Understanding Inter-Session Intentions via Complex Logical Reasoning

Jiaxin Bai*
Department of CSE, HKUST
Hong Kong SAR, China
jbai@connect.ust.hk

Chen Luo
Amazon.com Inc
Palo Alto, USA
cheluo@amazon.com

Zheng Li
Amazon.com Inc
Palo Alto, USA
amzzhe@amazon.com

Qingyu Yin
Amazon.com Inc
Palo Alto, USA
qingyy@amazon.com

Yangqiu Song†
Department of CSE, HKUST
Hong Kong SAR, China
yqsong@cse.ust.hk

ABSTRACT

Understanding user intentions is essential for improving product recommendations, navigation suggestions, and query reformulations. However, user intentions can be intricate, involving multiple sessions and attribute requirements connected by logical operators such as And, Or, and Not. For instance, a user may search for Nike or Adidas running shoes across various sessions, with a preference for purple. In another example, a user may have purchased a mattress in a previous session and is now looking for a matching bed frame without intending to buy another mattress. Existing research on session understanding has not adequately addressed making product or attribute recommendations for such complex intentions. In this paper, we present the task of logical session complex query answering (LS-CQA), where sessions are treated as hyperedges of items, and we frame the problem of complex intention understanding as an LS-CQA task on an aggregated hypergraph of sessions, items, and attributes. This is a unique complex query answering task with sessions as ordered hyperedges. We also introduce a new model, the Logical Session Graph Transformer (LSGT), which captures interactions among items across different sessions and their logical connections using a transformer structure. We analyze the expressiveness of LSGT and prove the permutation invariance of the inputs for the logical operators. By evaluating LSGT on three datasets, we demonstrate that it achieves state-of-the-art results.

CCS CONCEPTS

• Information systems → Data mining; **• Computing methodologies → Semantic networks**; **Logic programming and answer set programming**.

* Work done during an internship at Amazon.
† Visiting academic scholar at Amazon.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08…$15.00
https://doi.org/10.1145/3637528.3671808

KEYWORDS

knowledge graph, complex query answering, session recommendation

ACM Reference Format:
Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, and Yangqiu Song. 2024. Understanding Inter-Session Intentions via Complex Logical Reasoning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671808

1 INTRODUCTION

Understanding user intention is a critical challenge in product search. A user's intention can be captured in many ways during the product search process. Some intentions can be explicitly given through search keywords. For example, a user may use keywords like "Red Nike Shoes" to indicate the desired product type, brand, and color. However, search keywords may not always accurately reflect the user's intention, especially when they are unsure of what they want initially. To address this issue, session-based recommendation methods have been proposed to leverage user behavior information to make more accurate recommendations [13, 22].

User intentions are usually complex. Users often have several explicit requirements for desired items, such as brand names, colors, sizes, and materials. For example, in Figure 1, query q_1 shows a user desiring Nike or Adidas products in the current session. On the other hand, users may spend multiple sessions before making a purchasing decision. For query q_2, the user spends two sessions searching for a desired product with an explicit requirement of purple color. Moreover, these requirements can involve logical structures. For instance, a user explicitly states that they do not want products similar to a previous session. In query q_3, a user has purchased a mattress in a previous session and is now looking for a wooden bed frame, without any intention of buying another mattress. With the help of logical operators like `AND` $\wedge$, `OR` $\vee$, and `NOT` $\neg$, we can describe the complex intentions by using a complex logical session query, like q_1, q_2, and q_3 in Figure 1.

Furthermore, there are scenarios where we are interested in obtaining product attributes based on sessions. For example, in query q_4 shown in Figure 1, we aim to identify the material types of the products desired in the session. Similarly, in query q_5, when given two sessions, we want to determine the brand names of the products desired in both sessions. To address these scenarios, we can describe these queries using logic expressions and variables. For instance, we can use the variable V_1 to represent the products and $V_?$ to represent the attribute associated with the product V_1. As a result, the task of recommending attributes based on complex user intentions can be formulated as logical query answering. We inquire about the attribute $V_?$ such that there exists a product V_1 in the given sessions, and the product attribute $V_?$ is desired.

To systematically answer queries with complex user intentions, we formally propose the task of logical session complex query answering (LS-CQA). This can be seen as an extension of the complex query answering (CQA) problem to relational hypergraph data, where sessions are treated as ordered hyperedges of items. The task of product or attribute recommendation under complex intention is reformulated as a task of answering logical queries on an aggregated hypergraph of sessions, items, and attributes. Figure 2 (C) provides an example of such an example, where each session is represented as a hyperedge connecting the corresponding items.

In addition to utilizing CQA methods with N-ary facts, such as NQE proposed by [26], another more reasonable approach to LS-CQA is to employ a session encoder. Recent studies [17, 21, 44] have shown the effectiveness of session encoders in encoding sessions and generating session representations. However, the neural session encoders tend to conduct implicit abstraction of products during the session encoding process [44]. The logical query encoder can only access the abstracted session representations, resulting in a lack of capturing the interactions between items in different sessions during the query encoding.

Motivated by this, we introduce the Logical Session Graph Transformer (LSGT) as an approach for encoding complex query sessions as hypergraphs[1]. Building upon the work by [19], we transform items, sessions, relation features, session structures, and logical structures into tokens, and they are then encoded using a standard transformer model. This transformation enables us to effectively capture interactions among items in different sessions through the any-to-any attention mechanisms in transformer models. By analyzing the Relational Weisfeiler-Lehman by [9, 16], we provide theoretical justification for LSGT, demonstrating that it possesses the expressiveness of at least 1-RWL, and has at least same expressiveness as existing logical query encoders that employ message-passing mechanisms for logical query encoding in WL test. Meanwhile, LSGT maintains the property of operation-wise permutation invariance, similar to other logical query encoders. To evaluate LSGT, we have conducted experiments on three evaluation datasets: Amazon, Diginetica, and Dressipi. Results demonstrate that LSGT achieves state-of-the-art performance on these datasets. In general, the contribution of this paper can be summarized as follows:

- We extend complex query answering (CQA) to hypergraphs with sessions as ordered hyperedges (LS-CQA) for describing and solving the product and attribute recommendations with complex user intentions. We also constructed three corresponding scaled datasets with the full support of first-order logical operators (intersection, union, negation) for evaluating CQA models on hypergraphs with ordered hyperedges and varied arity.
- We propose a new method, logical session graph transformer (LSGT). We use tokens and identifiers to uniformly represent the items, sessions, logical operators, and their relations. Then we use a transformer structure to encode them.
- We conducted experiments on Amazon, Digintica, and Dressipi to show that existing Transformer-based models show similar results on 3 benchmarks despite different linearization strategies. Meanwhile, We also find the linearization of LSGT leads to improvements in queries with negations and unseen query types. Meanwhile, We theoretically justify the expressiveness in the Weisfeiler-Lehman (WL) test and the Relational Weisfeiler-Lehman (RWL) test. We also prove the operator-wise permutation invariance of LSGT.

[1] Code available: https://github.com/HKUST-KnowComp/SessionCQA

2 PROBLEM FORMULATION

2.1 logical session complex query Answering

In previous work, complex query answering is usually conducted on a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R})$. However, on our aggregated hypergraph, there are items, sessions, and attribute values. Because of this, the graph definition is $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{S})$. The $\mathcal{V}$ is the set of vertices v, and the $\mathcal{R}$ is the set of relation r. The $\mathcal{S}$ is the set of sessions regarded as directed hyperedges. To describe the relations in logical expressions, the relations are defined in functional forms. Each relation r is defined as a function, and each relation has two arguments, which are two items or attributes v and v'. The value of function $r(v, v') = 1$ if and only if there is a relation between the items or attributes v and v'. Each session $s \in \mathcal{S}$ is the sequence of vertices where $s(v_1, v_2, ..., v_n) = 1$ if and only if $v_1, v_2, ..., v_n$ appeared in the same session.

