Learning Algebraic Recombination for Compositional Generalization

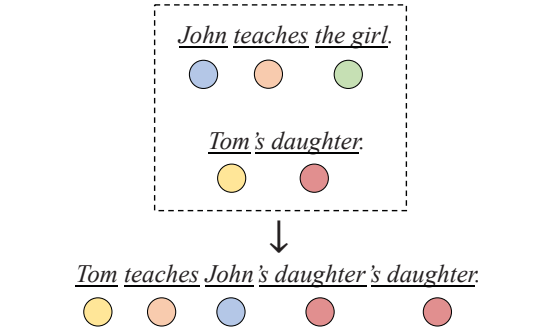
Chenyao Liu[1*] **Shengnan An**[2*] **Zeqi Lin**[3†] **Qian Liu**[4*] **Bei Chen**[3]
Jian-Guang LOU[3] **Lijie Wen**[1†] **Nanning Zheng**[2] **Dongmei Zhang**[3]
[1] School of Software, Tsinghua University [2] Xi'an Jiaotong University
[3] Microsoft Research Asia [4] Beihang University
{liucy19@mails, wenlj@}.tsinghua.edu.cn
{an1006634493@stu, nnzheng@mail}.xjtu.edu.cn
{Zeqi.Lin, beichen, jlou, dongmeiz}@microsoft.com
qian.liu@buaa.edu.cn

Abstract

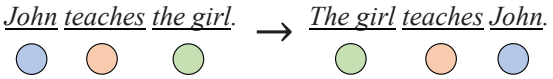
Neural sequence models exhibit limited *compositional generalization* ability in semantic parsing tasks. Compositional generalization requires *algebraic recombination*, i.e., dynamically recombining structured expressions in a recursive manner. However, most previous studies mainly concentrate on recombining lexical units, which is an important but not sufficient part of algebraic recombination. In this paper, we propose LEAR, an end-to-end neural model to learn algebraic recombination for compositional generalization. The key insight is to model the semantic parsing task as a homomorphism between a latent syntactic algebra and a semantic algebra, thus encouraging algebraic recombination. Specifically, we learn two modules jointly: a Composer for producing latent syntax, and an Interpreter for assigning semantic operations. Experiments on two realistic and comprehensive compositional generalization benchmarks demonstrate the effectiveness of our model. The source code is publicly available at https://github.com/microsoft/ContextualSP.

1 Introduction

The principle of compositionality is an essential property of language: the meaning of a complex expression is fully determined by its structure and the meanings of its constituents (Pelletier, 2003; Szabó, 2004). Based on this principle, human intelligence exhibits *compositional generalization* — the algebraic capability to understand and produce a potentially infinite number of novel expressions by dynamically recombining known components (Chomsky, 1957; Fodor and Pylyshyn, 1988; Fodor and Lepore, 2002). For example, people who know the meaning of "*John teaches the girl*" and "*Tom's daughter*" must know the meaning of "*Tom teaches John's daughter's daughter*" (Figure 1a), even though they have never seen such complex sentences before.



(a) Compositional generalization requires **algebraic recombination**, i.e., dynamically recombining structured expressions in a recursive manner.



(b) Most previous studies mainly concentrate on recombining lexical units, which is an important but not sufficient part of algebraic recombination.

Figure 1: Compositional generalization.

In recent years, there has been accumulating evidence that end-to-end deep learning models lack such ability in semantic parsing (i.e., translating natural language expressions to machine interpretable semantic meanings) tasks (Lake and Baroni, 2018; Keysers et al., 2019; Kim and Linzen, 2020; Tsarkov et al., 2020).

Compositional generalization requires **algebraic recombination**, i.e., dynamically recombining structured expressions in a recursive manner. In the example in Figure 1a, understanding "*John's daughter's daughter*" is a prerequisite for understanding "*Tom teaches John's daughter's daughter*", while "*John's daughter's daughter*" is also a novel compound expression, which requires recom-

[*] Work done during an internship at Microsoft Research. The first two authors contributed equally to this paper.
[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1129–1144

bining "*John*" and "*Tom's daughter*" recursively.

Most previous studies on compositional generalization mainly concentrate on recombining lexical units (e.g., words and phrases) (Lake, 2019; Li et al., 2019; Andreas, 2019; Gordon et al., 2020; Akyürek et al., 2020; Guo et al., 2020a; Russin et al., 2019), of which an example is shown in Figure 1b. This is a necessary part of algebraic recombination, but it is not sufficient for compositional generalization. There have been some studies on algebraic recombination (Liu et al., 2020; Chen et al., 2020). However, they are highly specific to a relative simple domain SCAN (Lake and Baroni, 2018) and can hardly generalize to more complex domains.

In this paper, our main point to achieve algebraic recombination is to **model semantic parsing as a homomorphism between a latent syntactic algebra and a semantic algebra** (Montague, 1970; Marcus, 2019). Based on this formalism, we focus on learning the high-level mapping between latent syntactic operations and semantic operations, rather than the direct mapping between expression instances and semantic meanings.

Motivated by this idea, we propose LEAR (**Le**arning **A**lgebraic **R**ecombination), an end-to-end neural architecture for compositional generalization. LEAR consists of two modules: a *Composer* and an *Interpreter*. Composer learns to model the latent syntactic algebra, thus it can produce the latent syntactic structure of each expression in a bottom-up manner; Interpreter learns to assign semantic operations to syntactic operations, thus we can transform a syntactic tree to the final composed semantic meaning.

Experiments on two realistic and comprehensive compositional generalization benchmarks (*CFQ* (Keysers et al., 2019) and *COGS* (Kim and Linzen, 2020)) demonstrate the effectiveness of our model: CFQ $67.3\% \rightarrow 90.9\%$, COGS $35.0\% \rightarrow 97.7\%$.

2 Compositionality: An Algebraic View

A semantic parsing task aims to learn a meaning-assignment function $m : L \rightarrow M$, where L is the set of (simple and complex) expressions in the language, and M is the set of available semantic meanings for the expressions in L. Many end-to-end deep learning models are built upon this simple and direct formalism, in which the principle of compositionality is not leveraged, thus exhibiting limited compositional generalization.

To address this problem, in this section we put forward the formal statement that "*compositionality requires the existence of a homomorphism between the expressions of a language and the meanings of those expressions*" (Montague, 1970).

Let us consider a language as a partial algebra $\mathbf{L} = \langle L, (f_\gamma)_{\gamma \in \Gamma} \rangle$, where Γ is the set of underlying syntactic (grammar) rules, and we use $f_\gamma : L^k \rightarrow L$ to denote the syntactic operation with a fixed arity k for each $\gamma \in \Gamma$. Note that f_γ is a partial function, which means that we allow f_γ be undefined for certain expressions. Therefore, $\mathbf{L}$ is a partial algebra, and we call it a **syntactic algebra**. In a semantic parsing task, $\mathbf{L}$ is latent, and we need to model it by learning from data.

