SyGNS: A Systematic Generalization Testbed Based on Natural Language Semantics

Hitomi Yanaka[1,2], **Koji Mineshima**[3], **and Kentaro Inui**[4,2]
[1]The University of Tokyo, [2]RIKEN, [3]Keio University, [4]Tohoku University
`hyanaka@is.s.u-tokyo.ac.jp`, `minesima@abelard.flet.keio.ac.jp`, `inui@ecei.tohoku.ac.jp`

Abstract

Recently, deep neural networks (DNNs) have achieved great success in semantically challenging NLP tasks, yet it remains unclear whether DNN models can capture *compositional* meanings, those aspects of meaning that have been long studied in formal semantics. To investigate this issue, we propose a Systematic Generalization testbed based on Natural language Semantics (SyGNS), whose challenge is to map natural language sentences to multiple forms of scoped meaning representations, designed to account for various semantic phenomena. Using SyGNS, we test whether neural networks can systematically parse sentences involving novel combinations of logical expressions such as quantifiers and negation. Experiments show that Transformer and GRU models can generalize to unseen combinations of quantifiers, negations, and modifiers that are similar to given training instances in form, but not to the others. We also find that the generalization performance to unseen combinations is better when the form of meaning representations is simpler. The data and code for SyGNS are publicly available at `https://github.com/verypluming/SyGNS`.

1 Introduction

Deep neural networks (DNNs) have shown impressive performance in various language understanding tasks (Wang et al., 2019a,b, i.a.), including semantically challenging tasks such as Natural Language Inference (NLI; Dagan et al., 2013; Bowman et al., 2015). However, a number of studies to probe DNN models with various NLI datasets (Naik et al., 2018; Dasgupta et al., 2018; Yanaka et al., 2019; Kim et al., 2019; Richardson et al., 2020; Saha et al., 2020; Geiger et al., 2020) have reported that current DNN models have some limitations to generalize to diverse semantic phenomena, and it is still not clear whether DNN models obtain the ability to capture *compositional* aspects of meaning in natural language.

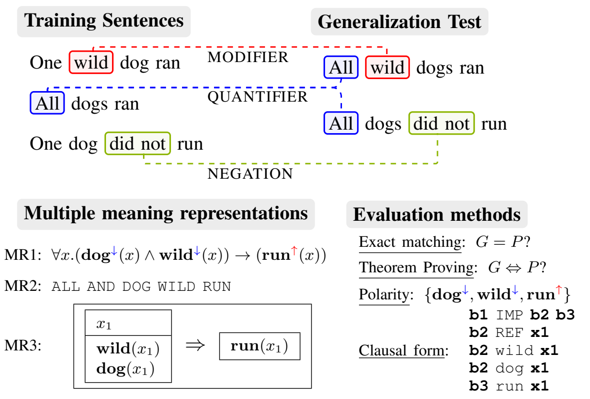

Training Sentences
Generalization Test
One wild dog ran
MODIFIER
All wild dogs ran
All dogs ran
QUANTIFIER
All dogs did not run
One dog did not run
NEGATION
Multiple meaning representations
Evaluation methods
MR1: ∀x.(dog↓(x) ∧ wild↓(x)) → (run↑(x))
MR2: ALL AND DOG WILD RUN
MR3: x1 wild(x1) dog(x1) ⇒ run(x1)
Exact matching: G = P?
Theorem Proving: G ⇔ P?
Polarity: {dog↓, wild↓, run↑}
Clausal form: b1 IMP b2 b3 / b2 REF x1 / b2 wild x1 / b2 dog x1 / b3 run x1

Figure 1: Illustration of our evaluation protocol using SyGNS. The goal is to map English sentences to meaning representations. The generalization test evaluates novel combinations of operations (modifier, quantifier, negation) in the training set. We use multiple meaning representations and evaluation methods.

There are two issues to consider here. First, recent analyses (Talmor and Berant, 2019; Liu et al., 2019; McCoy et al., 2019) have pointed out that the standard paradigm for evaluation, where a test set is drawn from the same distribution as the training set, does not always indicate that the model has obtained the intended generalization ability for language understanding. Second, the NLI task of predicting the relationship between a premise sentence and an associated hypothesis without asking their semantic interpretation tends to be *black-box*, in that it is often difficult to isolate the reasons why models make incorrect predictions (Bos, 2008).

To address these issues, we propose SyGNS (pronounced as *signs*), a Systematic Generalization testbed based on Natural language Semantics. The goal is to map English sentences to various meaning representations, so it can be taken as a sequence-to-sequence semantic parsing task.

Figure 1 illustrates our evaluation protocol using SyGNS. To address the first issue above, we probe the generalization capability of DNN models on two out-of-distribution tests: *systematicity* (Section 3.1) and *productivity* (Section 3.2), two concepts treated as hallmarks of human cognitive capacities in cognitive sciences (Fodor and Pylyshyn, 1988; Calvo and Symons, 2014). We use a train-test split controlled by each target concept and train models with a minimally sized training set (Basic set) involving primitive patterns composed of semantic phenomena such as quantifiers, modifiers, and negation. If a model learns different properties of each semantic phenomenon from the Basic set, it should be able to parse a sentence with novel combination patterns. Otherwise, a model has to memorize an exponential number of combinations of linguistic expressions.

To address the second issue, we use multiple forms of meaning representations developed in formal semantics (Montague, 1973; Heim and Kratzer, 1998; Jacobson, 2014) and their respective evaluation methods. We use three *scoped* meaning representation forms, each of which preserves the same semantic information (Section 3.3). In formal semantics, it is generally assumed that scoped meaning representations are standard forms for handling diverse semantic phenomena such as quantification and negation. Scoped meaning representations also enable us to evaluate the compositional generalization ability of the models to capture semantic phenomena in a more fine-grained way. By decomposing an output meaning representation into constituents (e.g., words) in accordance with its structure, we can compute the matching ratio between the output representation and the gold standard representation. Evaluating the models on multiple meaning representation forms also allows us to explore whether the performance depends on the complexity of the representation forms.

This paper provides three main contributions. First, we develop the SyGNS testbed to test model ability to systematically transform sentences involving linguistic phenomena into multiple forms of scoped meaning representations. The data and code for SyGNS are publicly available at `https://github.com/verypluming/SyGNS`. Second, we use SyGNS to analyze the systematic generalization capacity of two standard DNN models: Gated Recurrent Unit (GRU) and Transformer. Experiments show that these models can generalize to unseen combinations of quantifiers, negations, and modifiers to some extent. However, the generalization ability is limited to the combinations whose forms are similar to those of the training instances. In addition, the models struggle with parsing sentences involving nested clauses. We also show that the extent of generalization depends on the choice of primitive patterns and representation forms.

2 Related Work

The question of whether neural networks obtain the systematic generalization capacity has long been discussed (Fodor and Pylyshyn, 1988; Marcus, 2003; Baroni, 2020). Recently, empirical studies using NLI tasks have revisited this question, showing that current models learn undesired biases (Glockner et al., 2018; Poliak et al., 2018; Tsuchiya, 2018; Geva et al., 2019; Liu et al., 2019) and heuristics (McCoy et al., 2019), and fail to consistently learn various inference types (Rozen et al., 2019; Nie et al., 2019; Yanaka et al., 2019; Richardson et al., 2020; Joshi et al., 2020). In particular, previous works (Goodwin et al., 2020; Yanaka et al., 2020; Geiger et al., 2020; Yanaka et al., 2021) have examined whether models learn the systematicity of NLI on monotonicity and veridicality. While this line of work has shown certain limitations of model generalization capacity, it is often difficult to figure out why the NLI model fails and how to improve it, partly because NLI tasks depend on multiple factors, including semantic interpretation of target phenomena and acquisition of background knowledge. By focusing on semantic parsing rather than NLI, one can probe to what extent models systematically interpret the meaning of sentences according to their structures and the meanings of their constituents.

