Normalizing Compositional Structures Across Graphbanks

Lucia Donatelli, Jonas Groschwitz, Alexander Koller,
Matthias Lindemann, Pia Weißenhorn
Department of Language Science and Technology
Saarland Informatics Campus
Saarland University, Germany
{donatelli,jonasg,koller,mlinde,piaw}@coli.uni-saarland.de

Abstract

The emergence of a variety of graph-based meaning representations (MRs) has sparked an important conversation about how to adequately represent semantic structure. MRs exhibit structural differences that reflect different theoretical and design considerations, presenting challenges to uniform linguistic analysis and cross-framework semantic parsing. Here, we ask the question of which design differences between MRs are meaningful and semantically-rooted, and which are superficial. We present a methodology for normalizing discrepancies between MRs at the compositional level (Lindemann et al., 2019), finding that we can normalize the majority of divergent phenomena using linguistically-grounded rules. Our work significantly increases the match in compositional structure between MRs and improves multi-task learning (MTL) in a low-resource setting, serving as a proof of concept for future broad-scale cross-MR normalization.

1 Introduction

Graph-based representations of sentence meaning offer an expressive and flexible means of modeling natural language semantics. In recent years, a number of different *graphbanks* have annotated large corpora with graph-based semantic representations of various types (Oepen and Lønning, 2006; Ivanova et al., 2012; Banarescu et al., 2013; Abend and Rappoport, 2013). Because of differences in graphbank design principles, individual graphs can differ greatly and often in fundamental strategies (Oepen et al., 2019). Fig. 1 illustrates distinct graphs from the three graphbanks of the SemEval 2015 Shared Task on Semantic Dependency Parsing, which we take as our focus in this paper. The graphs visibly differ with respect to edge labels, edge directions, the treatment of a periphrastic verb construction, and coordination.

In this paper, we develop a methodology to (i) understand the nature of the differences across graphbanks and (ii) normalize annotations from different graphbanks at the level of their compositional structures. This approach allows us to identify which design differences between graphbanks are linguistically meaningful and important for the design of future corpora; as well as to identify more unified approaches to certain structures to potentially facilitate multi-task learning (MTL). In Section 2 we build upon Lindemann et al. (2019), who used *AM dependency trees* to represent the compositional structure of graph-based meaning representations (MRs) based on the *AM algebra* (Groschwitz et al., 2018). We detail a new methodology for identifying and quantifying mismatches between the compositional structures assigned to the same sentence by different graphbanks; we then present our extended *AM+ algebra*, with which we can systematically reshape these compositional structures to align them across graphbanks (Section 3). Finally, we provide key examples of how our methodology normalizes specific linguistic phenomena that pose challenges to MR design and differ in representation across graphbanks (Section 4).

Using our methods, we increase the match between the AM dependency trees (compositional structure) of the different graphbanks to 76.3 (DM-PAS), 78.8 (DM-PSD), and 82.0 (PAS-PSD) directed unlabeled F-score, compared to 63.5, 55.7, and 57.0 for Lindemann et al.'s AM dependency trees (Section 5). This is

Proceedings of the 28th International Conference on Computational Linguistics, pages 2991–3006
Barcelona, Spain (Online), December 8-13, 2020

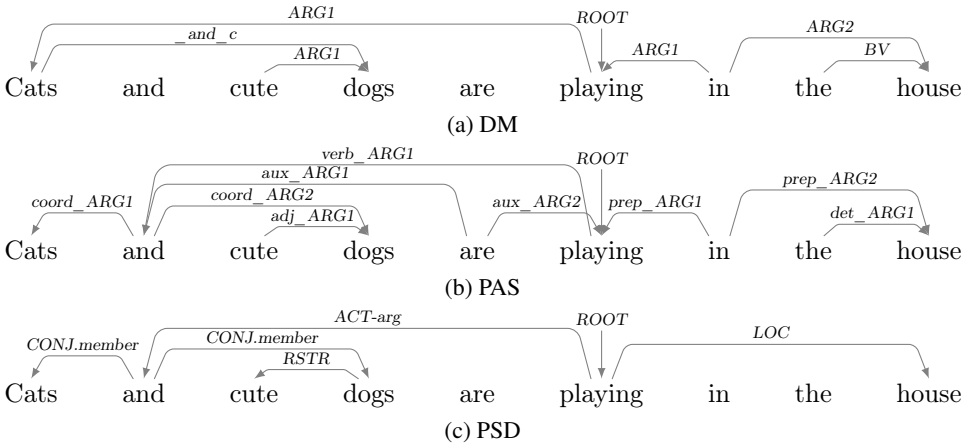


Figure 1: Examples for DM, PAS and PSD graphs. The PSD graph includes Lindemann et al.'s (2019) reversible preprocessing of coordination.

a drastic improvement over the unlabeled F-scores of 64.2, 26.1, and 29.6 for the original graphs (Oepen et al., 2015). We additionally demonstrate that when training a graph parser on a tiny graphbank combined with larger graphbanks using multi-task learning (MTL), our methods improve parsing accuracy by up to 2.6 points F-score over MTL without normalized AM dependency trees. Our work serves as a proof of concept that linguistically-informed methods to normalize compositional structure across graphbanks is successful and can be applied at broad-scale and potentially to more complex graphbanks (Section 6).

2 Background

2.1 The Graphbanks

We focus on the three graphbanks of the SemEval 2015 Shared Task on Semantic Dependency Parsing (SDP): (i) DELPH-IN MRS-Derived Semantic Dependencies (DM), (ii) Enju Predicate–Argument Structures (PAS), and (iii) Prague Semantic Dependencies (PSD) (Oepen et al., 2015). All graphbanks have parallel semantic annotations over the same English texts: the Wall Street Journal (WSJ) and Brown segments of the Penn Treebank (PTB; Marcus et al. (1993)). The graphbanks are bilexical, node-ordered, labeled, directed graphs, representing core semantic information like predicate-argument relations.

The SDP graphbanks make different choices about which linguistic information to represent and how to represent it (first columns of Table 2) (Oepen et al., 2016). Representative graphs of the three graphbanks containing linguistic phenomena of interest (modification, coordination, copula, prepositional phrase) are shown in Fig. 1. While in all three graphs each token of the sentence corresponds to at most one node, there may also be tokens which are not part of the graph. These 'ignored' tokens differ across graphbanks; for instance, in DM coordinators ("and") and temporal auxiliaries ("are") are not nodes in the graph, while PSD ignores temporal auxiliaries, prepositions, and determiners. The edges are also different; for instance, edges that point from adjectives to nouns in DM and PAS go the other way in PSD.