Consider now $\mathbf{M} = \langle M, G \rangle$, where G are semantic operations upon M. $\mathbf{M}$ is also a partial algebra, and we call it a **semantic algebra**. In a semantic parsing task, we can easily define this algebra (by enumerating all available semantic primitives and semantic operations), since $\mathbf{M}$ is a machine-interpretable formal system.

The key to compositionality is that the meaning-assignment function m should be a homomorphism from $\mathbf{L}$ to $\mathbf{M}$. That is, for each k-ary syntactic operation f_γ in $\mathbf{L}$, there exists a k-ary semantic operation $g_\gamma \in G$ such that whenever $f_\gamma(e_1, ..., e_k)$ is defined,

$$m(f_\gamma(e_1, ..., e_k)) = g_\gamma(m(e_1), ..., m(e_k)). \quad (1)$$

Based on this formal statement, the task of learning the meaning-assignment function m can be transformed as two sub-tasks: (1) learning latent syntax of expressions (i.e., modeling the syntactic algebra $\mathbf{L}$); (2) learning the operation assignment function $(f_\gamma)_{\gamma \in \Gamma} \rightarrow G$.

Learning latent syntax. We need to learn a syntactic parser that can produce the syntactic structure of each given expression. To ensure compositional generalization, there must be an underlying grammar (i.e., Γ), and we hypothesize that Γ is a context-free grammar.

Learning operation assignment. In the syntax tree, for each nonterminal node with k nonterminal children, we assign a k-ary semantic operation to it. This operation assignment entirely depends on the underlying syntactic operation γ of this node.

In semantic parsing tasks, we do not have respective supervision for these two sub-tasks. Therefore, we need to jointly learning these two sub-tasks only from the end-to-end supervision $\mathcal{D} \subset L \times M$.

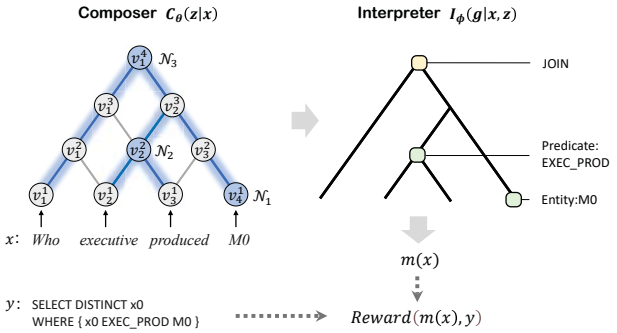


Figure 2: An overview of LEAR: (1) Composer $C_\theta(z|x)$ is a neural network based on latent Tree-LSTM, which produces the latent syntax tree z of input expression x; (2) Interpreter $I_\phi(g|x,z)$ is a neural network that assigns a semantic operation for each nonterminal node in z.

3 Model

We propose a novel end-to-end neural model LEAR (**Le**arning **A**lgebraic **R**ecombination) for compositional generalization in semantic parsing tasks. Figure 2 shows its overall architecture. LEAR consists of two parts: (1) *Composer* $C_\theta(z|x)$, which produces the latent syntax tree z of input expression x; (2) Interpreter $I_\phi(g|x,z)$, which assigns a semantic operation for each nonterminal node in z. θ and ϕ refers to learnable parameters in them respectively. We generate a semantic meaning $m(x)$ according to the predicted z and g in a symbolic manner, then check whether it is semantic equivalent to the ground truth semantic meaning y to produce rewards for optimizing θ and ϕ.

3.1 Composer

We use $x = [x_1, ..., x_T]$ to denote an input expression of length T. Composer $C_\theta(z|x)$ will produce a latent binary tree z given x.

3.1.1 Latent Tree-LSTM

We build up the latent binary tree z in a bottom-up manner based on Tree-LSTM encoder, called latent Tree-LSTM (Choi et al., 2018; Havrylov et al., 2019).

Given the input sequence x of length T, latent Tree-LSTM merges two nodes into one parent node at each merge step, constructing a binary tree after $T-1$ merge steps. The merge process is implemented by selecting the adjacent node pair which has the highest merging score.

At the t-th ($1 \le t < T$) merge step, we have:

$$\hat{i}_t = \arg\max_{1 \le i \le T-t} \text{Linear}(\text{Tree-LSTM}(\mathbf{r}_i^t, \mathbf{r}_{i+1}^t)) \quad (2)$$

Here "Tree-LSTM" is the standard child-sum tree-structured LSTM encoder (Tai et al., 2015). We use v_i^t to denote the i-th cell at layer t (the t-th merge step is determined by the t-th layer), and use $\mathbf{r}_i^t$ to denote the representation of v_i^t:

$$\mathbf{r}_i^1 = \text{Linear}(\text{Emb}(x_i)) \quad (3)$$

$$\underset{t>1}{\mathbf{r}_i^t} = \begin{cases} \mathbf{r}_i^{t-1} & i < \hat{i}_{t-1} \\ \text{Tree-LSTM}(\mathbf{r}_i^{t-1}, \mathbf{r}_{i+1}^{t-1}) & i = \hat{i}_{t-1} \\ \mathbf{r}_{i+1}^{t-1} & i > \hat{i}_{t-1} \end{cases} \quad (4)$$

Then we can obtain a unlabeled binary tree, in which $\{v_1^1, v_2^1, ..., v_T^1\}$ are leaf nodes, and $\{v_{\hat{i}_1}^2, v_{\hat{i}_2}^3 ..., v_{\hat{i}_{T-1}}^T\}$ are non-leaf nodes.

3.1.2 Abstraction by Nonterminal Symbols

As discussed in Section 2, our hypothesis is that the underlying grammar Γ is context-free. Therefore, each syntactic rule $\gamma \in \Gamma$ can be expressed in the form of:

$$A \to B, \qquad A \in \mathcal{N}, B \in (\mathcal{N} \cup \Sigma)^+$$

where $\mathcal{N}$ is a finite set of nonterminals, and Σ is a finite set of terminal symbols.

Abstraction is an essential property of context-free grammar: each compound expression e will be abstracted as a simple nonterminal symbol $\mathcal{N}(e)$, then it can be combined with other expressions to produce more complex expressions, no matter what details e originally has. This setup may benefit the generalizability, thus we want to incorporate it as an inductive bias into our model.