The queries are defined in the first-order logical (FOL) forms. In a first-order logical expression, there are logical operations such as existential quantifiers $\exists$, conjunctions $\wedge$, disjunctions $\vee$, and negations $\neg$. In such a logical query, there are anchor items or attribute $V_a \in \mathcal{V}$, existential quantified variables $V_1, V_2, ...V_k \in \mathcal{V}$, and a target variable $V_? \in \mathcal{V}$. The knowledge graph query is written to find the answer $V_? \in \mathcal{V}$, such that there exist $V_1, V_2, ...V_k \in \mathcal{V}$ satisfying the logical expression in the query. For each query, it can be converted to a disjunctive normal form, where the query is expressed as a disjunction of several conjunctive expressions:

$$q[V_?] = V_?.\exists V_1, ..., V_k : c_1 \vee c_2 \vee ... \vee c_n, \quad (1)$$

$$c_i = e_{i1} \wedge e_{i2} \wedge ... \wedge e_{im}. \quad (2)$$

Each c_i represents a conjunctive expression of literals e_{ij}, and each e_{ij} is an atomic or the negation of an atomic expression in any of the following forms: $e_{ij} = r(v_a, V)$, $e_{ij} = \neg r(v_a, V)$, $e_{ij} = r(V, V')$, or $e_{ij} = \neg r(V, V')$. The atomics e_{ij} can be also hyper N-ary relations between vertices indicating that there exists a session among them. In this case, the $e_{ij} = s(v_1, v_2, ..., v_n, V)$ or its negations $e_{ij} = \neg s(v_1, v_2, ..., v_n, V)$. Here v_a and $v_i \in V_a$ is one of the anchor nodes, and $V, V' \in \{V_1, V_2, ..., V_k, V_?\}$ are distinct variables satisfying $V \neq V'$. When a query is an existential positive first-order (EPFO) query, there are only conjunctions $\wedge$ and disjunctions $\vee$ in the expression (no negations $\neg$). When the query is a conjunctive query, there are only conjunctions $\wedge$ in the expressions (no disjunctions $\vee$ and negations $\neg$).

Complex Queries	Interpretations
$q_1 = V_? : Session(Item_1, Item_2, \dots, Item_k, V_?) \wedge (Brand(V_?, Nike) \vee Brand(V_?, Adidas))$	Find the desired next item of a session with the brand Nike or Adidas.
$q_2 = V_? : Session(Item_{1,1}, Item_{1,2}, \dots, Item_{1,m}, V_?) \wedge Session(Item_{2,1}, Item_{2,2}, \dots, Item_{2,n}, V_?) \wedge Color(V_?, Purple)$	Find the desired next item of both $session_1$ and $session_2$ with purple color.
$q_3 = V_? : Session(Item_{1,1}, Item_{1,2}, \dots, Item_{1,m}, V_?) \wedge \neg Session(Item_{2,1}, Item_{2,2}, \dots, Item_{2,n}, V_?) \wedge Material(V_?, Wood)$	Find the item with wooden material that is desired by the $session_1$ but is not desired by $session_2$.
$q_4 = V_?, \exists V_1: Session(Item_1, Item_2, \dots, Item_k, V_1) \wedge Material(V_1, V_?)$	Find the material type of the product that a session desires.
$q_5 = V_?, \exists V_1: Session(Item_{1,1}, Item_{1,2}, \dots, Item_{1,m}, V_1) \wedge Session(Item_{2,1}, Item_{2,2}, \dots, Item_{2,n}, V_1) \wedge Brand(V_1, V_?)$	Find the brand name the product that is desired by both the $session_1$ and by the $session_2$.

Figure 1: Example complex queries involving varied numbers of sessions, products, and product attributes.

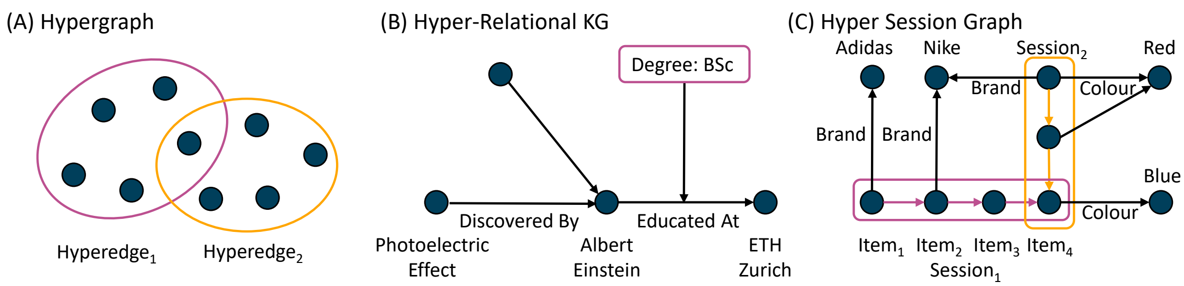

Figure 2: This figure shows the connections and differences between general hypergraphs, hyper-relational knowledge graphs, and the hyper-session graph in our problem.

(A) $q_1 = V_?, \exists V_1: Session(Item_{1,1}, \dots, Item_{1,m}, V_1) \wedge Session(Item_{2,1}, \dots, Item_{2,n}, V_1) \wedge Brand(V_1, V_?)$

(B) Find the brand name the product that is desired by both the $session_1$ and by the $session_2$.

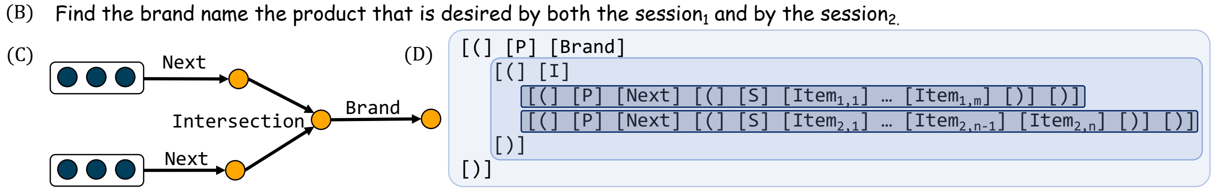

Figure 3: This figure shows the illustration of different query embedding methods. (A) The logical session complex query is expressed in the first-order logic form. (B) The interpretations on the logical session complex query. (C) The computational graph of the complex query proposed by [12]; (D) The linearization of the computational graph to token proposed by [6].

3 RELATED WORK

3.1 Hyper-Relational Graph Reasoning

The reasoning over hyper-relational KG proposed by [1], they extend the multi-hop reasoning problem to hyper-relational KGs and propose a method, StarQE, to embed and answer hyper-relational conjunctive queries using Graph Neural Networks and query embedding techniques. The StarQE conducts message-passing over the quantifier of the hyper-relations in KG, which cannot be directly used for encoding the hyper-relations in this task. [26] propose a novel Nary Query Embedding (NQE) model for complex query answering over hyper-relational knowledge graphs, which can handle more general n-ary FOL queries including existential quantifiers, conjunction, disjunction, and negation. The encoder design of NQE is more general to N-ary facts, thus it can be directly used for encoding the sessions as hyper-edges.

3.2 Complex Query Answering

Previous literature on logical query answering mainly focuses on knowledge graphs [2–5, 12, 14, 15, 23, 29, 35, 43]. Various methods are proposed to deal with the incompleteness issue of knowledge graphs by using the existing facts in the KG to generalize to the facts that are not in the KG but highly likely to be true. Recent advances in query embedding [12, 29, 30] methods have shown promise in answering logical queries on large-scaled graph-structured data effectively and efficiently. However, they cannot be directly used for answering queries with hyperedges, or in other words N-ary facts. Meanwhile, there are also methods [1, 26] that can perform robust reasoning on hyper-relational knowledge graphs, which is illustrated in Figure 2 (B). Because of the fundamental differences between the hyper-relational knowledge graphs and the hypergraphs of sessions, not all of them can be directly adopted for this task. Recently, there is also new progress on query encoding that is