Meanwhile, datasets for analysing the compositional generalization ability of DNN models in semantic parsing have been proposed, including SCAN (Lake and Baroni, 2017; Baroni, 2020), CLUTRR (Sinha et al., 2019), and CFQ (Keysers et al., 2020). For example, the SCAN task is to investigate whether models trained with a set of primitive instructions (e.g., *jump* → `JUMP`) and modifiers (e.g., *walk twice* → `WALK WALK`) generalize to new combinations of primitives (e.g., *jump twice* → `JUMP JUMP`). However, these datasets deal with artificial languages, where the variation of linguistic expressions is limited, so it is not clear to what extent the models systematically interpret various semantic phenomena in natural language, such as quantification and negation.

Regarding the generalization capacity of DNN models in natural language, previous studies have focused on syntactic and morphological generalization capacities such as subject-verb agreement tasks (Linzen et al., 2016; Gulordava et al., 2018; Marvin and Linzen, 2018, i.a.). Perhaps closest to our work is the COGS task (Kim and Linzen, 2020) for probing the generalization capacity of semantic parsing in a synthetic natural language fragment. For instance, the task is to see whether models trained to parse sentences where some lexical items only appear in subject position (e.g., *John ate the meat*) can generalize to structurally related sentences where these items appear in object position (e.g., *The kid liked John*). In contrast to this work, our focus is more on semantic parsing of sentences with logical and semantic phenomena that require scoped meaning representations. Our study also improves previous work on the compositional generalization capacity in semantic parsing in that we compare three types of meaning representations and evaluate them at multiple levels, including logical entailment, polarity assignment, and partial clause matching (Section 3.3).

3 Overview of SyGNS

We use two evaluation concepts to assess the systematic capability of models: systematicity (Section 3.1) and productivity (Section 3.2). In evaluating these two concepts, we use synthesized pairs of sentences and their meaning representations to control a train-test split (Section 3.4). The main idea is to analyze models trained with a minimum size of a training set (Basic set) involving primitive patterns composed of various semantic phenomena; if a model systematically learns primitive combination patterns in the Basic set, it should parse a new sentence with different combination patterns. We target three types of scoped meaning representations and use their respective evaluation methods, according to the function and structure of each representation form (Section 3.3).

3.1 Systematicity

Table 1 illustrates how we test systematicity, i.e., the capacity to interpret novel combinations of primitive semantic phenomena. We generate Basic set 1 by combining various quantifiers with sentences without modifiers. We also generate Basic set 2 by setting an arbitrary quantifier (e.g., *one*) to a primitive quantifier and combining it with various types of modifiers. We then evaluate whether models trained with Basic sets 1 and 2 can parse sentences involving unseen combinations of quantifiers and modifiers. We also test the combination of quantifiers and negation in the same manner; the detail is given in Appendix D.

Pattern		Sentence
	Train	
Primitive quantifier		**One** tiger ran
Basic 1	EXI	**A** tiger ran
	NUM	**Two** tigers ran
	UNI	**Every** tiger ran
Basic 2	ADJ	**One** *small* tiger ran
	ADV	**One** tiger ran *quickly*
	CON	**One** tiger ran *or came*
	Test	
EXI+ADJ		**A** *small* tiger ran
NUM+ADV		**Two** tigers ran *quickly*
UNI+CON		**Every** tiger ran *or came*

Table 1: Training and test instances for systematicity.

To provide a controlled setup, we use three quantifier types: existential quantifiers (EXI), numerals (NUM), and universal quantifiers (UNI). Each type has two patterns: *one* and *a* for EXI, *two* and *three* for NUM, and *all* and *every* for UNI. We consider three settings where the primitive quantifier is set to the type EXI, NUM, or UNI.

For modifiers, we distinguish three types — adjectives (ADJ), adverbs (ADV), and logical connectives (CON; conjunction *and*, disjunction *or*) — and ten patterns for each. Note that each modifier type differs in its position; an adjective appears inside a noun phrase (e.g., one *small* tiger), while an adverb (e.g., *quickly*) and a coordinated phrase with a logical connective (e.g., *or came*) appears at the end of a sentence. Although Table 1 only shows the pattern generated by the primitive quantifier *one* and the noun *tiger*, the noun can be replaced with ten other nouns (e.g., *dog*, *cat*, etc.) in each setting. See Appendix A for more details on the fragment of English considered here.

3.2 Productivity

Productivity refers to the capacity to interpret an indefinite number of sentences with recursive operations. To test productivity, we use embedded relative clauses, which interact with quantifiers to gen-

Pattern		Sentence
Train (Basic 1: depth 0, Basic 2: depth 1)		
Basic 1	Non	Two dogs loved Ann
Basic 2	Per	Bob liked a bear [that chased all polite cats]
	Cen	Two dogs [that all cats kicked] loved Ann
Test (examples: depth 2)		
Per+Per		Bob liked a bear [that chased all polite cats [that loves Ann]]
Per+Cen		Two dogs [that a bear [that chased all polite cats] kicked] loved Ann

Table 2: Training and test instances for productivity.

erate logically complex sentences. Table 2 shows examples. We provide two Basic sets; Basic set 1 consists of sentences without embedded clauses (Non) and Basic set 2 consists of sentences with a single embedded clause, which we call sentences with depth one. We then test whether models trained with Basic sets 1 and 2 can parse a sentence involving deeper embedded clauses, i.e., sentences whose depth is two or more. As Table 2 shows, we consider both peripheral-embedding (Per) and center-embedding (Cen) clauses.

3.3 Meaning representation and evaluation

Overview To evaluate generalization capacity in semantic parsing at multiple levels, we use three types of scoped meaning representations: (i) First-Order Logic (FOL) formulas, (ii) Discourse Representation Structures (DRSs; Kamp and Reyle, 1993), and (iii) Variable-Free (VF) formulas (Baader et al., 2003; Prat-Hartmann and Moss, 2009). DRSs can be converted to *clausal forms* (van Noord et al., 2018a) for evaluation. For instance, the sentence (1) is mapped to the FOL formula in (2), the DRS in (3a), its clausal form in (3b), and the VF formula in (4).

(1) One white dog did not run.

(2) $\exists x_1.(\mathbf{white}(x_1) \wedge \mathbf{dog}(x_1) \wedge \neg\mathbf{run}(x_1))$

(3) a. $\begin{array}{|l|}\hline x_1 \\ \hline \mathbf{white}(x_1) \\ \mathbf{dog}(x_1) \\ \neg\ \boxed{\mathbf{run}(x_1)} \\ \hline \end{array}$ b.
```
b1 REF x1
b1 white x1
b1 dog x1
b1 NOT b2
b2 run x1
```

(4) `EXIST AND WHITE DOG NOT RUN`

Using these multiple forms enables us to analyze whether the difficulty in semantic generalization depends on the format of meaning representations.

Previous studies for probing generalization capacity in semantic parsing (e.g., Lake and Baroni, 2017; Sinha et al., 2019; Keysers et al., 2020; Kim and Linzen, 2020) use a fixed type of meaning representation, with its evaluation method limited to the exact-match percentage, where an output is considered correct only if it exactly matches the gold standard. However, this does not properly assess whether models capture the structure and function of meaning representation. First, exact matching does not directly take into account whether two meanings are logically equivalent (Blackburn and Bos, 2005): for instance, schematically two formulas $A \wedge B$ and $B \wedge A$ are different in form but have the same meaning. Relatedly, scoped meaning representations for natural languages can be made complex by including parentheses and variable renaming mechanism (the so-called α-conversion in λ-calculus). For instance, we want to identify two formulas which only differ in variable naming, e.g., $\exists x_1.F(x_1)$ and $\exists x_2.F(x_2)$. It is desirable to compare exact matching with alternative evaluation methods, and to consider alternative meaning representations that avoid these problems. Having this background in mind, below we will describe each type of meaning representation in detail.