Concretely, we assume that there are at most N latent nonterminals in language $\mathbf{L}$ (i.e., $\mathcal{N} = \{\mathcal{N}_1, ..., \mathcal{N}_N\}$, where N is a hyper-parameter). For each node v_i^t in tree z, we perform a $(N+1)$-class classification:

$$\hat{c}_{v_i^t} = \arg\max_{0 \le c \le N} \text{Linear}(\mathbf{r}_i^t) \quad (5)$$

We assign the nonterminal $\mathcal{N}_{\hat{c}_{v_i^t}}$ to v_i^t when $\hat{c}_{v_i^t} > 0$. The collection of such nonterminal nodes

are denoted as V'_z. Then we modify Equation 4:

$$\mathbf{r}_i^t \underset{t>1}{=} \begin{cases} \mathbf{r}_i^{t-1} & i < \hat{i}_{t-1} \\ \text{Tree-LSTM}(\overline{\mathbf{r}}_i^{t-1}, \overline{\mathbf{r}}_{i+1}^{t-1}) & i = \hat{i}_{t-1} \\ \mathbf{r}_{i+1}^{t-1} & i > \hat{i}_{t-1} \end{cases} \quad (6)$$

$$\overline{\mathbf{r}}_i^t = \begin{cases} \text{Linear}(\text{Emb}(\mathcal{N}(v_i^t))) & v_i^t \in V'_z \\ \mathbf{r}_i^t & v_i^t \notin V'_z \end{cases}$$

Equation 6 means that: in nonterminal nodes, the bottom-up message passing will be reduced from $\mathbf{r}_i^t$ to a nonterminal symbol $\mathcal{N}(v_i^t)$, thus mimicking the abstraction setup in context-free grammar.

3.2 Interpreter

For each nonterminal node $v \in V'_z$, Interpreter $I_\phi(g|x, z)$ assigns a semantic operation g_v to it.

We divide nonterminal nodes into two categories: (1) *lexical nodes*, which refer to those containing no any other nonterminal node in the corresponding sub-trees; (2) *algebraic nodes*, which refer to the rest of nonterminal nodes.

Interpreting Lexical Nodes For each lexical node v, Interpreter assigns a semantic primitive (i.e., 0-ary semantic operation) to it. Take the CFQ benchmark as an example: it uses SPARQL queries to annotate semantic meanings, thus semantic primitives in CFQ are entities (e.g., *m.0gwm_wy*), predicates (e.g., *ns:film.director.film*) and attributes (e.g., *ns:people.person.gender m_05zppz*).

We use a classifier to predict the semantic primitive:

$$g_v = \underset{g \in G_{lex}}{\arg\max}\, \text{Linear}(\mathbf{h}_{v,x}) \quad (7)$$

where G_{lex} is the collection of semantic primitives in the domain, and $\mathbf{h}_{v,x}$ is the contextual representation of the span corresponding to v (implemented using Bi-LSTM). *Contextually conditioned variation* is an important phenomenon in language: the meaning of lexical units varies according to the contexts in which they appear (Allwood, 2003). For example, "*editor*" means a predicate "*film.editor.film*" in expression "*Is M0 an editor of M1?*", while it means an attribute "*film.editor*" in expression "*Is M0 an Italian editor?*". This is the reason why we use contextual representation in Equation 7.

Interpreting Algebraic Nodes For each algebraic node v, Interpreter assigns a semantic operation to it. The collection of all possible semantic operations G_{opr} also depends on the domain. Take the CFQ benchmark as an example[1], this domain has two operations (detailed in Table 1): $\wedge$ (conjunction) and JOIN.

Operation	Args[t_1, t_2]→ Result Type	Example
$\wedge(t_1, t_2)$	[P, P]→P	Who [**direct** and **act**] M0?
	[E, E]→E	Who direct [**M0** and **M1**]?
	[A, A]→A	Is M0 an [**Italian female**]?
	[A, E]→E [E, A]→E	Is [**M0** an **Italian female**]?
	[A, P]→P [P, A]→P	Is M0 M3's [**Italian editor**]?
JOIN(t_1, t_2)	[E, P]→E [P, E]→E	Is M0 an [**editor** of **M1**]?
	[A, P]→E [P, A]→E	Who [**marries** an **Italian**]?

Table 1: Semantic operations in CFQ. A/P/E represents Attribute/Predicate/Entity.

We also use a classifier to predict the semantic operation of v:

$$g_v = \underset{g \in G_{opr}}{\arg\max}\, \text{Linear}(\mathbf{r}_v) \quad (8)$$

where $\mathbf{r}_v$ is the latent Tree-LSTM representation of node v (see Equation 6).

In Equation 8, we do not use any contextual information from outside v. This setup is based on the assumption of **semantic locality**: each compound expression should mean the same thing in different contexts.

4 Training

Denote $\tau = \{z, g\}$ as the trajectory produced by our model where z and g are actions produced from Composer and Interpreter, respectively, and $R(\tau)$ as the reward of trajectory τ (elaborated in Sec. 4.1). Using policy gradient (Sutton et al., 2000) with the likelihood ratio trick, our model can be optimized by ascending the following gradient:

$$\nabla \mathcal{J}(\theta, \phi) = \mathbb{E}_{\tau \sim \pi_{\theta,\phi}}\, R(\tau) \nabla \log \pi_{\theta,\phi}(\tau)\,, \quad (9)$$

where θ and ϕ are learnable parameters in Composer and Interpreter respectively and ∇ is the abbreviation of $\nabla_{\theta,\phi}$. Furthermore, the REINFORCE algorithm (Williams, 1992) is leveraged to approximate Eq. 9 and the mean-reward baseline (Weaver and Tao, 2001) is employed to reduce variance.

[1] It is not difficult to define G_{lex} and G_{opr} for each domain, as semantic meanings are always machine-interpretable. The semantic operations of another compositional generalization benchmark, COGS, are listed in the Appendix.

4.1 Reward Design

The reward $R(\tau)$ combines two parts as:

$$R(\tau) = \alpha \cdot R_1(\tau) + (1-\alpha) \cdot R_2(\tau), \quad (10)$$

Logic-based Reward $R_1(\tau)$. We use $m(x)$ and y to denote the predicted semantic meaning and the ground truth semantic meaning respectively. Each semantic meaning can be converted to a conjunctive normal form[2]. We use $S_{m(x)}$ and S_y to denote conjunctive components in $m(x)$ and y, then define $R_1(\tau)$ based on Jaccard similarity (i.e., intersection over union):

$$R_1(\tau) = \text{Jaccard-Sim}(S_{m(x)}, S_y) \quad (11)$$

Primitive-Based Reward $R_2(\tau)$. We use $S'_{m(x)}$ and S'_y to denote semantic primitives ocurred in $m(x)$ and y. Then we define $R_2(\tau)$ as:

$$R_2(\tau) = \text{Jaccard-Sim}(S'_{m(x)}, S'_y) \quad (12)$$

4.2 Reducing Search Space

To reduce the huge search space of τ, we make two constraints as follows.

Parameter Constraint. Consider v, a tree node with $n(n > 0)$ nonterminal children. Composer will never make v a nonterminal node, if no semantic operation has n parameters.

Phrase Table Constraint. Following the strategy proposed in Guo et al. (2020b), we build a "phrase table" consisting of lexical units (i.e., words and phrases) paired with semantic primitives that frequently co-occur with them[3]. Composer will never produce a lexical node outside of this table, and Interpreter will use this table to restrict candidates in Equation 7.

