and their corresponding logical forms (*e.g.*, NL ⇔ SPARQL).

The results are listed in Table 5. The semantic distance from natural language utterances to GraphQ IR is significantly closer than that to different logical forms by at most 25.28%. We also use t-SNE (Van der Maaten and Hinton, 2008) to reduce the dimension and visualize the embeddings. Figure 3 (a) and (b) respectively shows the visualized feature space on KQA PRO and OVERNIGHT datasets. The computation and visualization results affirm our hypothesis that GraphQ IR can effectively close the semantic gap and ease the learning of neural semantic parser.

5.1 Error Analysis

To investigate GraphQ IR's potentials and bottlenecks, we look into the failures of our approach when incorrect logical forms are generated. Out of the total 979 errors in KQA PRO's test set, we randomly sampled 100 cases and categorized them into 4 types as shown in Table 6.

Inaccurate data annotation (28%). The reference logical form (*e.g.*, `v_1 != "110"`) may contain inconsistent or misinterpreted information that contradicts to the corresponding natural language utterance (*e.g.*, *last 110 minutes*). We attribute this type of error to the dataset rather than the failure of our approach.

Ambiguous query expression (27%). The semantics of the user utterance may be present in more than one way (*e.g.*, *kid film* or *children's film*) due to the ambiguity in natural language, whereas the schema of the knowledge base is pre-defined (*e.g.*, only `children's film` is considered a valid entity). This category of error can be fixed by incorporating explicit schema linking modules, which are orthogonal to the implementation of our GraphQ IR and semantic parser.

Unspecified graph structure (13%). Logical forms of different structures (*e.g.*, `(Uzbekistan` `capital Tashkent)` and `(Tashkent capital_of Uzbekistan)`) can convey the same semantics in a directed cycle graph, but some of them contain structures that are absent in a knowledge base. This type of error is due to the incompleteness of the knowledge base.

Nonequivalent semantics (32%). The output includes incorrect query element (*e.g.*, string and numerical values) or structure (*e.g.*, edges and properties) that conveys nonequivalent semantics, such as misinterpreting *graduate* to `start_time`.

Overall, 89% of the sampled errors can be simply fixed by the revision of annotation or one-step correction on the IR element, demonstrating that our proposed method with GraphQ IR can generate high-quality logical forms that are easy to debug.

6 Related Work

6.1 Semantic Parsing

Semantic parsing aims to translate natural language utterances into executable logical forms, such as CCG (Zettlemoyer and Collins, 2005), Lambda-DCS (Liang, 2013; Pasupat and Liang, 2015), SQL (Zhong et al., 2017; Yu et al., 2021), AMR (Banarescu et al., 2013), SPARQL (Sun et al., 2020) and KoPL (Cao et al., 2022a,b).

Most recent works take semantic parsing as a Seq2Seq translation task via an encoder-decoder framework, which is challenging due to the semantic and structural gaps between natural utterances and logical forms. To overcome such issues, current semantic parsers usually (1) rely on a large amount of labeled data (Cao et al., 2022a); or (2) leverage external resources for mini the structural mismatch, *e.g.*, injecting grammar rules during decoding (Wu et al., 2021; Shin et al., 2021); or (3) employ synthetic data to diminish the semantic mismatch (Xu et al., 2020; Wu et al., 2021).

Compared with previous works, our proposed GraphQ IR allows the semantic parser to adapt to different downstream formal query languages without extra efforts and demonstrates promising performance under the compositional generalization and few-shot settings.

6.2 Intermediate Representation

Intermediate representations (IR) are usually generated for the internal use of compilers and represent the code structure of input programs (Aho et al., 1986). Good IR designs with informative and distinctive mid-level features can provide huge benefits for optimization, translation, and downstream code generation (Lattner and Adve, 2004), especially in areas like deep learning (Chen et al., 2018b; Cyphers et al., 2018) and heterogeneous computing (Lattner et al., 2020).

