ReCOGS: How Incidental Details of a Logical Form Overshadow an Evaluation of Semantic Interpretation

Zhengxuan Wu   **Christopher D. Manning**   **Christopher Potts**
Stanford University, USA
{wuzhengx, manning, cgpotts}@stanford.edu

Abstract

Compositional generalization benchmarks for semantic parsing seek to assess whether models can accurately compute *meanings* for novel sentences, but operationalize this in terms of *logical form* (LF) prediction. This raises the concern that semantically irrelevant details of the chosen LFs could shape model performance. We argue that this concern is realized for the COGS benchmark (Kim and Linzen, 2020). COGS poses generalization splits that appear impossible for present-day models, which could be taken as an indictment of those models. However, we show that the negative results trace to incidental features of COGS LFs. Converting these LFs to semantically equivalent ones and factoring out capabilities unrelated to semantic interpretation, we find that even baseline models get traction. A recent variable-free translation of COGS LFs suggests similar conclusions, but we observe this format is not semantically equivalent; it is incapable of accurately representing some COGS meanings. These findings inform our proposal for ReCOGS, a modified version of COGS that comes closer to assessing the target semantic capabilities while remaining very challenging. Overall, our results reaffirm the importance of compositional generalization and careful benchmark task design.

1 Introduction

Compositional generalization benchmarks have emerged as powerful tools for diagnosing whether models have learned to systematically interpret natural language (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020; Ruis et al., 2020; Wu et al., 2021). The core task is to map sentences to logical forms (LFs), and the goal is to make accurate predictions for held-out examples that include novel grammatical combinations of elements. These tasks are guided by the assumption that natural languages are governed by the *principle of compositionality*: the meanings of complex phrases are determined by the meanings of their parts and the way those parts combine syntactically, and so the meanings of novel combinations of familiar elements are fully determined as well (Montague and Thomason, 1974; Partee, 1984; Janssen and Partee, 1997).

The COGS (**CO**mpositional **G**eneralization Challenge based on **S**emantic Interpretation) benchmark of Kim and Linzen (2020) is among the most widely used compositional generalization benchmarks at present, and it is noteworthy for containing assessment splits that almost no present-day models get traction on (Table 1). The phenomena are remarkably simple. For instance, in the COGS Object-to-Subject modification split (Table 2), models are trained on modified nouns like *the cake on the plate* only in grammatical object positions and asked to make predictions about such phrases when they are grammatical subjects. The average score for this split in prior work is roughly 0. Similar patterns obtain for splits requiring generalization to deeper clausal embedding. This looks like strong evidence that present-day models cannot handle basic matters of semantic interpretation.

We argue that this conclusion is hasty. The core issue is one of semantic representation. The goal of COGS is to assess whether models can compute *meanings* compositionally, but this is operationalized as the task of predicting LFs in a specific format. There will in general be innumerable LF formats that express the desired meanings, and we have no reason to privilege any particular one.

In this paper, we focus on three incidental details of syntactic form that profoundly impact model performance and thus challenge the validity of COGS. First, we find that when we make trivial, meaning-preserving modifications to the LFs by removing redundant symbols, we see substantial improvements in model performance (Section 4.2).

Input Sentence: Mia ate a cake .

COGS LF: eat.agent(x_1,Mia) AND eat.theme(x_1,x_3) AND cake(x_3)

↓

Redundant Token Removal

↓

Length Augmentation via Example Concatenation

↓

Meaning Preserving Syntactic Transformations

ReCOGS LF: Mia(3) ; cake(21) ; eat(6) AND agent(6,3) AND theme(6,21)

Performance
- LEX
- STRUCT

Figure 1: Converting COGS LFs into semantically equivalent LFs greatly impacts model performance: removing redundant tokens increases performance on the lexical (LEX) tasks, while length augmentation and meaning-preserving syntactic transformations help on the harder structural (STRUCT) tasks. ReCOGS incorporates these lessons while also decoupling variable names from linear position. The result is a more purely semantic task that remains extremely challenging for present-day models.

We also identify subtler issues related to variable binding. In COGS, all variables are bound.[1] They appear unbound in the LFs but they are interpreted as existentially closed, as in many theories of dynamic semantics (Kamp, 1981; Heim, 1982, 1983). As bound variables, they can be freely renamed with no change to the interpretation as long as the renaming is consistent; $truck(x)$ and $truck(y)$ are semantically identical in COGS because both variables are implicitly bound with existentials. However, COGS currently requires models to predict the exact identity of variables. This goes well beyond capturing semantics. For neural models that rely on token embeddings, this poses an artificial challenge for test LFs that happen to contain novel variable names or familiar variable names that happen to appear in new contexts. This affects COGS splits involving novel clausal embedding depths (Section 4.3) and novel modification patterns (Section 4.4). Again, meaning-preserving adjustments to the COGS LFs address these issues and allow even baseline models to succeed.

At the same time, we emphasize that COGS does

already contain instances of variable binding relationships that are challenging for all present-day models. A recent proposal by Qiu et al. (2022) to map all COGS LFs to variable-free forms is not meaning preserving precisely because it cannot handle these binding relationships properly (Section 3). Thus, the high scores models have obtained for this COGS variant are partly illusory.