4.3 Curriculum Learning

To help the model converge better, we use a simple curriculum learning (Bengio et al., 2009) strategy to train the model. Specifically, we first train the model on samples of input length less than a cut-off N_{CL}, then further train it on the full train set.

[2]For example, the semantic meaning of "*Who directed and edited* M_0 *'s prequel and* M_1*?*" can be converted to a conjunctive normal form with four components: "$x_0 \cdot$ DIRECT $\cdot$ $x_1 \cdot$ PREQUEL $\cdot$ M_0", "$x_0 \cdot$ EDIT $\cdot$ $x_1 \cdot$ PREQUEL $\cdot$ M_0", "$x_0 \cdot$ DIRECT $\cdot$ M_1", and "$x_0 \cdot$ EDIT $\cdot$ M_1".

[3]Mainly based on statistical word alignment technique in machine translation, detailed in the Appendix.

CFQ

x "Did a male film director edit and direct M0?"

y
```
SELECT count ( * ) WHERE {
    ?x0 ns:film.director.film M0 .
    ?x0 ns:film.editor.film M0 .
    ?x0 ns:people.person.gender m_05zppz }
```

COGS

x "Charlotte was given the cake on a table."

y
```
cake(x_4) ; give.recipient (x_2, Charlotte)
AND give.theme(x_2,x_4)
AND cake.nmod.on(x_4, x_7)
AND table(x_7)
```

Figure 3: Examples of CFQ and COGS.

Statistics	CFQ	COGS
Train Size	95,743	24,155
Dev Size	11,968	3,000
Test Size	11,968	21,000
Vocab Size	96	740
Avg Input Len (Train/Test)	13.5/15.1	7.5/9.8
Avg Output Len (Train/Test)	27.7/34.0	43.6/67.6
Input Pattern Coverage[a]	0.022	0.783
Output Pattern Coverage	0.045	0.782

Table 2: Dataset statistics.

[a]Input/output pattern coverage is the percentage of test x/y whose patterns occur in the train data. Output patterns are determined by anonymizing semantic primitives, and input patterns are determined by anonymizing their lexical units.

5 Experimental Setup

Benchmarks. We mainly evaluate LEAR on *CFQ* (Keysers et al., 2019) and *COGS* (Kim and Linzen, 2020), two comprehensive and realistic benchmarks for measuring compositional generalization. They use different semantic formulations: CFQ uses SPARQL queries, and COGS uses logical queries (Figure 3 shows examples of them). We list dataset statistics in Table 2. The input/output pattern coverage indicates that: CFQ mainly measures the algebraic recombination ability, while COGS measures both lexical recombination ($\sim 78\%$) and algebraic recombination ($\sim 22\%$).

In addition to these two compositional generalization benchmarks in which utterances are synthesized by formal grammars, we also evaluate LEAR on *GEO* (Zelle and Mooney, 1996), a widely used semantic parsing benchmark, to see whether LEAR can generalize to utterances written by real users. We use the variable-free FunQL (Kate et al., 2005) as the semantic formalism, and we follow the compositional train/test split (Finegan-Dollak et al., 2018) to evaluate compositional generalization.

Baselines. For CFQ, we consider 3 groups of models as our baselines: (1) sequence-to-sequence mod-

Models	MCD-MEAN	MCD1	MCD2	MCD3
LSTM+Attention (Keysers et al., 2019)	14.9±1.1	28.9±1.8	5.0±0.8	10.8±0.6
Transformer (Keysers et al., 2019)	17.9±0.9	34.9±1.1	8.2±0.3	10.6±1.1
Universal Transformer (Keysers et al., 2019)	18.9±1.4	37.4±2.2	8.1±1.6	11.3±0.3
Evolved Transformer (Furrer et al., 2020)	20.8±0.7	42.4±1.0	9.3±0.8	10.8±0.2
T5-11B (Furrer et al., 2020)	40.9±4.3	61.4±4.8	30.1±2.2	31.2±5.7
T5-11B-mod (Furrer et al., 2020)	42.1±9.1	61.6±12.4	31.3±12.8	33.3±2.3
Neural Shuffle Exchange (Furrer et al., 2020)	2.8±0.3	5.1±0.4	0.9±0.1	2.3±0.3
CGPS (Furrer et al., 2020; Li et al., 2019)	7.1±1.8	13.2±3.9	1.6±0.8	6.6±0.6
HPD (Guo et al., 2020b)	67.3±4.1	72.0±7.5	66.1±6.4	63.9±5.7
LEAR	**90.9±1.2**	**91.7±1.0**	**89.2±1.9**	**91.7±0.6**
w/o Abstraction	85.4±4.5	88.4±1.6	80.0±11	87.9±0.8
w/o Semantic locality	87.9±2.7	89.8±1.7	87.3±1.8	86.5±4.6
w/o Primitive-based reward	85.3±7.8	77.0±19	89.2±2.2	89.7±2.1
w/o Curriculum learning	71.9±15.4	59.7±23	77.2±13.5	78.8±9.6
w/o Tree-LSTM	30.4±3.2	40.1±1.9	25.6±6.1	25.4±1.8

Table 3: Accuracy on three splits (MCD1/MCD2/MCD3) of CFQ benchmark.

els based on deep encoder-decoder architecture, including LSTM+Attention (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014), Transformer (Vaswani et al., 2017), Universal Transformer (Dehghani et al., 2018) and Evolved Transformer (So et al., 2019); (2) deep models with large pretrained encoder, such as T5 (Raffel et al., 2019); (3) Models that are specially designed for compositional generalization, which include Neural Shuffle Exchange Network (Freivalds et al., 2019), CGPS (Li et al., 2019), and state-of-the-art model HPD (Guo et al., 2020b). For COGS, we quote the baseline results in the original paper (Kim and Linzen, 2020). For GEO, we take the baseline results reported by Herzig and Berant (2020), and also compare with two specially designed methods: SpanBasedSP (Herzig and Berant, 2020) and PDE (Guo et al., 2020c).

Evaluation Metric. We use accuracy as the evaluation metric, i.e., the percentage test samples of which the predicted semantic meaning $m(x)$ is semantically equivalent to the ground truth y.

Hyper-Parameters. We set $N = 3/2/3$ (the number of nonterminal symbols), and $\alpha = 0.5/1.0/0.9$ for CFQ/COGS/GEO respectively. In CFQ, the curriculum cut-off N_{CL} is set to 11, as we statistically find that this is the smallest curriculum that contains the complete vocabulary. We do not apply curriculum learning strategy to COGS and GEO, as LEAR can work well without curriculum learning in both benchmarks. Learnable parameters (θ and ϕ) are optimized with AdaDelta (Zeiler, 2012), and the setting of learning rate is discussed in Section 6.1. We take the model that performs best on the validation set for testing, and all results are obtained by averaging over 5 runs with different random seeds. See Appendix for more implementation details.