Recently, IR has also become common in many semantic parsing works that include an auxiliary representation between natural language and logical form. Most of them take a top-down approach and adopt IR similar to natural language (Su and Yan, 2017; Herzig and Berant, 2019; Shin et al., 2021). In contrast, another category of works constructs IR based on the key structure of target logical forms in a bottom-up manner (Wolfson et al., 2020; Marion et al., 2021). For example, Herzig et al. designed CFQ IR that rewrites SPARQL by grouping the triples of identical elements (2021).

Although these works partially mitigate the mismatch between natural and formal language, they either failed in removing the formal representations that are unnatural to the language models or neglected the structural information requisite for downstream compilation. In this work, we omit those IRs that cannot be losslessly converted into downstream logical forms.

7 Conclusion and Future Work

This paper proposes a novel intermediate representation, namely GraphQ IR, for bridging the semantic gap between natural language and graph query languages. Evaluation results show that our approach with GraphQ IR consistently surpasses the baselines on several benchmarks covering multiple formal languages, *i.e.*, SPARQL, KoPL, Lambda-DCS, and Cypher. Moreover, GraphQ IR also demonstrates superior generalization ability and robustness under the out-of-distribution and low-resource settings.

As an early step towards the unification of semantic parsing, our work opens up several future directions. For example, many code optimization techniques (*e.g.*, common subexpression elimination) can be incorporated into IR to improve performance further. By bringing in multiple levels of IR, our framework may also be extended to support relational database query languages like SQL. Moreover, since the current designs of GraphQ IR still require non-trivial manual efforts, the automation of such procedure, *e.g.*, in prompt-like manners, is worth future exploration.

Limitations

The major limitations of this work include: (a) the composition rules of GraphQ IR are closely aligned with interrogative sentences. Therefore, our current formalism may not be applicable to general-domain semantic parsing; (b) for the semantic parsing of an input language whose syntax significantly differs from English (*e.g.*, Arabic, Chinese, Hindi, etc.), the benefits of GraphQ IR may be limited; (c) our experiments fine-tuned a neural semantic parser on top of a pretrained model with ∼139 million parameters, thus cannot be easily reproduced without adequate GPU resources.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. This work is partially supported by the National Key R&D Program of China (2021ZD0110104), the National Natural Science Foundation of China (U20A20226), the NSFC Youth Project (62006136), and a grant from the Institute for Guo Qiang, Tsinghua University (2019GQB0003). Jidong Zhai is the corresponding author of this paper.

References

Lakshya A. Agrawal, Nikunj Singhal, and Raghava Mutharaju. 2022. A SPARQL to cypher transpiler: Proposal and initial results. In *CODS-COMAD 2022: 5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD), Bangalore, India, January 8 - 10, 2022*, pages 312–313. ACM.

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley series in computer science / World student series edition. Addison-Wesley.

Renzo Angles. 2012. A comparison of current graph database models. In *Workshops Proceedings of the IEEE 28th International Conference on Data Engineering, ICDE 2012, Arlington, VA, USA, April 1-5, 2012*, pages 171–177. IEEE Computer Society.

Renzo Angles. 2018. The property graph database model. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018*, volume 2100 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5):68:1–68:40.

Renzo Angles, Harsh Thakkar, and Dominik Tomaszuk. 2019. RDF and property graphs interoperability: Status and issues. In *Proceedings of the 13th Alberto Mendelzon International Workshop on Foundations of Data Management, Asunción, Paraguay, June 3-7, 2019*, volume 2369 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Pablo Barceló Baeza. 2013. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 175–188. ACM.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, pages 178–186. The Association for Computer Linguistics.

Artur Baranowski and Nico Hochgeschwender. 2021. Grammar-constrained neural semantic parsing with LR parsers. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1275–1279. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1415–1425. The Association for Computer Linguistics.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Mihaela Bornea, Ramón Fernandez Astudillo, Tahira Naseem, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Pavan Kapanipathi, Radu Florian, and Salim Roukos. 2021. Learning to transpile AMR into SPARQL. *CoRR*, abs/2112.07877.

