Scope-enhanced Compositional Semantic Parsing for DRT

Anonymous ACL submission

Abstract

Discourse Representation Theory (DRT) distinguishes itself from other semantic representation frameworks by its ability to model complex semantic and discourse phenomena through structural nesting and variable binding. While seq2seq models hold the state of the art on DRT parsing, their accuracy degrades with the complexity of the sentence, and they sometimes struggle to produce well-formed DRT representations. We introduce the AMS parser, a compositional, neurosymbolic semantic parser for DRT. It rests on a novel mechanism for predicting quantifier scope. We show that the AMS parser reliably produces well-formed outputs and performs well on DRT parsing, especially on complex sentences.

1 Introduction

Among current semantic representation formalisms used in NLP, Discourse Representation Theory (DRT; Kamp and Reyle, 1993) stands out in its systematic use of structural nesting and variable binding to represent meaning in detail. Originating from linguistic theory, DRT has been designed to capture subtle semantic and discourse phenomena such as anaphora, presupposition, and discourse structure, as well as tense and aspect (see Fig. 1). This structural and semantic richness distinguishes DRT from other popular frameworks in semantic parsing, such as Abstract Meaning Representation (AMR; Banarescu et al., 2013).

With the availability of the broad-coverage Parallel Meaning Bank (PMB; Abzianidze et al., 2017), DRT has become an active target for the development of semantic parsing methods. The current state of the art is held by purely neural seq2seq models (Zhang et al., 2024). However, due to the structural complexity of typical DRT representations, these models do not always generate well-formed meaning representations. They also struggle on long sentences; length generalization is a known challenge for transformers in semantic parsing settings (Hupkes et al., 2020; Yao and Koller, 2022). Existing compositional semantic parsers for DRT significantly lag behind the seq2seq models in terms of parsing accuracy.

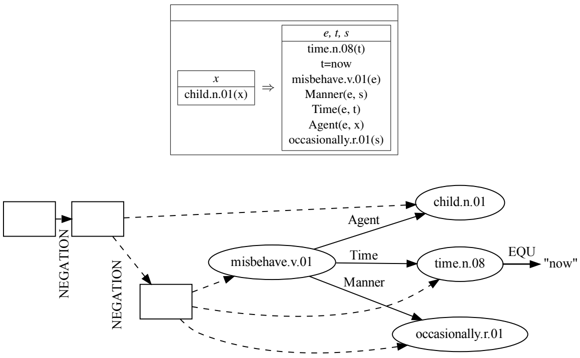


Figure 1: DRS (top) and DRG (bottom) for the sentence *Every child misbehaves occasionally*; dashed lines represent scope assignments of connectives.

In this paper, we introduce the *AMS parser*, an accurate compositional DRT parser. The AMS parser extends the AM parser (Groschwitz et al., 2018), which predicts meaning representations compositionally and has achieved high accuracy across a range of sembanks (Lindemann et al., 2019; Weißenhorn et al., 2022). The AM parser by itself struggles to predict structural nesting in DRT. The key challenge is to predict *scope:* how to assign each atomic formula in Fig. 1 to one of the three boxes. Differences in scope assignment affect the represented meaning significantly.

The technical contribution of this paper is to extend the AM parser with an innovative mechanism for predicting scope. We train a dependency parser to predict scope relations between word tokens and project this information into the DRT representation using word-to-box alignments. We show that this dependency mechanism can predict correct scope assignments at very high accuracy. The overall parser always predicts well-formed DRT representations (in contrast to all seq2seq models) and is almost on par with the best models in parsing accuracy. On the PMB TestLong split, which contains particularly long sentences, it outperforms all other DRT parsers that are trained on the PMB gold dataset. Thus, the strength of the AMS parser is its ability to remain accurate as sentences grow complex.

2 Background and Related Work

Discourse Representation Theory (DRT; Kamp and Reyle, 1993) is a well-developed framework for dynamic semantics that aims to interpret meanings from the context. It can model diverse linguistic phenomena ranging from anaphora (Kamp, 1981; Haug, 2014) to rhetorical structures (Lascarides and Asher, 2007). In DRT, meanings are traditionally represented by Discourse Representation Structures (DRS), which are composed of nested boxes that contain discourse referents (the entities talked about in the discourse) and propositions about these discourse referents. Fig. 1 (top) is an example of DRS representing *Every child misbehaves occasionally*. The boxes act as logical quantifiers that bind variables, and they can be connected with logical operators such as implication.

Bos (2023) recently proposed an equivalent, variable-free notation for DRSs in the form of directed acyclic graphs, called Discourse Representation Graphs (DRGs; see Fig. 1, bottom). A DRG contains nodes representing boxes, predicate symbols, and constants. Some edges (drawn solid in Fig. 1) connect predicates to arguments with semantic roles. Others (drawn dashed) represent the structural nesting of boxes and propositions: A dashed edge means that its target node is inside the box from which the edge emanates. Universal quantification, disjunction, and implication are represented in DRGs as logically equivalent structures using only negation and conjunction.

The main resource for DRS and DRG is the Parallel Meaning Bank (PMB; Abzianidze et al. (2017)), which is a multilingual parallel corpus comprising sentences and texts paired with meaning representations. In this paper, we use the latest version (PMB release 5.1.0, English) for evaluation. It includes three distinct splits based on the quality and method of annotation: Gold (manually verified), Silver (partially corrected), and Bronze (automatically generated by Boxer). As our objective is to address challenges within a limited data setting, our experiments specifically focus on utilizing gold-annotated data.

2.1 DRS parsing

Deriving DRSs from sentences compositionally is a nontrivial challenge. Efforts towards this goal include λ-DRT (Muskens, 1994; Kohlhase et al., 1996, 1998), Compositional DRT (Muskens, 1996), and bottom-up DRT (Asher, 1993). All of these approaches use lambda calculus to compositionally combine partial meaning representations, which is intractable in broad-coverage semantic parsing (see e.g. the discussion by Artzi et al. (2015)).