The fundamental question we address in this paper is to what extent these variations are superficial and artifacts of graph design choice, and to what extent they are deeper and structurally rooted in complex linguistic phenomena. To this end, we make the standard assumption from theoretical semantics that MRs are derived *compositionally* by systematically computing MRs for each part of the sentence from its immediate subparts. We say that a difference between two graphs is 'superficial' if it can be derived using the same compositional structure. To give an example, Fig. 4(a, b) show the compositional structure which Lindemann et al. (2019) assign to the DM and PSD graphs in Fig. 1. We explain the technical details in Section 2.2; for now, observe that certain differences between DM and PSD have been normalized (e.g. both dependency trees analyze the noun as the head and the adjective as the modifier), while others are still present (e.g. ignored tokens).

2.2 Compositional Semantic Parsing with the AM Algebra

The *Apply-Modify (AM) Algebra* was initially presented in Groschwitz et al. (2018) for compositional AMR parsing. It provides the latent compositional structures we aim to normalize, in the form of *AM dependency trees* (explained below). Lindemann et al. (2019) (hereafter referred to as L19) extended the AM algebra to parsing four additional graph-based MRs (DM, PAS, PSD, and EDS); Donatelli et al. (2019) extended this work to UCCA. L19 achieved state of the art results for all formalisms in question with pretrained BERT embeddings and manual heuristics tailored to each graphbank. We adapt these heuristics to find uniform structures that generalize across graphbanks.

To illustrate the algebra, consider the example AM dependency tree in Fig. 3e, which constructs the PAS graph in Fig. 3b. The edges correspond to AM algebra operations, and each word is assigned a *graph constant*, specifically an *as-graph*, written below the word (as a visual help, Fig. 2c shows the PAS graph of Fig. 3b in a style more compatible with the graph constants). As-graphs possess two kinds of markers: (i) an obligatory *root source* on the *root node* (indicated with bold outline); and (ii) optional *sources* (marked in red, e.g. S and O) on other nodes, which allow as-graphs to be combined. Every node can bear at most one source, and each source can occur at most once in an as-graph. Moving forward, we will write G_{the} for the graph constant at "the", G_{cat} for "cat", and so on.

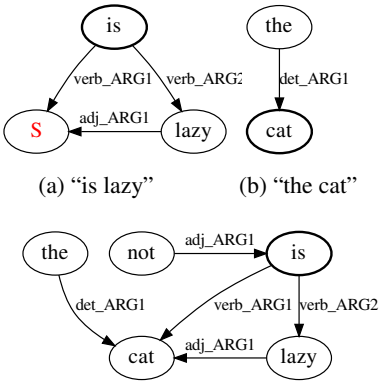


Figure 2: Partial results of the Fig. 3e AM dependency tree and final PAS graph for "the cat is not lazy".

The two kinds of edge labels in the AM dependency tree reflect the two operations of the AM algebra. First, the *Apply* operation APP_{O} on the edge from the head "is" to the argument "lazy" fills the O-source of G_{is} with the root of G_{lazy}; in other words, it plugs the root of the argument into the specified source of the head.[1] The result is shown in Fig. 2a. If there are nodes in the two graphs that have the same source (like here for the source S), those nodes are merged. The type annotation $[\text{S}]$ at the O-source of G_{is} requests the argument to have an S-source (which G_{lazy} does). This way, type annotations lead to reentrancies in the graph, such as the triangle structure here.

Second, the *Modify* operation allows an as-graph to modify a head. For example, the MOD_{DET} operation encoded by the edge from "cat" to "the" yields the as-graph shown in Fig. 2b. Here, G_{cat} is the head and G_{the} the modifier; G_{the} attaches with its DET-source at G_{cat} and loses its own root source.

The APP_{S} edge from "is" to "cat" combines these two intermediate results, filling the S-source of Fig. 2a with the root of Fig. 2b. The final graph in Fig. 2c is then the result of the MOD_{M} operation between "not" and "is". Note that AM dependency trees do not always cover all words in a sentence: in the AM dependency tree for this sentence's DM graph (Fig. 3d), the word "is" is not covered, reflecting the fact that it has no incident edges in the DM graph (Fig. 3a). If a token is not covered by the AM dependency tree, we assign it no graph constant and write '$\perp$' instead.

3 Quantifying and Normalizing Compositional Mismatches

The AM dependency trees of L19 (henceforth, *L19 AM trees*) naturally normalize some of the differences of the original MRs we consider in this paper (Fig. 3). Here, we go further and ask: how far can this normalization be pushed? Given our corpus, if we can find a single compositional structure for each sentence that corresponds to all three MR graphs, this suggests that the designs and annotators of the three graphbanks tacitly agree on compositional structure; surface differences between the graphs can then be explained by low-level representations of individual words. In this work, we take a first step towards this goal, devising linguistically motivated changes to the compositional structures that we can apply automatically to all sentences in the corpus to normalize them.

[1] Note: "lazy" is represented in Fig. 3e with an S-source for ease of understanding here; L19 analyze it as a M-source. We use S-sources for adjectives in PAS in the presentation of all examples.

Fig. 3(g, h, i) shows a successful normalization by our method: the edges of the AM trees are the same across all three MRs; all differences are delegated to the graph constants. Take for example the determiner "the", which is originally ignored in PSD but not in DM or PAS. In the normalized PSD AM tree (Fig. 3i), it has a constant that consists of an unlabeled root marked with a DET-source, and has an incoming MOD_{DET} edge from "cat". When this semantically empty graph modifies the graph for "cat" G_{cat} (the MOD_{DET} operation), it does not in fact change that graph G_{cat} but just merges with its root without leaving a trace. This way, the normalized AM tree in Fig. 3i still evaluates to the original graph in Fig. 3c but includes the added MOD_{DET} edge to match DM and PAS. The trees also normalize the copula and negation, which requires more subtle methods detailed in Section 4.