We close with a proposal for a revised version of COGS, ReCOGS, that incorporates the above insights (Section 5). ReCOGS is easier than COGS in some respects and harder in others. Through ablation studies in Section 5, we show that the original linear variable binding of COGS actually make some aspects of COGS artificially easier, and can prevent us from accurately accessing compositional generalization of models. As we noted above, any particular choice of LFs will be somewhat arbitrary relative to our goals, but we feel that ReCOGS comes closer to assessing whether our models possess the compositional generalization capabilities that Kim and Linzen (2020) identified as essential, and our findings suggest that present-day models continue to struggle in these areas.[2]

2 Background: COGS Benchmark

COGS consists of input–output pairs mapping English sentences to LFs. The dataset is generated using a rule-based approach, which allows COGS to maintain systematic gaps between training and different evaluation splits. COGS LFs are based on a Neo-Davidsonian view of verbal arguments (Parsons, 1990), in which verbs introduce event arguments and participants are linked to those events via thematic role predicates. As discussed in the original paper (Kim and Linzen, 2020), the LFs are created by post-processing the simplified ones defined in Reddy et al. (2017). The LFs are purely conjunctive (conjunction is denoted by ; and AND), and conjuncts are sorted by their variable names, which are determined by the position of the head phrase in the sentence (see Table 2 for examples). Event predicates for nominals are not included. Definite and indefinite phrases are formally distinguished. All COGS variables are bound; the definiteness operator * binds variables locally to its conjunct, and all other variables are interpreted as bound by implicit widest-scope existential quantifiers.

By convention, the subscripts on variables in

---

[1]That is, the values that each variable can take on are specified by a variable-binding operator such as a quantifier.

Model	Obj PP → Subj PP	STRUCT CP Recursion	PP Recursion	LEX	Overall %
BART (Lewis et al., 2020)	0	0	12	91	79[†]
BART+syn (Lewis et al., 2020)	0	5	8	80	80[†]
T5 (Raffel et al., 2020)	0	0	9	97	83[†]
Kim and Linzen 2020	0	0	0	73	63
Ontanon et al. 2022	0	0	0	53	48
Akyurek and Andreas 2021	0	0	1	96	82
Conklin et al. 2021	0	0	0	88	75
Csordás et al. 2021	0	0	0	95	81
Zheng and Lapata 2022	0	25	35	99	88[‡]

Table 1: Results on the COGS benchmark for different generalization splits, including recent seq2seq models specialized for COGS. [†]Models use pretrained weights, and their results are copied from Yao and Koller (2022). [‡]Model uses pretrained weights and is hyperparameter tuned using data sampled from the generalization splits. Our focus is on the factors behind the strikingly bad performance of all models, but especially the models that are not pretrained, on the structural generalization splits.

COGS correspond to the 0-indexed position of the corresponding word in the input sentence. Thus, the LF for *A cat rolled Lina* includes a conjunct `cat(x_1)`, which indicates *cat* is the first word in the sentence, whereas the LF for *Lina rolled a cat* includes a conjunct `cat(x_3)`, which indicates *cat* is the third word in the sentence.

The COGS evaluation metric is percent exact string identity of logical forms. COGS provides a single training split as well as standard in-distribution (IID) validation splits. In addition, the dataset includes *generalization splits* designed around types of examples that were not seen in training but seem to be natural extrapolations of examples seen in training under assumptions of compositionality, as exemplified in Table 2. The generalization splits cover five scenarios:

1. Interpreting novel pairings of primitives and grammatical roles (e.g., Subj → Obj Proper).

2. Verb argument structure alternation (e.g., Active → Passive).

3. Sensitivity to verb class (e.g., Agent NP → Unaccusative subject).

4. Interpreting novel combinations of modified phrases and grammatical roles (e.g., Object PP → Subject PP).

5. Generalizing phrase nesting to unseen depths (e.g., CP Recursion).

The first three fall under lexical generalization, and the final two require structural generalization.

It is noteworthy that, for the splits in group 1, there is only a single training example for the primitive with a single input and output token (e.g., "Paula" → `Paula`). This leads to higher variance across these related splits; we suspect that model performance for these cases is affected by when this single example is seen by the model during training.

3 Related Work

Approaches to COGS Researchers have adopted a variety of approaches to solving COGS, including grammar-based rules (Herzig and Berant, 2021), lexicons or lexicon-style alignments incorporated into seq2seq models (Akyurek and Andreas, 2021; Zheng and Lapata, 2021), modified Transformer models for better-structured representations (Oren et al., 2020; Zheng and Lapata, 2022; Bergen et al., 2021; Csordás et al., 2021), meta-learning (Conklin et al., 2021), tree-like neural parsers (Weißenhorn et al., 2022), a grammar-enhanced seq2seq learner (Wang et al., 2022), and various data augmentation techniques (Qiu et al., 2022).

COGS Artifacts A number of recent papers investigate artifacts in compositional generalization benchmarks such as SCAN (Patel et al., 2022) and ReaSCAN (Sikarwar et al., 2022). Csordás et al. (2021) focus on COGS. They examine potential pitfalls that might lead us to underestimate a model's ability to generalize. By carefully exploring the effects of relative positional embeddings and training