Model	Acc
Transformer (Kim and Linzen, 2020)	35 ± 6
LSTM (Bi) (Kim and Linzen, 2020)	16 ± 8
LSTM (Uni) (Kim and Linzen, 2020)	32 ± 6
LEAR	**97.7 ± 0.7**
w/o Abstraction	94.5 ± 2.8
w/o Semantic locality	94.0 ± 3.6
w/o Tree-LSTM	80.7 ± 4.3

Table 4: Accuracy on COGS benchmark.

Model	Acc
Seq2Seq (Herzig and Berant, 2020)	46.0
BERT2Seq (Herzig and Berant, 2020)	49.6
GRAMMAR (Herzig and Berant, 2020)	54.0
PDE (Guo et al., 2020c)	81.2
SpanBasedSP (Herzig and Berant, 2020)	82.2
LEAR	**84.1**

Table 5: Accuracy on GEO benchmark.

6 Results and Discussion

Table 3 shows average accuracy and 95% confidence intervals on three splits of CFQ. LEAR achieves an average accuracy of 90.9% on these three splits, outperforming all baselines by a large margin. We list some observations as follows.

Methods for lexical recombination cannot generalize to algebraic recombination. Many methods for compositional generalization have been

proved effective for lexical recombination. Neural Shuffle Exchange and CGPS are two representatives of them. However, experimental results show that they cannot generalize to CFQ, which focus on algebraic recombination.

Knowledge of semantics is important for compositional generalization. Seq2seq models show poor compositional generalization ability ($\sim 20\%$). Pre-training helps a lot ($\sim 20\% \rightarrow \sim 40\%$), but still not satisfying. HPD and LEAR incorporate knowledge of semantics (i.e., semantic operations) into the models, rather than simply model semantic meanings as sequences. This brings large profit.

Exploring latent compositional structure in a bottom-up manner is key to compositional generalization. HPD uses LSTM to encode the input expressions, while LEAR uses latent Tree-LSTM, which explicitly explores latent compositional structure of expressions. This is the key to the large accuracy profit ($67.3\% \rightarrow 90.9\%$).

Table 4 shows the results on COGS benchmark. It proves that LEAR can well generalize to domains which use different semantic formalisms, by specifying domain-specific G_{lex} (semantic primitives) and G_{opr} (semantic operations). Table 5 shows the results on GEO benchmark. It proves that LEAR can well generalize to utterances written by real users (i.e., non-synthetic utterances).

6.1 Ablation Study

Table 3 and 4 also report results of some ablation models. Our observations are as follows.

Abstraction by nonterminal symbols brings profit. We use "*w/o abstraction*" to denote the ablation model in which Equation 6 is disabled. This ablation leads to $5.5\%/3.2\%$ accuracy drop on CFQ/COGS.

Incorporating semantic locality into the model brings profit. We use "*w/o semantic locality*" to denote the ablation model in which a Bi-LSTM layer is added before the latent Tree-LSTM. This ablation leads to $3.0\%/3.7\%$ accuracy drop on CFQ/COGS.

Tree-LSTM contributes significantly to compositional generalization. In the ablation "*w/o Tree-LSTM*", we replace the Tree-LSTM encoder with a span-based encoder, in which each span is represented by concatenating its start and end LSTM representations (similar to Herzig and Berant (2020)). In Table 3 and 4, we can see that span-based encoder severely affects the performance and even much worse than the results of "*w/o abstraction*" and "*w/o semantic locality*". This ablation hints that Tree-LSTM is the main inductive bias of compositionality in our model.

Ratio	MCD-MEAN	MCD1	MCD2	MCD3
1:1:1	87.4±7.1	91.5±2.1	**89.4±2.3**	81.2±17
1:0.5:0.1	**90.9±1.2**	**91.7±1.0**	89.2±1.9	**91.7±0.6**
1:0.1:0.1	86.7±3.9	89.4±1.6	85.8±2.7	84.9±7.5

Table 6: Results of different learning rate ratios of lexical Interpreter, Composer, and algebraic Interpreter.

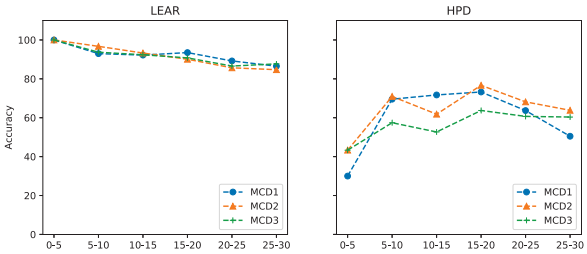
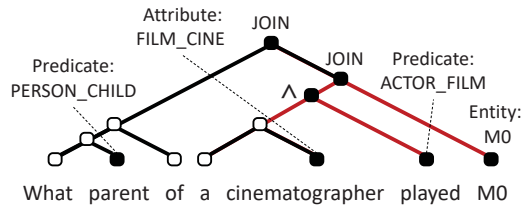


Figure 4: Performance by input length.

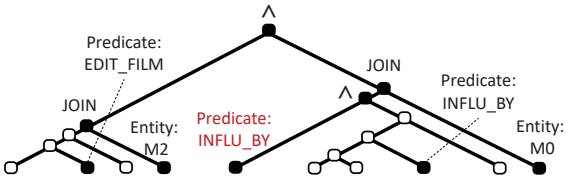
Primitive-based reward helps the model converge better. The ablation "*w/o primitive-based reward*" leads to 5.6% accuracy drop on CFQ, and the model variance has become much larger. The key insight is: primitive-based reward guides the model to interpret polysemous lexical units more effectively, thus helping the model converge better.

Curriculum learning helps the model converge better. The ablation "*w/o curriculum learning*" leads to 19% accuracy drop on CFQ, and the model variance has become much larger. This indicates the importance of curriculum learning. On COGS, LEAR performs well without curriculum learning. We speculate that there are two main reasons: (1) expressions of COGS is much shorter than CFQ; (2) the input/output pattern coverage of COGS is much higher than CFQ.

Higher component with smaller learning rate. Inspired by the differential update strategy used in Liu et al. (2020)(i.e., the higher level the component is positioned in the model, the slower the parameters in it should be updated), we set three different learning rates to three different components in LEAR (in bottom-up order): lexical Interpreter, Composer, and algebraic Interpreter. We fix the learning rate of lexical Interpreter to 1, and adjust the ratio of the learning rates of Composer and algebraic Interpreter to lexical Interpreter. Table 6 shows the results on CFQ. The hierarchical learning rate setup ($1 : 0.5 : 0.1$) achieves the best performance.



(a) Composer error. A correct syntax tree should compose "*parent of a cinematographer*" as a constituent, while the predicted syntax tree incorrectly composes "*a cinematographer played M0*".



(b) Interpreter error. In this expression, the first "*influenced*" should be assigned a semantic primitive "*influence.influence_node.influenced*", while Interpreter incorrectly assigns "*influence.influence_node.influenced_by*" (abbreviated as "*INFLU_BY*" in this figure) to it.