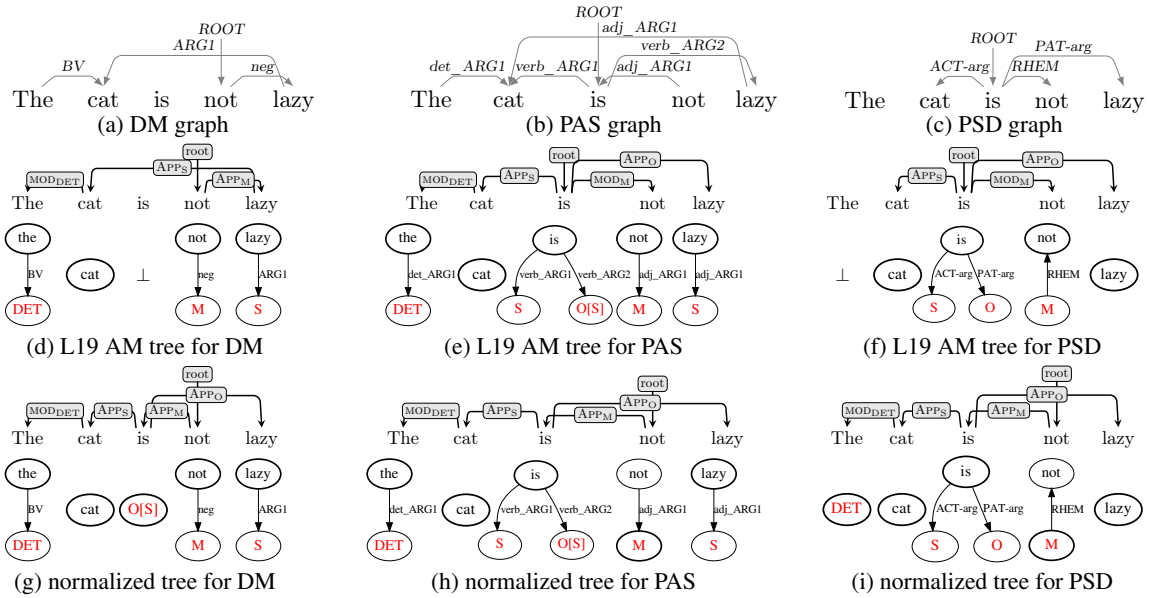


Figure 3: SDP graphs, L19 AM trees and normalized trees for the sentence "The cat is not lazy"

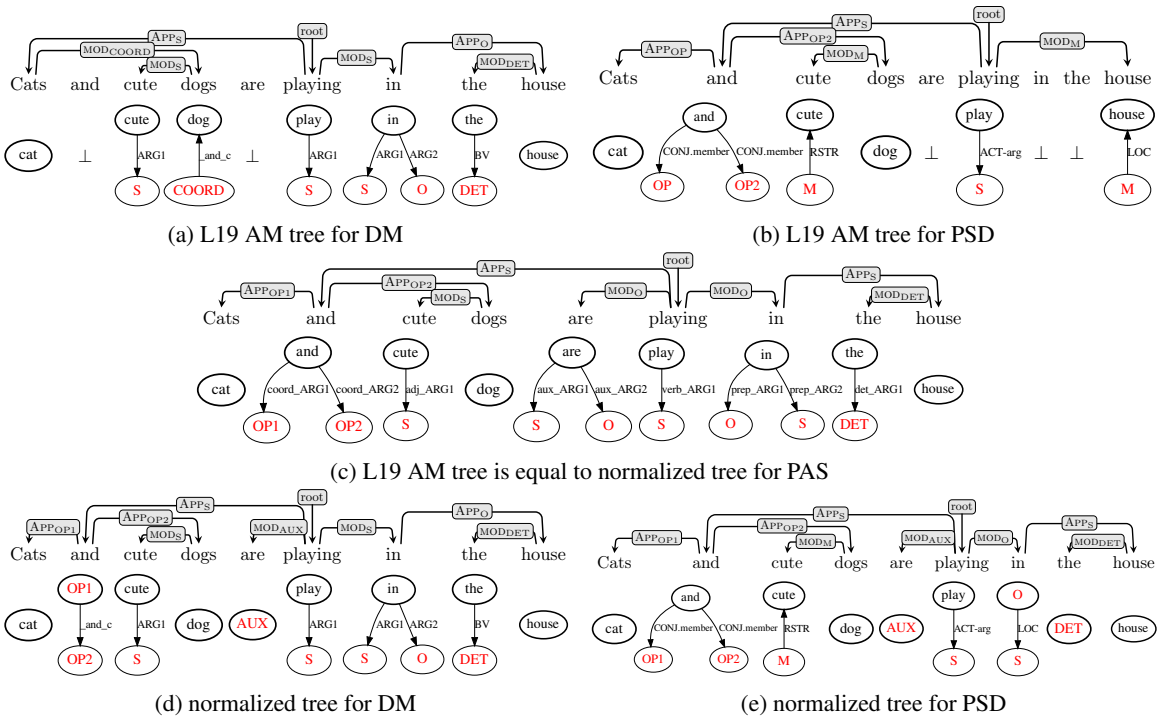


Figure 4: L19 AM trees and normalized trees for SDP graphs presented in Fig. 1.

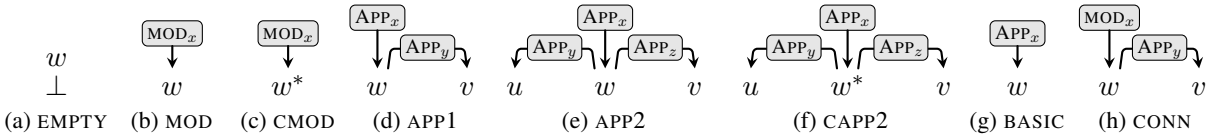


Figure 5: The set of local patterns for w we distinguish based on their structures in L19 AM trees.

3.1 The AM+ Algebra

The graph constant we introduced for "the" poses a technical challenge: technically, it is invalid because its root node is marked with another source (DET). This is necessary to make the constant semantically vacuous, but the AM algebra does not allow root nodes to also have another source besides the root source. We therefore extend the AM algebra to the *AM+ algebra*, which is defined exactly like the AM algebra but allows root nodes to contain additional sources. The concept of an AM dependency tree thus generalizes naturally to the AM+ algebra. Fig. 3(g-i) become valid AM dependency trees of the AM+ algebra and evaluate to the original graphs in Fig. 3(a-c).

The seemingly innocent step from the AM algebra to the AM+ algebra has both formal and practical consequences. On a technical level, the AM algebra is grounded in the HR graph algebra (Courcelle and Engelfriet, 2012). In this translation, root nodes in the AM algebra are mapped to sources with a special name ("root") in the HR algebra, and a node in a graph of the HR algebra may not be marked with more than one source at once. Formally, we thus base the AM+ algebra on the extension of the HR algebra presented in van Harmelen and Groschwitz (2020) that allows multiple sources at one node. The AM+ algebra also makes it possible to create loops and multiple edges between two nodes, from constants that do not have them (detailed in Appendix A). This is undesirable here since SDP graphs only contain single non-loop edges; we find in Section 5 that while loops and multiple edges can be created accidentally during parsing, this occurs only rarely. From a modeling perspective, the extra flexibility of the AM+ algebra means that the same graph can be described with more terms of the AM+ algebra than with terms of the AM algebra. This can be useful – in fact, we rely on this flexibility here – but it also means that the choice of compositional structure must be made with care and, ideally, in a linguistically informed manner. We therefore first classify and quantify the compositional mismatches.