Case	Split	Example
Subj → Obj Proper	*Train* (LF)	**Lina** gave the bottle to John . `*bottle(x_3) ; give.agent(x_1,Lina) AND give.theme(x_1,x_3) AND give.recipient(x_1,John)`
	Gen. (LF)	A cat rolled **Lina** . `cat(x_1) AND roll.agent (x_2,x_1) AND roll.theme(x_2,Lina)`
Prim → Subj Proper	*Train* (LF)	**Paula** `Paula`
	Gen. (LF)	**Paula** painted a cake . `paint.agent(x_1,Paula) AND paint.theme(x_1,x_3) AND cake(x_3)`
Prim → Obj Proper	*Train* (LF)	**Paula** `Paula`
	Gen. (LF)	James rolled **Paula** . `agent(x_1,James) AND roll.theme(x_1,Paula)`
Obj PP → Subj PP	*Train* (LF)	Emma ate **the cake on the table** . `*cake(x_3) ; *table(x_6) ; eat.agent(x_1,Emma) AND eat.theme(x_1,x_3) AND cake.nmod.on(x_3,x_6)`
	Gen. (LF)	**The cake on the table** burned . `*cake(x_1) ; *table(x_4) ; cake.nmod.on(x_1,x_4) AND burn.theme(x_5,x_1)`
CP Recursion	*Train* (LF)	Noah knew **that** Emma said **that** the cat painted . `*cat(x_7) ; know.agent(x_1,Noah) AND know.ccomp(x_1,Emma) AND say.agent(x_4,Emma) AND say.ccomp(x_4,x_7) AND paint.agent(x_8,x_7)`
	Gen. (LF)	Noah knew **that** Emma said **that** John saw **that** the cat painted . `*cat(x_10) ; know.agent(x_1,Noah) AND know.ccomp(x_1,Emma) AND say.agent(x_4,Emma) AND say.ccomp(x_4,x_7) AND see.agent(x_7,John) AND see.ccomp(x_7,x_10) AND paint.agent(x_11,x_10)`
PP Recursion	*Train* (LF)	John saw the ball **in** the bottle **in** the box . `*ball(x_3) ; *bottle(x_6) ; *box(x_9) ; see.agent(x_1,John) AND see.theme(x_1,x_3) AND ball.nmod.in(x_3,x_6) AND bottle.nmod.in(x_6,x_9)`
	Gen. (LF)	John saw the ball **in** the bottle **in** the box **on** the floor . `*ball(x_3) ; *bottle(x_6) ; *box(x_9) ; *floor(x_12) ; see.agent(x_1,John) AND see.theme(x_1,x_3) AND ball.nmod.in(x_3,x_6) AND bottle.nmod.in(x_6,x_9) AND box.nmod.on(x_9,x_12)`

Table 2: Representative COGS generalization (*Gen.*) splits with logical forms (LFs). Due to space constraints, we use simplified versions of examples included in the dataset. LFs are detokenized for readability; tokenized examples can be found in Table 3.

time, they are able to increase overall model performance from 35% to 81% on the generalization splits. Our work complements these findings by focusing on issues of semantic representation.

A Flawed Variable-Free COGS Representation Qiu et al. (2022) propose a variable-free version of COGS and show that models score much better on the format (which is used by Drozdov et al. (2023) in experiments with GPT-3). However, the representations of Qiu et al. do not preserve the meaning of COGS examples. COGS embeds variable-binding challenges that the variable-free forms artificially side-step. For instance, their formalism represents *A zebra needs to walk* as

Variant (Token Removal Set)	Logical Form
COGS	`* boy ( x _ 3 ) ; hold . agent ( x _ 1 , Liam ) AND hold . theme ( x _ 1 , x _ 3 ) AND boy . nmod . beside ( x _ 3 , x _ 6 ) AND table ( x _ 6 )`
Token Removal ({x _})	`* boy ( 3 ) ; hold . agent ( 1 , Liam ) AND hold . theme ( 1 , 3 ) AND boy . nmod . beside ( 3 , 6 ) AND table ( 6 )`
Token Removal ({x _ ( )})	`* boy 3; hold . agent 1 , Liam AND hold . theme 1 , 3 AND boy . nmod . beside 3 , 6 AND table 6`
Token Removal ({x _ ( ) ,})	`* boy 3; hold . agent 1 Liam AND hold . theme 1 3 AND boy . nmod . beside 3 6 AND table 6`

Table 3: Removing redundant tokens from the LF for the sentence *Liam held the boy beside a table* .

`need(agent=zebra,xcomp=walk(agent=zebra))`

which means 'a zebra needs a zebra to walk', with two unlinked occurrences of the indefinite *a zebra*. Consider a model where zebra a needs zebra b to walk, for $a \neq b$, and no zebra a is such that a needs a to walk. The above LF is true in this model but our original sentence is not (McCawley, 1968). In addition, these variable-free forms would be unable to represent quantifier-binding relationships like *Every zebra ate its meal* as well as simple reflexives like *A zebra saw itself*. Since variable binding arguably reflects one of the deepest challenges of natural language interpretation, we argue that these phenomena should not be marginalized.[3]

4 Experiments

We report on experiments studying semantic representations for COGS. For any modifications we apply to the dataset, we ensure there is no data leakage, as such leakage trivializes the generalization tests (as evidenced by the consistently very high results for COGS IID test splits in prior work).

4.1 Methods

Architectures   Following the original COGS paper (Kim and Linzen, 2020), we train encoder–decoder models with two model architectures: LSTMs (Hochreiter and Schmidhuber, 1997) and Transformers (Vaswani et al., 2017). We adopt similar configurations to the original paper. For the LSTMs, we use a 2-layer LSTM as the encoder with global attention and a dot-product scoring

function, and a 2-layer LSTM as the decoder with a hidden dimension size of 512. For the Transformers, we use two 2-layer Transformer blocks with 4 attention heads and learnable absolute positional embeddings[4] with a hidden dimension size of 300.

The LSTM architecture has approximately 9M parameters whereas the Transformer architecture has approximately 4M parameters.

Training Details   We use cross-entropy loss and a fixed number of training epochs (200 for LSTMs and 300 for Transformers), as previous work suggests that early stopping hurts model performance (Csordás et al., 2021). We set the batch size to 128 for the Transformers and 512 for the LSTMs. We train with a fixed learning rate of 8×10^{-4} for the LSTMs, and 1×10^{-4} for the Transformers.

We use a single NVIDIA GeForce RTX 3090 24GB GPU to train our models. For LSTMs, the training time ranges from 0.5 hours to 5 hours. For Transformers, it ranges from 0.3 hours to 3 hours. We run each experiment with 20 distinct random seeds. Additional training details can be found in our code repository.

No Pretraining   We train all of our models from scratch, without any pretraining, to ensure that we are not introducing outside information that could be relevant to the COGS generalization tasks (q.v. Kim et al., 2022).

4.2 Experiment 1: Removing Redundant Tokens from LFs

Our first experiments involve only very trivial modifications to COGS LFs: we remove some redundant

---

[3]The results of Curry et al. (1958) ensure that there is some variable-free representation scheme that can capture these phenomena. It may be worthwhile to develop and explore such representations as alternative LFs for COGS.