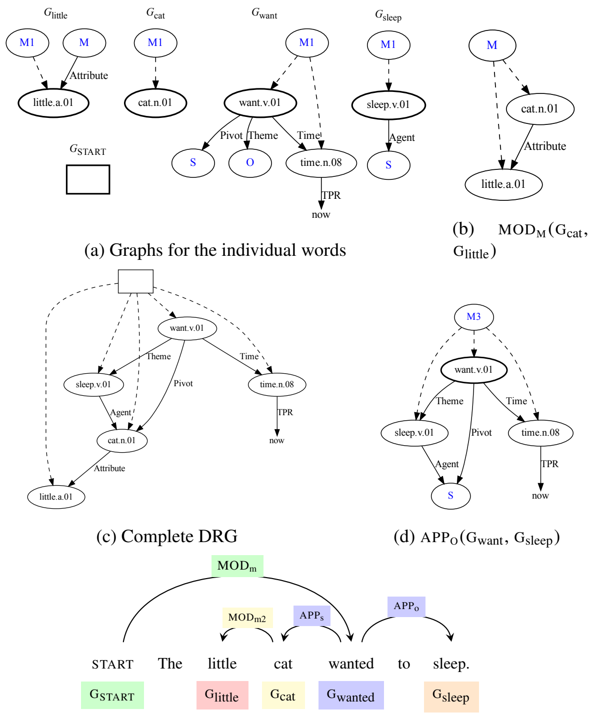
To date, the most accurate broad-coverage DRT parsers are based on neural sequence-to-sequence models (e.g., Liu et al., 2018; Fancellu et al., 2019; Van Noord et al., 2018; van Noord et al., 2020). They achieve impressive performances, especially when the models are trained on additional silver or bronze training data (Wang et al., 2023a) or use additional features (van Noord et al., 2019, 2020). However, due to the structure-unaware design of these models, they sometimes struggle to generate well-formed DRT representations (see Poelman et al. (2022)).

Existing compositional semantic parsers for DRT rely on syntactic dependency parsers (Le and Zuidema, 2012; Poelman et al., 2022) or CCG parsers (Bos, 2008, 2015). These models reliably generate well-formed DRSs, but are not competitive with seq2seq models in terms of parsing accuracy.

2.2 AM Parsing

The DRT parser we present here is based on the AM Parser (Groschwitz et al., 2018), a neurosymbolic compositional semantic parser that has previously been shown to be fast and accurate both on broad-coverage parsing, e.g. on AMR (Lindemann et al., 2019), and in compositional generalization tasks (Weißenhorn et al., 2022).

Apply and Modify The AM parser uses a neural dependency parser and tagger to predict terms over the AM algebra (Groschwitz et al., 2017), which combines graphs into bigger graphs using the operations *Apply* and *Modify*. To this end, nodes of the graphs can be decorated with *sources* (Courcelle and Engelfriet (2012), marked in blue), which assign names to nodes at which the graph can be combined with other graphs. Every graph has a special source called ROOT, drawn with a bold out-



(a) Graphs for the individual words

(b) $\text{MOD}_\text{M}(G_{\text{cat}}, G_{\text{little}})$

(c) Complete DRG

(d) $\text{APP}_\text{O}(G_{\text{want}}, G_{\text{sleep}})$

(e) AM dependency tree

Figure 2: Relevant graphs for sentence *The little cat wanted to sleep.*

line, which is where the graph inserts into others when used as an argument.

In the example of Fig. 2a, the graph G_{WANT} has sources S and O indicating where the arguments supplied by the subject and object should be inserted. It also has a source M1 which allows it to attach to some other graph as a modifier.

The *Apply* operation (APP) models the combination of a complement (i.e. argument) with its head. For example in Fig. 2d, the APP_O operation combines the head G_{WANT} with its argument G_{SLEEP}, plugging the root of G_{SLEEP} into the O source of G_{WANT} (Fig. 2d). Because every graph may only contain one node decorated with each source name, the S and M1 source nodes of G_{SLEEP} and G_{WANT} get merged. This allows the AM algebra to generate nontrivial graph structures.

The *Modify* operation (MOD) models the combination of a head with a modifier. For example, the MOD_M operation in our example attaches the adjunct G_{LITTLE} to the root of its head G_{CAT}, using the adjunct's M source (Fig. 2b). Again, both graphs have an M1 source that gets merged.

AM dependency trees and AM parsing The AM parser predicts a graph from a sentence by computing an *AM dependency tree*, as in Fig. 2e. It uses a neural tagger to predict a *lexical graph* for each word (drawn below the sentence) and a neural dependency parser to predict APP and MOD edges. The AM dependency tree can be unraveled into a term of APP and MOD operations over the AM algebra, which deterministically evaluates into a graph; for instance, the AM dependency tree in Fig. 2e evaluates to the graph in Fig. 2c. Words that do not lexically contribute to the meaning representation, such as the determiner *the*, are not assigned incoming dependency edges and thus ignored in the construction of the graph.

In order to train the AM parser, one needs to construct an AM dependency tree for every sentence-graph instance in the training data. *Decomposing* the graph into an AM dependency tree is a nontrivial task, which can fail: Depending on the alignments between word tokens and nodes in the graph, an AM dependency tree that evaluates to the given graph may not exist. We call such training instances *non-decomposable*.

3 Scope in DRT is hard for the AM parser

We start with an attempt to directly apply the AM parser to DRT. As we will see, the dashed scope edges in a DRG are difficult to handle with the AM parser. We will solve this problem in the AMS parser, presented in Section 4.

3.1 A baseline AM parser for DRG

We construct AM dependency trees for the DRGs in the PMB using the approach proposed by Groschwitz et al. (2021), which learns decomposition jointly with training the neural parsing model. The learning algorithm represents the latent space of possible AM dependency trees for each graph compactly, allowing training on the whole latent space. This leads to the parser converging on AM dependency trees that are consistent across the corpus.

This largely unsupervised method still requires two inputs beyond the graph. First, node-token alignments (every node must be aligned), for which we use the alignments given in PMB5.1. For the top box in each DRG, which is always unaligned, we introduce a special START token to align it to (cf. Fig. 2e).

Second, each edge must be assigned to a graph constant, to fully partition the DRG into lexical graphs for the individual words. This often makes the difference between an Apply or Modify op-

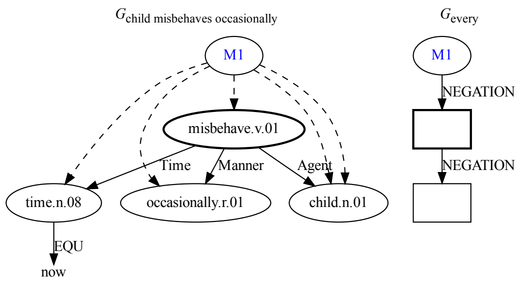


Figure 3: Failed combination of graphs for Fig 1

	NoPrep	CPT	SCPL
APP	0.7	84.5	94.4
MOD	76.7	77.7	78.0

Table 1: Decomposable graphs in PMB5 (%). APP: member edges grouped with the box; resulting in Apply operations in the AM dependency tree. MOD: member edges grouped with the content nodes, resulting in Modify operations.

eration. For example in Fig. 2, the `Attribute` edge between `little` and `cat` is grouped with the `little` node, making `little` a modifier of `cat`, a linguistically plausible analysis. The edge could also be grouped with the `cat` node, effectively making `little` an argument of `cat` (the two would be combined with an APP operation), an implausible analysis. We follow the linguistically-informed principle to group edges between a head and an argument with the head, and edges between a head and a modifier (Lindemann et al., 2019); see Appendix C for our full heuristics. The scope edges do not fall into these categories and provide a unique challenge, see below.