3.2 Classifying compositional mismatches with local patterns

To normalize the compositional discrepancies across the large graphbanks, it is imperative that we can apply our changes automatically to each sentence. At the same time, we want to ensure that the changes we make are linguistically reasonable to keep the flexibility of the AM+ algebra in check. We therefore develop a set of *local patterns* in the L19 AM trees that correspond to individual tokens. We then compare these local patterns across the L19 AM trees for the three graphbanks to uncover a *pattern signature*, with which we can identify and normalize differences across the graphbanks.

The patterns we use are based on incoming and outgoing AM dependency edges of a token, and on a property of the graph constant; Fig. 5 lists all patterns. For example, the pattern EMPTY matches tokens that are ignored in the AM dependency tree, the pattern MOD matches tokens with an incoming MOD edge (e.g. adjectives), and APP1 matches heads with one argument (e.g. intransitive verbs). Patterns where w has an incoming APP$_x$ edge also apply when w is the root of the AM tree. The patterns do not take outgoing MOD edges of w (i.e. modifiers of w) into account. The star at w in the 'complex' CMOD and CAPP2 patterns indicates that the graph constant of w causes a reentrancy, like the [S] annotation of G_{is} in Fig. 3e causing the reentrancy "is"–"lazy"–"cat" in Fig. 3b as seen in 2.2. This is an example of CAPP2; an example of CMOD is "are" in Fig. 4c. The distinctions MOD/CMOD and APP2/CAPP2 do not reflect differences in the AM dependency edges and do not require normalization, but they are useful in distinguishing linguistic phenomena. Tokens matching none of the patterns are classified as OTHER.

Pattern signatures inform us how tokens are represented across the graphbanks. As an example, consider the determiner "the" in Fig. 3. In the L19 AM trees (d–f): the DM (d) and PAS (e) trees combine the determiner with a MOD relation, while the PSD tree (f) leaves it absent ($\perp$). That is, we have the patterns MOD for DM and PAS, and EMPTY for PSD. This creates a pattern signature with the triple [DM / PAS / PSD]: [MOD / MOD / EMPTY]. We use this order for all pattern signatures. In the next section, we analyze

Rank	Pattern signature ([DM / PAS / PSD])	Most commonly associated phenomena	Count	Percentage of total
1	[MOD / MOD / EMPTY]	Determiners	49 965	13.38%
2	[EMPTY / MOD / EMPTY]	Punctuation*	45 544	12.20%
3	[CONN / CONN / EMPTY]	Prepositions	42 395	11.35%
4	[BASIC / BASIC / MOD]	NPs with prep.*	30 202	8.09%
5	[EMPTY / CONN / EMPTY]	Prepositions	22 324	5.98%
8	[EMPTY / CMOD / EMPTY]	Temporal auxiliaries	14 074	3.77%
10	[APP1 / MOD / MOD]	Verbal Negation*	7215	1.93%
14	[EMPTY / APP2 / APP2]	Coordination	5709	1.53%
20	[EMPTY / CAPP2 / APP2]	Copula (adjectival)	2915	0.78%
21	[EMPTY / OTHER / OTHER]	Coordination	2828	0.76%

Table 1: Pattern signatures we fix. Asterisks indicate that additional phenomena comprise a significant portion of the pattern. See Appendix B for complete list of pattern signatures we detect.

the pattern signatures obtained with this method and normalize the compositional structures based on that analysis.

4 Normalizing Key Compositional Structures Across Graphbanks

Examining the pattern signatures of distinct tokens, we make a key finding: about half of the nodes exhibit the same pattern in all three graphbanks. These *aligned* patterns are typically in the form of open-class words and predicate-argument structure, core components of MR design. For example, adjectives often display the signature [MOD / MOD / MOD], and nouns [BASIC / BASIC / BASIC]. Uniform pattern signatures represent a priori shared compositional structure across graphbanks that we do not need to normalize.

For *unaligned* pattern signatures, i.e. ones that display a pattern mismatch across graphbanks, our goal is to align the pattern signature so the patterns for each graphbank are equivalent. This normalization process follows a three-step process: (i) identify the pattern signatures with distinct local patterns for the three MRs; (ii) match these patterns to meaningful linguistic phenomena; and (iii) manipulate the compositional structures to align the pattern signature across graphbanks, using the AM+ algebra. Table 1 shows the pattern signatures we have investigated in detail and normalized; we chose them as a combination of the most frequent unaligned pattern signatures, and signatures that correspond to interesting and challenging linguistic phenomena. A full list of signatures we detect is in Appendix B. As with aligned pattern signatures, these unaligned signatures map quite precisely to linguistic phenomena—specifically, phenomena involving closed-class functional words, elements often peripheral in MR design.

A close investigation of the unaligned pattern signatures reveals three main disparities between the graphbanks that we can now address with the AM+ algebra: (i) 'ignored' elements; (ii) disagreement in headedness of constructions; and (iii) relational constructions. Here we detail representative examples for each case; Appendix C has additional examples. For all normalizations, we apply transformations using the AM+ algebra directly to the AM dependency trees of L19. The transformations are carefully designed not to change the graph the AM dependency trees evaluate to. We do this for all AM dependency trees by iterating over the sentences in the graphbanks and applying a transformation whenever its corresponding pattern signature matches. The transformations are always applied in the same order (Table 2).

4.1 Copula and "Ignored" Elements

A common difference between graphbanks is the 'ignoring' of tokens, when certain tokens do not contribute any edges or nodes to the final graph. Among the SDP graphs, PSD consistently 'ignores' the most token types; PAS includes nearly all tokens in the graph; and DM falls somewhere in the middle. Importantly, in the corresponding L19 AM trees, ignored tokens are not part of the dependency tree.

Section 3.2 already discussed how we normalize determiners like "the" in Fig. 1. Copula constructions with adjectival predicates are another prime example of this category, exhibiting the [EMPTY / CAPP2 / APP2] pattern. This is seen for the sentence "The cat is not lazy" in Fig. 3(a-c)). DM treats this

construction like an adnominal adjectival construction ("cute dogs" in Fig. 1a); as such, the copula does not contribute directly to the graph (EMPTY pattern). PSD and PAS choose an alternative strategy: the copula verb functions as a transitive verb. PAS includes an edge from the adjective to the subject, resulting in a reentrancy in the graph (Section 2.2) and the CAPP2 pattern in the AM tree. PSD expresses the relation between the adjective and the subject only via the copula. This results in the APP2 pattern in the AM tree. Though APP2 pattern (PSD) differs from CAPP2 pattern (PAS), both result in same AM dependency tree: the verb is the root with outgoing APP_S and APP_O edges to the subject and the predicate respectively (Fig. 3(e, f)). Thus, only DM exhibits a different compositional structure.