Giovanni Campagna, Silei Xu, Mehrad Moradshahi, Richard Socher, and Monica S. Lam. 2019. Genie: a generator of natural language semantic parsers for virtual assistant commands. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, pages 394–410. ACM.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 51–64. Association for Computational Linguistics.

Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. 2020. Unsupervised dual paraphrasing for two-stage semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6806–6817. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022a. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6101–6119. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022b. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8128–8140. Association for Computational Linguistics.

Dhivya Chandrasekaran and Vijay Mago. 2021. Evolution of semantic similarity - A survey. *ACM Comput. Surv.*, 54(2):41:1–41:37.

Bo Chen, Le Sun, and Xianpei Han. 2018a. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 766–777. Association for Computational Linguistics.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. Retrack: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 325–336. Association for Computational Linguistics.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Q. Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018b. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*, pages 578–594. USENIX Association.

Jianpeng Cheng, Siva Reddy, Vijay A. Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 44–55. Association for Computational Linguistics.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning, CoNLL 2010, Uppsala, Sweden, July 15-16, 2010*, pages 18–27. ACL.

Scott Cyphers, Arjun K. Bansal, Anahita Bhiwandiwalla, Jayaram Bobba, Matthew Brookhart, Avijit Chakraborty, William Constable, Christian Convey, Leona Cook, Omar Kanawi, Robert Kimball, Jason Knight, Nikolay Korovaiko, Varun Kumar Vijay, Yixing Lao, Christopher R. Lishka, Jaikrishnan Menon, Jennifer Myers, Sandeep Aswath Narayana, Adam Procter, and Tristan J. Webb. 2018. Intel ngraph: An intermediate representation, compiler, and executor for deep learning. *CoRR*, abs/1801.08058.

Marco Damonte and Emilio Monti. 2021. One semantic parser to parse them all: Sequence to sequence multi-task learning on semantic parsing datasets. In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics, *SEM 2021, Online, August 5-6, 2021*, pages 173–184. Association for Computational Linguistics.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 69–78. Springer.

George H. L. Fletcher, Marc Gyssens, Dirk Leinders, Dimitri Surinx, Jan Van den Bussche, Dirk Van Gucht, Stijn Vansummeren, and Yuqing Wu. 2015. Relative expressive power of navigational querying on graphs. *Inf. Sci.*, 298:390–406.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the*

2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018, pages 1433–1445. ACM.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *CoRR*, abs/2007.08970.

Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: three levels of generalization for question answering on knowledge bases. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. *CoRR*, abs/2204.08109.

Jonathan Herzig and Jonathan Berant. 2019. Don't paraphrase, detect! rapid and effective data collection for semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3808–3818. Association for Computational Linguistics.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *CoRR*, abs/2104.07478.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2007. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley.

Sen Hu, Lei Zou, and Xinbo Zhang. 2018. A state-transition framework to answer complex questions over knowledge base. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2098–2108. Association for Computational Linguistics.

Xin Huang, Jung-Jae Kim, and Bowei Zou. 2021. Unseen entity handling in complex question answering over knowledge base via language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 547–557. Association for Computational Linguistics.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander G. Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois P. S. Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G. P. Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging abstract meaning representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3884–3894. Association for Computational Linguistics.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1516–1526. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1223–1233. ACL.

Chris Lattner and Vikram S. Adve. 2004. LLVM: A compilation framework for lifelong program analysis & transformation. In *2nd IEEE / ACM International Symposium on Code Generation and Optimization (CGO 2004), 20-24 March 2004, San Jose, CA, USA*, pages 75–88. IEEE Computer Society.

Chris Lattner, Jacques A. Pienaar, Mehdi Amini, Uday Bondhugula, River Riddle, Albert Cohen, Tatiana Shpeisman, Andy Davis, Nicolas Vasilache, and Oleksandr Zinenko. 2020. MLIR: A compiler infrastructure for the end of moore's law. *CoRR*, abs/2002.11054.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020a. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9119–9130. Association for Computational Linguistics.