All remaining aspects of the AM dependency tree, including the source names, are then learned during training.

3.2 The Challenge of Scope Prediction

The scope edges of DRGs are not something that the Apply and Modify operations were designed for. In particular, the scope edges do not fall straightforwardly into the head/argument/modifier paradigm. The design of the AM algebra forces us into an inconvenient choice: (1) include scope edges in the lexical graph that contains the box and insert the contents of the box with Apply operations; or (2) include scope edges in the lexical graphs of the contents of the box and insert them into the box using Modify operations.

The first approach fails completely, with only

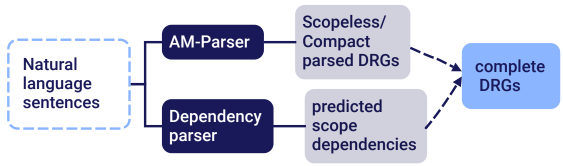


Figure 4: Overall structure of the AMS parser.

0.7% of DRGs in PMB5 being decomposable (see Table 1, NoPrep/APP; see also Appendix B). The second approach works better, with 76% graphs being decomposable (Table 1, NoPrep/MOD). For example, Fig. 2e shows a valid AM dependency tree for the graph in Fig. 2c under this paradigm. However, this success is limited to graphs with only a single box: only 30% of all multibox DRGs, i.e. DRGs that contain more than one box node, can be decomposed into AM dependency trees.

To illustrate the challenge, consider the DRG in Fig. 1. Fig. 3 shows two partial graphs in an attempt to build the full graph with the AM algebra, the left representing *child misbehaves occasionally*, and the right representing *every*. The lexical graph G_{EVERY} introduces two boxes, and to obtain the DRG in Fig. 1, we need to draw a scope edge from the upper box to the *child* node on the left and, simultaneously, scope edges from the lower box to the *misbehave*, *time*, and *occasionally* nodes. We can use a MOD_{M1} operation to unify the M1-source of the left graph with the root of G_{EVERY} (the upper box); but this will put *child* into the wrong box. The problem is that both boxes are introduced by the same lexical graph (a consequence of the alignments in the PMB), and only one of them can receive outgoing edges through a single Modify operation. Other attempts at decomposing the DRG in Fig. 1 fail in similar ways.

4 Scope-enhanced AM Parsing

We will address this scope challenge through a two-step process. First, we simplify the DRGs by removing scope edges, such that over 94% of DRGs can be decomposed for training. Second, we recover the scope information at parsing time through an independent scope prediction mechanism. The overall structure of our parser is sketched in Fig. 4.

4.1 Simplifying DRGs

We identified two effective DRG simplification strategies: *Compact DRG* and *Scopeless DRG*.

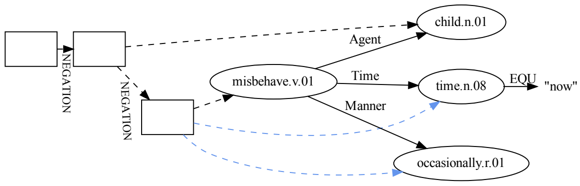


Figure 5: Compact DRG for Fig 1.

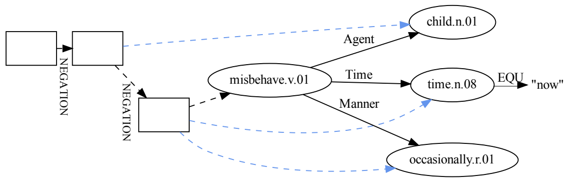


Figure 6: Scopeless DRG for Fig 1.

Compact DRG The Compact DRG representation (CPT), inspired by Abzianidze et al. (2020), makes use of the fact that many nodes share the same scope as their parent node, i.e. are members of the same box. In this representation, we thus remove all scope edges for nodes that are in the same scope as their parents (if there are multiple parents, we only remove the scope edge if the node and *all* its parents are in the same box). This method removes around 70% of scope edges, and the full scope information can be losslessly recovered with the rule-based method in Section 4.2. The compact DRG for Fig. 1 is shown in Fig. 5 with the removed edges marked in light blue.

Scopeless DRG While Compact DRGs maintain at least one connection between a scope box and a node within its scope, Scopeless DRGs (SCPL) remove all scope edges as long as the graph remains connected. This results in graphs that are mostly reduced to their predicate-argument structure, facilitating a more straightforward decomposition with the AM Algebra, at the cost of losing some information. An example is shown in Fig. 6. More complex examples are detailed in Appendix I.

Both Compact and Scopeless DRGs show much higher decomposability rates compared to the full DRGs, see Table 1. This effect is particularly strong in the setting where membership edges are grouped with the boxes (see row "APP"), where Compact and Scopeless DRGs achieve decomposability rates of 84.5% and 94.4% respectively.

4.2 Scope Prediction

To recover the scope information, we designed two scope resolvers: one rule-based, and the other relying on a dependency parser to predict the scope edges.

Rule-based Scope Resolver The rule-based scope resolver is the inverse of our Compact DRG simplification method, but can also be applied to Scopeless DRG. This resolver traverses the predicted graph top-down; if it encounters a node with no incoming scope edge, it assigns the node the same scope as its parent. If a node has multiple parents with conflicting scope, an arbitrary parent is chosen (this only occurs with Scopeless DRG). For Compact DRG, this method recovers the full scope information losslessly.

This rule-based approach is easy to implement, transparent and fully explainable. However, it is imperfect for Scopeless DRG, and even for Compact DRG it may propagate parsing errors into the recovered scope edges.