To address this discrepancy between DM on one hand and PAS/PSD on the other, we use the AM+ algebra to create vacuous lexical as-graphs that do not change the graph when combined. Specifically, we align the AM dependency tree of DM to that of PSD and PAS by adding a vacuous graph constant for the DM copula verb. This new constant consists of one root node with an O-source, requiring an S-source in its argument (see "is" in Fig. 3g). To evaluate this new AM tree, the adjective ("lazy") is first plugged into the O-source of the copula ("is"), resulting in an as-graph identical to the lexical as-graph of the adjective. The S-source is then filled, plugging the subject into the correct slot of the adjective. As an added perk, this solution allows us to use the same DM adjective as-graph for adnominal and predicative uses. We restrict our copula fix to instances which have a node with the lemma "be" in PSD plus exactly two outgoing edges APP_S and APP_O where the target node of the latter has an adjectival POS tag.

In addition to determiner and copula constructions, we use the above methods to normalize punctuation, particles, and temporal and aspectual auxiliary verbs (see Appendix C). Together these cases account for roughly 30% of the tokens that exhibit a pattern difference.

4.2 Verbal Negation and Graph Headedness Challenges

A second discrepancy between graphbanks arises from disagreement about which token is the head of a given construction. One such construction is verbal negation (roughly 2% of tokens that exhibit a pattern difference). Verbal negation follows the [APP1 / MOD / MOD] pattern: while PSD and PAS treat the negation as modifier, DM usually considers the negation to be the head (Fig. 3(a-f)).

We address this challenge by switching the headedness between graphbanks, making the negation the head in the AM dependency trees for PAS and PSD graphs. This transformation resembles type raising in lambda calculus where functor and argument are exchanged. An example is shown for the PSD dependency tree in Fig. 3f and its transformation in Fig. 3i (the process is the same for PAS). The AM+ algebra allows us to change the graph constant for "not" by moving the root source to the node with the NEG-source. As a result, the dependency relation between the former head ("is" in Fig. 3f) and the negation is flipped and the label changed from MOD to APP. We restrict our negation fix to instances which have a node with the lemma "#Neg" or "never" in PSD. This covers 35% of the pattern; other phenomena adhering to this pattern include some discourse connectors and adverbs.

Besides conceptual differences in headedness like negation, some phrases simply do not have an obvious head, such as a date like "Nov. 29". In such cases, the MRs also often disagree on whether "Nov." or "29" is the head. We leave the automatic detection and transformation of such cases to future work.

4.3 Binary Coordination and Relational Elements

As a third category, the SDP graphbanks differ in how they mark certain linguistic relations such as in coordination or prepositional phrases. We observe two strategies: (i) the relation is marked using only an edge between the two more 'contentful' elements, ignoring the functional word; or (ii) the relation is marked using a node for the functional word with outgoing edges to the two elements. These decisions permeate through to the AM dependency trees. We address the two most common phenomena of this category here, coordination and prepositions, which account for about 20% of all divergences. In both cases, we normalize the AM dependency trees towards strategy (ii).

In coordination, DM uses an edge (strategy (i)) to connect two conjuncts while PAS and PSD[2] make the conjunction "and" the head of the phrase with outgoing edges to the conjuncts (strategy (ii)). This

[2] As preprocessed by L19.

can be seen in Fig. 1 for the coordinate phrase "cats and cute dogs", where the edge in DM is '_and_c'. PAS and PSD exhibit the same APP2 pattern in the L19 AM trees (Fig. 4(b, c)), while DM exhibits the EMPTY pattern. If both conjuncts share an argument, PAS and PSD exhibit the OTHER pattern due to the additional APP child for this argument (for an example, see Appendix C.2). Binary coordination can thus have either the [EMPTY / APP2 / APP2] or [EMPTY / OTHER / OTHER] signature.

Binary coordination makes up approximately 86% of all coordinations. We leave automatic normalization of coordination of three or more conjuncts to future work. We restrict our fix to nodes with *CC* POS tag and exactly two outgoing '*.member' edges in PSD. This covers 89% of the two patterns. Our fix consists of adding a lexical as-graph for the conjunction token (e.g. "and") in DM containing the dedicated conjunction edge that was previously part of one of the conjuncts graph constants (compare Fig. 4a and 4d). The incident nodes use OP1- and OP2-sources to indicate the open positions for the conjuncts, and we connect the conjuncts as APP_{OP1} and APP_{OP2} children accordingly. One of the conjuncts was previously the head of the coordination in DM (here "cats"), we shift its incoming edge (here APP_{S}) to the conjunction instead. The root of the conjunction's lexical as-graph is placed at the OP1-source where "cats" will be placed, such that all outside graph edges attach correctly. For coordination with a common argument, source annotations at OP1- and OP2-sources are added for this argument (see Appendix C.2).

For prepositions, DM and PAS use strategy (ii) for most prepositions and encode them as nodes, whereas PSD uses strategy (i) and encodes them as edges. See the preposition "in" in Fig. 1, to which the PSD 'LOC' edge corresponds. In the AM trees, these prepositions have the [CONN / CONN / EMPTY] signature. As a fix we normalize the PSD AM trees to match the others by introducing a new lexical as-graph for the preposition that consists of only the graph edge corresponding to the preposition. For example in Fig. 4e, we use the 'LOC' edge as the lexical as-graph for the preposition "in". We update the rest of the graph accordingly, in particular removing the 'LOC' edge from G_{house}, which changes the MOD pattern at "house" (Fig. 4b) to the more consistent BASIC pattern (Fig. 4e). In total, we have changed the [CONN / CONN / EMPTY] signature at "in" and the [BASIC / BASIC / MOD] signature at "house" to [CONN / CONN / CONN] and [BASIC / BASIC / BASIC], addressing two of the most common pattern mismatches at once. More details on our treatment prepositions can be found in Appendix C.

5 Evaluation

We evaluate the quality of the normalized AM dependency trees by measuring cross-formalism similarity and suitability for parsing on the SDP corpora of the 2015 shared task (Oepen et al., 2015). We apply L19's preprocessing of coordination for PSD. Due to the similarities of the graphbanks, it is appealing to use multi-task learning (MTL), which shares a subset of the model weights across graphbanks to allow training data for one graphbank influence the prediction on another. We expect this to help particularly in scenarios where there is limited data for one graphbank but large amounts for others.

5.1 Structural Similarity

Table 2 summarizes the structural comparison of the graphs and AM dependency trees, respectively. We also calculate the percentage of lexical as-graphs that change by graphbank as a result of our normalization: 13.53% (DM), .35% (PAS), and 33.22% (PSD).