Zhuang Li, Lizhen Qu, and Gholamreza Haffari. 2020b. Context dependent semantic parsing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2509–2521. International Committee on Computational Linguistics.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *CoRR*, abs/1309.4408.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Mohamed Nadjib Mami, Damien Graux, Harsh Thakkar, Simon Scerri, Sören Auer, and Jens Lehmann. 2019. The query translation landscape: a survey. *CoRR*, abs/1910.03118.

Pierre Marion, Pawel Krzysztof Nowak, and Francesco Piccinno. 2021. Structured context and high-coverage grammar for conversational question answering over knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8813–8829. Association for Computational Linguistics.

Alberto O. Mendelzon and Peter T. Wood. 1995. Finding regular simple paths in graph databases. *SIAM J. Comput.*, 24(6):1235–1258.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409. The Association for Computational Linguistics.

Ezequiel José Veloso Ferreira Moreira and José Carlos Ramalho. 2020. SPARQLing Neo4J. In *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*, volume 83 of *OpenAccess Series in Informatics (OASIcs)*, pages 17:1–17:10, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Zifan Nan, Hui Guan, and Xipeng Shen. 2020. Hisyn: human learning-inspired natural language programming. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 75–86. ACM.

Jianmo Ni, Gustavo Hernandez Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1864–1874. Association for Computational Linguistics.

Terence Parr. 2013. *The definitive ANTLR 4 reference*. Pragmatic Bookshelf.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics.

Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. 2009. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3):16:1–16:45.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Trans. Assoc. Comput. Linguistics*, 4:127–140.

Marko A. Rodriguez. 2015. The gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages, Pittsburgh, PA, USA, October 25-30, 2015*, pages 1–10. ACM.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Philipp Seifer, Johannes Härtel, Martin Leinberger, Ralf Lämmel, and Steffen Staab. 2019. Empirical study on the usage of graph query languages in open source java projects. In *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2019, Athens, Greece, October 20-22, 2019*, pages 152–166. ACM.

Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and

Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7699–7715. Association for Computational Linguistics.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1235–1246. Association for Computational Linguistics.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. SPARQA: skeleton-based semantic parsing for complex questions over knowledge bases. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8952–8959. AAAI Press.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics.

Russell S Tomlin. 2014. *Basic Word Order (RLE Linguistics B: Grammar): Functional Principles*. Routledge.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Oskar van Rest, Sungpack Hong, Jinha Kim, Xuming Meng, and Hassan Chafi. 2016. PGQL: a property graph query language. In *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24 - 24, 2016*, page 7. ACM.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Learning from executions for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2747–2759. Association for Computational Linguistics.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1332–1342. The Association for Computer Linguistics.

Tomer Wolfson, Mor Geva, Ankit Gupta, Yoav Goldberg, Matt Gardner, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Trans. Assoc. Comput. Linguistics*, 8:183–198.

Peter T. Wood. 2012. Query languages for graph databases. *SIGMOD Rec.*, 41(1):50–60.

Shan Wu, Bo Chen, Chunlei Xin, Xianpei Han, Le Sun, Weipeng Zhang, Jiansong Chen, Fan Yang, and Xunliang Cai. 2021. From paraphrasing to semantic parsing: Unsupervised semantic parsing via synchronous semantic decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5110–5121. Association for Computational Linguistics.

Silei Xu, Sina J. Semnani, Giovanni Campagna, and Monica S. Lam. 2020. Autoqa: From databases to QA semantic parsers with only synthetic training data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 422–434. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6032–6043. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 440–450. Association for Computational Linguistics.

Pengcheng Yin, John Wieting, Avirup Sil, and Graham Neubig. 2022. On the ingredients of an effective zero-shot semantic parser. pages 1455–1474.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076. AAAI Press.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A GraphQ IR Grammar

We present GraphQ IR's non-terminals and production rules in Table 7.