FOL formula FOL formulas are standard forms in formal and computational semantics (Blackburn and Bos, 2005; Jurafsky and Martin, 2009), where content words such as nouns and verbs are represented as predicates, and function words such as quantifiers, negation, and connectives are represented as logical operators with scope relations (cf. the example in (2)). To address the issue on evaluation, we consider two ways of evaluating FOL formulas in addition to exact matching: (i) automated theorem proving (ATP) and (ii) monotonicity-based polarity assignment.

First, FOL formulas can be evaluated by checking the logical entailment relationships that directly consider whether two formulas are logically equivalent. Thus we evaluate predicted FOL formulas by using ATP. We check whether a gold formula G entails prediction P and vice versa, using an off-the-shelf FOL theorem prover[1]. To see the logical relationship between G and P, we measure the accuracy for unidirectional and bidirectional

[1]We use a state-of-the-art FOL prover Vampire available at `https://github.com/vprover/vampire`

One dog$^\uparrow$ ran$^\uparrow$:	$\exists x.(\mathbf{dog}^\uparrow(x) \wedge \mathbf{run}^\uparrow(x))$
All dogs$^\downarrow$ ran$^\uparrow$:	$\forall x.(\mathbf{dog}^\downarrow(x) \rightarrow \mathbf{run}^\uparrow(x))$
All dogs$^\downarrow$ did not run$^\downarrow$:	$\forall x.(\mathbf{dog}^\downarrow(x) \rightarrow \neg\mathbf{run}^\downarrow(x))$

Table 3: Examples of monotonicity-based polarity assignments for FOL formulas.

entailment: $G \Rightarrow P$, $G \Leftarrow P$, and $G \Leftrightarrow P$.

Second, the polarity of each content word appearing in a sentence can be extracted from the FOL formula using its monotonicity property (van Benthem, 1986; MacCartney and Manning, 2007). This enables us to analyze whether models can correctly capture entailment relations triggered by quantifier and negation scopes. Table 3 shows some examples of monotonicity-based polarity assignments. For example, existential quantifiers such as *one* are upward monotone (shown as $\uparrow$) with respect to the subject NP and the VP, because they can be substituted with their hypernyms (e.g., *One dog ran* $\Rightarrow$ *One animal moved*). These polarities can be extracted from the FOL formula because $\exists$ and $\wedge$ are upward monotone operators in FOL. Universal quantifiers such as *all* are downward monotone (shown as $\downarrow$) with respect to the subject NP and upward monotone with respect to the VP. Expressions in downward monotone position can be substituted with their hyponymous expressions (e.g., *All dogs ran* $\Rightarrow$ *All white dogs ran*). The polarity can be reversed by embedding another downward entailing context like negation, so the polarity of *run* in the third case in Table 3 is flipped to downward monotone.[2] For evaluation based on monotonicity, we extract a polarity for each content word in a gold formula and a prediction and calculate the F-score for each monotonicity direction (upward and downward).

DRS A DRS is a form of scoped meaning representations proposed in Discourse Representation Theory, a well-studied formalism in formal semantics (Kamp and Reyle, 1993; Asher, 1993; Muskens, 1996). By translating a box notation as in (3a) to the clausal form as in (3b), one can evaluate DRSs by COUNTER[3], which is a standard tool for evaluating neural DRS parsers (Liu et al., 2018; van Noord et al., 2018b). COUNTER searches for the best variable mapping between predicted DRS clauses and gold DRS clauses and calculates an F-score over matching clause, which is similar to SMATCH (Cai and Knight, 2013), an evaluation metric designed for Abstract Meaning Representation (AMR; Banarescu et al., 2013). COUNTER alleviates the process of variable renaming and correctly evaluates the cases where the order of clauses is different from that of gold answers.

VF formula FOL formulas in our fragment have logically equivalent forms in a variable-free format, which does not contain parentheses nor variables as in the example (4). Our format is similar to a variable-free form in Description Logic (Baader et al., 2003) and Natural Logic (Prat-Hartmann and Moss, 2009). VF formulas alleviate the problem of parentheses and variable renaming, while preserving semantic information (cf. Wang et al., 2017). Due to the equivalence with FOL formulas, it is possible to extract polarities from VF formulas. See Appendix A for more examples of VF formulas.

3.4 Data generation

To provide synthesized data, we generate sentences using a context-free grammar (CFG) associated with semantic composition rules in the standard λ-calculus (see Appendix A for details). Each sentence is mapped to an FOL formula and VF formula by using the semantic composition rules specified in the CFG. DRSs are converted from the generated FOL formulas using the standard mapping (Kamp and Reyle, 1993). To generate controlled fragments for each train-test split, the CFG rules automatically annotate the types of semantic phenomena involved in sentences generated. We annotate seven types: the positions of quantifiers (subject or object), negation, adjectives, adverbs, conjunction, disjunction, and embedded clause types (peripheral or center embedding).

To test systematicity, we generate sentences using the CFG, randomly select 50,000 examples, and then split them into 12,000 training examples and 38,000 test examples according to a primitive quantifier. To test productivity, we apply up to four recursive rules and randomly select 20,000 examples for each depth.

4 Experiments and Analysis

Using SyGNS, we test the performance of Gated Recurrent Unit (GRU; Cho et al., 2014) and Transformer (Vaswani et al., 2017) in an encoder-decoder setup. These are widely used models

[2] We follow the surface order of NPs and take it that the subject NP always take scope over the VP.

[3] https://github.com/RikVN/DRS_parsing

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: existential quantifier *one*								
EXI	96.1	99.5	99.9	99.7	99.9	99.8	99.9	100.0
NUM	7.6	12.7	86.0	37.0	18.1	96.9	99.7	20.7
UNI	3.1	4.4	56.8	39.5	8.3	2.2	74.2	17.7
Valid	98.2	99.7	100.0	99.6	100.0	100.0	100.0	100.0
primitive quantifier: numeral *two*								
EXI	11.6	42.1	91.4	45.3	34.0	84.5	98.3	10.5
NUM	59.5	83.6	98.7	42.8	99.9	97.4	99.8	80.9
UNI	2.5	1.8	68.6	39.2	0.0	0.1	72.3	90.9
Valid	84.3	99.7	100.0	98.9	100.0	100.0	100.0	100.0
primitive quantifier: universal quantifier *every*								
EXI	1.6	0.3	43.8	61.3	2.1	0.2	70.8	20.8
NUM	1.4	0.3	75.9	69.3	0.1	0.1	76.8	99.7
UNI	33.8	96.5	99.4	100.0	100.0	100.0	100.0	99.9
Valid	93.4	100.0	100.0	99.3	100.0	100.0	100.0	99.9
primitive quantifiers: *one*, *two*, *every*								
EXI	99.7	99.0	100.0	100.0	100.0	100.0	100.0	100.0
NUM	91.2	96.4	99.2	99.3	100.0	100.0	100.0	100.0
UNI	95.7	97.6	99.4	100.0	99.9	100.0	100.0	100.0
Valid	98.4	100.0	100.0	99.3	100.0	100.0	100.0	100.0

Table 4: Accuracy by quantifier type. "DRS (cnt)" columns show the accuracy of predicted DRSs by COUNTER, and "Valid" rows show the validation accuracy. Each accuracy is measured by exact matching, except for "DRS (cnt)" columns.

to perform well on hierarchical generalization tasks (McCoy et al., 2018; Russin et al., 2020).