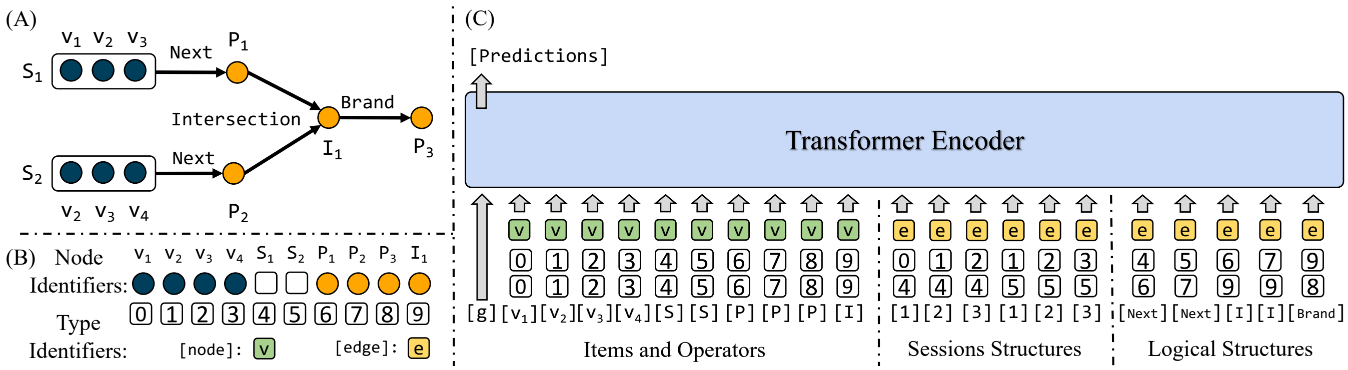

Figure 4: This figure shows the method of LSGT. (A) The computational graph indicates finding the brand of the products that are designed by both session S_1 and S_2. (B) The node identifiers and type identifiers for the tokens and each of the identifiers is associated with its corresponding embedding vector. (C) The transformer encoder is used for encoding the tokens.

orthogonal to this paper, which puts a focus on the neural encoders for complex queries. Xu et al. [41] propose a neural-symbolic entangled method, ENeSy, for query encoding. Yang et al. [42] propose to use Gamma Embeddings to encode complex logical queries. Liu et al. [25] propose to use pre-training on the knowledge graph with kg-transformer and then conduct fine-tuning on the complex query answering. Meanwhile, query decomposition [2] is another way to deal with the problem of complex query answering. In this method, the probabilities of atomic queries are first computed by a link predictor, and then continuous optimization or beam search is used to conduct inference time optimization. Moreover, [36] propose an alternative to query encoding and query decomposition, in which they conduct message passing on the one-hop atomics to conduct complex query answering. Recently a novel neural search-based method QTO [8] is proposed. QTO demonstrates impressive performance CQA. There are also neural-symbolic query encoding methods proposed [41, 45]. In this line of research, their query encoders refer back to the training knowledge graph to obtain symbolic information from the graph. LogicRec [33] discusses recommending highly personalized items based on complex logical requirements, which current recommendation systems struggle to handle.

3.3 Session Encoders

In recent literature, various methods have been proposed to reflect user intentions and build better recommendation systems using session history. Because of the nature of sequence modeling, various methods utilize recurrent neural networks (RNNs) and convolutions neural networks (CNNs) to model session data [13, 22, 24, 32]. Recent developments in session-based recommendation have focused on using Graph Neural Networks (GNNs) to extract relationships and better model transitions within sessions [11, 17, 21]. Wu et al. [38] were the first to propose using GNNs to capture complex transitions with graph structures, and subsequent research has incorporated position and target information, global context, and highway networks to further improve performance [28, 39]. However, previous efforts have focused more on the message-passing part and less on designing effective readout operations to aggregate embeddings to the session-level embedding. According to [44], current readout operations have limited capacity in reasoning over sessions, and the performance improvement of GNN models is not significant enough to justify the time and memory consumption of sophisticated models. So [44] proposed a pure attention-based method Atten-Mixer to conduct session recommendations.

4 LOGICAL SESSION GRAPH TRANSFORMER

In this session, we describe the logical session graph transformer (LSGT) for encoding logical queries involving sessions. In LSGT, the node and edge features, session structures, and logical structures are all converted into tokens and identifiers. Subsequently, they serve as input to a standard transformer encoder model.

4.1 Items, Sessions, and Operators Tokens

The first step in LSGT involves assigning node identifiers to items, sessions, and operators. For instance, in Figure 4, there are two sessions, S_1 and S_2, with items $[v_1, v_2, v_3]$ and $[v_2, v_3, v_4]$, respectively. The computational graph then uses relational projection operators P_1 and P_2 to find the two sets of next items desired by S_1 and S_2, respectively. Once all items, sessions, and operators have been identified, each is assigned a unique node identifier. For example, v_1 to v_4 are assigned identifiers from 0 to 3, S_1 and S_2 are assigned identifiers 4 and 5, projections from P_1 to P_3 are assigned identifiers from 6 to 8, and intersection operation I_1 is assigned to 9.

In general, when there are n nodes denoted by their identifiers as $\{p_0, p_1, ..., p_n\}$, their node features are assigned as follows: if p_i is an item, its features are assigned to its item embedding. If p_i is a session S_j, it is assigned an embedding of [S]. If p_i is a neural operator, it is assigned the operator embedding from [P], [I], [N], or [U] based on its operation type. The feature matrix for these n nodes is then denoted as $\boldsymbol{X_p} \in \mathbb{R}^{n \times d_1}$. Additionally, each node identifier is associated with random orthonormal vectors [19], denoted as $\boldsymbol{P_p} \in \mathbb{R}^{n \times d_2}$. All nodes are assigned the type identifier of [node], which means that they are the nodes in the computational graph. The token type embedding for vertices is denoted as $T_{[node]} \in \mathbb{R}^{d_3}$. The input vectors for the transformer are concatenations of node features, the random orthonormal vectors, and token type embeddings, where node identifiers vectors are repeated twice: $X_u^v = [X_p, P_p, P_p, T_{[node]}] \in \mathbb{R}^{n\times(d_1+2d_2+d_3)}$.

4.2 Session Structure Tokens

In this part, we describe the process of constructing the input tokens to indicate the session structure, namely which items are in which session in which position. Suppose the item p is from session q and at the position of r, and there are m item-session correspondences in total. First, we use positional encoding $Pos(\mathrm{r}) \in \mathbb{R}^{d_1}$ to describe the positional information. Meanwhile, as the item and sessions are associated with their node identifiers p and q, we use the node identifier vectors $P_p \in \mathbb{R}^{d_2}$ and $P_q \in \mathbb{R}^{d_2}$ to represent them. Meanwhile, this token represents a correspondence between two nodes, so we use the `[edge]` token type embedding to describe this $T_{[edge]} \in \mathbb{R}^{d_3}$. As there are in total m of item-session correspondences, we concatenate them to obtain the input vectors for the tokens representing session structures: $X^s_{(p,q,r)} = [Pos(\mathrm{r}), P_p, P_q, T^{[edge]}] \in \mathbb{R}^{m\times(d_1+2d_2+d_3)}$.

4.3 Logical Structure Tokens

In this part, we describe the process of constructing the input for tokens to indicate the logical structures. As shown in Figure 4, in an edge representing a logical operation, there are two nodes p and q respectively. If the logical operation is projection, then the edge feature is assigned with relation embedding `[Rel]`. Otherwise, the edge feature is assigned with the operation embedding from `[P]`, `[I]`, `[N]`, and `[U]` accordingly. The edge feature is denoted as $R_r \in \mathbb{R}^{d_1}$. Similarly, we use the node identifier vectors $P_p \in \mathbb{R}^{d_2}$ and $P_q \in \mathbb{R}^{d_2}$ to represent involved nodes p and q. Meanwhile, this token represents an edge in the computational graph, so we also associate it with token type embedding $T_{[edge]} \in \mathbb{R}^{d_3}$ to describe it. Suppose there are in total w such logical edges, we concatenate them together to obtain the input vectors for the tokens representing logical structure: $X^l_{(p,q,r)} = [R_r, P_p, P_q, T_{[edge]}] \in \mathbb{R}^{w\times(d_1+2d_2+d_3)}$.

4.4 Training LSGT

After obtaining the three parts describing the items, session structures, and logical structures, we concatenate them together $X = [X_{[\mathrm{graph}]}, X^v, X^s, X^l] \in R^{(m+n+w+1)\times(d_1+2d_2+d_3)}$, and use this matrix as the input for a standard transformer encoder for compute the query encoding of this complex logical session query. Then we append a special token `[graph]` with embedding $X_{[\mathrm{graph}]} \in \mathbb{R}^{d_1+2d_2+d_3}$ at the beginning of the transformer and use the token output of the `[graph]` token as the embedding of the complex logical session query. To train the LSGT model, we compute the normalized probability of the vertice a being the correct answer of query q by using the `softmax` function on all similarity scores,

$$p(q,a) = \frac{e^{<e_q,e_a>}}{\sum_{a'\in V} e^{<e_q,e_{a'}>}}. \tag{3}$$

Then we construct a cross-entropy loss to maximize the log probabilities of all correct pairs:

$$L = -\frac{1}{N}\sum_i \log p(q^{(i)}, a^{(i)}). \tag{4}$$

Each $(q^{(i)}, a^{(i)})$ denotes one of the positive query-answer pairs, and there are N pairs.