There is a consistent increase in similarity by at least 12.3 points F-score of the normalized AM dependency trees in comparison to the L19 AM trees over all metrics. The A/M F-score is the upper bound on how much similarity can be achieved by systematically making the source names more uniform as well. Comparing the similarities of the normalized AM dependency trees to the similarities of the graphs, there is an even larger gain in the unlabeled, directed F-score of up to 52.7 points. Importantly, we only compute F-scores on the edges of the AM dependency trees, not on the as-graph constants. The higher similarity in the tree structure is achieved primarily by moving discrepancies to the lexicon, particularly for edge directions and edge labels.

		O15		L19		ΔDet		ΔAux		ΔPrep		ΔCoord		ΔCopula		ΔNeg		Δ Punct		All changes	
		PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD	PAS	PSD
DM	UUF	67.2	56.8	68.6	58.5	0.0	6.7	1.2	0.8	3.3	9.8	2.8	3.2	0.7	0.8	0.3	0.2	4.0	1.1	**80.9**	**81.1**
	UF	64.2	26.1	63.5	55.7	0.0	6.8	1.3	0.8	3.3	9.9	2.8	3.2	0.6	0.6	0.5	0.5	4.3	1.3	**76.3**	**78.8**
	A/M F			57.8	46.7	0.0	7.2	1.3	1.0	3.4	11.3	2.8	3.2	0.6	0.7	0.5	0.5	4.5	1.8	**70.9**	**72.4**
	LF			29.7	26.2	0.0	8.1	1.6	1.4	3.6	1.3	2.6	2.9	0.6	0.7	0.5	0.4	5.3	3.9	**43.9**	**44.9**
PAS	UUF		54.9		59.9		5.7		1.3		13.4		0.0		0.0		0.0		4.3		**84.6**
	UF		29.6		57.0		5.9		1.3		13.4		0.0		0.0		0.0		4.4		**82.0**
	A/M F				50.1		6.1		1.4		13.7		0.0		0.0		0.0		4.6		**75.9**
	LF				36.8		6.7		1.5		14.2		1.1		0.0		0.0		4.9		**65.2**

Table 2: F-scores between representations (with punct.). O15 are F-scores on SDP graphs (Oepen et al., 2015); else F-scores on AM dependency trees at different stages of normalization. UUF is undirected and unlabeled F, UF is unlabeled F, A/M F is labeled F without source names and LF is labeled F.

Training data		L19 AM trees					Normalized AM dependency trees				
		100			full		100			full	
		Single	MTL	Δ	Single	MTL	Single	MTL	Δ	Single	MTL
DM	id F	65.6±1.4	72.2±0.9	6.6	92.8±0.1	92.9±0.1	66.1±1.2	72.9±0.2	6.8	92.3±0.2	92.2±0.1
	ood F	63.3±1.3	69.1±1.4	5.8	88.9±0.1	88.9±0.1	63.0±1.2	69.5±0.5	6.5	88.0±0.3	87.7±0.3
PAS	id F	72.6±1.2	81.5±0.2	8.9	94.7±0.1	94.6±0.1	72.5±1.4	84.1±0.8	11.5	94.6±0.1	94.5±0.1
	ood F	69.2±2.0	78.9±0.7	9.7	92.6±0.1	92.6±0.1	69.0±1.6	80.7±1.0	11.7	92.5±0.1	92.6±0.1
PSD	id F	51.1±0.7	58.9±1.2	7.8	81.2±0.1	80.9±0.3	52.9±1.8	60.2±0.9	7.4	80.5±0.1	80.3±0.1
	ood F	50.3±2.2	58.2±1.8	7.9	80.3±0.1	80.1±0.2	51.6±1.2	59.1±0.8	7.5	79.7±0.1	79.5±0.1

Table 3: Means and standard deviations of labeled F-scores on test sets for varying amounts of training data and use of MTL. Computed over 3 data samples with replacement and different random initializations.

5.2 Suitability for Parsing

An important question to ask is if the normalization of the AM dependency trees has an impact on how accurate parsing is. In particular, we expect the high similarity to be beneficial for projecting from a large corpus to a smaller one. Therefore, we conduct parsing experiments in the usual high-resource scenario and also in a simulated low-resource scenario, for which we sampled 3 subsets of the training data with 100 instances each and used those subsets throughout all low-resource experiments.

We use the parser of L19 but, in contrast, we use only those training instances for which we have AM dependency trees in *all* formalisms, resulting in 29,182 training instances. Moreover, we do not use embeddings for lemmas because of data sparsity in the low-resource condition. We follow L19 in how we perform MTL, that is, we share an LSTM over all tasks in addition to using task-specific ones. Scores for edges and lexical as-graphs are produced from single-layer feedforward networks that take the concatenation of the output of the shared and task-specific LSTM as input. MTL experiments use the full training data of the non-target formalism and the specified amount for the target task; no data beyond the SDP corpora was used for the MTL experiments. For full details of hyperparameters, see Appendix D.

Table 3 summarizes the results. MTL in the low-resource scenario is hugely beneficial and works better with the normalized AM dependency trees. The slight to moderate decrease in accuracy of the normalized trees in the full data condition likely comes from the noise we introduce, as not all phenomena are detected perfectly and our fixes are sometimes restricted to specific instances. As noted in Section 3.1, the AM+ algebra can produce multiple edges between pairs of nodes and self-loops. This is rare in practice except for PSD in the low-resource condition without MTL, where up to 1.7% of predicted edges are affected.

6 Discussion

The methodology we detail in Sections 3 and 4 allows us to normalize roughly 60% of differences across compositional structures of the SDP graphbanks. Our work establishes that a small set of linguistically-grounded transformations are quite powerful towards creating uniform compositional structure; in fact, we find that *all* of the divergent patterns across graphbanks that we detect are 'superficial' in the sense that, given proper attention and methodology, we can normalize them.

The individual phenomena that create inconsistencies across graph representations and their corresponding L19 AM trees predictably adhere to a Zipfian distribution (Appendix B). To manually develop automatic normalizations for 100% of the unaligned patterns may be infeasible. These observations seem to indicate that the remaining differences between our normalized AM trees are due to practical, implementation concerns rather than conceptual ones. A possible approach to solving the practical issues would be a fully automated method with statistical rather than human guidance. In a low resource scenario, manual annotations could also be a feasible solution. It is to note though that the AM+ algebra is not all-powerful; the L19 preprocessing step for coordination in PSD remains necessary.

Despite the fact that all phenomena we examine can be normalized, *how* they are normalized provides insights into the differences among the underlying MRs. While structural differences like those seen in negation and coordination essentially reduce to mildly different graph constants, ignored words or underspecified attachment that may create ambiguity are a different story. Namely, they make explicit the different kinds of information that some MRs include and others do not, forcing us to choose the most informative compositional structure (such as including determiners for PSD). Curiously, our parser does not seem to care much about this added, more-explicit information, but we believe future work in cross-framework parsing with more distinct graphbanks may provide more meaningful insight on this fact.