[4]We found that using relative embeddings or initializing embeddings with periodic functions did not improve performance, corroborating findings of Zheng and Lapata (2022).

Model (Token Removal Set)	LEX			
	Subj → Obj Proper	Prim → Obj Proper	Prim → Subj Proper	Overall
LSTM	5.3 [1.4, 9.3]	21.9 [10.9, 32.9]	69.0 [51.6, 86.3]	32.1 [28.2, 36.0]
+ Tokens Removal ({x _})	5.1 [0.2, 10.0]	18.1 [8.7, 27.4]	76.5 [60.9, 92.0]	34.9 [31.5, 38.2]
+ Tokens Removal ({x _ ( )})	7.2 [1.6, 12.9]	24.6 [14.5, 34.8]	88.1 [78.2, 98.1]	39.6 [36.7, 42.5]
+ Tokens Removal ({x _ ( ) , })	5.5 [1.2, 9.8]	14.7 [5.8, 23.6]	65.4 [45.6, 85.3]	36.5 [32.8, 40.2]
Transformer	74.4 [71.9, 76.9]	62.4 [60.1, 64.6]	97.6 [96.4, 98.8]	81.3 [80.7, 81.9]
+ Tokens Removal ({x _})	91.9 [90.0, 93.9]	81.1 [74.5, 87.7]	99.2 [98.6, 99.7]	83.6 [82.9, 84.3]
+ Tokens Removal ({x _ ( )})	86.0 [83.2, 88.9]	71.2 [61.3, 81.1]	97.8 [95.5, 100.2]	82.8 [82.1, 83.4]
+ Tokens Removal ({x _ ( ) , })	88.3 [84.4, 92.2]	73.4 [65.8, 81.0]	99.0 [98.7, 99.4]	83.4 [82.8, 84.0]

Table 4: Results on the COGS lexical generalization splits for different token removal scheme. We report means (over 20 evaluations) with bootstrapped 95% confidence intervals.
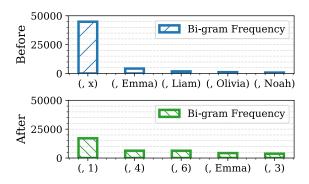

Figure 2: The frequencies of bigrams in the training data starting with , become more balanced after removing two incidental tokens {x _}.
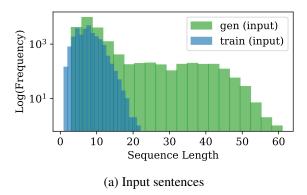
tokens from the LFs, with no other modifications to the examples, and we study the effects this has on the lexical generalization splits in COGS. (These modifications alone do not significantly improve results on the structural generalization splits.)
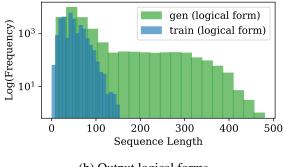
LF Modifications   Table 3 summarizes our redundant token removal strategies. The first row gives a COGS example. Each noun or verb is associated with a variable representing its token position in the input sentence. Concretely, these variables are given as three whitespace-separated elements: x _ N, where N is a numeral. The initial x, the underscore, and the spaces do not contribute to variable identity in any way. Thus, we can remove these prefixes without changing the semantics of the LFs. The second row of Table 3 depicts removal of the underscore, and the third row depicts removal of the entire prefix. Additionally, we experiment with different token removal schemes by removing redundant punctuation in the set {, ( )}.

Results   Table 4 illustrates model performance on the three most demanding lexical generalization splits when models are trained and assessed with these minor variants of COGS LFs. Our primary finding is that model performance is highly sensitive to redundant token removal. By removing the prefix x _, the Transformer achieves nearly 30% better performance on average for the Primitive → Object Proper Noun split, and the LSTM's performance improves 27% for the Primitive → Subject Proper Noun split after removing the prefix x _ and parenthesis tokens.

Our findings show that model performance can vary substantially across different semantically equivalent LFs. In addition, while model performance is stable in the Overall evaluation, some of the splits show a high degree of sensitivity to the random seed. Nonetheless, even in these splits, the performance increases that stem from redundant token removal are consistently large enough to be robust to this variance.

We hypothesize that the improvements that come from removing these redundant tokens derive from simple considerations of how sequence models operate. As shown in Figure 2, the bigram , x appears 44,846 times, whereas the bigram , Emma appears 4,279 times (Emma is the most frequent proper noun in the training split). Thus, the conditional bigram probability $P(x \mid ,)$ is significantly larger than $P(\text{Emma} \mid ,)$. Removing the prefix balances the dataset in this respect because we then mostly care about $P(N \mid ,)$ for different values of N. This helps the decoder to generate less skewed label distributions. This corroborates the findings of Yao and Koller (2022), who show that the decoder is heavily biased towards generating seen n-grams.

(a) Input sentences

(b) Output logical forms

Figure 3: Sequence length distributions for the COGS training split and the generalization splits. The generalization split has inputs and logical forms with lengths completely unseen in the training set.
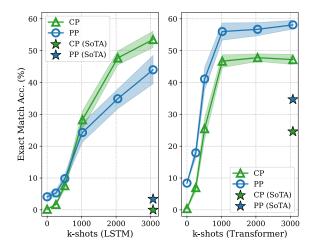

Figure 4: Adding k items with concatenated training examples to give exposure to long sequences greatly improves structural generalization on COGS for both LSTM-based and transformer-based models. (Transformer-based) SoTA performance is taken from Zheng and Lapata (2021). The plots show means (of 20 runs) with 95% confidence interval.

4.3 Experiment 2: Separating Structural and Length Generalization

The splits that focus on recursive phrase embedding are among the hardest problems in COGS. When training, models see multiply embedded examples like [*John knows that* [*Noah knows that . . .*]CP]CP containing nestings of depth 0–2. At test time, they see examples involving strictly greater depth (3–12). COGS includes two constructions that allow nesting: sentential complements (nested CPs) and nominal PP modifiers (nested PPs).