Table 1: The statistics of the constructed hypergraph on sessions, items, and their attribute values are shown.

Dataset	Edges	Vertices	Sessions	Items	Attributes	Relations
Amazon	8,004,984	2,431,747	720,816	431,036	1,279,895	10
Diginetica	1,387,861	266,897	12,047	134,904	125,204	3
Dressipi	2,698,692	674,853	668,650	23,618	903	74

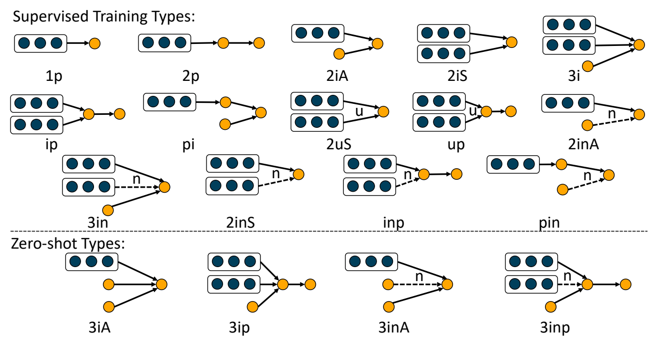

Figure 5: The query structures are used for training and evaluation. For brevity, the p, i, n, and u represent the projection, intersection, negation, and union operations. The query types are trained and evaluated under supervised settings.

4.5 Theoretical Properties of LSGT

In this part, we analyze the theoretical properties of LSGT, focusing on two perspectives. First, we analyze the expressiveness of LSGT compared to baseline methods in Theorem 1 and 2. Second, we analyze whether LSGT has operator-wise permutation invariant, and this is important in query encoding as operators like `Intersection` and `Union` are permutation invariant to inputs in Theorem 3. We prove the following theorems in the Appendix A of LSGT:

THEOREM 1. *When without considering the relation types in the query graph, the expressiveness of the LSGT encoder is at least the same as that of the encoder that combines a session encoder followed by a logical query encoder under Weisfeiler-Lehman tests[27].*

THEOREM 2. *When considering the query graphs are multi-relational graphs with edge relation types, the expressiveness of the LSGT encoder is also at least as powerful as 1-RWL, namely the expressiveness of R-GCN and CompGCN [9, 16].*

THEOREM 3. *LSGT can approximate a logical query encoding model that is operator-wise input permutation invariant.*

Table 2: The performance in the mean reciprocal ranking of LSGT compared with the baseline models of SQE and NQE with different backbone modules. Statistical significance is denoted by *.

Dataset	Query Encoder	Session Encoder	1p	2p	2ia	2is	3i	pi	ip	2u	up	Average EPFO
Amazon	FuzzQE	GRURec	11.99	6.96	54.79	88.47	68.13	16.73	14.49	10.02	6.87	30.94
		SRGNN	13.52	7.93	54.12	85.45	67.56	18.62	**20.46**	10.82	7.24	31.75
		Attn-Mixer	16.79	7.96	55.76	**89.64**	69.16	14.87	9.93	13.83	7.20	31.68
	Q2P	GRURec	11.60	6.83	34.73	67.64	42.18	16.66	13.82	8.54	5.82	23.09
		SRGNN	13.94	7.69	35.89	69.90	44.61	16.19	16.44	10.20	6.46	24.59
		Attn-Mixer	15.93	8.53	46.67	68.62	61.13	16.95	15.78	12.43	7.41	28.16
	NQE	-	5.60	2.50	48.40	77.98	63.06	2.20	1.80	4.20	3.00	23.19
	SQE-Transformer	-	16.09	8.30	53.90	72.26	64.48	17.54	16.80	13.86	7.35	30.07
	SQE-LSTM	-	16.59	7.45	55.60	86.81	69.11	17.86	19.04	13.46	6.87	32.53
	LSGT (Ours)		**17.73***	**9.10***	**56.73***	84.62	**69.39**	**19.39***	19.47	**15.40***	**7.86***	**33.26*** **(+0.73)**
Diginetica	FuzzQE	GRURec	24.10	12.29	82.48	89.19	86.26	11.64	23.34	18.19	11.18	39.85
		SRGNN	22.53	12.33	83.19	88.35	86.26	12.55	29.56	19.76	11.48	40.67
		Attn-Mixer	**33.87**	11.89	82.94	88.94	**86.36**	12.28	28.21	24.78	10.81	42.23
	Q2P	GRURec	26.02	**23.73**	62.46	83.95	76.25	21.77	32.04	17.00	21.62	40.54
		SRGNN	18.76	22.29	52.94	84.67	58.72	21.93	30.34	13.04	**20.86**	35.95
		Attn-Mixer	34.87	24.36	55.00	87.09	58.46	**22.81**	31.26	25.76	21.60	40.13
	NQE	-	15.82	11.24	76.79	87.16	79.52	11.07	30.76	11.12	10.14	37.07
	SQE-Transformer	-	30.60	14.93	83.72	**90.87**	80.58	15.18	32.72	25.61	13.98	43.13
	SQE-LSTM	-	31.50	14.10	83.67	86.70	84.76	14.46	30.08	21.92	12.53	42.19
	LSGT (Ours)		32.00	15.27	**83.34***	90.61	86.05	15.62	**33.80***	**26.34***	14.45	**44.16*** **(+1.03)**
Dressipi	FuzzQE	GRURec	27.62	94.28	56.15	77.21	75.40	94.81	98.43	23.46	95.52	71.43
		SRGNN	30.18	94.90	52.41	74.63	73.38	95.37	98.32	25.09	95.69	71.11
		Attn-Mixer	30.60	94.80	57.17	78.14	75.94	94.83	**98.57**	24.39	95.69	72.24
	Q2P	GRURec	**35.93**	95.20	45.22	66.62	51.20	96.27	92.58	25.46	95.45	67.10
		SRGNN	35.48	95.95	46.05	64.01	52.58	95.75	92.81	25.28	95.68	67.07
		Attn-Mixer	37.92	96.04	47.06	66.47	50.91	96.22	94.88	26.16	95.75	67.93
	NQE	-	11.52	95.62	21.19	52.79	48.28	96.08	98.04	13.39	95.80	59.19
	SQE-Transformer	-	27.01	95.37	62.38	**80.55**	**79.72**	96.02	97.99	24.55	95.95	73.28
	SQE-LSTM	-	25.84	94.81	62.23	64.19	70.43	95.39	96.91	25.23	95.62	70.07
	LSGT (Ours)		31.12	**96.16***	**64.26***	76.85	78.66	**98.02***	96.98	**28.83***	**96.04***	**74.10*** **(+0.82)**

5 EXPERIMENT

We use three public datasets from KDD-Cup [2] [18], Diginetica [3], and Dressipi [4] for evaluation. The number of items, sessions, and relations are reported in Table 1. Following previous work [7, 30, 37], we use eighty percent of the edges for training, ten percent of edges for validation, and the rest of the edges as testing edges. As shown in Figure 5, we conduct sampling of fourteen types of logical session queries by using the sampling algorithm described by [7]. The number of queries is shown in Table 7. Each of the queries has a concrete meaning. For example, the 1p queries are vanilla session-based product recommendations, and the 2p queries aim to recommend product attributes based on a single session history. A detailed explanation of the query types is shown in Appendix B.

[2]https://www.aicrowd.com/challenges/amazon-kdd-cup-23-multilingual-recommendation-challenge
[3]https://competitions.codalab.org/competitions/11161
[4]https://dressipi.com/downloads/recsys-datasets

5.1 Baseline Models

We briefly introduce the baseline query encoding models that use various neural networks to encode the query into embedding structures. Here are the baseline models for the complex query-answering models:

- NQE [26] is a method that can be used to encode N-ary facts from the KG;
- SQE [7] uses sequence encoders to encode linearized complex queries.