Dependency-based Scope Resolver For the dependency-based scope resolver, we make use of the fact that an AM dependency tree splits the graph into lexical graphs, each of which is linked to a specific word token in the sentence. This induces an alignment relation between nodes in the graph and tokens in the sentence: a node is aligned to the token if it is part of the lexical graph for that token. We project the scope edges in the DRG into edges between the word tokens by following this alignment relation from the nodes to the tokens; this creates a *scope dependency graph* over the sentence (see Fig. 7). The scope dependency graph is not necessarily a tree: it need not be connected, and a token might receive multiple incoming edges if the aligned lexical graph contains multiple nodes linked to different boxes (see Appendix F).

When the lexical graph for a token contains multiple nodes or boxes, we also encounter a further challenge. In such a case, the scope dependency graph, which connects only the two tokens, cannot fully specify which nodes in the lexical graphs the scope edge connects. An example of this can be seen in Fig. 7. Here, the lexical graphs G_{CHILD} and $G_{\text{MISBEHAVES}}$ are both children of G_{EVERY} in the scope dependency graph, but they should go in different boxes of G_{EVERY}.

To remove this ambiguity, we name the boxes in each lexical graph and encode the box to which each child in the scope dependency graph connects in the dependency edge label. For example, consider again the dependency graph in Fig. 7. The

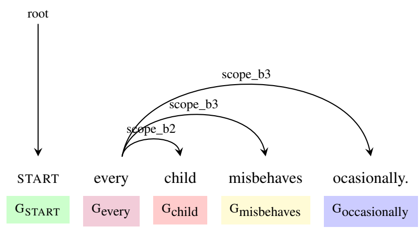


Figure 7: Scope dependency graph for *Every child misbehaves occasionally.*

relationship between the tokens *every* and *child* is annotated as `scope_b2`, indicating that *child* goes into the upper box (`b2`[1]). By contrast, the edge into *misbehaves* has the label `scope_b3`, indicating that it goes into the lower box. We use a similar method if different nodes from the same constant are members of different boxes, see Appendix F. In this way, the labeled scope dependency graphs unambiguously specify scope edges.

This method allows us to use standard dependency parsing techniques for scope prediction. We adopted the biaffine dependency graph parser of Dozat and Manning (2018), which is simple and accurate. We use ordinary supervised training, based on the annotated node-token alignments in the PMB5. Hyperparameter details can be found in Appendix E.

Since the AM parser also predicts some scope edges (in particular for Compact DRG, but also a bare minimum in Scopeless DRG), there can be conflicts between the dependency-based scope predictions and scope edges already present in the predicted simplified DRG. We use the following rules to resolve mismatches: (1) We only use a dependency-based edge if its target has no scope edge in the predicted simplified DRG; i.e. the AM parser predictions take precedence. (2) Any remaining node without a scope edge inherits its scope from its parent (as in the rule-based resolver).

5 Results & Discussion

5.1 Data & Evaluation

We evaluated on the latest Parallel Meaning Bank 5.1.0 (Abzianidze et al., 2017). Apart from the normal train, dev, and test split, the PMB 5.1.0 also provides an extra TestLong set that contains 40 lengthy (average length: 39.7 tokens) sentences. Statistics can be found in Appendix A.

For the evaluation metric, following Poelman et al. (2022), we convert DRGs to condensed Penman notations[2] (Wang et al., 2023b) and adopt the SMATCH F1 score (Cai and Knight, 2013) to assess DRGs. We also report the percentage of test instances for which a parser generated ill-formed DRGs.

5.2 Handling coreference

The PMB contains coreference annotations; these are non-compositional by design and thus very tricky for a compositional system like the AM parser. We reduce the impact of coreference on our evaluation through a simple pre- and postprocessing method. We remove all edges indicating coreference in the DRG and introduce a new tag `p` to the label of all coreferent nodes. In postprocessing, we then simply add coreference edges between all nodes marked as coreferent. This method further increases decomposability, up to 94% (see Table 1). Details are in Appendix D.

This method has the advantage of only using information from the predicted DRG, but it only really works when there is just one instance of coreference in the graph. This is frequently the case in the PMB 5.1.0, but in a different setting, more complex coreference resolution methods would likely be needed (see e.g. Anikina et al. (2020)).

5.3 Experiment details

We use the implementation of Groschwitz et al. (2021) in all our AM Parser experiments. Hyperparameter settings can be found in Appendix H.

We compare the AMS parser against the strongest published models for DRG parsing listed in Zhang et al. (2024): `byT5` (Xue et al., 2022), `mT5` (Xue et al., 2021) and `mBART` (Liu et al., 2020). All of these are sequence-to-sequence models with no built-in awareness of semantic structure, compositionality, or scope.

We also trained the AM Parser on the DRGs with the original scope annotations. To make the root box easier to learn for the parser, we introduced a new token `START` to the beginning of each input sentence. Finally, we fine-tuned `T5-Base`, `T5-Large` (Raffel et al., 2020) as two further robust baselines.

[1]The labeling of the boxes is decided by the hierarchy of the boxes in the whole graph: the parent box is assigned by a smaller number than the children, the root box is assigned with `b1`.

[2]Examples can be found in Appendix G.

Dev		Test		TestLong	
UAS	LAS	UAS	LAS	UAS	LAS
98.7	96.4	98.3	95.7	67.0	55.4

Table 2: Accuracy of scope dependency parsing.

Models	Test		TestLong	
	F1	Err	F1	Err
Baselines (gold only)				
ByT5$_{(G)}$	**86.7**	5.4	27.1	38.3
mT5$_{(G)}$	61.2	11.3	16.5	25.0
mBART$_{(G)}$	82.8	6.3	30.5	12.5
T5-base$_{(G)}$	76.4	20.0	13.9	77.5
T5-large$_{(G)}$	84.2	3.9	18.1	67.5
AM Parser$_{(G)}$	81.9	**0.0**	47.2	**0.0**
without scope resolution				
AMS Parser$_{scpl(G)}$	73.5	**0.0**	39.3	**0.0**
AMS Parser$_{cmpt(G)}$	73.1	**0.0**	37.3	**0.0**
with scope resolution				
AMS Parser$_{scpl+h(G)}$	84.6	**0.0**	48.4	**0.0**
AMS Parser$_{cmpt+h(G)}$	83.1	**0.0**	44.9	**0.0**
AMS Parser$_{scpl+d(G)}$	85.3	**0.0**	**48.8**	**0.0**
AMS Parser$_{cmpt+d(G)}$	83.3	**0.0**	46.0	**0.0**
Baselines (gold + silver)				
byT5$_{(G+S)}$	93.4	0.7	36.6	40.0
T5-base$_{(G+S)}$	86.0	1.6	44.3	37.5
mT5$_{(G+S)}$	93.1	0.8	55.8	15.0
mBART$_{(G+S)}$	86.2	4.4	7.8	12.5

Table 3: Accuracy and error rates for DRG parsing.