B Implementation Details

B.1 Model Hyperparameters

For the neural semantic parser, we used the BART-base model (Lewis et al., 2020) released by Facebook on HuggingFace[1]. 12 special tokens (*e.g.*, `<E>`) were added to the tokenizer vocabulary as the structure indicators for GraphQ IR. We used the AdamW optimizer (Loshchilov and Hutter, 2017) with the learning rate set to $3e^{-5}$ and weight decay set to $1e^{-5}$ following the default settings.

B.2 Environmental Configurations

In our implementation of the compiler, we used ANTLR (Parr, 2013) version 4.9.2 for analyzing our specified grammar rules and building up the corresponding lexer and parser toolkit. For evaluation, we used Virtuoso 7.20[2], SEMPRE 2.4[3], Neo4j 4.4[4] and KoPL 0.3[5] as the back-ends respectively for executing the SPARQL, Lambda-DCS, Cypher, and KoPL queries.

Our whole experiments were performed on a single machine with 8 NVIDIA Tesla V100 (32GB memory) GPUs on CUDA 11.

C Supplementary Study

C.1 KQA Pro Compositional Generalization

Compositional generalization refers to a model's capability of generalizing from the known components to produce novel combinations (Pasupat and Liang, 2015; Keysers et al., 2020; Furrer et al., 2020). To measure our IR's compositional generalization ability, we also create a new KQA Pro data split based on the logical form length and test the parsers to generate long queries (KoPL queries with > 7 functions) based on the short query components seen in the training data (KoPL queries with ≤ 7 functions).

The results are listed in Table 8. Compared with the plain-BART baseline and the CFQ IR (Herzig et al., 2021) that is specially designed for improving the compositional generalization on SPARQL,

[1]https://huggingface.co/facebook/bart-base
[2]https://github.com/openlink/virtuoso-opensource
[3]https://github.com/percyliang/sempre
[4]https://github.com/neo4j/neo4j
[5]https://pypi.org/project/KoPL/

Non-terminal	**Productions**
S	→ EntityQuery | AttributeQuery | RelationQuery | QualifierQuery | CountQuery | VerifyQuery | ValueQuery
EntityQuery	→ *what is* EntitySet
AttributeQuery	→ *what is the attribute* Attribute *of* EntitySet
RelationQuery	→ *what is the relation from* EntitySet *to* EntitySet
QualifierQuery	→ *what is the qualifier* Qualifier *of* EntitySet Constraint
CountQuery	→ *how many* EntitySet
VerifyQuery	→ *whether* EntitySet Constraint
ValueQuery	→ *what is* Value
EntitySet	→ <ES> EntitySet LOP EntitySet </ES> | <ES> EntitySet Constraint </ES> | <ES> Concept EntitySet </ES> | Concept | Entity | *ones*
Constraint	→ AttributeConstraint QualifierConstraint? | RelationConstraint QualifierConstraint?
AttributeConstraint	→ *whose* Attribute COP Value | *that have* SOP Attribute
RelationConstraint	→ *that* Relation DIR *to* (COP Value?) EntitySet | *that* Relation DIR *to* SOP EntitySet
QualifierConstraint	→ Qualifier COP Value
Entity	→ <E> *entity* </E>
Concept	→ <C> *concept* </C>
Attribute	→ <A> *attribute* </A>
Relation	→ <R> *relation* </R>
Qualifier	→ <Q> *qualifier* </Q>
Value	→ VTYPE <V> Value LOP Value | VOP *of* Value | Attribute *of* Entity | VTYPE <V> *value* </V>
LOP	→ *and* | *or* | *not*
VOP	→ *sum* | *average* | *maximum* | *minimum*
COP	→ *is* | *is not* | *larger than* | *smaller than* | *at least* | *at most*
SOP	→ *largest* | *smallest*
DIR	→ *forward* | *backward*
VTYPE	→ *string* | *numeric* | *year* | *month* | *date* | *time*

Table 7: GraphQ IR grammar rules that cover the common graph query patterns. "|" separates multiple productions at the same level, and "?" denotes that the preceding expression is optional. Italic words refer to the terminal symbols. Here we omit the corner case production rules for simplicity.