Figure 5: Two error cases. We use solid nodes to denote predicted nonterminal nodes. Incorrect parts are colored red.

6.2 Closer Analysis

We also conduct closer analysis to the results of LEAR as follows.

6.2.1 Performance by Input Length

Intuitively, understanding longer expressions requires stronger algebraic recombination ability than shorter examples. Therefore, we expect that our model should keep a good and stable performance with the increasing of input length.

Figure 4 shows the performance of LEAR and HPD (the state-of-the-art model on CFQ) under different input lengths. Specifically, test instances are divided into 6 groups by length: $[1, 5], [6, 10], ..., [26, 30]$), and we report accuracy on each group separately. The results indicate that **LEAR has stable high performance for different input lengths**, with only a slow decline as length increases. Even on the group with the longest input length, LEAR can maintain an average 86.3% accuracy across three MCD-splits.

6.2.2 Error Analysis

To understand the source of errors, we take a closer look at the failed test instances of LEAR on CFQ. These failed test instances account for 9.1% of the test dataset. We category them into two error types:

Error Type	MCD1	MCD2	MCD3
CE	45.70%	32.05%	39.83%
IE	54.30%	67.95%	60.17%

Table 7: Distribution of CE (Composer Error) and IE (Interpreter Error).

Composer error (CE), i.e., test cases where Composer produces incorrect syntactic structures (only considering nonterminal nodes). Figure 5a shows an example. As we do not have ground-truth syntactic structures, we determine whether a failed test instance belongs to this category based on handcraft syntactic templates.

Interpreter error (IE), i.e., test cases where Composer produces correct syntactic structures but Interpreter assigns one or more incorrect semantic primitives or operations. Figure 5b shows an example, which contains an incorrect semantic primitive assignment.

Table 7 shows the distribution of these two error types. On average, 39.19% of failed instances are composer errors, and the remaining 60.81% are interpreter errors.

6.3 Limitations

Our approach is implicitly build upon the assumption of **primitive alignment**, that is, each primitive in the meaning representation can align to at least one span in the utterance. This assumption holds in most cases of various semantic parsing tasks, including CFQ, COGS, and GEO. However, for robustness and generalizability, we also need to consider cases that do not meet this assumption. For example, consider this utterance "*Obama's brother*", of which the corresponding meaning representation is "$Slibing(People[Obama]) \wedge Gender[Male]$". Neither "$Slibing$" nor "$Gender[Male]$" can align to a span in the utterance, as the composed meaning of them is expressed by a single word ("*brother*"). Therefore, LEAR is more suitable for formalisms where primitives can better align to natural language.

In addition, while our approach is general for various semantic parsing tasks, the collection of semantic operations needs to be redesigned for each task. We need to ensure that these semantic operations are k-ary projections (as described in Section 2), and all the meaning representations are covered by the operations collection. This is tractable, but still requires some efforts from domain experts.

7 Related Work

7.1 Compositional Generalization

Recently, exploring compositional generalization (CG) on neural networks has attracted large attention in NLP community. For SCAN (Lake and Baroni, 2018), the first benchmark to test CG on seq2seq models, many solutions have been proposed, which can be classified into two tracks: data augmentation (Andreas, 2019; Akyürek et al., 2020; Guo et al., 2020a) and specialized architecture (Lake, 2019; Li et al., 2019; Gordon et al., 2020). However, most of these works only focus on lexical recombination. Some works on SCAN have stepped towards algebraic recombination (Liu et al., 2020; Chen et al., 2020), but they do not generalize well to other tasks such as CFQ (Keysers et al., 2019) and COGS (Kim and Linzen, 2020).

Before our work, there is no satisfactory solution on CFQ and COGS. Previous works on CFQ demonstrated that MLM pre-training (Furrer et al., 2020) and iterative back-translation (Guo et al., 2020d) can improve traditional seq2seq models. HPD (Guo et al., 2020b), the state-of-the-art solution before ours, was shown to be effective on CFQ, but still far from satisfactory. As for COGS, there is no solution to it to the best of our knowledge.

7.2 Compositional Semantic Parsing

In contrast to neural semantic parsing models which are mostly constructed under a fully seq2seq paradigm, compositional semantic parsing models predict partial meaning representations and compose them to produce a full meaning representation in a bottom-up manner (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2012; Liang et al., 2013; Berant et al., 2013; Berant and Liang, 2015; Pasupat and Liang, 2015; Herzig and Berant, 2020). Our model takes the advantage of compositional semantic parsing, without requiring any handcraft lexicon or syntactic rule.

7.3 Unsupervised Parsing

Unsupervised parsing (or grammar induction) trains syntax-dependent models to produce syntactic trees of natural language expressions without direct syntactic annotation (Klein and Manning, 2002; Bod, 2006; Ponvert et al., 2011; Pate and Johnson, 2016; Shen et al., 2018; Kim et al., 2019; Drozdov et al., 2020). Comparing to them, our model learns both syntax and semantics jointly.

8 Conclusion

In this paper, we introduce LEAR, a novel end-to-end neural model for compositional generalization in semantic parsing tasks. Our contribution is 4-fold: (1) LEAR focuses on algebraic recombination, thus it exhibits stronger compositional generalization ability than previous methods that focus on simpler lexical recombination. (2) We model the semantic parsing task as a homomorphism between two partial algebras, thus encouraging algebraic recombination. (3) We propose the model architecture of LEAR, which consists of a Composer (to learn latent syntax) and an Interpreter (to learn operation assignments). (4) Experiments on two realistic and comprehensive compositional generalization benchmarks demonstrate the effectiveness of our model.

Acknowledgments

The work was supported by the National Key Research and Development Program of China (No. 2019YFB1704003), the National Nature Science Foundation of China (No. 71690231), Tsinghua BNRist and Beijing Key Laboratory of Industrial Bigdata System and Application.

Ethical Consideration

The experiments in this paper are conducted on existing datasets. We describe the model architecture and training method in detail, and provide more explanations in the supplemental materials. All the data and code will be released with the paper. The resources required to reproduce the experiments is a Tesla P100 GPU, and for COGS benchmark even one CPU is sufficient. Since the compositional generalization ability explored in this paper is a fundamental problem of artificial intelligence and has not yet involved real applications, there are no social consequences or ethical issues.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resample data for compositional generalization.

Jens Allwood. 2003. Meaning potentials and context: Some consequences for the analysis of variation in meaning. *Cognitive approaches to lexical semantics*, pages 29–66.

Jacob Andreas. 2019. Good-enough compositional data augmentation. *CoRR*, abs/1904.09545.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

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*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Noam Chomsky. 1957. Syntactic structures (the hague: Mouton, 1957). *Review of Verbal Behavior by BF Skinner, Language*, 35:26–58.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with s-diora: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4832–4845.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

J. A. Fodor and Z. W. Pylyshyn. 1988. Connectionism and cognitive architecture - a critical analysis. *Cognition*, 28(1-2):3–71.