In the hyper-relational session-product-attribute graph, each session can be formulated as a directed hyper-relation among various entities. Because of this, we construct the relation of NEXT connecting the items that are browsed in the session following the corresponding order. We employed a state-of-the-art session encoder to model the item history within a session. The session encoder takes into account the temporal dependencies and context of products, effectively creating a contextual representation of the entire session:

Table 3: The performance in the mean reciprocal ranking of LSGT compared with the baseline models of SQE and NQE with different backbone modules on the queries involving negations. The statistical significance is denoted by *.

Dataset	Query Encoder	Session Encoder	2ina	2ins	3in	inp	pin	Average Negative
Amazon	FuzzQE	GRURec	10.11	10.39	50.83	30.11	3.72	21.03
		SRGNN	12.02	11.08	51.37	30.79	6.06	22.26
		Attn-Mixer	17.28	17.47	53.77	31.96	4.55	25.00
	Q2P	GRURec	9.56	10.21	18.59	30.83	3.87	14.61
		SRGNN	11.57	11.97	20.08	35.07	4.42	16.62
		Attn-Mixer	18.75	20.68	51.52	**37.04**	6.78	26.95
	NQE	-	5.00	5.10	48.16	30.26	2.10	18.12
	SQE-Transformer	-	18.15	18.88	55.83	34.76	8.21	27.16
	SQE-LSTM	-	18.42	19.10	56.99	33.67	7.45	27.13
	LSGT (Ours)		**20.98***	**22.00***	**60.70***	35.95	**8.84***	**29.69* (+2.93)**
Diginetica	FuzzQE	GRURec	16.15	9.09	81.65	14.07	10.69	26.33
		SRGNN	16.62	15.77	82.30	14.92	10.69	28.06
		Attn-Mixer	22.49	23.99	82.33	13.87	9.17	30.37
	Q2P	GRURec	11.42	9.92	34.33	10.94	15.58	16.44
		SRGNN	9.17	8.90	26.28	11.01	14.84	14.04
		Attn-Mixer	19.44	23.84	26.72	11.05	15.12	19.23
	NQE	-	9.71	11.05	73.10	11.76	8.60	22.84
	SQE-Transformer	-	23.81	25.07	77.64	18.97	14.57	32.01
	SQE-LSTM	-	23.05	18.56	81.22	16.77	13.68	30.66
	LSGT (Ours)		**24.15***	**28.69***	**83.04***	**19.21***	**15.62***	**34.14* (+2.13)**
Dressipi	FuzzQE	GRURec	20.73	20.97	50.50	97.37	92.69	56.45
		SRGNN	23.50	23.68	50.47	97.36	92.89	57.58
		Attn-Mixer	22.70	21.75	51.81	97.20	93.69	57.43
	Q2P	GRURec	20.75	25.64	24.75	97.97	63.86	46.59
		SRGNN	20.04	24.35	26.11	97.70	64.04	46.45
		Attn-Mixer	**26.74**	**37.09**	49.58	**97.98**	95.22	61.32
	NQE	-	8.58	10.60	14.49	97.40	94.56	45.13
	SQE-Transformer	-	21.15	25.08	63.23	97.59	95.41	60.49
	SQE-LSTM	-	21.03	24.76	63.14	97.73	94.50	60.23
	LSGT (Ours)		25.58	30.66	**65.93***	97.74	**96.30***	**63.24* (+1.92)**

- Q2P [6] uses multiple vectors to encode the queries;
- FuzzQE [10] use fuzzy logic to represent logical operators.

Meanwhile, the previous query encoder cannot be directly used for encoding session history as hyper-relations, so we incorporate them with session encoders. For the session encoders, we leverage the following session encoders:

- Sequence-based encoder GRURec [31];
- GNN-based session encoder SR-GNN [38];
- Attention-based session encoder Attention-Mixer [44].

5.2 Evaluation

To precisely describe the metrics, we use the q to represent a testing query and $\mathcal{G}_{val}$, $\mathcal{G}_{test}$ to represent the validation and the testing knowledge graph. Here we use $[q]_{val}$ and $[q]_{test}$ to represent the answers of query q on the validation graph $\mathcal{G}_{val}$ and testing graph $\mathcal{G}_{test}$ respectively. Equation 5 describes how to compute the `Inference` metrics. When the evaluation metric is mean reciprocal ranking (MRR), then the $m(r)$ is defined as $m(r) = \frac{1}{r}$.

$$\texttt{Inference}(q) = \frac{\sum_{v \in [q]_{test}/[q]_{val}} m(\texttt{rank}(v))}{|[q]_{test}/[q]_{val}|}. \quad (5)$$

5.3 Experiment Details

We maintain a consistent hidden size of 384 for all models. This hidden size also corresponds to the size of session representation from session encoders in the baselines, as well as the query embedding size for the entire logical session query. We use the AdamW to train the models with a batch size of 512. The models are optimized with a learning rate of 0.001, except for those with transformer structures, namely NQE, SQE-Transformer, and LSGT. These models are trained with a learning rate of 0.0001 with a warm-up of 10000 steps. The SQE and LSGT models employ two layers of encoders. All models can be trained on GPU with 24GB memory.

Table 4: The ablation study on the logical structure and item orders in each session.

Dataset	Encoder	Average	1p	2p	2ia	2is	3i	pi	ip	2u	up	2ina	2ins	3in	inp	pin
Amazon	LSGT	31.99	17.73	9.10	56.73	84.26	69.39	19.39	19.47	15.40	7.86	20.98	22.00	60.70	35.95	8.84
	w/o Logic Structure	15.98	5.41	2.31	30.31	50.21	45.21	3.75	5.49	4.88	2.56	16.32	15.77	38.19	2.13	1.11
	w/o Session Order	8.45	6.29	2.59	17.22	13.85	19.34	14.07	3.23	3.49	1.73	5.50	4.75	17.54	4.92	3.73
Diginetica	LSGT	40.59	32.00	15.27	83.34	90.61	86.05	15.62	33.80	26.34	14.45	24.15	28.69	83.04	19.21	15.62
	w/o Logic Structure	27.17	18.61	3.84	68.40	62.80	64.87	10.13	20.22	16.08	8.49	17.38	14.17	60.37	9.21	5.74
	w/o Session Order	17.07	5.08	9.71	45.49	34.42	43.23	9.69	21.71	3.66	7.92	4.39	2.56	35.80	9.98	5.36
Dressipi	LSGT	70.22	31.12	96.16	64.26	76.85	78.66	98.02	96.98	28.83	96.04	25.58	30.66	65.93	97.74	96.30
	w/o Logic Structure	25.13	14.87	2.45	42.03	59.63	67.62	9.27	17.71	18.05	7.64	19.62	24.67	59.01	1.95	7.29
	w/o Session Order	39.78	9.21	42.80	21.57	19.57	23.28	88.31	61.47	6.31	68.27	7.41	6.87	15.96	96.53	89.35

Table 5: The out-of-distribution query types evaluation. We further evaluate four types of queries with types that are unseen during the training process.

Dataset	Query Encoder	3iA	3ip	3inA	3inp	Ave.
Amazon	FuzzQE + Attn-Mixer	66.72	29.67	54.33	48.76	49.87
	Q2P + Attn-Mixer	33.51	11.42	51.47	41.46	34.47
	NQE	61.72	1.98	46.47	34,04	36.72
	SQE + Transformers	66.03	28.41	55.61	51.28	50.33
	LSGT (Ours)	**68.44**	**34.22**	**58.50**	**51.49**	**53.16**
Diginetica	FuzzQE + Attn-Mixer	88.30	32.88	82.75	34.50	59.61
	Q2P + Attn-Mixer	40.28	**43.93**	54.31	**48.20**	46.68
	NQE	86.25	20.79	64.74	20.93	48.18
	SQE + Transformers	88.05	31.33	81.77	35.83	59.25
	LSGT (Ours)	**91.71**	35.24	**83.30**	41.05	**62.83**
Dressipi	FuzzQE + Attn-Mixer	65.43	95.64	53.36	97.75	78.05
	Q2P + Attn-Mixer	60.64	96.78	52.22	97.28	76.73
	NQE	31.96	96.18	9.89	97.80	58.96
	SQE + Transformers	72.61	97.12	55.20	98.14	80.77
	LSGT (Ours)	**74.34**	**97.30**	**58.30**	**98.23**	**82.04**

5.4 Experiment Results

Table 2 compares the performance of different models with various backbones and configurations. Based on the experimental results, we can draw the following conclusions.