The nature of these COGS splits creates a strict relationship between recursion depth and sequence length: deeper recursion leads to longer sequences. Figure 3 summarizes the situation when it comes to input sentences and output logical forms. As expected, we see non-overlapping long-tail distributions over longer sequences.

This is perfectly well-posed as a generalization task simultaneously assessing models on both longer lengths and deeper recursion. However, COGS binds these two tasks together in a way that takes us outside of the goal of assessing semantic generalization. Recall that COGS variables are named according to their linear position. This virtually guarantees that there will be variables whose numerical components are encountered only at test time. For instance, if 47 is the largest variable index seen at train time, then 48 and above will be encountered only in testing. For models that rely on embeddings, these new variables will have random representations at test time. In other words, COGS is underspecified in a way that prevents models from learning novel positional embeddings or embeddings for novel positional indices. However, there is nothing privileged about this particular variable naming scheme; as we discussed earlier, bound variables can be freely renamed as long as this does not change any binding relationships.

Relatedly, for models like Transformers with token positional embeddings, some of the position representations will be encountered only at test time. It is fair to say that this is a limitation of these models that COGS is exposing, but it also further reinforces the concern that length generalization may be totally overshadowing the challenge of generalizing to novel recursion depths.

LF Modifications To overcome the challenge of length generalization, we augment the training

data by concatenating existing examples together, reindexing the output LFs with higher positional indexes, and gluing them together as conjunctive terms. In doing this, we do not add any semantically new claims to the COGS training set, and we do not create any new recursive structures. We simply ensure that the relevant token and position indices are not random at test time. Similar concatenation experiments have been shown effective in understanding model artifacts for language model acceptability tasks (Sinha et al., 2023).

Results   Figure 4 presents our results. We gradually introduce k augmented examples into our training set, where $k \in \{256, 512, 1024, 2048, 3072\}$. Our results indicate that the previously seen catastrophic failure of structural generalization over nested clauses is largely due to the fact that models are not trained with longer sequences. When the sequence length issue is addressed, the models appear to be very capable at handling novel recursion depths. Indeed, our models now far surpass published state-of-the-art results on these tasks.

We further note that the failures are not due to our setup of relative positional embeddings, as fine-tuning a T5-base model with fixed window-based relative positional embeddings remains 0.0% on both splits, as shown in Table 1.

4.4   Experiment 3: Variable Name Binding Prevents Generalization

The hardest COGS split based on published numbers seems to be the structural generalization task that involves interpreting novel combinations of modified phrases and grammatical role – e.g., interpreting subject noun phrases with PP modifiers when the train set includes only object noun phrases with such modifiers (*Noah ate **the cake on the plate**.* → ***The cake on the plate** burned*). To the best of our knowledge, all prior seq2seq models have completely failed to get traction here (Table 1). Our goal in this section is to understand more deeply why this split has proven recalcitrant.

To start, we observe that this split is arguably different conceptually from the others. The train set contains only object-modifying PPs. It is quite reasonable for a learner to infer from this situation that PPs are allowed only in this position. Natural languages manifest a wide range of subject/object asymmetries, and learners presumably induce these at least in part from an absence of certain kinds of inputs in their experience. Thus, there is a case

to be made that this split is not strictly speaking *fair* in the sense of Geiger et al. (2019): we have a generalization target in mind as analysts, but this target is not uniquely defined by the available data in a way that would invariably lead even an ideal learner to the desired conclusion.

That said, we feel it is reasonable to explore whether models can learn the sort of theory that would naturally allow them to make correct predictions about these particular novel inputs, even if their training experiences run counter to that. However, COGS erects another obstacle to this goal by numbering variables using a word's position index in the sentence. At train time, the model sees only PP modifiers like the one in the first row of Table 3. This associates PP modifiers with a particular range of tokens (variable numerals). As before, this indexing scheme is not a semantic matter, but rather a superficial matter of representation. However, it can also be seen as very direct supervision about the limited distribution of these modifiers: they can associate only with relatively high variable indices.

Additionally, for models with learned positional indices, all modifier phrase tokens are associated with a particular set of relatively high positional index values. These models thus further reinforce the distributional inference that such phrases cannot appear in subjects.

Both of the above concerns are supported by error analyses on vanilla models trained on the original COGS LFs. For instance, to correctly generate the LF for subject-modifying PPs, the model has to predict the LF term for a PP modifier as the first conjunctive clause after the semicolon (e.g, `cake .nmod .on`). However, 0% of the model's predictions follow this pattern, as this never happens during training. Likewise, since no subject-modifying PPs are seen in training, the variables mentioned in PP clauses are never associated with the subject. The effect of these patterns on model behavior is clear: the first variable of any generated modified phrase clause always refers to a token position seen during training.

To isolate these effects from the structural generalization, we propose three data modification strategies. None of these strategies change the set of meanings expressed by COGS. Rather, they make adjustments to the syntax that, according to the COGS indexing rules, automatically expand the range of variables and positional indices associated with modifier phrases. Examples for each strategy

Variant	Sentence	Logical Form (LF)
Preposing + Filler Words	**The box in the tent** Emma was **um um** lended .	`*box(x_1) ; *tent(x_4) ; box.nmod.in(x_1 ,x_4) AND lend.theme(x_7,x_1) AND lend .recipient( x_7,Emma)`
Participial Verb Phrase (*Subj*)	A leaf **painting the spaceship** froze .	`*spaceship(x_4) ; leaf(x_1) AND leaf.acl .paint(x_1,x_4) AND freeze.theme(x_5,x_1)`

Table 5: Modifications of COGS input and output sequences that we use to diagnose artifacts in the original semantic representation. LFs are detokenized for readability; cf. Table 3 for tokenized examples.

are given in Table 5.