	Overall	**Qualifier**	**Comparison**	**Logical**
BART	50.58	21.55	87.66	50.60
CFQ IR	50.70	25.33	93.77	50.73
GraphQ IR	**54.91**	**40.46**	**95.19**	**54.90**

Table 8: Experimental results on KQA PRO compositional generalization data split.

GraphQ IR achieves the best performance in overall data as well as in complex task settings, which can be again credited to our IR designs that simplify the redundant semantics and preserve the key structural features.

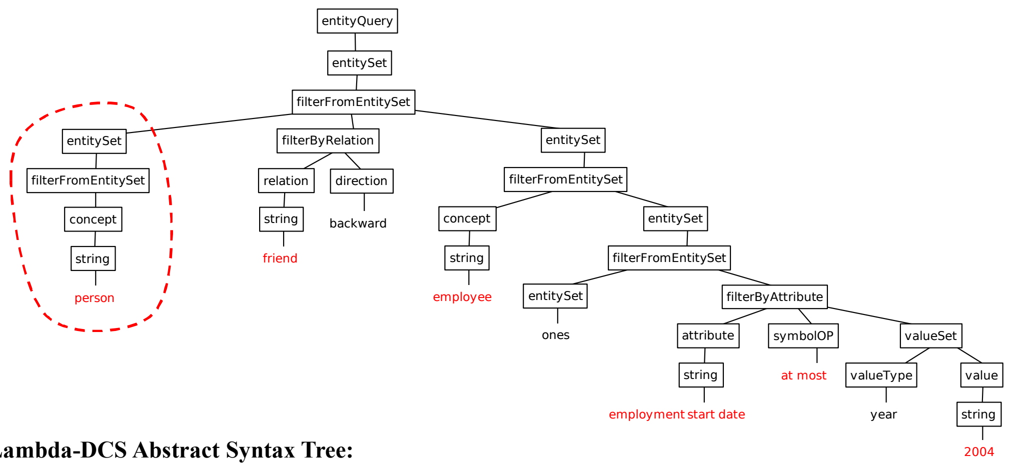
Natural Language Utterance:

friends of people who joined their jobs before 2005

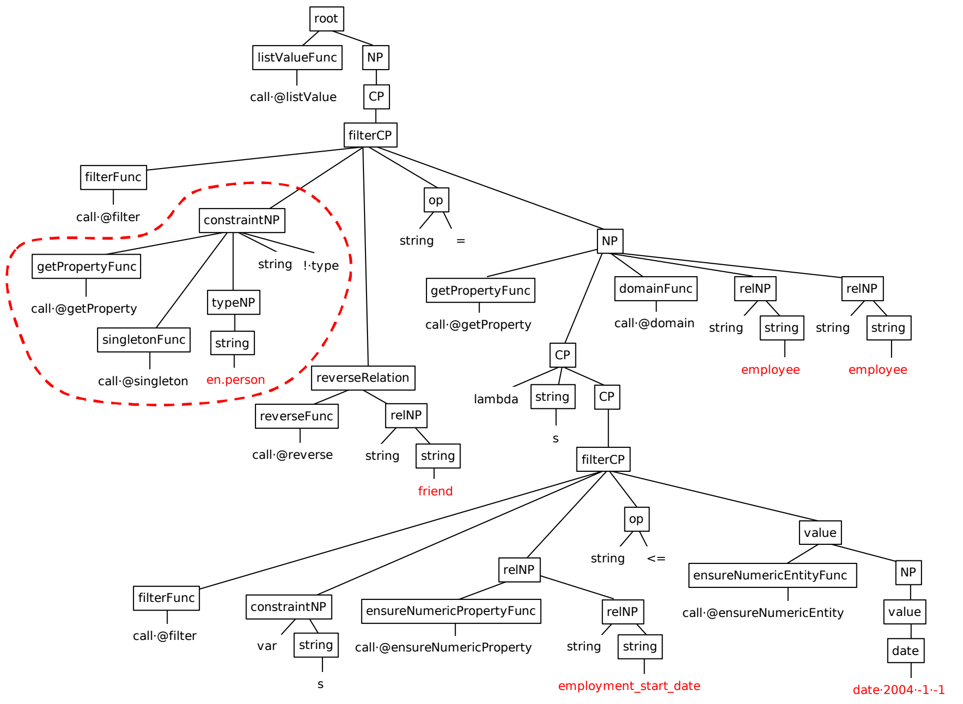
GraphQ IR Sequence:

```
what is <ES> <ES> <C> person </C> </ES> that <R> friend </R> backward to
<ES> <C> employee </C> <ES> ones
whose <A> employment start date </A> at most year <V> 2004 </V> </ES> </ES> </ES>
```