We found that the proposed LSGT method not only outperforms all other models but also represents the current state-of-the-art for the task. In comparison to models that solely rely on session encoders followed by query encoders, LSGT possesses the ability to leverage item information across different sessions, which proves to be critical for achieving superior performance. Additionally, LSGT demonstrates better capability in encoding graph structural inductive bias due to its operation-wise permutation invariance property, resulting in improved performance compared to other transformer-based models like SQE.

We compare the baseline of SQE [7] in Tables 2 and 3, which uses a simple prefix linearization strategy [20] to represent logical queries. However, because SQE is doing sequence modeling instead of graph modeling, it is not permutation invariant to logical operations, and thus cannot perform well in the query types that are sensitive to logical graph structure, like negation queries (Table 3) and out-of-distribution queries (Table 5).

Furthermore, LSGT exhibits greater effectiveness in handling queries involving negations when compared to the baseline models. It achieves more significant improvements on negation queries than on EPFO queries, surpassing the performance of the best baseline.

For `2is` and `ip` query types, the any-to-any attention mechanism may not be necessary as the session embedding adequately reflects the "concentrated" intentions. However, for multi-hop questions involving negations and disjunctions, the any-to-any mechanisms demonstrate advantages. In the Dressipi dataset, query types asking for attribute values, such as `ip`, `pi`, and others, consistently yield high performances. Compared to other datasets, which cover broad domains, the Dressipi dataset focuses specifically on dressing, consequently leading to a low diversity of attribute values. All the reasoning model performances reflect this property, thereby strengthening the validity of the query sampling and dataset construction.

Moreover, our observations indicate that neural models can generate more accurate results when presented with additional information or constraints. This highlights the significance of effectively modeling complex user intentions and underscores the potential for enhancing service quality in real-world usage scenarios.

5.5 Compositional Generalization

We conducted additional experiments on compositional generalization, and the results are presented in Table 5. In this particular setting, we evaluated the performance of our model on query types that were not encountered during the training process. These query types, namely `3iA`, `3ip`, `3inA`, and `3inp` (as illustrated in Fig. 5), were selected due to their complexity, involving three anchors, encompassing both EPFO queries and queries with negations, and incorporating 1-hop and 2-hop relational projections in the reasoning process. These query types were not included in the training data and were evaluated in a zero-shot manner.

By comparing the performance of our proposed method with the baselines, we observed that our approach demonstrated stronger compositional generalization on these previously unseen query types. Across the three datasets, our method improves in Mean Reciprocal Rank (MRR) ranging from 1.28 to 3.22.

5.6 Ablation Study

The results of the ablation study are presented in Table 4. In the first ablation study, we removed the tokens that represent the logical structures, and in the second ablation study, we eliminated the order information within the hypergraph by excluding the positional encoding features of item tokens in each session. When we removed the logical structure information, a significant drop in the model's performance was observed, particularly for queries involving negations and multi-hop reasoning, such as `ip`, `pi`, `inp`, and `pin`. Without the logical structure, the model was restricted to utilizing co-occurrence information, such as "bag of sessions" and "bag of items," for ranking candidate answers. While this information may be useful for simple structured queries, its effectiveness diminished for complex structured queries. Likewise, when we removed the order information within each session, a notable decrease in the overall performance was observed. This highlights two important findings: First, the item orders within each session play a crucial role in this task. Second, the LSGT model effectively utilizes the order information for this specific task.

6 CONCLUSION

In this paper, we presented a framework that models user intent as a complex logical query over a hyper-relational graph that describes sessions, products, and their attributes. Our framework formulates the session understanding problem as a logical session complex query answering (LS-CQA) on this graph and trains complex query-answering models to make recommendations based on the logical queries. We also introduced a novel method of logical session graph transformer (LSGT) and demonstrated its expressiveness and operator-wise permutation invariance. Our evaluation of fourteen intersection logical reasoning tasks showed that our proposed framework achieves better results on unseen queries and queries involving negations. Overall, our framework provides a flexible and effective approach for modeling user intent and making recommendations in e-commerce scenarios. Future work could extend our approach to other domains and incorporate additional sources of information to improve recommendation accuracy.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers and the area chair for their constructive comments. The authors of this paper were supported by the NSFC Fund (U20B2053) from the NSFC of China, the RIF (R6020-19 and R6021-20), and the GRF (16211520 and 16205322) from RGC of Hong Kong. We also thank the support from the UGC Research Matching Grants (RMGS20EG01-D, RMGS20CR11, RMGS20CR12, RMGS20EG19, RMGS20EG21, RMGS23CR05, RMGS23EG08).

REFERENCES

[1] Dimitrios Alivanistos, Max Berrendorf, Michael Cochez, and Mikhail Galkin. 2022. Query Embedding on Hyper-Relational Knowledge Graphs. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. https://openreview.net/forum?id=4rLw09TgRw9

[2] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex Query Answering with Neural Link Predictors. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=Mos9F9kDwkz

[3] Jiaxin Bai, Xin Liu, Weiqi Wang, Chen Luo, and Yangqiu Song. 2023. Complex Query Answering on Eventuality Knowledge Graph with Implicit Logical Constraints. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/6174c67b136621f3f2e4a6b1d3286f6b-Abstract-Conference.html

[4] Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, and Yangqiu Song. 2023. Understanding Inter-Session Intentions via Complex Logical Reasoning. *CoRR* abs/2312.13866 (2023). https://doi.org/10.48550/ARXIV.2312.13866 arXiv:2312.13866

[5] Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, Bing Yin, and Yangqiu Song. 2023. Knowledge Graph Reasoning over Entities and Numerical Values. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 57–68. https://doi.org/10.1145/3580305.3599399

[6] Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. 2022. Query2Particles: Knowledge Graph Reasoning with Particle Embeddings. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, Seattle, United States, 2703–2714. https://doi.org/10.18653/v1/2022.findings-naacl.207

[7] Jiaxin Bai, Tianshi Zheng, and Yangqiu Song. 2023. Sequential Query Encoding for Complex Query Answering on Knowledge Graphs. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=ERqGqZzSu5

[8] Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. 2022. Answering Complex Logical Queries on Knowledge Graphs via Query Computation Tree Optimization. *CoRR* abs/2212.09567 (2022). https://doi.org/10.48550/arXiv.2212.09567 arXiv:2212.09567

[9] Pablo Barceló, Mikhail Galkin, Christopher Morris, and Miguel A. Romero Orth. 2022. Weisfeiler and Leman Go Relational. In *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event (Proceedings of Machine Learning Research, Vol. 198)*, Bastian Rieck and Razvan Pascanu (Eds.). PMLR, 46. https://proceedings.mlr.press/v198/barcelo22a.html

[10] Xuelu Chen, Ziniu Hu, and Yizhou Sun. 2022. Fuzzy Logic Based Logical Query Answering on Knowledge Graphs. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 3939–3948. https://ojs.aaai.org/index.php/AAAI/article/view/20310

[11] Jiayan Guo, Peiyan Zhang, Chaozhuo Li, Xing Xie, Yan Zhang, and Sunghun Kim. 2022. Evolutionary Preference Learning via Graph Nested GRU ODE for Session-based Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 624–634. https://doi.org/10.1145/3511808.3557314

[12] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 2030–2041. https://proceedings.neurips.cc/paper/2018/hash/ef50c335cca9f340bde656363ebd02fd-Abstract.html

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[14] Qi Hu, Weifeng Jiang, Haoran Li, Zihao Wang, Jiaxin Bai, Qianren Mao, Yangqiu Song, Lixin Fan, and Jianxin Li. 2024. FedCQA: Answering Complex Queries on Multi-Source Knowledge Graphs via Federated Learning. *CoRR* abs/2402.14609 (2024). https://doi.org/10.48550/ARXIV.2402.14609 arXiv:2402.14609

[15] Qi Hu, Haoran Li, Jiaxin Bai, and Yangqiu Song. 2023. Privacy-Preserving Neural Graph Databases. *CoRR* abs/2312.15591 (2023). https://doi.org/10.48550/ARXIV.2312.15591 arXiv:2312.15591

[16] Xingyue Huang, Miguel A. Romero Orth, İsmail İlkan Ceylan, and Pablo Barceló. 2023. A Theory of Link Prediction via Relational Weisfeiler-Leman. *CoRR* abs/2302.02209 (2023). https://doi.org/10.48550/ARXIV.2302.02209 arXiv:2302.02209

[17] Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. 2022. Going Deeper into Permutation-Sensitive Graph Neural Networks. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 9377–9409. https://proceedings.mlr.press/v162/huang22l.html

[18] Wei Jin, Haitao Mao, Zheng Li, Haoming Jiang, Chen Luo, Hongzhi Wen, Haoyu Han, Hanqing Lu, Zhengyang Wang, Ruirui Li, et al. 2023. Amazon-M2: A Multilingual Multi-locale Shopping Session Dataset for Recommendation and Text Generation. *arXiv preprint arXiv:2307.09688* (2023).