Preposing One strategy to disentangle the effects of variable names and positional indices is to move the modified phrases to the front of the sentence via preposing (topicalization). This has no effect on argument structure or the meaning of the corresponding LF. It simply ensures that the model sees concurrences of positional indices that would otherwise have only appeared in the testing set. In our experiments, we prepose the object in 5% of training examples containing at least one prepositional phrase.

Preposing brings a new challenge for parsing, since the model needs to learn how to parse preposed PP modifiers for object NPs as well as the regular ones. Despite this new challenge, preposing dramatically improves model performance.

Filler Words To further alleviate the problem with unseen occurrences of positional indices, we experiment with adding filler words ("um") into the input sentence. This shifts around the variable names and positional indices in the LF (see the corresponding example in Table 3). We sprinkle 1–3 filler words at random in 5% of training examples with one or more prepositional phrase. Filler words can be seen as mapping to the constant conjunct with meaning `True`, and thus they do not affect the overall meaning of the LF.

Participial Verb Phrases We augment the training set by providing examples for participial verb (PV) phrases for both objects and subjects. The original vocabulary of COGS does not have any participial verbs, so we added participial forms for all verbs.[5] To create full phrases, we randomly select a participial verb and a noun. Then, we randomly assign an article for the noun selected. For

---

[5] We did this using `ChatGPT`, prompted with "Convert the following verbs into participial verbs" followed by a list of verbs that exist in COGS and then hand-checked the results.

two participial verb phrases in a row, we repeat this process twice. The vocabulary files of the models are updated to include the new tokens required for parsing these PV phrases.

As shown in the corresponding row in Table 5, the model now needs to learn this new type of noun modifier `acl` with participial verbs. We argue that testing with PP modifiers within subject NPs becomes more reasonable in this new setting, as we completely isolate the potential artifacts due to the positional indices. We add PV phrases to 15% of training examples with maximally two PV phrases in a row. We include an *easy* version without adding the novel `acl` token and replace it with the existing `nmod` token in the LF.

Results As shown in Table 6, both the LSTM and the Transformer model start to show progress on this structural generalization split. Surprisingly, the LSTM model becomes much more effective in parsing subject NPs after including PV phrases, which contrasts with previous findings that the Transformer model is always better than the LSTM model at structural generalization on average. Our modifications isolate the effect of variable name bindings from structural generalization. The results suggest that the stagnation of model performance on this split is mostly due to the particular variable naming convention of the current semantic representation. We later show (in Section 5) that models overfit to the variable binding of COGS, which could make COGS artificially easier, and prevent us from accurately accessing compositional generalization of models.

5 ReCOGS: A Revised Version of COGS

We now propose a revision to COGS called ReCOGS. We assemble the insights provided by the above experiments and use them to inform a benchmark that comes closer to assessing models purely on their ability to handle semantic gener-

| Model | STRUCT | | | |
	Obj PP → Subj PP	CP Recursion	PP Recursion	Overall
LSTM	$0.0_{[0.0,\,0.0]}$	$0.2_{[0.1,\,0.3]}$	$4.1_{[2.9,\,5.3]}$	$32.1_{[28.2,\,36.0]}$
+ Preposing	$8.1_{[6.6,\,9.5]}$	$0.3_{[0.1,\,0.4]}$	$3.4_{[2.1,\,4.6]}$	$32.7_{[28.6,\,36.7]}$
+ Preposing + Filler Words	$8.7_{[7.1,\,10.2]}$	$0.8_{[0.6,\,1.0]}$	$0.1_{[0.0,\,0.2]}$	$33.9_{[29.1,\,38.6]}$
+ Participial Verb Phrase	$54.9_{[38.0,\,71.8]}$	$3.3_{[2.8,\,3.8]}$	$5.8_{[4.7,\,7.0]}$	$43.3_{[41.4,\,45.2]}$
+ Participial Verb Phrase (*easy*)	$88.7_{[86.2,\,91.3]}$	$4.9_{[4.3,\,5.4]}$	$5.6_{[4.6,\,6.7]}$	$44.5_{[42.4,\,46.5]}$
Transformer	$0.0_{[0.0,\,0.0]}$	$0.4_{[0.2,\,0.6]}$	$8.4_{[8.2,\,8.6]}$	$81.3_{[80.7,\,81.9]}$
+ Preposing	$18.6_{[16.2,\,21.1]}$	$0.5_{[0.4,\,0.6]}$	$8.6_{[8.3,\,8.9]}$	$81.6_{[81.0,\,82.2]}$
+ Preposing + Filler Words	$20.5_{[19.1,\,22.0]}$	$1.3_{[1.0,\,1.6]}$	$9.1_{[8.3,\,9.8]}$	$82.6_{[82.2,\,83.1]}$
+ Participial Verb Phrase	$24.7_{[11.8,\,37.6]}$	$4.2_{[3.7,\,4.7]}$	$10.2_{[9.7,\,10.7]}$	$83.6_{[82.8,\,84.3]}$
+ Participial Verb Phrase (*easy*)	$82.7_{[80.9,\,84.4]}$	$3.8_{[3.4,\,4.2]}$	$9.9_{[9.3,\,10.5]}$	$85.9_{[85.2,\,86.6]}$

Table 6: Results on the COGS structural generalization splits for different meaning preserving data augmentation methods designed to ensure that the full range of needed variable indices are seen during training. We report means (over 20 runs) with bootstrapped 95% confidence intervals. Training on these data augmentations greatly improves compositional generalization.