GraphQ IR Abstract Syntax Tree:



Lambda-DCS Abstract Syntax Tree:



Lambda-DCS Sequence:

```
( call @listValue ( call @filter ( call @getProperty ( call @singleton en.person )
( string ! type ) ) ( call @reverse ( string friend ) ) ( string = )
( call @getProperty ( ( lambda s ( call @filter ( var s ) ( call @ensureNumericProperty
( string employment_start_date ) ) ( string <= ) ( call @ensureNumericEntity ( date 2004
-1 -1 ) ) ) ) ) ( call @domain ( string employee ) ) ) ( string employee ) ) ) )
```

Figure 4: A user query in OVERNIGHT. The neural semantic parser first converts the input utterance into GraphQ IR. The compiler then parses the GraphQ IR sequence into an abstract syntax tree, which is subsequently transformed into the corresponding Lambda-DCS sequence along with a tree mapping process. To exemplify, the subtrees circled by red dash lines are carrying equivalent information that can be transformed with pre-defined rules. The red words are terminal nodes that correspond to the graph structure.

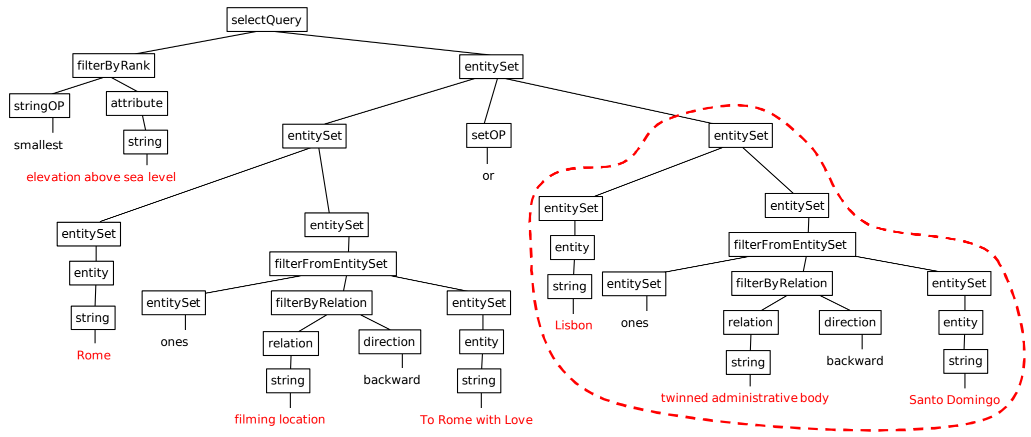
Natural Language Utterance:

Which has less elevation above sea level, Rome that is the filming location of To Rome with Love or Lisbon which is the twinned administrative body of Santo Domingo?