[19] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure Transformers are Powerful Graph Learners. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/5d84236751fe6d25dc06db055a3180b0-Abstract-Conference.html

[20] Guillaume Lample and François Charton. 2020. Deep Learning For Symbolic Mathematics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=S1eZYeHFDS

[21] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. AdsGNN: Behavior-Graph Augmented Relevance Modeling in Sponsored Search. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 223–232. https://doi.org/10.1145/3404835.3462926

[22] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[23] Lihui Liu, Zihao Wang, Jiaxin Bai, Yangqiu Song, and Hanghang Tong. 2024. New Frontiers of Knowledge Graph Reasoning: Recent Advances and Future Trends. In *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Roy Ka-Wei Lee, Ravi Kumar, and Hady W. Lauw (Eds.). ACM, 1294–1297. https://doi.org/10.1145/3589335.3641254

[24] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 1831–1839. https://doi.org/10.1145/3219819.3219950

[25] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. 2022. Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1120–1130. https://doi.org/10.1145/3534678.3539472

[26] Haoran Luo, Haihong E, Yuhao Yang, Gengxian Zhou, Yikai Guo, Tianyu Yao, Zichen Tang, Xueyuan Lin, and Kaiyang Wan. 2023. NQE: N-ary Query Embedding for Complex Query Answering over Hyper-Relational Knowledge Graphs. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 4543–4551. https://ojs.aaai.org/index.php/AAAI/article/view/25576

[27] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. 2019. Invariant and Equivariant Graph Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=Syx72jC9tm

[28] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star Graph Neural Networks for Session-based Recommendation. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 1195–1204. https://doi.org/10.1145/3340531.3412014

[29] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=BJgr4kSFDS

[30] Hongyu Ren and Jure Leskovec. 2020. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/e43739bba7cdb577e9e3e4e42447f5a5-Abstract.html

[31] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar Shalom, Haggai Roitman, Bracha Shapira, and Lior Rokach (Eds.). ACM, 17–22. https://doi.org/10.1145/2988450.2988452

[32] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (Eds.). ACM, 565–573. https://doi.org/10.1145/3159652.3159656

[33] Zhenwei Tang, Griffin Floto, Armin Toroghi, Shichao Pei, Xiangliang Zhang, and Scott Sanner. 2023. LogicRec: Recommendation with Users' Logical Requirements. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 2129–2133. https://doi.org/10.1145/3539618.3592012

[34] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BylA_C4tPr

[35] Zihao Wang, Weizhi Fei, Hang Yin, Yangqiu Song, Ginny Y. Wong, and Simon See. 2023. Wasserstein-Fisher-Rao Embedding: Logical Query Embeddings with Local Comparison and Global Transport. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 13679–13696. https://doi.org/10.18653/V1/2023.FINDINGS-ACL.864

[36] Zihao Wang, Yangqiu Song, Ginny Y. Wong, and Simon See. 2023. Logical Message Passing Networks with One-hop Inference on Atomic Formulas. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=SoyOsp7i_l

[37] Zihao Wang, Hang Yin, and Yangqiu Song. 2021. Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, Joaquin Vanschoren and Sai-Kit Yeung (Eds.). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/7eabe3a1649ffa2b3ff8c02ebfd5659f-Abstract-round2.html

[38] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[39] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 4503–4511. https://ojs.aaai.org/index.php/AAAI/article/view/16578

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[41] Zezhong Xu, Wen Zhang, Peng Ye, Hui Chen, and Huajun Chen. 2022. Neural-Symbolic Entangled Framework for Complex Query Answering. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 1806–1819. https://proceedings.neurips.cc/paper_files/paper/2022/file/0bcfb525c8f8f07ae10a93d0b2a40e00-Paper-Conference.pdf

[42] Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. 2022. GammaE: Gamma Embeddings for Logical Queries on Knowledge Graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 745–760. https://aclanthology.org/2022.emnlp-main.47

[43] Hang Yin, Zihao Wang, Weizhi Fei, and Yangqiu Song. 2023. EFO_k-CQA: Towards Knowledge Graph Complex Query Answering beyond Set Operation. *CoRR* abs/2307.13701 (2023). https://doi.org/10.48550/ARXIV.2307.13701 arXiv:2307.13701

[44] Peiyan Zhang, Jiayan Guo, Chaozhuo Li, Yueqi Xie, Jaeboum Kim, Yan Zhang, Xing Xie, Haohan Wang, and Sunghun Kim. 2023. Efficiently Leveraging Multi-level User Intent for Session-based Recommendation via Atten-Mixer Network. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, Tat-Seng Chua, Hady W. Lauw, Luo Si, Evimaria Terzi, and Panayiotis Tsaparas (Eds.). ACM, 168–176. https://doi.org/10.1145/3539597.3570445

[45] Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. 2022. Neural-Symbolic Models for Logical Queries on Knowledge Graphs. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 27454–27478. https://proceedings.mlr.press/v162/zhu22c.html

Table 6: The query types and their explanations.

Query Types	Explanations
1p	Predict the product that is desired by a given session.
2p	Predict the attribute value of the product that is desired by a given session.
2iA	Predict the product that is desired by a given session with a certain attribute value.
2iS	Predict the product that is desired by both given sessions.
3i	Predict the product that is desired by both given sessions with a certain attribute value.
ip	Predict the attribute value of the product that is desired by both of the given sessions.
pi	Predict the attribute value of the product that is desired by a given session, this attribute value is possessed by another given item.
2u	Predict the product that is desired by either one of the sessions.
up	Predict the attribute value of the product that is desired by either of the sessions.
2inA	Predict the product that is desired by a given session, but does not have a certain attribute.
2inS	Predict the product that is desired by a given session, but is not wanted by another session.
3in	Predict the product that is desired by a given session with a certain attribute, but is not wanted by another session.
inp	Predict the attribute value of the product that is desired by a given session, but is not wanted by another session.
pin	Predict the attribute value of the product that is desired by a given session, but is not possessed by another given item.

A PROOFS

We give the proofs of **Theorem** 1, 2, and **Theorem** 3. Before proving these two theorems, we define the proxy graph of the graph we used in this paper that involves N-ary facts.

Defintion (Proxy Graph). For each computational graph utilized by the query encoder, we can uniquely identify the corresponding proxy graph. This graph comprises binary edges without hyperedges and consists of vertices representing items, sessions, and operators. The edges in the proxy graph can be categorized into three types: session edges, which connect item vertices to session vertices and utilize their position, such as $1, 2, \ldots, k$ as edge types; relational projection edges, which connect two vertices and employ the relation type as the edge type; and logical edges, which utilize the corresponding logical operation type as the edge type. It is important to note that the proxy graph is distinct for different computational graphs with N-ary facts.

Defintion (Non-relational Argumented Proxy Graph). For each proxy graph, we create another graph called a Non-relational Argumented Proxy Graph. This graph includes all vertices in the original proxy. Meanwhile, the argument graph an additional node for each edge in the original graph, and it takes relation type as a node feature.