5.4 Parsing Results

Scope Dependency Parsing We first evaluate how accurately scope assignments can be predicted by dependency parsing (cf. Section 4.2), using the usual UAS and LAS evaluation measures for dependency parsing. Table 2 reveals high LAS and UAS of predicted scope dependency graphs across both development and test sets, indicating reliable scope prediction. This is remarkable, given the complexity of the scope prediction task.

On the TestLong set, the accuracy dropped significantly, indicating the difficulty of predicting scope as sentences grow in complexity. The much larger drop in LAS compared to UAS indicates the difficulty of reliably making scope assignment decisions within a lexical graph.

DRG Parsing For the task of DRG parsing itself, we compare the AMS parser to the baselines in Table 3. Our focus is on models that are trained on the hand-annotated gold dataset (G); we also include some models trained on gold and silver. The suffix *scpl* denotes Scopeless DRGs, *cpt* refers to Compact DRGs, *d* indicates the dependency-based scope resolver, and *h* signifies the heuristic scope resolver. "Without scope resolution" groups together variants of the AMS parser that directly predict compact or scopeless DRGs, without a mechanism for reconstructing scope edges in postprocessing. The best results among the gold-trained models are marked in bold. Three critical observations emerge from the table.

First, the AMS parser, especially the scopeless (SCPL) version, excels against the gold-data trained baselines. The only exception is `byT5`, which has a token-free architecture that makes it particularly good at processing short texts – a significant advantage given the very short average sentence length of 6.7 tokens in the regular test set. The AMS parser also outperforms the generic AM parser, indicating the effectiveness of our novel scope resolution mechanism.

Second, in contrast to all seq2seq models, the AMS parser maintains a 0% error rate, i.e. it never generated ill-formed DRGs. Furthermore, on the very long sentences of the TestLong set, all variants of the AMS parser outperform the gold-trained seq2seq baselines by a large margin, almost achieving parity with the best model trained on silver data.

Finally, Scopeless DRGs perform better than Compact DRGs. This could be attributed to the fact that Compact DRGs retain more scope edges, making the graph more complex to learn. The higher decomposability rate of Scopeless DRGs also means that we have more training data in that setting. The dependency-based scope resolver outperforms its heuristic-based counterpart in accuracy across in-domain development and test splits. This advantage makes sense given the scope dependency parser's high accuracy. It could also be that the dependency resolver is better able to handle initial parsing inaccuracies compared to the rule-based resolver, where AM Parsing errors can easily propagate into more scope errors.

Scaling to complex DRGs As we already saw in Section 3.2, scope prediction is easy when there are not many boxes. Table 3 therefore splits the test instances by number of boxes[3]. For each of these classes, we report the overall SMATCH score of our best model and the baselines, as well as the SMATCH score when considering only scope

[3]For the TestLong split, we evaluate the models only on multi-box DRGs, of which there are 33 out of 40.

Models	Test				TestL
# Box	#1	#2	#3	#≥4	#≥2
Count	972	136	75	10	33
T5-base(G)	86.2	17.5	17.5	16.7	9.0
	92.9	3.0	5.0	14.6	10.3
mBART(G)	84.8	81.5	83.1	76.6	17.2
	91.7	80.7	84.8	80.2	17.2
mT5(G)	65.6	61.6	55.8	49.9	13.5
	76.7	65.4	63.3	59.3	17.7
T5-large(G)	**87.9**	74.8	67.4	45.2	19.0
	94.1	73.7	68.5	48.3	21.5
byT5(G)	87.3	**84.3**	**86.9**	52.5	29.6
	93.6	85.2	**86.6**	55.6	33.0
AM Parser(G)	84.3	75.6	67.5	61.9	46.3
	92.1	78.9	72.0	66.5	56.0
AMS Parser$_{\text{scpl+d}}$(G)	86.0	83.8	81.9	**75.2**	**48.2**
	92.9	**86.5**	82.2	**85.2**	**58.7**
byT5(G+S)	89.1	89.9	88.8	83.6	48.0
	95.0	89.2	90.6	87.4	47.8
mT5(G+S)	89.1	89.9	88.8	85.6	61.7
	95.0	88.9	89.5	88.4	65.1
mBART(G+S)	84.8	81.5	83.1	75.7	17.2
	91.7	80.6	84.6	80.2	17.6

Table 4: SMATCH score for multi-box DRGs and corresponding scope score (highlighted in gray)

edges. This allows us to explore how the parsers scale to complex DRGs, and in particular how they maintain their ability to predict scope edges when there are many boxes.

Compared to other models trained on gold data, the AMS parser excels at maintaining its accuracy as the DRGs grow more complex. While the AM parser is almost on par with the AMS parser on single-box DRGs, the gap widens drastically with increasing complexity. For DRGs with four or more boxes, as well as on the TestLong set, the AMS parser also decisively outperforms all (gold) seq2seq baselines.

At the same time, we observe that the AMS parser maintains a very high accuracy on predicting scope edges even for complex DRGs. We observe that the difference between the AMS Parser and the baseline AM parser is small on single-box DRGs, but much larger on multi-box DRGs, showing that treating scope prediction separately pays off.

5.5 What makes long texts so hard?

Anil et al. (2022) found that simple fine-tuning of transformer models does not achieve length generalization, nor does scaling up the models. We conducted a detailed error analysis and identified two factors that might contribute to the limitations of the models in length generalization.

Structural Complexity As shown in Table 4, all models show a decreasing trend as the number of boxes increases. We find that a higher number of boxes generally results in longer sequences, especially in the TestLong split - we assume the box complexity brought by longer sequences could be a possible reason for length generalization limitation.

Furthermore, byT5 tends to generate shorter sequences, averaging 70 roles and relations in its predictions, in contrast to other models which average approximately 100. This discrepancy underscores byT5's limitation in handling long texts.

Sense Generalization Furthermore, longer sentences can introduce new word senses, which have to be predicted as node labels. 25% senses in the TestLong split are absent in the train split. All models show accuracies lower than 0.33 in predicting unseen senses with the AMS Parser performing the best at this rate.