4.1 Experimental setup

In all experiments, we trained each model for 25 epochs with early stopping (patience = 3). We performed five runs and reported their average accuracies. The input sentence is represented as a sequence of words, using spaces as a separator. The maximum input and output sequence length is set to the length of a sequence with maximum depths of embedded clauses. We set the dropout probability to 0.1 on the output and used a batch size of 128 and an embedding size of 256. Since incorporating pre-training would make it hard to distinguish whether the models' ability to perform semantic parsing comes from training data or from pre-training data, we did not use any pre-training.

For the GRU, we used a single-layer encoder-decoder with global attention and a dot-product score function. Since a previous work (Kim and Linzen, 2020) reported that unidirectional models are more robust regarding sentence structures than bi-directional models, we selected a unidirectional GRU encoder. For the Transformer, we used a three-layer encoder-decoder, a model size of 512, and a hidden size of 256. The number of model parameters was 10M, respectively. See Appendix B for additional training details.

4.2 Results on systematicity

Generalization on quantifiers Table 4 shows the accuracy by quantifier type. When the existential quantifier *one* was the primitive quantifier, the accuracy on the problems involving existential quantifiers, which have the same type as the primitive quantifier, was nearly perfect. Similarly, when the universal quantifier *every* was the primitive quantifier, the accuracy on the problems involving universal quantifiers was much better than that on the problems involving other quantifier types. These results indicate that models can easily generalize to problems involving quantifiers of the same type as the primitive quantifier, while the models struggle with generalizing to the others. We also experimented with larger models and observed the same trend (see Appendix C). The extent of generalization varies according to the primitive quantifier type and meaning representation forms. For example, when the primitive quantifier is the numeral expression *two*, models generalize to problems of VF formulas involving universal quantifiers. This can be explained by the fact that VF formulas involving universal quantifiers like (5b) have a similar form to those involving numerals as in (6b), whereas FOL formulas involving universal quantifiers have a different form from those involving numerals as in (5c) and (6c).

(5) a. All small cats chased Bob

b. `ALL AND CAT SMALL EXIST BOB CHASE`

c. $\forall x_1.(\mathbf{cat}(x_1) \wedge \mathbf{small}(x_1) \rightarrow \exists x_2.(\mathbf{bob}(x_2) \wedge \mathbf{chase}(x_1, x_2)))$

(6) a. Two small cats chased Bob

b. `TWO AND CAT SMALL EXIST BOB CHASE`

c. $\exists x_1.(\mathbf{two}(x_1) \wedge \mathbf{cat}(x_1) \wedge \mathbf{small}(x_1) \wedge \exists x_2.(\mathbf{bob}(x_2) \wedge \mathbf{chase}(x_1, x_2)))$

We also check the performance when three quantifiers *one*, *two*, and *every* are set as primitive quantifiers. This setting is easier than that for the systematicity in Table 1, since the models are exposed to combination patterns of all the quantifier types and all the modifier types. In this setting, the models achieved almost perfect performance on the test set involving non-primitive quantifiers (*a*, *three*, *all*).

Generalization on modifiers Table 5 shows the accuracy by modifier type where *one* is set to the primitive quantifier (see Appendix C for the results where other quantifier types are set to the primitive quantifier). No matter which quantifier is set as the primitive quantifier, the accuracy for problems involving logical connectives or adverbs is better than those involving adjectives. As in (8), an adjective is placed between a quantifier and a noun, so the position of the noun *dog* with respect to the quantifier *every* in the test set changes from the example in the training (Basic) set in (7). In contrast, adverbs and logical connectives are placed at the end of a sentence, so the position of the noun does not change from the training set, as in (9). This suggests that models can more easily generalize in problems involving unseen combinations of quantifiers and modifiers where the position of the noun is the same between the training and test sets.

(7)	Every $\underline{\text{dog}}$ ran	Train (Basic set)
(8)	Every $\underline{\text{large dog}}$ ran	Test (ADJ)
(9)	Every $\underline{\text{dog}}$ ran and cried	Test (CON)

Table 5 also shows that the accuracy is nearly the same regardless of the existence of negation. Basic set contains examples involving negation, and this indicates that the existence of complex phenomena like negation does not affect generalization performance of models on modifiers so long as such phenomena are included in the training set.

Meaning representation comparison Comparing forms of meaning representations, accuracy by exact matching is highest in the order of VF formulas, DRS clausal forms, and FOL formulas. This indicates that models can more easily generalize to unseen combinations where the form of meaning representation is simpler; VF formulas do not contain parentheses nor variables, DRS clausal forms contain variables but not parentheses, and FOL formulas contain both parentheses and variables.

4.3 Model comparison

Regarding the generalization capacity of models for decoding meaning representations, the left two figures in Figure 2 show learning curves on FOL prediction tasks by quantifier type. While GRU achieved perfect performance on the same quantifier type as the primitive quantifier, where the number of training data is 2,500, Transformer achieved the same performance when the number of training data is 8,000. The right two figures in Figure 2 show learning curves by modifier type. The GRU accuracy is unstable even when the number of training examples is maximal. In contrast, the Transformer accuracy is stable when the number of training data exceeds 8,000. These results indicate that GRU generalizes to unseen combinations of quantifiers and modifiers with a smaller training set than can Transformer, while the Transformer performance is more stable than that of GRU.

ATP-based evaluation Table 6 shows the ATP-based evaluation results on FOL formulas. For combinations involving numerals, both GRU and Transformer achieve high accuracies for $G \Rightarrow P$ entailments but low accuracies for $G \Leftarrow P$ entailments. Since both models fail to output the formulas corresponding to modifiers, they fail to prove $G \Leftarrow P$ entailments. Regarding combinations involving universal quantifiers, the GRU accuracy for both $G \Rightarrow P$ and $G \Leftarrow P$ entailments is low, and the Transformer accuracy for $G \Leftarrow P$ entailments is much higher than that for $G \Rightarrow P$ entailments. As indicated by examples shown in Table 7,

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
ADJ	18.9	20.1	78.1	42.3	26.8	59.1	91.3	27.6
ADJ+NEG	18.8	20.2	80.5	39.7	23.1	59.5	93.6	27.5
ADV	20.1	47.7	87.5	58.4	36.2	67.3	97.6	50.7
ADV+NEG	26.9	62.7	92.7	67.2	50.7	69.4	97.3	62.1
CON	28.9	58.3	84.7	72.9	54.3	66.8	88.3	65.9
CON+NEG	33.6	62.8	86.6	74.9	60.1	65.1	89.9	69.1
Valid	98.2	99.7	100.0	99.6	100.0	100.0	100.0	100.0

Table 5: Accuracy by modifier type (primitive quantifier: existential quantifier *one*). +NEG indicates problems involving negation. Each accuracy is measured by exact matching, except for "DRS (cnt)" columns.

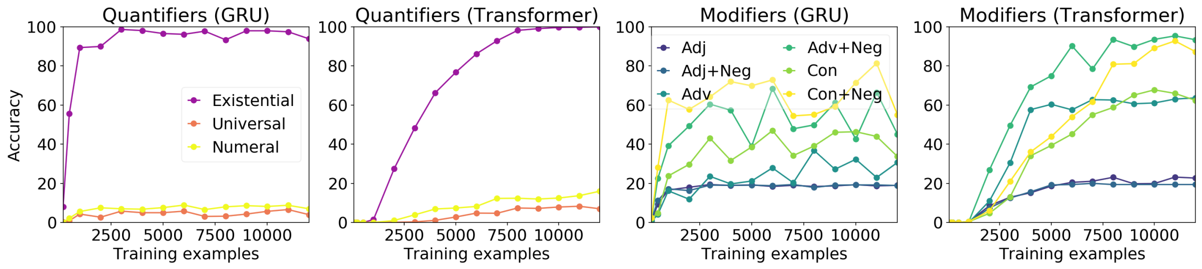


Figure 2: Learning curves on FOL formula generalization tasks (primitive quantifier: *one*).