Jerry A Fodor and Ernest Lepore. 2002. *The compositionality papers*. Oxford University Press.

Kārlis Freivalds, Emīls Ozoliņš, and Agris Šostaks. 2019. Neural shuffle-exchange networks–sequence processing in o (n log n) time. *arXiv preprint arXiv:1907.07897*.

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

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Demi Guo, Yoon Kim, and Alexander M. Rush. 2020a. Sequence-level mixed sample data augmentation.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020b. Hierarchical poset decoding for compositional generalization in language. *arXiv preprint arXiv:2010.07792*.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020c. Iterative utterance segmentation for neural semantic parsing. *arXiv preprint arXiv:2012.07019*.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2020d. Revisiting iterative back-translation from the perspective of compositional generalization.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative learning of disjoint syntax and semantics. *arXiv preprint arXiv:1902.09393*.

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *AAAI*, volume 5, pages 1062–1068.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019. Compound probabilistic context-free grammars for grammar induction. *arXiv preprint arXiv:1906.10225*.

Dan Klein and Christopher D Manning. 2002. Natural language grammar induction using a constituent-context model. *Advances in neural information processing systems*, 1:35–42.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems 32*, pages 9791–9801. Curran Associates, Inc.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. *arXiv preprint arXiv:1910.02612*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. Compositional generalization by learning analytical expressions.

Gary F Marcus. 2019. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press.

Richard Montague. 1970. Universal grammar. *1974*, pages 222–46.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

John K Pate and Mark Johnson. 2016. Grammar induction from (lots of) words alone. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 23–32.

Francis Jeffry Pelletier. 2003. Context dependence and compositionality. *Mind & Language*, 18(2):148–161.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *CoRR*, abs/1904.09708.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR.

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press.

Zoltán Gendler Szabó. 2004. Compositionality.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. 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 1556–1566. The Association for Computer Linguistics.

Dmitry Tsarkov, Tibor Tihon, Nathan Scales, Nikola Momchev, Danila Sinopalnikov, and Nathanael Schärli. 2020. *-cfq: Analyzing the scalability of machine learning on a compositional task. *arXiv preprint arXiv:2012.08266*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 538–545. Morgan Kaufmann.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

This is the Appendix for the paper: "Learning Algebraic Recombination for Compositional Generalization".

A Semantic Operations in COGS

The semantic primitives used in COGS benchmark are entities (e.g., *Emma* and *cat(x_1)*), predicates (e.g., *eat*) and propositions (e.g., *eat.agent(x_1, Emma)*). The semantic operations in COGS are listed in Table 8.

The operations with "$^{-1}$" (e.g., ON^{-1}) are right-to-left operations (e.g., ON^{-1}(cake, table)→table.ON.cake) while the operations without "-1" represent the left-to-right operations (e.g., ON(cake, table)→cake.ON.table). For operation FillFrame, the entity in its arguments will be filled into predicate/proposition as an AGENT, THEME or RECIPIENT, which is decided by model.

B Semantic Operations in GEO and Post-process

The semantic primitives used in GEO benchmark are entities (e.g., *var0*), predicates (e.g., *state()*) and propositions (e.g., *state(var0)*). The semantic operations in GEO are listed in Table 9.

To fit the FunQL formalism, we design two post-processing rules for the final semantics generated by the model. First, if the final semantic is a predicate (not a proposition), it will be converted in to a proposition by filling the entity *all*. Second, the predicate *most* will be shifted forward two positions in the final semantics.

C Policy Gradient and Differential Update

In this section, we will show more details about the formulation of our RL training based on policy gradient and how to use differential update strategy on it.

Denoting $\tau = \{z, g\}$ as the trajectory of our model where z and g are actions (or called results) produced from Composer and Interpreter, respectively, and $R(\tau)$ as the reward of a trajectory τ (elaborated in Sec. 4.1), the training objective of our model is to maximize the expectation of rewards as:

$$\max_{\theta,\phi} \mathcal{J}(\theta,\phi) = \max_{\theta,\phi} \mathbb{E}_{\tau\sim\pi_{\theta,\phi}}\, R(\tau), \qquad (13)$$

where $\pi_{\theta,\phi}$ is the policy of the whole model θ and ϕ are the parameters in Composer and Interpreter, respectively. Applying the likelihood ratio trick, θ and ϕ can be optimized by ascending the following gradient:

$$\nabla\mathcal{J}(\theta,\phi) = \mathbb{E}_{\tau\sim\pi_{\theta,\phi}}\, R(\tau)\nabla\log\pi_{\theta,\phi}(\tau)\,,$$

which is same with Eq. 9.

As described in Sec. 3 that the interpreting process can be divided into two stages: interpreting lexical nodes and interpreting algebraic nodes, the action g can also be split as the semantic primitives of lexical nodes g_l and the semantic operations of algebraic nodes g_a. In our implement, we utilize two independent neural modules for interpreting lexical nodes and interpreting algebraic nodes, with parameters ϕ_l and ϕ_a respectively. Therefore, $\nabla\log\pi_{\theta,\phi}(\tau)$ in Eq. 9 can be expanded via the chain rule as:

$$\begin{aligned}\nabla\log\pi_{\theta,\phi}(\tau) =& \nabla\log\pi_\theta(z|x) + \\ & \nabla\log\pi_{\phi_l}(g_l|x,z) + \\ & \nabla\log\pi_{\phi_a}(g_a|x,z,g_l)\,.\end{aligned} \qquad (14)$$

With Eq. 14, we can set different learning rates:

$$\begin{aligned}&\theta \leftarrow \theta + \alpha\cdot\mathbb{E}\; R(\tau)\nabla\log\pi_\theta(z|x),\\ &\phi_l \leftarrow \phi_l + \beta\cdot\mathbb{E}\; R(\tau)\nabla\log\pi_{\phi_l}(g_l|x,z),\\ &\phi_a \leftarrow \phi_a + \gamma\cdot\mathbb{E}\; R(\tau)\nabla\log\pi_{\phi_a}(g_a|x,z,g_l).\end{aligned} \qquad (15)$$

Furthermore, in our experiments, the AdaDelta optimizer (Zeiler, 2012) is employed to optimize our model.

D Phrase Table

The phrase table consists of lexical units (i.e., words and phrases) paired with semantic primitives that frequently co-occur with them. It can be obtained with statistical methods.