6 Conclusion and Future Work

In this work, we proposed a novel mechanism for predicting scope assignments in DRT parsing. By combining it with the compositional AM parser, we obtain the AMS parser, which outperforms existing DRT parsers trained on the same dataset, especially for complex sentences. It also avoids the prediction of ill-formed DRGs that plague other models. The prediction of scope information has been a longstanding challenge in computational semantics; our dependency parsing mechanism achieves very high accuracy on this task.

In the future, we plan to extend our work to tackle increasingly complex meaning representation frameworks, such as Uniform Meaning Representation (UMR) (Van Gysel et al., 2021). Since UMR-writer (Zhao et al., 2021), the UMR annotation tool, provides node-token alignment automatically, no more manual annotation is needed. Furthermore, our current system's architecture, which includes both the AM Parser and a dependency parser by Dozat and Manning (2018), presents opportunities for optimization. We aim to streamline the process by unifying these two models into a single framework that leverages joint learning.

Limitations

The AMS parser uses the AM parser to predict the predicate-argument relations in the DRGs. The AM parser has not kept pace in accuracy with the development of overall graph parsing models since it was published in 2019. This holds back the accuracy of the AMS parser. If a more accurate sentence-to-graph parser that induces node-token alignments became available, the AMS parser could be combined with it for increased accuracy. Note, however, that the AM parser shows strong performance with respect to the degradation of parsing accuracy for long and complex sentences.

Furthermore, the treatment of coreference in the paper is quite shallow. One might include the predictions of a coreference resolver into the parsing process. On the relatively short coreference chains in the PMB test sets, this would probably not make a significant impact on the evaluation.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Lasha Abzianidze, Johan Bos, and Stephan Oepen. 2020. DRS at MRP 2020: Dressing up discourse representation structures as graphs. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 23–32, Online. Association for Computational Linguistics.

Tatiana Anikina, Alexander Koller, and Michael Roth. 2020. Predicting coreference in Abstract Meaning Representations. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 33–38, Barcelona, Spain (online). Association for Computational Linguistics.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Nicholas Asher. 1993. *Reference to abstract objects in discourse*, volume 50. Springer Science & Business Media.

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.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Johan Bos. 2015. Open-domain semantic parsing with boxer. In *Proceedings of the 20th nordic conference of computational linguistics (NODALIDA 2015)*, pages 301–304.

Johan Bos. 2023. The sequence notation: Catching complex meanings in simple graphs. In *Proceedings of the 15th International Conference on Computational Semantics*, pages 195–208, Nancy, France. Association for Computational Linguistics.

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.

Bruno Courcelle and Joost Engelfriet. 2012. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Federico Fancellu, Sorcha Gilroy, Adam Lopez, and Mirella Lapata. 2019. Semantic graph parsing with recurrent neural network DAG grammars. 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 2769–2778, Hong Kong, China. Association for Computational Linguistics.

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with AMRs. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS) — Long papers*.

Jonas Groschwitz, Meaghan Fowlie, and Alexander Koller. 2021. Learning compositional structures for semantic graph parsing. In *Proceedings of the 5th Workshop on Structured Prediction for NLP (SPNLP 2021)*, pages 22–36, Online. Association for Computational Linguistics.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1831–1841, Melbourne, Australia. Association for Computational Linguistics.

Dag Trygve Truslew Haug. 2014. Partial dynamic semantics for anaphora: Compositionality without syntactic coindexation. *Journal of Semantics*, 31(4):457–511.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Hans Kamp. 1981. Evénements, représentations discursives et référence temporelle. *Langages*, (64):39–64.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic: An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

Michael Kohlhase, Susanna Kuschert, and Martin Müller. 1998. Dynamic lambda calculus. Manuscript.

Michael Kohlhase, Susanna Kuschert, and Manfred Pinkal. 1996. A type-theoretic semantics for-DRT. In *Proceedings of the 10th Amsterdam Colloquium*, pages 479–498.

Alex Lascarides and Nicholas Asher. 2007. Segmented Discourse Representation Theory: Dynamic semantics with discourse structure. In *Computing meaning*, pages 87–124. Springer.

Phong Le and Willem Zuidema. 2012. Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING 2012*, pages 1535–1552, Mumbai, India. The COLING 2012 Organizing Committee.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

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.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

R.A. Muskens. 1994. A compositional Discourse Representation Theory. In *Proceedings of the Ninth Amsterdam Colloquium*, volume 9 of *Proceedings of the Amsterdam Colloquium*, pages 467–486. ILLC/Department of Philosophy, University of Amsterdam. Pagination: 20.

Reinhard Muskens. 1996. Combining Montague semantics and discourse representation. *Linguistics and philosophy*, pages 143–186.

Wessel Poelman, Rik van Noord, and Johan Bos. 2022. Transparent semantic parsing with Universal Dependencies using graph transformations. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4186–4192, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Jens EL Van Gysel, Meagan Vigus, Jayeol Chun, Kenneth Lai, Sarah Moeller, Jiarui Yao, Tim O'Gorman, Andrew Cowell, William Croft, Chu-Ren Huang, et al. 2021. Designing a uniform meaning representation for natural language processing. *KI-Künstliche Intelligenz*, 35(3-4):343–360.

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

Rik van Noord, Antonio Toral, and Johan Bos. 2019. Linguistic information in neural semantic parsing with multiple encoders. In *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 24–31, Gothenburg, Sweden. Association for Computational Linguistics.

Rik van Noord, Antonio Toral, and Johan Bos. 2020. Character-level representations improve DRS-based semantic parsing even in the age of BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4587–4603, Online. Association for Computational Linguistics.

Chunliu Wang, Huiyuan Lai, Malvina Nissim, and Johan Bos. 2023a. Pre-trained language-meaning models for multilingual parsing and generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5586–5600, Toronto, Canada. Association for Computational Linguistics.

Chunliu Wang, Xiao Zhang, and Johan Bos. 2023b. Discourse representation structure parsing for Chinese. In *Proceedings of the 4th Natural Logic Meets Machine Learning Workshop*, pages 62–74, Nancy, France. Association for Computational Linguistics.

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*, pages 44–54, Seattle, Washington. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Yuekun Yao and Alexander Koller. 2022. Structural generalization is hard for sequence-to-sequence models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5048–5062, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiao Zhang, Chunliu Wang, Rik van Noord, and Johan Bos. 2024. Gaining more insight into neural semantic parsing with challenging benchmarks. In *Proceedings of the Fifth International Workshop on Designing Meaning Representations @ LREC-COLING 2024*, pages 162–175, Torino, Italia. ELRA and ICCL.