Test	GRU			Transformer		
	$G \Rightarrow P$	$G \Leftarrow P$	$G \Leftrightarrow P$	$G \Rightarrow P$	$G \Leftarrow P$	$G \Leftrightarrow P$
EXI	99.8	100.0	99.8	100.0	100.0	100.0
NUM	77.1	19.0	10.4	91.3	21.1	12.4
UNI	7.1	18.7	2.7	21.1	83.4	12.3

Table 6: ATP-based evaluation results on FOL formulas (primitive quantifier: *one*).

Input	Every wild cat escaped and ran
Gold	$\forall x.((\mathbf{cat}^{\downarrow}(x) \wedge \mathbf{wild}^{\downarrow}(x)) \rightarrow (\mathbf{escape}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x)))$
GRU	$\forall x.(\mathbf{cat}^{\downarrow}(x) \rightarrow (\mathbf{escape}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x)))$
Trans	$\forall x.(\mathbf{cat}^{\downarrow}(x) \rightarrow \mathbf{wild}^{\uparrow}(x) \wedge (\mathbf{escape}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x)))$

Table 7: Examples of typical errors.

Test	GRU		Transformer	
	Up	Down	Up	Down
EXI	99.9	100.0	100.0	100.0
NUM	84.8	96.8	88.1	97.5
UNI	90.9	40.7	94.9	73.4

Table 8: Monotonicity-based evaluation results on FOL (primitive quantifier: *one*). "Up" and "Down" columns show upward and downward accuracy, respectively.

GRU tends to fail to output the formula for a modifier, e.g., $\mathbf{wild}(x)$ in this case, while Transformer fails to correctly output the position of the implication ($\rightarrow$). The ATP-based evaluation results reflect such differences between error trends of models in problems involving different forms of quantifiers.

Monotonicity-based evaluation Table 8 shows accuracies for the monotonicity-based polarity assignment evaluation on FOL formulas. The accuracies were higher than those using exact matching (cf. Table 4). Monotonicity-based evaluation captures the polarities assigned to content words even for the problems that exact-matching judges as incorrect because of the differences in form. Table 7 shows examples of predicted polarity assignments. Here both models predicted correct polarities for three content words, $\mathbf{cat}^{\downarrow}$, $\mathbf{escape}^{\uparrow}$, $\mathbf{run}^{\uparrow}$. Exact-matching cannot take into account such partial matching. The downward monotone accuracies for problems involving universal quantifiers are low (40.7 and 73.4 in Table 8). In Table 7, both models failed to predict the downward monotonicity of $\mathbf{wild}^{\downarrow}$. The results indicate that both models struggle with capturing the scope of universal quantifiers. Appendix C shows the evaluation results on the polarities of VF formulas.

4.4 Results on productivity

Table 9 shows very low generalization accuracy for both GRU and Transformer at unseen depths. Although the evaluation results using COUNTER on DRS prediction tasks is much higher than those by exact matching, this is due to the fact that COUNTER uses partial matching; both models tended to correctly predict the clauses in the subject NP that are positioned at the beginning of the

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
Dep2	0.36	0.41	55.5	0.32	0.61	0.61	64.6	0.58
Dep3	0.04	0.07	45.6	0.04	0.11	0.12	46.6	0.12
Dep4	0.00	0.01	38.0	0.00	0.02	0.02	37.6	0.02
Valid	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 9: Accuracy for productivity. "Dep" rows show embedding depths, "DRS (cnt)" columns show accuracy of predicted DRSs by COUNTER, and "Valid" row shows the validation accuracy. Each accuracy is measured by exact matching, except for "DRS (cnt)" columns.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
Dep1	22.1	21.9	81.9	48.9	96.6	77.1	96.5	97.5
Dep2	3.52	3.89	59.1	12.3	76.3	54.6	90.5	85.4
Dep3	0.15	0.12	43.3	0.31	24.9	4.5	70.4	37.0
Dep4	0.08	0.15	37.7	0.46	4.4	1.6	60.3	5.57
Valid	94.3	94.9	100.0	96.0	97.6	98.1	100.0	97.8

Table 10: Evaluation results for systematicity involving embedding quantifiers. "Dep" rows show embedding depths.

sentence (see Appendix E for details).

We checked whether models can generalize to unseen combinations involving embedded clauses when the models are exposed to a part of training instances at each depth. We provide Basic set 1 involving non-embedding patterns like (10), where **Q** can be replaced with any quantifier. This Basic set 1 exposes models to all quantifier patterns. We also expose models to Basic set 2 involving three primitive quantifiers (*one*, *two*, and *every*) at each embedding depth, like (11) and (12). We provide around 2,000 training instances at each depth. We then test models on a test set involving the other quantifiers (*a*, *three*, and *all*) at each embedding depth, like (13) and (14). If models can distinguish quantifier types during training, they can correctly compose meaning representations involving different combinations of multiple quantifiers. Note that this setting is easier than that for productivity in Table 2, in that models are exposed to some instances at each depth.

(10) **Q** dog(s) liked Bob

(11) **One** dog liked Bob [that loved **two** rats]

(12) **One** dog liked Bob [that loved **two** rats [that knew **every** pig]]

(13) **A** dog liked Bob [that loved **three** rats]

(14) **A** dog liked Bob [that loved **three** rats [that knew **all** pigs]]

Table 10 shows that both models partially generalize to the cases where the depth is 1 or 2. However, both models fail to generalize to the cases where the depth is 3 or more. This suggests that even if models are trained with some instances at each depth, the models fail to learn distinctions between different quantifier types and struggle with parsing sentences whose embedding depth is 3 or more.

5 Conclusion

We have introduced an analysis method using SyGNS, a testbed for diagnosing the systematic generalization capability of DNN models in semantic parsing. We found that GRU and Transformer generalized to unseen combinations of semantic phenomena whose meaning representations are similar in forms to those in a training set, while the models struggle with generalizing to the others. In addition, these models failed to generalize to cases involving nested clauses. Our analyses using multiple meaning representation and evaluation methods also revealed detailed behaviors of models. We believe that SyGNS serves as an effective testbed for investigating the ability to capture compositional meanings in natural language.

Acknowledgement

We thank the three anonymous reviewers for their helpful comments and suggestions. This work was partially supported by JSPS KAKENHI Grant Number JP20K19868.

References

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Springer.

Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

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*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Marco Baroni. 2020. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B*, 375(1791):20190307.

Johan van Benthem. 1986. *Essays in Logical Semantics*. Springer.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Center for the Study of Language and Information.

Johan Bos. 2008. Let's not argue about semantics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Paco Calvo and John Symons. 2014. *The Architecture of Cognition: Rethinking Fodor and Pylyshyn's Systematicity Challenge*. MIT Press.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. 2018. Evaluating compositionality in sentence embeddings. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 1596–1601.

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

Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 163–173, Online. Association for Computational Linguistics.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Emily Goodwin, Koustuv Sinha, and Timothy J. O'Donnell. 2020. Probing linguistic systematicity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1958–1969, Online. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell.

Pauline Jacobson. 2014. *Compositional Semantics: An Introduction to the Syntax/Semantics Interface*. Oxford University Press.

Pratik Joshi, Somak Aditya, Aalok Sathe, and Monojit Choudhury. 2020. TaxiNLI: Taking a ride up the NLU hill. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 41–55, Online. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Brenden M. Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of International Conference on Machine Learning (ICML)*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia. Association for Computational Linguistics.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2171–2179, Minneapolis, Minnesota. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Gary Marcus. 2003. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society, CogSci 2018, Madison, WI, USA, July 25-28, 2018*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius M. E. Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language*, pages 189–224. Reidel, Dordrecht. Reprinted in Richmond H. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, 247–270, 1974, New Haven: Yale University Press.