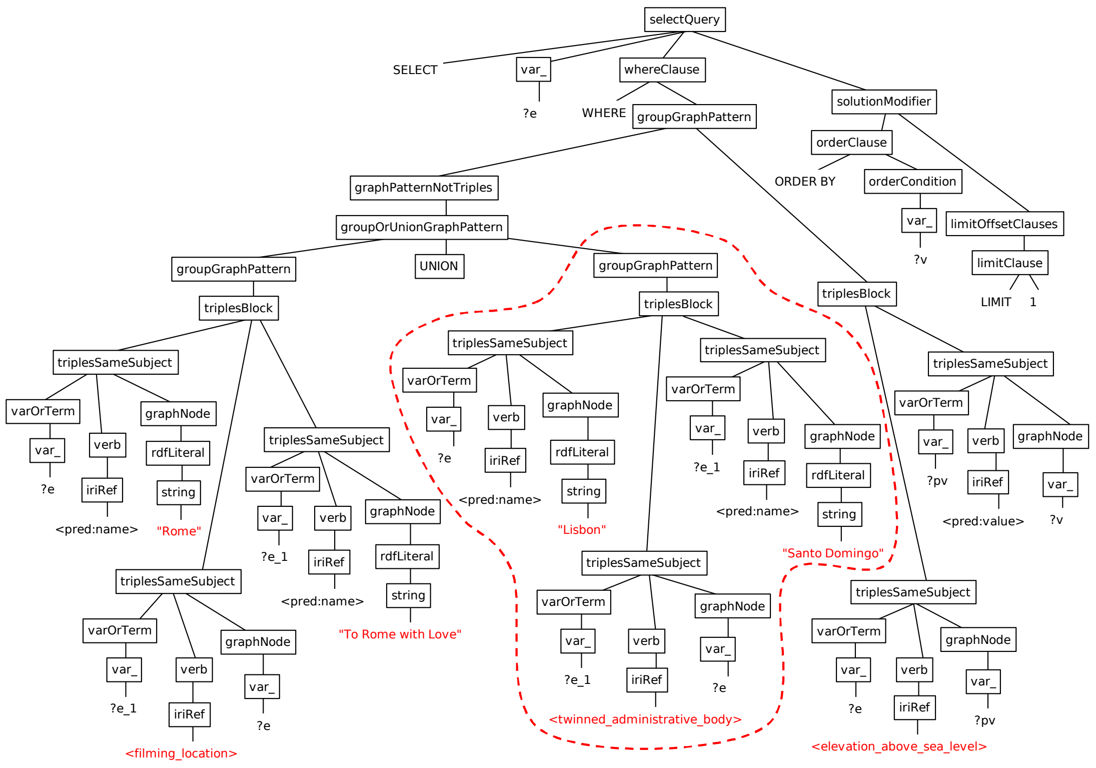
GraphQ IR Sequence:

```
which one has the smallest <A> elevation above sea level </A> among <ES> <ES> <E> Rome
</E> (<ES> ones that <R> filming location </R> backward to <E> To Rome with Love </E>
</ES>) </ES> or <ES> <E> Lisbon </E> (<ES> ones that <R> twinned administrative body </R>
backward to <E> Santo Domingo </E> </ES>) </ES> </ES>
```

GraphQ IR Abstract Syntax Tree:



SPARQL Abstract Syntax Tree:



SPARQL Sequence:

```
SELECT ?e WHERE { { ?e <pred:name> “Rome” . ?e_1 <filming_location> ?e . ?e_1 <pred:name>
“To Rome with Love” .   } UNION { ?e <pred:name> “Lisbon” .
?e_1 <twinned_administrative_body> ?e . ?e_1 <pred:name> "Santo Domingo" .   }
?e <elevation_above_sea_level> ?pv . ?pv <pred:value> ?v .  } ORDER BY ?v LIMIT 1
```

Figure 5: A user query in KQA PRO. Similarly, the compiler parses the generated GraphQ IR sequence into an abstract syntax tree, then transform its tree structure into the corresponding SPARQL sequence.