Extending our work to other graphbanks will shed light on the effect of graph flavours; all three SDP formalisms investigated in this paper share the same graph 'flavour' possibly leading to very uniform graph constants and facilitating normalization. Adding other formalisms with different graph flavours is therefore an interesting future research direction, but limited by the availability of parallel texts.

7 Conclusion

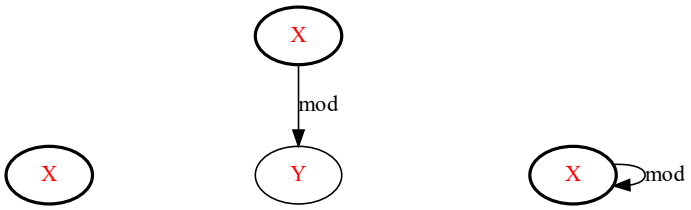
We have shown how annotations from different graphbanks (specifically, DM, PAS, and PSD) can be normalized at the level of their compositional structure using principled linguistic reasoning. We achieve this by updating the AM algebra to the AM+ algebra, which allows more flexibility in adapting compositional structures across MRs. By working at the compositional level of the graphs using AM dependency trees, we are able to quantify mismatches between different graphbanks, systematically reshape these mismatches to make them more uniform across graphbanks, and thereby increase the match between the compositional structures for the three graphbanks. Such work serves as a proof of concept that normalization is possible and contributes to a broader effort to increase parallel MR parsing accuracy.

Acknowledgements

Thanks go to Meaghan Fowlie for insightful discussions. This research was in part funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project KO 2916/2-2.

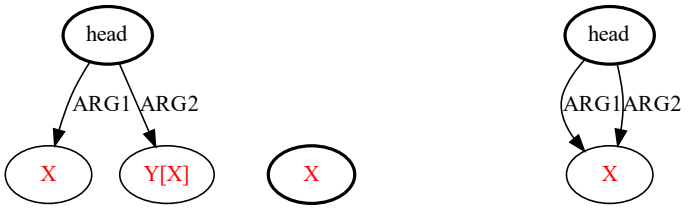
Appendix

A Details on the AM+ algebra



(a) G_1 (b) G_2 (c) Result of MOD_Y (G_1, G_2)

Figure 6: (a) Head G_1 and (b) modifier G_2; the result of the MOD_Y operation in (c) has a loop.



(a) G_3 (b) G_4 (c) Result of APP_Y (G_3, G_4)

Figure 7: (a) Head G_3 and (b) argument G_4; the result of the APP_Y operation in (c) has only two nodes but multiple edges between them.

Fig. 6 shows an example of how the AM+ algebra can lead to loops. The MOD_Y operation attaches the Y source of the modifier in (b) to the root of the head in (a), and merges the X sources. Since the root and X source of the head are on the same node, a loop results.

Fig. 7 shows an example of how the AM+ algebra can lead to multiple edges between two nodes. When the APP_Y operation fills the Y source of the head in (a) with the argument in (b), both argument nodes of the head would have an X source; they merge. This results in a graph with two edges between the same two nodes.

Neither loops nor such duplicate edges are part of the design of SDP graphs; they occur in no graphs in the corpus. During evaluation, we treat loops and multiple edges like any other incorrect edge. For example, if the gold graph has one edge from some node v to some node w with label ℓ, and we predict three edges from node v to w among which one is labeled correctly, we count one correct edge and two incorrect edges, obtaining full recall but reduced precision.

In terms of theoretic consequences of the transition from the AM to the AM+ algebra, the proof of Groschwitz (2019) that each well-typed AM dependency tree evaluates to a unique graph still applies. In fact, none of the arguments in the proof are affected except that the proof uses the commutativity of the merge operation of the HR algebra. However, in the extension of van Harmelen and Groschwitz (2020) the merge operation is still commutative, so the proof goes through.

B Additional Pattern Signatures and Frequencies

In Table 1 we presented the pattern signatures we fixed, whereas Table 4 includes all of the 30 most frequent pattern signatures containing different patterns.

Rank	Pattern signature ([DM / PAS / PSD])	Most commonly associated phenomena	Count	Percentage	Cumulative percentage	We fix
1	[MOD / MOD / EMPTY]	Determiners	49 965	13.38%	13.38%	✓
2	[EMPTY / MOD / EMPTY]	Punctuation	45 544	12.20%	25.58%	✓
3	[CONN / CONN / EMPTY]	Prepositions	42 395	11.35%	36.93%	✓
4	[BASIC / BASIC / MOD]	NPs with prep.*	30 202	8.09%	45.02%	✓
5	[EMPTY / CONN / EMPTY]	Prepositions	22 324	5.98%	51.00%	✓
6	[MOD / BASIC / BASIC]	Named Entities*	15 835	4.24%	55.24%	–
7	[MOD / BASIC / MOD]	Named Entities*	14 234	3.81%	59.05%	–
8	[EMPTY / CMOD / EMPTY]	Temporal auxiliaries	14 074	3.77%	62.82%	✓
9	[MOD / MOD / BASIC]	Named Entities*	7844	2.10%	64.93%	–
10	[APP1 / MOD / MOD]	Verbal Negation*	7215	1.93%	66.86%	✓
11	[BASIC / BASIC / APP1]	Named Entities*	6670	1.79%	68.64%	–
12	[APP1 / BASIC / BASIC]	Named Entities*	6595	1.77%	70.41%	–
13	[BASIC / MOD / MOD]	Named Entities*	5714	1.53%	71.94%	–
14	[EMPTY / APP2 / APP2]	Coordination	5709	1.53%	73.47%	✓
15	[BASIC / MOD / BASIC]	$ signs*	5008	1.34%	74.81%	–
16	[EMPTY / MOD / BASIC]	Relative pronouns	3703	0.99%	75.80%	–
17	[APP2 / CONN / EMPTY]	Subordinating conjs.	3609	0.97%	76.77%	–
18	[APP1 / CMOD / EMPTY]	Modal auxiliaries	3431	0.92%	77.69%	–
19	[EMPTY / CONN / APP2]	Punctuation	2952	0.79%	78.48%	–
20	[EMPTY / CAPP2 / APP2]	Copula (adjectival)	2915	0.78%	79.26%	✓
21	[EMPTY / APP2 / OTHER]	Coordination	2848	0.76%	80.02%	✓
22	[EMPTY / OTHER / OTHER]	Coordination	2828	0.76%	80.78%	✓
23	[APP1 / BASIC / APP1]	NCP	2712	0.73%	81.51%	–
24	[APP2 / APP2 / OTHER]	Particle verbs	2612	0.70%	82.21%	–
25	[CONN / MOD / EMPTY]	Determiners*	2234	0.60%	82.80%	–
26	[EMPTY / MOD / MOD]	Adverbs*	2119	0.57%	83.37%	–
27	[BASIC / BASIC / CONN]	Named Entities*	1992	0.53%	83.90%	–
28	[CONN / CONN / OTHER]	NCP	1851	0.50%	84.40%	–
29	[MOD / CONN / EMPTY]	Subordinating conjs.	1845	0.49%	84.89%	–
30	[EMPTY / APP2 / EMPTY]	Punctuation	1842	0.49%	85.39%	–