Jin Zhao, Nianwen Xue, Jens Van Gysel, and Jinho D. Choi. 2021. UMR-writer: A web application for annotating uniform meaning representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 160–167, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Statistics of Parallel Meaning Bank Release 5.1.0

In our experiment, we excluded all ill-formed DRGs from the gold split of the PMB5.1.0 dataset. Detailed statistics of the modified gold data as well as the silver and bronze splits are presented below.

Gold				**Silver**	**Bronze**
Train	**Dev**	**Test**	**TestLong**		
9560	1195	1193	40	146,718	141,435

Table 5: Number of sentences across different splits

B Challenges Brought by Scope

In this section, we show that AM-Algebra struggles with even one-box DRG when scope is taken as an argument of the root box, with sentence *The little cat wanted to sleep* as an example.

The graphs corresponding to the sentences are illustrated in Fig. 8. As depicted in Fig. 9, these graphs can be merged to form a scopeless lexical graph. However, integrating this lexical graph with a box requiring four arguments proves problematic for constructing the AM-tree. This is due to AM-Algebra's restriction against multiple APPs (applications) between two sub-graphs, a constraint that mirrors linguistic principles in English, where different parts of one constituent cannot play unique roles relative to another constituent.

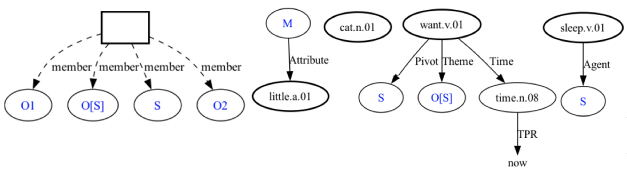


Figure 8: graphs for DRGs with scope as argument

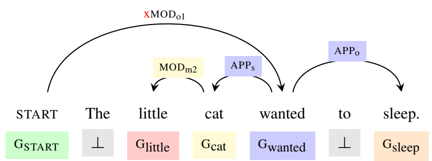


Figure 9: Failed combinations of graphs

C Heuristics on Edge Directions

The heuristics on edge directions can be found in Table 6.

Operation	Edge Labels
APP	Agent, Bearer, Participant, Creator, Proposition, Stimulus, Beneficiary, Co-Agent, Co-Patient, Co-Theme, Experiencer, Patient, Pivot, Product, Recipient, Theme, Owner, OF, User, Role, NEQ, APX, EQU, TPR
Mod	Consumer, Topic, Result, member, Sub, Source, Destination, Goal, Product, ALTERNATION, ATTRIBUTION, CONDITION, CONSEQUENCE, CONTINUATION, CONTRAST, EXPLANATION, NECESSITY, NEGATION, POSSIBILITY, PRECONDITION, RESULT, SOURCE

Table 6: Mapping of operation types to edge labels in the DRG-to-graph conversion process.

D Coreference Resolution

PMB5.1.0 explicitly marks coreference: two nodes that refer to the same entity are connected with an `ANA` edge.

In our approach, we leverage the AM Parser's supertagger for coreference resolution. In PMB, node labels are annotated with lexical categories like `n` (noun), `a` (adjective), `r` (adverb), and `v` (verb), such as `female.n.02` in Fig. 10. To allow coreference resolution via supertagging, we introduce a new category, denoted as `p` (pronoun). During preprocessing, this category is assigned to nodes involved in coreference, identified by the `ANA` edge linking them. For example, Fig. 10 shows the resulting penman notation after preprocessing and postprocessing steps. The two nodes, `s0` and `s3` (bot labeled `female.n.01`) are relabeled as `female.p.01`. This encodes the fact that the two entities corefer is now encoded in the node labels, allowing us to remove the `ANA` edge. While this is not always a lossless transformation when there are multiple instances of coreference in the graph, we find it to work well in practice (see Section 5). And crucially, this removes a reentrancy from the DRG, making it more likely to be decomposable by the AM algebra. At training time, the AM Parser's supertagger can then learn to distinguish regular nouns (i.e., `n`) and coreferent nouns (i.e., `p`).

At evaluation time, we reconstruct coreference information in a postprocessing step. This step begins with identifying nodes marked as `p` in predicted DRGs. However, if a DRG contains only one such `p`-tagged node, we do not treat it as coreferent, since coreference involves multiple entities. In most cases, the parser flags either one or two nodes as potential coreference candidates within

Preprocessing:
(b0 / box :member (s1 / unscrew.v.01 :Agent (s0 / female.~~n~~p.02 :Name (c0 / "Mary")) :Time (s2 / time.n.08 :TPR (c1 / "now")) :Patient (s4 / lipstick.n.01 :User (s3 / female.~~n~~**p**.02 ~~:ANA s0~~))))
Postprocessing:
(b0 / box :member (s1 / unscrew.v.01 :Agent (s0 / female.**n**.02 :Name (c0 / "Mary"))) :Time (s2 / time.n.08 :TPR (c1 / "now")) :Patient (s4 / lipstick.n.01 :User (s3 / female.**n**.02 **:ANA s0**))))

Figure 10: An example of coreference after preprocessing and postprocessing for the sentence *She$_i$ unscrewed her$_i$ lipstick.*

a single DRG. When two nodes are both tagged as p, we compare their node concepts to see if they are identical. In our example (Fig. 10), since both nodes are labeled `female.p.02`, indicating a match, we create an `ANA` edge linking them. This edge is directed from node with a larger number on the node label (like `s3`) to the one with a smaller node label (like `s1`). The final step is to change the nodes' categories from `p` back to `n`.

E Implementation details of the scope dependency parser

The original implementation of Dozat and Manning (2018) uses POS tags, lemma-, and character-level word embeddings, processed through a BiLSTM and a Forward Network (FNN), to predict if there is an edge between two tokens as well as the corresponding edge label. Then a biaffian classifier is used to predict the existence of an edge and the edge label.

In our experiment, we fine-tune `roberta-large` (Liu et al., 2019) and take POS tags and characters as feature embeddings. All the linguistic information is provided by spaCy[4] (Honnibal et al., 2020). We keep all other hyperparameters the same as the best model reported in their paper.

F Scope Annotation of a Complex Example

As discussed in Section 4, when a single token aligns with a lexical graph that contains multiple nodes or boxes, it creates a complex scenario where different nodes within the same lexical graph are linked to distinct boxes and complicates the establishment of straightforward one-to-one dependency relations between tokens. Our annotation method is straightforward: as long as an aligned lexical graph contains multiple nodes or boxes, we make the scope assignment of each node explicit in a top-down order.