Reinhard Muskens. 1996. Combining Montague semantics and discourse representation. *Linguistics and philosophy*, 19(2):143–186.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yixin Nie, Yicheng Wang, and Mohit Bansal. 2019. Analyzing compositionality-sensitivity of NLI models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6867–6874.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Ian Prat-Hartmann and Lawrence S. Moss. 2009. Logics for the relational syllogistic. *The Review of Symbolic Logic*, 2(4):647–683.

Kyle Richardson, Hai Hu, Lawrence S. Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ohad Rozen, Vered Shwartz, Roee Aharoni, and Ido Dagan. 2019. Diversify your datasets: Analyzing generalization via controlled variance in adversarial datasets. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 196–205, Hong Kong, China. Association for Computational Linguistics.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.

Swarnadeep Saha, Yixin Nie, and Mohit Bansal. 2020. ConjNLI: Natural language inference over conjunctive sentences. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8240–8252, Online. Association for Computational Linguistics.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jan Van Eijck. 2005. Natural logic for natural language. In *International Tbilisi Symposium on Logic, Language, and Computation*, pages 216–230. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. 2017. Premise selection for theorem proving by deep graph embedding. In *Proceedings of Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2786–2796.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. 2020. Do neural models learn systematicity of monotonicity inference in natural language? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6105–6117, Online. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, and Kentaro Inui. 2021. Exploring transitivity in neural NLI models through veridicality. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 920–934.

A Data generation details

Table 12 shows a set of context-free grammar rules and semantic composition rules we use to generate a fragment of English annotated with meaning representations in the SyGNS dataset. Each grammar rule is associated with two kinds of semantic composition rules formulated in λ-calculus. One is for deriving first-order logic (FOL) formulas, and the other is for deriving variable-free (VF) formulas. For FOL, semantic composition runs in the standard Montagovian fashion where all NPs (including proper nouns) are treated as generalized quantifiers (Heim and Kratzer, 1998; Jacobson, 2014). From FOL formulas, we can extract the *polarity* of each content word using the monotonicity calculus (Van Eijck, 2005). Table 13 shows some examples of polarized FOL formulas. The derivation of VF formulas runs in two steps. To begin with, a sentence is mapped to a variable-free form by semantic composition rules. For instance, the sentence *a small dog did not swim* is mapped to a variable-free formula `EXIST(AND(SMALL,DOG),NOT(SWIM))` by the rules in Table 12. Second, since this form is in prefix notation, all brackets can be eliminated without causing ambiguity. This produces the resulting VF formula `EXIST AND SMALL DOG NOT SWIM`. Some other examples are shown in Table 13. DRSs are converted from FOL formulas in the standard way (Kamp and Reyle, 1993).

B Training details

We implemented the GRU model and the Transformer model using PyTorch. Both models were optimized using Adam (Kingma and Ba, 2015) at an initial learning rate of 0.0005. The hyperparameters (batch size, learning rate, number of epochs, hidden units, and dropout probability) were tuned by random search. In all experiments, we trained models on eight NVIDIA DGX-1 Tesla V100 GPUs. The runtime for training each model was about 1-4 hours, depending on the size of the training set. The order of training instances was shuffled for each model. We used 10% of the training set for a validation set.

C Detailed evaluation results

Effect of Model Size The results we report are from a model with 10M parameters. How does the number of parameters affect the systematic generalization performance of models? Table 11 shows the performance of three models of varying size (large: 27M, medium: 10M, small: 4M). The number of parameters did not have a large impact on the generalization performance; all runs of the models achieved higher than 90% accuracy on the validation set and the test set involving quantifiers of the same type as the primitive quantifier, while they did not work well on the test set involving the other types of quantifiers.

Test	GRU			Transformer		
	4M	10M	27M	4M	10M	27M
EXI	96.8	99.9	97.1	99.9	99.8	99.3
NUM	7.1	11.5	10.4	12.3	12.2	12.4
UNI	6.0	4.9	2.9	7.8	5.9	7.9
Valid	97.2	99.9	97.6	100.0	99.8	97.2

Table 11: The effect of model size on generalization performance (primitive quantifier: existential quantifier *one*, representation form: FOL).

Modifier type Table 14 shows all evaluation results by modifier types where *two* or *every* is set to the primitive quantifier. Regardless of primitive quantifier type, accuracies for problems involving logical connectives or adverbs were better than those for problems involving adjectives.

Monotonicity Table 15 shows all evaluation results of predicted FOL formulas and VF formulas based on monotonicity. We evaluate the precision, recall, and F-score for each monotonicity direction (upward and downward). Regardless of meaning representation forms, downward monotone accuracy on problems involving universal quantifiers is low. This indicates that both models struggle with learning the scope of universal quantifiers.

D Evaluation on systematicity of quantifiers and negation

We also analyze whether models can generalize to unseen combinations of quantifiers and negation. Here, we generate Basic set 1 by setting an arbitrary quantifier to a primitive quantifier and combining it with negation. As in (15b), we fix the primitive quantifier to the existential quantifier *one* and generate the negated sentence *One tiger did not run*. Next, as in (16a) and (16b), we generate Basic set 2 by combining a primitive term (e.g., *tiger*) with various quantifiers. If a model has the ability to systematically understand primitive combinations in Basic set, it can represent a new mean-