Lemma 1. *Encoding the complex session query by following the computational graph using a session encoder followed by query encoding is equivalent to performing message passing on the corresponding proxy graph.*

Proof. To prove this, we must analyze each operation in the original N-ary computational graph. For the session encoder part, the session representation is computed from the items it contains, which is equivalent to a message passing on the proxy graph with a unique aggregation function, namely the session encoder. For the intersection and union operations, the computational graph utilizes various specially designed logical operations to encode them, and they can be considered as messages passing over the proxy graph. Similarly, for the relational projection, the tail node aggregates information from the head node and relation type, which is also a message-passing process on the proxy graph. □

Lemma 2. *The encoding process of LSGT is equivalent to using TokenGT to encode the proxy graph.*

Proof. The encoding process of LSGT consists of three parts. First, the node tokens are used to identify and represent the items, sessions, and operators. Secondly, the logical structure tokens are employed to represent the logical connections between items and sessions. Finally, LSGT utilizes positional embedding as the token feature to describe the positional information of an item in a session. This process is equivalent to building an edge between the item and session and assigning its edge feature as the corresponding position embedding, which is done in the proxy graph. Therefore, encoding logical session graphs using LSGT is equivalent to using TokenGT on the proxy graph. □

Lemma 3. *Suppose the G_1 and G_2 are two proxy graphs, and G_1' and G_2' are two non-relational argument proxy graphs converted from G_1 and G_2 respectively. Then $G_1 = G_2 \longleftrightarrow G_1' = G_2'$.*

Proof. The direction $G_1 = G_2 \rightarrow G_1' = G_2'$ is trivial because according to the definition, the conversion process is deterministic. We focus on the reverse side: $G_1 = G_2 \leftarrow G_1' = G_2'$. We try to prove it by contradiction. Suppose $G_1 \neq G_2$ but $G_1' = G_2'$. Without losing generality, we can suppose there is an edge $(u, v, r) \in G_1$ but (u, v, r) is not in G_2 where u, v are vertices and r is the relation. Because of this, suppose w is a node with feature r connected that is linked to both u, v in the argument graph for both $G_1' = G_2'$. Namely both (u, w) and (w, v) are in $G_1' = G_2'$. Because the (u, v, r) is not in G_2, (w, v) is not constructed by the edge (u, v, r), thus it must be constructed by another edge (u', v, r). This suggests w is connected with at least three vertices u, v and u'. This is contradictory to the definition of the non-relational argument proxy graph. □

Proof of Theorem 1.

Proof. Based on Lemma 1, as the baseline models perform message passing on the proxy graph, their expressiveness is as powerful as the 1-WL graph isomorphism test [40]. Additionally, according to Lemma 1, the encoding process of LSGT on the session query graph is equivalent to using order-2 TokenGT on the proxy graph. Order-2 TokenGT can approximate the 2-IGN network [19], and the 2-IGN network is at least as expressive as the 2-WL graph isomorphism test [27]. Since the 2-WL test is equivalent to the 1-WL test, we can conclude that LSGT has at least the same expressiveness as the baseline models.

□

Proof of Theorem 2.

Proof. To prove the expressiveness of LSGT on the multirelational proxy graph is at least 1-RWL, we need to show that for two non-isomorphic multi-relational graphs G and H, if they can be distinguished by 1-RWL or equivalently CompGCN, then it also can be distinguished by LSGT.

According to the CompGCN definition and the definition of the Non-Relational Argument Proxy Graph of G' and H' which

Table 7: The query structures are used for training and evaluation. For brevity, the p, i, n, and u represent the projection, intersection, negation, and union operations. The query types are trained and evaluated under supervised settings.

	Train Queries		Validation Queries	Test Queries
Dataset	Item-Attribute	Others	All Types	All Types
Amazon	2,535,506	720,816	36,041	36,041
Diginetica	249,562	60,235	3,012	3,012
Dressipi	414,083	668,650	33,433	33,433

are constructed from G and H respectively, CompGCN computed on G and H can be regarded as a message passing on the non-relational message passing on G' and H'. We will give a more detailed justification as follows.

The formula of CompGCN [34] is as follows:

$$h_v^{k+1} = f\Big(\sum_{(u,r)\in N(v)} W^k_{\lambda(r)}\phi(h_u^k, h_r^k)\Big), \tag{6}$$

here h_u^k, h_r^k denotes features for node u and relation r at the k-th layer respectively, h_v^{k+1} denotes the updated representation of node v, and $W_{\lambda(r)} \in \mathbb{R}^{d\times d}$ is a relation-type specific parameter. f is an activation function (such as the ReLU). In CompGCN, we use direction-specific weights, i.e., $\lambda(r) = dir(r)$.

On the other hand, the general form of non-relational message passing is expressed as follows:

$$h_v^{k+1} = \gamma^{k+1}\Big(h_v^k, \bigoplus_{u\in N(v)} \psi^{k+1}(h_v^k, h_u^k)\Big), \tag{7}$$

Where denotes in the k-th layer, the $\bigoplus$ is a differentiable, permutation invariant function, e.g., sum, mean or max, and γ and ψ denote differentiable functions such as MLPs (Multi Layer Perceptrons).

With these two formulas we are going to prove the CompGCN computed on relational graph G with Equation (6) is equivalent to a message passing on the non-relational argumented graph G' with Equation (7). To prove this, we can use a constructive method to show that each step of CompGCN with equation (6) on G is identical to two steps message passing on G' with equation (7). Suppose for each edge (u, v) in graph G, there is a in-between node w with node label of relation type r in the augmented graph G'. The computation of equation 6 can be separated into the following two steps. First for each neighbor u of v, we use $g^k_{u,r}$ to represent the result of the result of a composition of the following two functions:

$$g^k_{u,r} = \phi(h_u^k, h_r^k) \tag{8}$$

$$h_v^{k+1} = f\Big(\sum_{(u,r)\in N(v)} W^k_{\lambda(r)} g^k_{u,r}\Big) \tag{9}$$

Then we are going to prove that these two steps each equivalent to one message passing step on the argumented graph G'. First we show the equation 8 is equivalent to a message passing process on G' to the augmented node w that contains the relation feature $h_w = h_r$ between u and v. We can let the $\gamma^{k+1}(x, y) = y$ and $\psi^{k+1}(x, y) = \phi(y, x)$ in the general message passing in equation 7, and we got the following equation

$$h_w^{k+1} = \gamma^{k+1}\Big(h_w^k, \bigoplus_{u\in N(w)} \psi^{k+1}(h_w^k, h_u^k)\Big) \tag{10}$$

$$= \bigoplus_{u\in N(w)} \psi^{k+1}(h_w^k, h_u^k) \tag{11}$$

$$= \psi^{k+1}(h_w^k, h_u^k) = \phi(h_u^k, h_w^k) \tag{12}$$

$$= \phi(h_u^k, h_r^k) = g^k_{u,r} \tag{13}$$

Then, we show that in the second step of CompGCN in equation 9 is equivalent to another step of message passing in G' from each of its neighbors w to v. According to the Equation 7, we can write

$$h_v^{k+1} = \gamma^{k+1}\Big(h_v^k, \bigoplus_{u\in N(w)} \psi^{k+1}(h_w^k, h_u^k)\Big) \tag{14}$$

$$= f\Big(\bigoplus_{u\in N(w)} \psi^{k+1}(h_w^k, h_u^k)\Big) \tag{15}$$

$$= f\Big(\sum_{u\in N(w)} \psi^{k+1}(h_w^k, h_u^k)\Big) \tag{16}$$

$$= f\Big(\sum_{u\in N(w)} W^k_{\lambda(r)} h_w^k\Big) \tag{17}$$

$$= f\Big(\sum_{u\in N(w)} W^k_{\lambda(r)} g^k_{u,r}\Big). \tag{18}$$

As a result, the CompGCN over G is able to write as two steps of message passing over the augmented graph G'.

Thus, if G and H can be distinguished by CompGCN, then G' and H' can be distinguished by a certain non-relational message-passing algorithm. Thus G' and H' can be distinguished by the 1-WL test. As Shown in previous proof LSGT is at least as powerful as the 2-WL test, and 1-WL and 2-WL tests are equivalent. We can conclude that LSGT is able to distinguish G' and H'. According to Lemma 3, if LSGT is able to distinguish G' and H' then it is able to distinguish G and H.

□

Proof of Theorem 3.

Proof. Operation-wise permutation invariance mainly focuses on the `Intersection` and `Union` operations. Suppose the input vertices for such an operator are $\{p_1, p_2, \ldots, p_n\}$. If an arbitrary permutation over these vertices is denoted as $\{p'_1, p'_2, \ldots, p'_n\}$, a global permutation of token identifiers can be constructed, where vertices p_i are mapped to p'_i and the rest are mapped to themselves. As per Lemma 2, LSGT can approximate 2-IGN [27], which is permutation invariant. Therefore, LSGT can approximate a query encoder that achieves operation-wise permutation invariance. □

B THE CONCRETE MEANINGS OF VARIOUS QUERY TYPES

In this session, we describe concrete meanings of the query types shown in Figure 5. The meanings are listed in the Table 6.