We illustrate our method with two other possibilities when we build the dependency edges between lexical graphs aligned with tokens.

(1) the two lexical graphs aligned with the token have multiple nodes and boxes respectively, and each node is assigned a different scope box. An example can be found in the scope assignment between the lexical graph aligned with *born* (G_{born}) and the lexical graph aligned with *all* (G_{all}). We can see that the bottom node of G_{born} receives the scope from the top box of G_{all}, while the top node of G_{born} receives the scope from the bottom box of G_{all}. In this case, the dependency edge between *born* and *all* is `scope_b3_b2`.

(2) the two lexical graphs aligned with the token have multiple nodes and boxes respectively, and each node is assigned the same scope box. This case can be found in the scope assignment between the lexical graph aligned with *children* (G_{children})and that with *all*. Although both nodes of G_{children} receives the same scope, we still explicitly annotating the scope for each node as shown in `scope_b2_b2`.

G Evaluation Format

In evaluation, we use a more compact format following Wang et al. (2023b). This strict format integrates synset nodes' information into a single entity and eliminates variables representing constants, thereby avoiding inflated scores. An example is shown in Fig. 12.

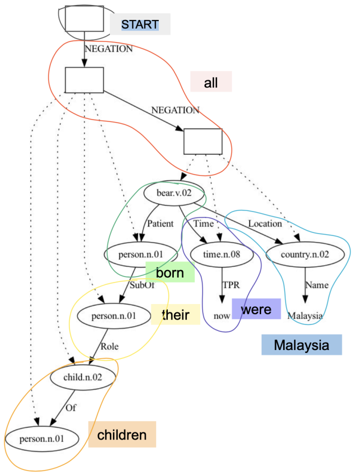
H Hyperparameters in AM Parser

The hyperparameters used in the experiments that show the best performance on the scopeless SBN training data are summarized in Table 7.

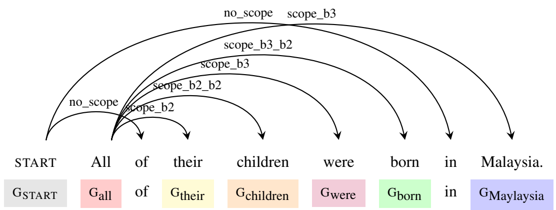
I More examples of Complex DRGs

In this section, we show examples of more complex scope assignments in Fig 13.

[4]We use version 3.7.2



(a) Token Alignments: lexical graphs are color-coded to indicate alignment with distinct tokens, denoted beneath each respective circle



(b) The converted dependency graph based on the scope information represented in dashed lines

Figure 11: Complex DRG and dependency graph for *All of their children were born in Malaysia.*

```
(b0 / "box"
:member (s0 / "synset"
    :lemma "person"
    :pos "n"
    :sense "01"
    :Name (c0 / "?"))
:member (s1 / "synset"
    :lemma "time"
    :pos "n"
    :sense "08"
    :TPR (c1 / "now"))
:member (s2 / "synset"
    :lemma "male"
    :pos "n"
    :sense "02"
  :Name (c2/"William W"))
:member (s3 / "synset"
    :lemma "defeat"
    :pos "v"
    :sense "01"
    :Co-Agent s0
    :Time s1
    :Agent s2))
```

(a) Lenient Format used in (Poelman et al., 2022)

```
(b0 / box
:member (s0 / person.n.01
    :Name "?")
:member (s1 / time.n.08
    :TPR "now")
:member (s2 / male.n.02
   :Name "William Wallace")
:member (s3 / defeat.v.01
    :Co-Agent s0
    :Time s1
    :Agent s2))
```

(b) Strict Format

Figure 12: Comparison of DRG Representation in Lenient and Strict Formats for the sentence *Who did William Wallace defeat?*

Hyperparameter	Value
Activation function	tanh
Optimizer	Adam
Learning rate	0.001
Epochs	100
Early Stopping	20
Dim of lemma embeddings	64
Dim of POS embeddings	32
Dim of NE embeddings	16
Minimum lemma frequency	7
Hidden layers in all MLPs	1
Hidden units in LSTM (per direction)	256
Hidden units in edge existence MLP	256
Hidden units in edge label MLP	256
Hidden units in supertagger MLP	1024
Hidden units in lexical label tagger MLP	1024
Layer dropout in LSTMs	0.35
Recurrent dropout in LSTMs	0.4
Input dropout	0.35
Dropout in edge existence MLP	0.0
Dropout in edge label MLP	0.0
Dropout in supertagger MLP	0.4
Dropout in lexical label tagger MLP	0.4

Table 7: Common hyperparameters used in all experiments in AM Parser.

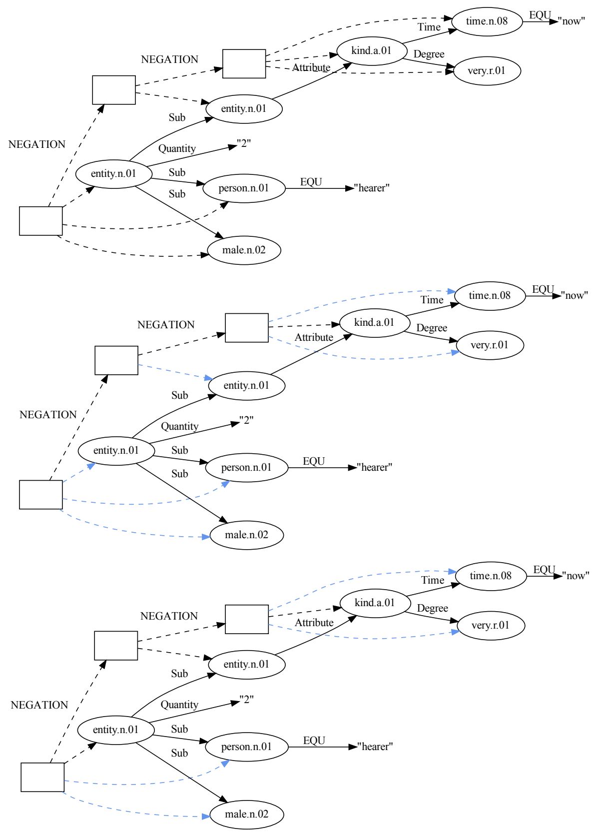


Figure 13: Examples of complete DRG (top), scopeless DRG(middle), and simplified DRG (bottom) for the sentence *You and he both are very kind.*