Grammar rules	Semantic composition rules: FOL	Semantic composition rules: VF
S $\rightarrow$ NP VP	$[\![\mathrm{S}]\!] = [\![\mathrm{NP}]\!]([\![\mathrm{VP}]\!])$	$[\![\mathrm{S}]\!] = [\![\mathrm{NP}]\!]([\![\mathrm{VP}]\!])$
S $\rightarrow$ NP *did not* VP	$[\![\mathrm{S}]\!] = [\![\mathrm{NP}]\!](\lambda x.\neg[\![\mathrm{VP}]\!](x))$	$[\![\mathrm{S}]\!] = [\![\mathrm{NP}]\!](\texttt{NOT}([\![\mathrm{VP}]\!]))$
NP $\rightarrow$ PN	$[\![\mathrm{NP}]\!] = [\![\mathrm{PN}]\!]$	$[\![\mathrm{NP}]\!] = [\![\mathrm{PN}]\!]$
NP $\rightarrow$ Q N	$[\![\mathrm{NP}]\!] = [\![\mathrm{Q}]\!]([\![\mathrm{N}]\!])$	$[\![\mathrm{NP}]\!] = [\![\mathrm{Q}]\!]([\![\mathrm{N}]\!])$
NP $\rightarrow$ Q ADJ N	$[\![\mathrm{NP}]\!] = [\![\mathrm{Q}]\!](\lambda x.([\![\mathrm{N}]\!](x) \wedge [\![\mathrm{ADJ}]\!](x)))$	$[\![\mathrm{NP}]\!] = [\![\mathrm{Q}]\!](\texttt{AND}([\![\mathrm{N}]\!], [\![\mathrm{ADJ}]\!]))$
NP $\rightarrow$ Q N $\overline{\mathrm{S}}$	$[\![\mathrm{NP}]\!] = [\![\mathrm{Q}]\!](\lambda x.([\![\mathrm{N}]\!](x) \wedge [\![\overline{\mathrm{S}}]\!](x)))$	$[\![\mathrm{NP}]\!] = [\![\mathrm{Q}]\!](\texttt{AND}([\![\mathrm{N}]\!], [\![\overline{\mathrm{S}}]\!]))$
VP $\rightarrow$ IV	$[\![\mathrm{VP}]\!] = [\![\mathrm{IV}]\!]$	$[\![\mathrm{VP}]\!] = [\![\mathrm{IV}]\!]$
VP $\rightarrow$ IV ADV	$[\![\mathrm{VP}]\!] = \lambda x.([\![\mathrm{IV}]\!](x) \wedge [\![\mathrm{ADV}]\!](x))$	$[\![\mathrm{VP}]\!] = \texttt{AND}([\![\mathrm{IV}]\!], [\![\mathrm{ADV}]\!]))$
VP $\rightarrow$ IV or IV$'$	$[\![\mathrm{VP}]\!] = \lambda x.([\![\mathrm{IV}]\!](x) \vee [\![\mathrm{IV}']\!](x))$	$[\![\mathrm{VP}]\!] = \texttt{OR}([\![\mathrm{IV}]\!], [\![\mathrm{IV}']\!]))$
VP $\rightarrow$ IV and IV$'$	$[\![\mathrm{VP}]\!] = \lambda x.([\![\mathrm{IV}]\!](x) \wedge [\![\mathrm{IV}']\!](x))$	$[\![\mathrm{VP}]\!] = \texttt{AND}([\![\mathrm{IV}]\!], [\![\mathrm{IV}']\!]))$
VP $\rightarrow$ TV NP	$[\![\mathrm{VP}]\!] = \lambda x.[\![\mathrm{NP}]\!](\lambda y.[\![\mathrm{TV}]\!](x, y))$	$[\![\mathrm{VP}]\!] = [\![\mathrm{NP}]\!]([\![\mathrm{TV}]\!])$
$\overline{\mathrm{S}}$ $\rightarrow$ *that* VP	$[\![\overline{\mathrm{S}}]\!] = [\![\mathrm{VP}]\!]$	$[\![\overline{\mathrm{S}}]\!] = [\![\mathrm{VP}]\!]$
$\overline{\mathrm{S}}$ $\rightarrow$ *that did not* VP	$[\![\overline{\mathrm{S}}]\!] = \lambda x.\neg[\![\mathrm{VP}]\!](x)$	$[\![\overline{\mathrm{S}}]\!] = \texttt{NOT}([\![\mathrm{VP}]\!])$
$\overline{\mathrm{S}}$ $\rightarrow$ *that* NP TV	$[\![\overline{\mathrm{S}}]\!] = \lambda y.[\![\mathrm{NP}]\!](\lambda x.[\![\mathrm{TV}]\!](x, y))$	$[\![\overline{\mathrm{S}}]\!] = [\![\mathrm{NP}]\!](\texttt{INV}([\![\mathrm{TV}]\!]))$
$\overline{\mathrm{S}}$ $\rightarrow$ *that* NP *did not* TV	$[\![\overline{\mathrm{S}}]\!] = \lambda y.[\![\mathrm{NP}]\!](\lambda x.\neg[\![\mathrm{TV}]\!](x, y))$	$[\![\overline{\mathrm{S}}]\!] = [\![\mathrm{NP}]\!](\texttt{NOT}(\texttt{INV}([\![\mathrm{TV}]\!])))$
Q $\rightarrow$ {*every, all, a, one, two, three*}	$[\![\textit{every}]\!] = [\![\textit{all}]\!] = \lambda F\lambda G.\forall x(F(x) \rightarrow G(x))$	$[\![\textit{every}]\!] = [\![\textit{all}]\!] = \lambda F\lambda G.\texttt{ALL}(F, G)$
	$[\![\textit{a}]\!] = [\![\textit{one}]\!] = \lambda F\lambda G.\exists x.(F(x) \wedge G(x))$	$[\![\textit{a}]\!] = [\![\textit{one}]\!] = \lambda F\lambda G.\texttt{EXIST}(F, G)$
	$[\![\textit{two}]\!] = \lambda F\lambda G.\exists x.(\mathbf{two}(x) \wedge F(x) \wedge G(x))$	$[\![\textit{two}]\!] = \lambda F\lambda G.\texttt{TWO}(F, G)$
	$[\![\textit{three}]\!] = \lambda F\lambda G.\exists x.(\mathbf{three}(x) \wedge F(x) \wedge G(x))$	$[\![\textit{three}]\!] = \lambda F\lambda G.\texttt{THREE}(F, G)$
N $\rightarrow$ {*dog, rabbit, cat, bear, tiger,...*}	$[\![\textit{dog}]\!] = \lambda x.\mathbf{dog}(x)$	$[\![\textit{dog}]\!] = \texttt{DOG}$
PN $\rightarrow$ {*ann, bob, fred, chris, eliott,...*}	$[\![\textit{ann}]\!] = \lambda F.F(\mathbf{ann})$	$[\![\textit{ann}]\!] = \lambda F.\texttt{EXIST}(\texttt{ANN}, F)$
IV $\rightarrow$ {*ran, walked, swam, danced, dawdled,...*}	$[\![\textit{ran}]\!] = \lambda x.\mathbf{run}(x)$	$[\![\textit{ran}]\!] = \texttt{RUN}$
IV$'$ $\rightarrow$ {*laughed, groaned, roared, screamed,...*}	$[\![\textit{laugh}]\!] = \lambda x.\mathbf{laugh}(x)$	$[\![\textit{laugh}]\!] = \texttt{LAUGH}$
TV $\rightarrow$ {*kissed, kicked, cleaned, touched,...*}	$[\![\textit{kissed}]\!] = \lambda y\lambda x.\mathbf{kiss}(x, y)$	$[\![\textit{kissed}]\!] = \texttt{KISS}$
ADJ $\rightarrow$ {*small, large, crazy, polite, wild,...*}	$[\![\textit{small}]\!] = \lambda x.\mathbf{small}(x)$	$[\![\textit{small}]\!] = \texttt{SMALL}$
ADV $\rightarrow$ {*slowly, quickly, seriously, suddenly,...*}	$[\![\textit{slowly}]\!] = \lambda x.\mathbf{slowly}(x)$	$[\![\textit{slowly}]\!] = \texttt{SLOWLY}$

Table 12: A set of context-free grammar rules and semantic composition rules used to generate the SyGNS dataset.

Sentence	FOL	VF
a small dog did not swim	$\exists x.(\mathbf{small}^{\uparrow}(x) \wedge \mathbf{dog}^{\uparrow}(x) \wedge \neg\mathbf{swim}^{\downarrow}(x))$	$\texttt{EXIST AND SMALL}^{\uparrow}\ \texttt{DOG}^{\uparrow}\ \texttt{NOT SWIM}^{\downarrow}$
all tigers ran or swam	$\forall x.(\mathbf{tiger}^{\downarrow}(x) \rightarrow \mathbf{run}^{\uparrow}(x) \vee \mathbf{swim}^{\uparrow}(x))$	$\texttt{ALL TIGER}^{\downarrow}\ \texttt{OR RUN}^{\uparrow}\ \texttt{SWIM}^{\uparrow}$
ann did not chase two dogs	$\neg\exists x.(\mathbf{two}(x) \wedge \mathbf{dog}^{\downarrow}(x) \wedge \mathbf{chase}^{\downarrow}(\mathbf{ann}, x))$	$\texttt{EXIST ANN NOT TWO DOG}^{\downarrow}\ \texttt{CHASE}^{\downarrow}$

Table 13: Example of (polarized) FOL formulas and VF formulas.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: numeral *two*								
ADJ	10.7	16.8	74.9	22.8	34.4	58.2	91.6	52.0
ADJ+NEG	10.1	22.3	79.7	24.3	33.4	58.7	94.4	51.0
ADV	12.8	29.3	79.8	46.9	40.3	56.1	89.1	60.5
ADV+NEG	14.1	33.9	83.8	58.6	34.4	56.4	93.3	65.8
CON	18.3	37.9	80.2	64.8	34.3	52.4	83.4	67.2
CON+NEG	24.4	40.6	82.6	68.7	31.3	50.8	88.0	68.5
primitive quantifier: universal quantifier *every*								
ADJ	7.7	19.5	70.6	58.5	20.7	20.5	89.9	62.2
ADJ+NEG	6.9	19.2	75.5	56.8	20.3	20.2	92.2	63.6
ADV	9.2	18.2	82.3	70.1	19.7	19.7	85.1	70.4
ADV+NEG	14.8	18.1	79.4	76.1	22.7	19.8	89.3	75.5
CON	14.7	18.0	70.3	79.6	21.5	19.2	68.8	75.8
CON+NEG	18.3	18.2	80.1	81.2	22.7	19.1	80.5	76.8

Table 14: Accuracy by modifier type where *two* or *every* is the primitive quantifier. "DRS (cnt)" columns show accuracies of predicted DRSs by COUNTER.