Variant	Logical Form (LF)
COGS	zebra(x_1) AND need.agent(x_2,x_1) AND need.xcomp(x_2,x_4) AND walk.agent(x_4,x_1)
ReCOGS	zebra(47) ; need(13) AND agent(13,47) AND xcomp(13,48) AND walk(48) AND agent(48,47)

Table 7: Semantic representations for *A zebra needed to walk.* in COGS and ReCOGS. LFs are detokenized for readability; tokenized examples can be found in Table 3.

alizations. Table 7 provides an example. We first implement the following changes, supported by our experiments, to create ReCOGS:

1. We remove redundant prefix tokens including x and _ (motivated by Section 4.2).

2. We replace each token position index with a random integer available in the model vocabulary file, maintaining consistent coreference. For instance, every appearance of x_3 may be replaced with x_46, which makes the indices irrelevant to their token positions in the sentence. For each original COGS example, we create 5 different versions by randomly sampling 5 distinct sets of indices. We provide a ReCOGS variant, ReCOGS_POS, without this change to show its effect.

3. We augment the current training split with longer examples by concatenating existing training examples. In total, we add in 15,360 (3,072 for each set of indices) new examples. We make sure only unique examples are kept (motivated by Section 4.3).

4. We prepose the object for 5% of the training examples containing an object with at least one prepositional phrase, and we randomly add filler words ("um") into these examples (motivated by Section 4.4).

Additionally, we make the following changes that are not experimentally supported but help make ReCOGS LFs more consistent and expressive:

1. We treat proper nouns as predicates, which prevents collisions when multiple distinct entities share the same name (e.g., Mia(4) and Mia(5) can refer to different people). This also leads to a more uniform semantic treatment of noun phrases in general.

2. We prepose proper nouns and nouns with indefinite articles in the LF, parallel to definites. This makes LFs more consistent.

3. We separate situations and semantic roles, and rely on variable binding to link them (e.g., reformat agent.eat(6,7) as eat(6) AND agent(6,7)). The resulting LFs more closely
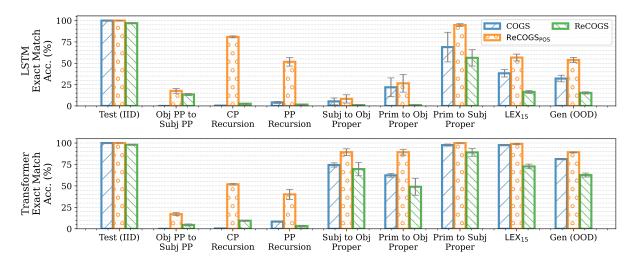
Figure 5: Model performance over different testing splits in COGS, ReCOGS$_{POS}$ (original variable name bindings are kept), and ReCOGS. We report means (of 20 runs) with bootstrapped 95% confidence intervals.

	Train	Dev	Test	Gen
COGS	24,155	3,000	3,000	21,000
ReCOGS	135,547	3,000	3,000	21,000

Table 8: Dataset statistics.

resemble the Neo-Davidsonian view of verbal arguments (Parsons, 1990).

Table 8 provides basic dataset statistics for COGS and ReCOGS.

We emphasize that ReCOGS continues to use variables in its representations. As we noted in Section 3, the variable-free proposal of Qiu et al. (2022) does not always preserve the meanings of the original COGS representations. COGS embeds variable-binding challenges that this variable-free form artificially side-steps, something that we feel is inappropriate when variable binding arguably reflects one of the deepest challenges of natural language interpretation.

As a metric for ReCOGS, we propose Semantic Exact Match (SEM). For SEM, we exhaustively check whether there is a semantically consistent conversion of the variables in the predicted LFs into those of the actual LF. For COGS, this amounts to checking whether there is bijective map between variables in the predicted and actual LFs, where corresponding variables participate in exactly the same predications.[6] For instance, the predicted LF

table(46) AND sturdy(46) can be converted to table(1) AND sturdy(1) via the mapping $[46 \mapsto 1]$, but the LF table(46) AND sturdy(7) cannot be converted in this way because 46 and 7 would have to both map to 1. The reverse conversion is also blocked because 1 would have to map to both 46 and 7. In addition, we treat conjunctions in the LF as an order-free set, and exact match is evaluated as set equality after variable conversion. For the COGS language, these steps amount to checking for semantic equivalence.[7]

We evaluate ReCOGS with the same two model architecture setups as in previous sections. Figure 5 reports initial results. Overall, we find that ReCOGS is a more challenging compositional generalization benchmark than COGS, but models are able to get traction on all splits, suggesting we have avoided the worrisome pattern of 0s seen in Table 1.

While both our models still achieve near 100% performance on the IID testing split, they struggle on the structural and lexical generalization tests. However, they now show signs of life. For instance, the Object PP → Subject PP split of ReCOGS is now tractable while remaining challenging. This shows how the meaning-preserving modifications of "um" insertion and preposing can remove artifacts that were preventing models from learning before. In addition, our results show that LSTMs perform better than Transformers on the same split

---

[6] SEM also resembles SMATCH score (Cai and Knight, 2013) used for abstract meaning representation (AMR) pars-

ing (Banarescu et al., 2013).

[7] Some straightforward modifications to the SEM checking procedure would be needed if COGS included tautologies, contradictions, or equality statements between variables.

(13.4% vs. 4.5% on average). Future work may investigate the reason why LSTMs seem to generalize better in this scenario.

We also see the effects of breaking the relationship between variable names and linear position. Without this non-semantic pattern to rely on, models show degraded performance. To verify this, we revert ReCOGS back to the original variable name bindings of COGS where the order of LFs mirrors the word order in the input sequence, and generate a new variant, ReCOGS$_{POS}$. As shown in Figure 5, performance of both models increases significantly on ReCOGS$_{POS}$. Consequently, we believe that ReCOGS poses additional challenges in terms of long-term coindexing over entities, challenges that were partly obscured by the variable naming scheme chosen for COGS.