For CFQ, we leverage GIZA++[4] (Och and Ney, 2003) toolkit to extract alignment pairs from training examples. We obtain 109 lexical units, each of which is paired with 1.7 candidate semantic primitives on average. Some examples in phrase table are shown in Table 10

As to COGS, for each possible lexical unit, we first filter out the semantic primitives that exactly co-occur with it, and delete lexical units with no semantic primitive. Among the remaining lexical units, for those only contain one semantic primitive, we record their co-occurring semantic primitives

[4]https://github.com/moses-smt/giza-pp.git

Operation	Arguments	Result Type	Example
ON(t_1, t_2)	[t_1: Entity, t_2: Entity]	Entity	Emma ate [**the cake** on **a table**] .
IN(t_1, t_2)			A girl was awarded [**a cake** in **a soup**] .
BESIDE(t_1, t_2)			Amelia dusted [**the girl** beside **a stage**] .
ON^{-1}, IN^{-1}, BESIDE^{-1}			NONE
REC-THE(t_1, t_2)	[t_1: Entity, t_2: Entity]	Entity	Lily gave [**Emma a strawberry**] .
THE-REC(t_1, t_2)			A girl offered [**a rose** to **Isabella**] .
AGE-THE, THE-AGE, REC-AGE, AGE-REC			-
FillFrame(t_1, t_2)	[t_1: Entity, t_2: Pred/Prop] [t_1: Pred/Prop, t_2: Entity]	Proposition	A cat [**disintegrated a girl**] .
CCOMP(t_1, t_2)	[t_1: Pred/Prop, t_2: Pred/Prop]	Proposition	[**Emma liked** that **a girl saw**] .
XCOMP(t_1, t_2)			David [**expected** to **cook**] .
CCOMP^{-1}, XCOMP^{-1}			NONE

Table 8: Semantic operations in COGS. "Pred" and "Prop" are abbreviations of "Predicate" and "Proposition", respectively. "AGE", "THE" and "REC" are abbreviations of "AGENT", "THEME" and "RECIPIENT", respectively. "-" omits similar examples. Some operations contain "NONE" example, indicating that no example utilize these operations in dataset.

Operation	Arguments	Result Type	Example
UNION(t_1, t_2)	[t_1: Entity/Prop, t_2: Entity/Prop]	Proposition	what is the population of [**var0 var1**]
INTER(t_1, t_2)			how many [**cities named var0 in the usa**]
EXC(t_1, t_2) $\text{EXC}^{-1}(\text{t}_1, t_2)$			which [**capitals** are not **major cities**]
CONCAT(t_1, t_2) $\text{CONCAT}^{-1}(\text{t}_1, t_2)$	[t_1: Pred, t_2: Pred]	Pred	what is the [**capital of** var0]
FillIn(t_1, t_2)	[t_1: Entity/Prop, t_2: Pred] [t_1: Pred, t_2: Entity/Prop]	Proposition	how many [**citizens** in **var0**]

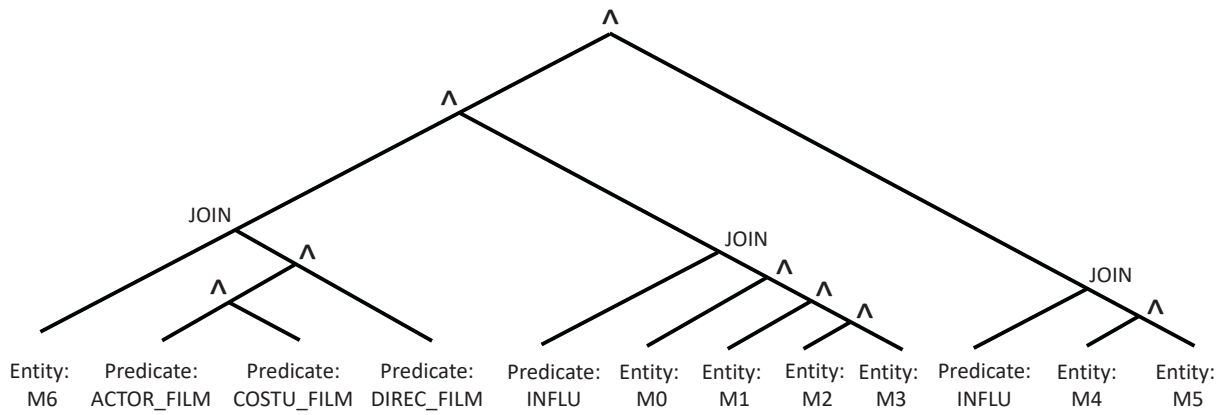
Table 9: Semantic operations in GEO. "Pred" and "Prop" are abbreviations of "Predicate" and "Proposition", respectively. "INTER", "EXC" and "CONCAT" are abbreviations of "INTERSECTION", "EXCLUDE" and "CONCATENATION", respectively.

as ready semantic primitives. For lexical units with more than one semantic primitives, we delete the ready semantic primitives from their co-occurring semantic primitives. Finally, we obtain 731 lexical units and each lexical unit is paired with just one semantic primitive.

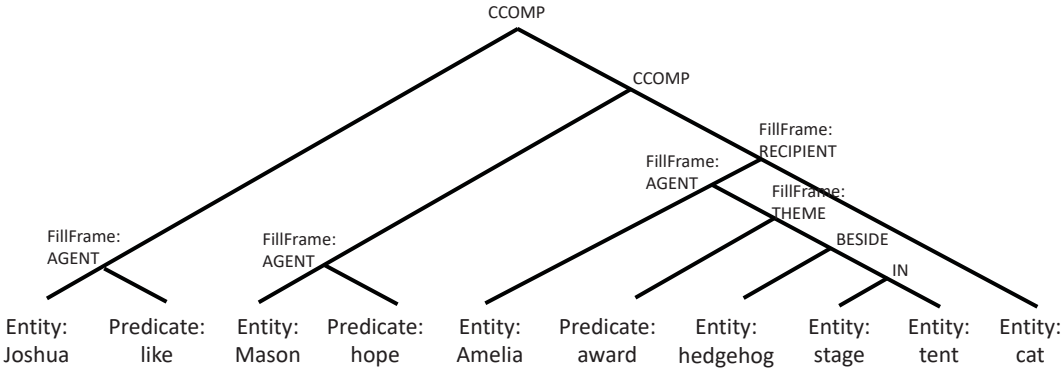
As GEO is quite small, we obtain its phrase table by handcraft.

E More Examples

We show more examples of generated tree-structures and semantics in Figure 6.



(a) An example of generated results in CFQ benchmark with the input "Did M6' s star, costume designer, and director influence M0, M1, M2, and M3 and influence M4 and M5 ".



(b) An example of generated results in COGS benchmark with the input "Joshua liked that Mason hoped that Amelia awarded the hedgehog beside the stage in the tent to a cat".

Figure 6: Examples of generated tree-structures and semantics in CFQ and COGS benchmarks.

Lexical Unit	Semantic Primitive(s)	Type
M0	M0	Entity
executive producer	film.film.executive_produced_by	Predicate
	film.producer.films_executive_produced	Predicate
editor	a film.editor	Attribute
	film.editor.film	Predicate
	film.film.edited_by	Predicate
Italian	people.person.nationality m_03rjj	Attribute

Table 10: Some examples in CFQ phrase table.