Test	GRU							Transformer						
	Exact	Upward			Downward			Exact	Upward			Downward		
	Match	Prec	Rec	F	Prec	Rec	F	Match	Prec	Rec	F	Prec	Rec	F
FOL formula														
EXI	96.1	100.0	99.9	99.9	100.0	100.0	100.0	99.9	100.0	100.0	100.0	100.0	100.0	100.0
NUM	7.6	99.2	75.5	84.8	99.6	94.7	96.8	18.1	100.0	79.5	88.1	100.0	95.6	97.5
UNI	3.1	92.6	89.9	90.9	42.9	39.4	40.7	8.3	97.3	93.4	94.9	79.4	70.0	73.4
VF formula														
EXI	99.7	100.0	99.9	99.9	99.9	99.9	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0
NUM	37.0	70.2	54.4	59.8	68.6	58.5	62.0	20.7	99.2	77.0	85.4	99.3	95.4	97.0
UNI	39.5	91.1	80.7	84.4	49.0	35.2	39.7	17.7	99.9	97.4	98.4	98.6	72.7	82.3

Table 15: Evaluation results on monotonicity. "Prec", "Rec", "F" indicate precision, recall, and F-score.

ing representation with different combinations of quantifiers and negations, like (17a) and (17b).

(15) a. **One** tiger ran
     b. **One** tiger *did not* run

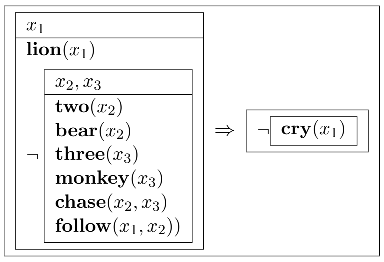
(16) a. **Every** tiger ran
     b. **Two** tigers ran

(17) a. **Every** tiger *did not* run
     b. **Two** tigers *did not* run

Table 16 shows the accuracy on combinations of quantifiers and negations by quantifier type. Similar to the results with unseen combinations of quantifiers and modifiers, models can easily generalize to problems involving quantifiers of the same type as the primitive quantifier. Table 17 shows the accuracy on combinations of quantifiers and negations by modifier types. Similar to the results in Table 14, the accuracies on problems involving logical connectives or adverbs were slightly better than those on problems involving adjectives.

E Error analysis of predicted DRSs

In the productivity experiments, the evaluation results using COUNTER on DRS prediction tasks are much higher than those by exact matching. Table 18 shows an example of predicted DRSs for the sentence *all lions that did not follow two bears that chased three monkeys did not cry*. This sentence contains embedded clauses with depth two, having the following gold DRS:



Both GRU and Transformer tend to correctly predict some of the clauses for content words, implication, and negation that appear at the beginning of the input sentence, while they fail to capture long-distance dependencies between subject nouns and verbs (e.g., *all lions ... did not cry*). Also, COUNTER correctly evaluates the cases where the order of clauses is different from that of gold answers.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: existential quantifier *one*								
Exi	65.9	88.0	98.1	94.5	99.9	96.6	97.6	45.0
Num	48.4	71.0	96.7	54.5	65.8	86.6	98.8	26.3
Uni	22.3	0.0	52.0	53.1	0.0	0.0	70.2	26.6
Valid	96.6	98.1	100.0	99.8	100.0	100.0	100.0	100.0
primitive quantifier: numeral *two*								
Exi	36.4	63.6	89.8	13.9	14.4	49.6	86.9	11.5
Num	40.0	66.7	94.1	21.5	33.0	59.1	87.6	14.3
Uni	15.4	0.0	50.7	0.0	0.0	0.0	71.0	12.4
Valid	96.4	97.8	100.0	99.3	100.0	100.0	100.0	100.0
primitive quantifier: universal quantifier *every*								
Exi	12.8	0.0	73.7	78.6	0.8	0.5	52.2	59.4
Num	17.1	0.0	75.3	78.1	0.0	0.6	59.2	65.1
Uni	91.1	88.3	97.4	94.9	86.6	70.1	92.3	76.6
Valid	98.8	98.1	100.0	98.8	100.0	100.0	100.0	100.0

Table 16: Accuracy on combinations of quantifiers and negation by quantifier type. “DRS (cnt)” columns show accuracies of predicted DRSs by Counter. “Valid” row shows the validation accuracy.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: existential quantifier *one*								
Adj+Neg	34.6	33.2	68.3	45.8	19.4	45.1	84.1	25.3
Adv+Neg	38.0	36.3	77.9	53.0	43.2	54.8	88.1	37.5
Con+Neg	36.8	33.2	73.2	54.7	47.4	52.4	85.4	37.0
primitive quantifier: numeral *two*								
Adj+Neg	21.2	18.2	69.3	8.0	11.6	29.3	75.9	1.2
Adv+Neg	26.5	28.5	71.4	10.3	19.3	36.5	81.8	17.8
Con+Neg	21.9	28.1	68.5	9.1	11.7	34.6	78.7	16.3
primitive quantifier: universal quantifier *every*								
Adj+Neg	26.7	12.4	63.9	60.5	13.9	14.6	58.0	50.7
Adv+Neg	25.9	13.2	69.2	66.1	21.9	15.2	60.9	63.3
Con+Neg	28.5	18.8	71.4	65.9	20.2	14.6	59.7	62.8

Table 17: Accuracy on combinations of quantifiers and negation by modifier type.

(a) Gold answer	(b) GRU (F: 0.45)	(c) Transformer (F: 0.42)
b1 IMP **b2 b4**	**b1** IMP **b2 b3**	**b1** IMP **b2 b3**
b2 REF **x1**	**b2** REF **x1**	**b2** REF **x1**
b2 lion **x1**	**b2** lion **x1**	**b2** lion **x1**
b2 NOT **b3**	**b2** NOT **b3**	**b2** NOT **b3**
b3 REF **x2**	**b3** REF **x2**	**b3** REF **x2**
b3 REF **x3**	**b3** two **x2**	**b3** two **x2**
b3 two **x2**	**b3** bear **x2**	**b3** monkey **x2**
b3 bear **x2**	**b3** follow **x2 x2**	**b3** follow **x2 x1**
b3 three **x3**	**b3** REF **x3**	**b3** REF **x3**
b3 monkey **x3**	**b3** three **x3**	**b3** john **x3**
b3 chase **x3 x2**	**b4** monkey **x3**	**b3** chase **x1 x3**
b3 follow **x2 x1**	**b4** like **x3 x2**	
b4 NOT **b5**	**b4** like **x1 x2**	
b5 cry **x1**		

Table 18: Error analysis of DRSs for the sentence “all lions that did not follow two bears that chased three monkeys did not cry”. Clauses in green are correct and those in red are incorrect. “F” shows F-score over matching clause.