6 Discussion and Conclusion

With compositional generalization benchmarks, we hope to gain reliable information about whether models have learned to construct the *meanings* of natural language sentences. However, meanings are highly abstract entities that we do not know how to specify directly, so we are compelled to conduct these assessments using LFs, which are syntactic objects that we presume can themselves be mapped to meanings in the relevant sense. The central difficulty here is that there is no single privileged class of LFs to use for this purpose, and different LFs are likely to pose substantially different learning problems and thus may lead to very different conclusions about our guiding question.

We explored these issues in the context of COGS, a prominent compositional generalization benchmark. COGS includes sub-tasks that even our best present-day models cannot get any traction on. Our central finding is that there are two major factors contributing to this abysmal performance: redundant symbols in COGS LFs and the requirement imposed by COGS that models predict the exact numerical values of bound variables. These details cannot be justified semantically, and they play a large role in shaping model performance. We showed that simple, meaning-preserving modifications to COGS fully address these problems and allow models to succeed. In turn, we propose ReCOGS, a modified version of COGS that incorporates these insights. Our models are able to get some traction on all the sub-tasks within ReCOGS, but it remains a very challenging benchmark.

One of our reviewers raised an important question relating to ReCOGS and benchmarking efforts that might follow from it: could this encourage an unproductive sort of "LF hacking" in which people continually reformat the data in an effort to incrementally boost performance? This is certainly a risk. However, for now, we venture that such experimentation can be productive. If the modified LFs are truth-conditionally identical to the originals, we can be confident that the generalization splits are not being compromised by this LF hacking. Thus, consistent improvements in performance may lead to lasting insights about effective meaning representation for our models, and persistent failures are likely to point to deep limitations. Overall, then, some LF hacking could help us get incrementally closer to truly evaluating the capacity of models to compute *meaning*.

Our results also raise important questions about compositional generalization itself. It is in the nature of compositional generalization tasks that models will be assessed on examples that are meaningfully different from those they have seen in training. When is this fair, and when does it implicitly contradict the train set itself? Our discussion focused on the distribution of PP modifiers across different grammatical positions. COGS models are trained on examples that might suggest these are limited to object noun phrases. This may seem implausible for PP modifiers, but many natural languages do display subject/object asymmetries with regard to phenomena like case marking, definiteness, and pronominalization. In our experiments, we indirectly instructed our model about our intended generalization via meaning preserving modifications to the training data, but ultimately these issues may call for deeper changes to how we pose compositional generalization tasks.

Acknowledgements

References

Ekin Akyurek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. In *Association for Computational Linguistics (ACL)*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Leon Bergen, Timothy O'Donnell, and Dzmitry Bahdanau. 2021. Systematic generalization with edge transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Association for Computational Linguistics (ACL)*.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Association for Computational Linguistics (ACL)*.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Haskell Brooks Curry, Robert Feys, William Craig, J Roger Hindley, and Jonathan P Seldin. 1958. *Combinatory Logic*. North-Holland Amsterdam.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional semantic parsing with large language models. In *International Conference on Learning Representations (ICLR)*.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2019. Posing fair generalization tasks for natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Irene Heim. 1982. *The Semantics of Definite and Indefinite Noun Phrases*. Ph.D. thesis, University of Massachusetts.

Irene Heim. 1983. File change semantics and the familiarity theory of definiteness. In *Meaning, Use, and Interpretation of Language*. De Gruyter.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Association for Computational Linguistics (ACL)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Theo M. Janssen and Barbara H. Partee. 1997. Compositionality. In *Handbook of logic and language*. Elsevier.

Hans Kamp. 1981. A theory of truth and semantic representation. In *Formal Methods in the Study of Language*. Mathematical Centre, Amsterdam.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Najoung Kim, Tal Linzen, and Paul Smolensky. 2022. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models. *arXiv preprint arXiv:2212.10769*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Association for Computational Linguistics (ACL)*.

James D. McCawley. 1968. Lexical insertion in a transformational grammar without deep structure. In *Papers from the Fourth Meeting of the Chicago Linguistic Society*.

Richard Montague and Richmond H. Thomason. 1974. Formal philosophy: selected papers of Richard Montague. *Erkenntnis*.

Santiago Ontanon, Joshua Ainslie, Zachary Fisher, and Vaclav Cvicek. 2022. Making transformers solve compositional tasks. In *Association for Computational Linguistics (ACL)*.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Terence Parsons. 1990. Events in the semantics of english: A study in subatomic semantics. *MIT press Cambridge*.

Barbara H. Partee. 1984. Compositionality. In *Varieties of Formal Semantics*. Wiley-Blackwell.

Arkil Patel, Satwik Bhattamishra, Phil Blunsom, and Navin Goyal. 2022. Revisiting the compositional generalization abilities of neural sequence models. In *Association for Computational Linguistics (ACL)*.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022. Improving compositional generalization with latent structure and data augmentation. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *The Journal of Machine Learning Research (JMLR)*.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. A benchmark for systematic generalization in grounded language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*.

Ankur Sikarwar, Arkil Patel, and Navin Goyal. 2022. When can transformers ground and compose: Insights from compositional generalization benchmarks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Koustuv Sinha, Jon Gauthier, Aaron Mueller, Kanishka Misra, Keren Fuentes, Roger Levy, and Adina Williams. 2023. Language model acceptability judgements are not always robust to context. In *Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bailin Wang, Ivan Titov, Jacob Andreas, and Yoon Kim. 2022. Hierarchical phrase-based sequence-to-sequence learning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pia Weißenhorn, Lucia Donatelli, and Alexander Koller. 2022. Compositional generalization with a broad-coverage semantic parser. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*.

Zhengxuan Wu, Elisa Kreiss, Desmond Ong, and Christopher Potts. 2021. ReaSCAN: Compositional reasoning in language grounding. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yuekun Yao and Alexander Koller. 2022. Structural generalization is hard for sequence-to-sequence models. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Hao Zheng and Mirella Lapata. 2022. Disentangled sequence to sequence learning for compositional generalization. In *Association for Computational Linguistics (ACL)*.