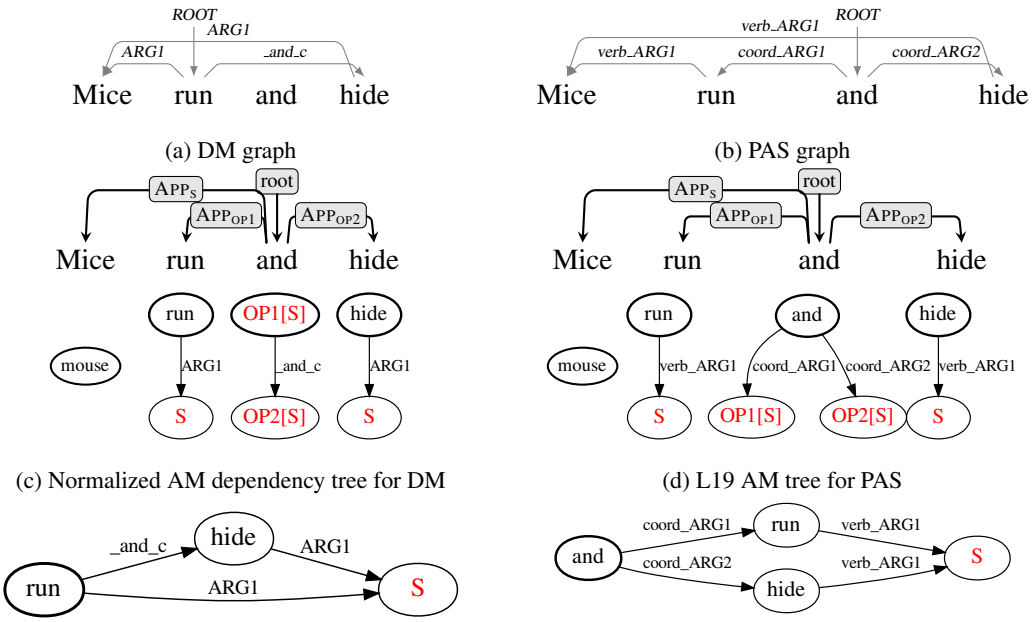
Table 4: Most frequent pattern signature differences across the three graphbanks. Asterisks indicate that additional phenomena comprise a significant portion of the pattern. NCP denotes 'no common pattern' or patterns that lack a single most common corresponding phenomenon.

C Normalizing Key Compositional Structures: Additional Examples

C.1 Category 1: Ignored Elements

Determiners. As seen in Section 3.2, determiners show up as modifiers in the AM dependency trees for DM and PAS, but are absent in PSD, yielding the [MOD / MOD / EMPTY] signature. To align the pattern signature, we modify the AM dependency tree for PSD to match the edges of the other AM trees (Fig. 3i). To do this, we use a semantically vacuous constant for the determiner "the" that consists of only an unlabeled node; this node is marked with a DET-source and functions as the root at the same time. Performing the MOD_{DET} operation with this constant does not change the graph of the noun (here "cat"), and thus the AM dependency tree in Fig. 3i still evaluates to the graph in Fig. 3c.

This construction makes sense linguistically and with our algebra: the MOD_{DET} operation reflects a relation on the sentence surface, and the empty determiner constant for PSD reflects the fact that the information carried by the determiner is ignored in the deeper PSD graph structure. To ensure we fix determiners in exactly this way, the token belonging to the local pattern must have the POS tag *DT*. This covers 95% of the pattern. Note that since this empty lexical as-graph does not change the graph, we could attach it anywhere and still obtain the same graph as evaluation result. Thus, we must look at determiner attachement in DM to guide our attachment in PSD; the PSD graph alone is not enough to tell us where



(a) DM graph (b) PAS graph

(c) Normalized AM dependency tree for DM (d) L19 AM tree for PAS

(e) Partial result of (c) after APP_{OP1} and APP_{OP2} operations (f) Partial result of (d) after APP_{OP1} and APP_{OP2} operations

Figure 8: Coordination of two verbs with a common subject.

the determiner should attach.

Punctuation, Particles & Co.. These elements together exhibit the pattern [EMPTY / MOD / EMPTY]. PAS is the only SDP MR to broadly annotate tokens that are mostly syntactic markers. This pattern includes punctuation, infinitival "to", passive "by", and particles of particle verbs. As with determiners, we add empty modifiers, here to DM and PSD. We restrict our fix to POS tags that indicate punctuation or those of *IN*, *TO* or *RP*, which covers 97% of pattern.

Temporal and Aspectual Auxiliary Verbs. These adhere to the [EMPTY / CMOD / EMPTY] pattern. An example of this is "are" in Fig. 4. These auxiliaries are annotated only in PAS, where the L19 AM trees (Fig. 4c) use complex modification to create the reentrancy observed in Fig. 1b. While the CMOD pattern is different from the MOD pattern as far as the resulting graph is concerned, both patterns simply have an incoming MOD edge when only looking at the AM tree edges. Thus, we can apply the same fix here as for the punctuation case. We restrict our fix to POS tags that start with *V*, which covers 82% of pattern.

C.2 Category 3: Relational Elements

Coordination with a common argument (further details): If both conjuncts have a common argument, such as in Fig. 8b, the source for the common argument is first passed up through the coordination before being filled with the argument, resulting in an additional APP child for the coordination (see Fig. 8(d, f)). In such cases, PAS and PSD L19 AM trees exhibit the OTHER pattern. If the conjuncts share a common argument like the subject "mice" in Fig. 8a, we add respective source annotations to the OP1- and OP2-sources (Fig. 8c). Fig. 8e shows the partial result after evaluating the APP_{OP1} and APP_{OP2} operations in this example.

Prepositions and Their Effects. Though typically understood as a single morphosyntactic class in English, prepositions are notoriously polysemous (Schneider et al., 2018). This is reflected in their varied pattern signature: [CONN / CONN / EMPTY], [EMPTY / CONN / EMPTY], [BASIC / BASIC / MOD]. We find that most prepositions have the CONN pattern or are ignored in the graph (compare for example the preposition "in" in Fig. 1 in DM and PAS versus PSD).

DM and PAS use strategy (ii) (see Section 4.3) for most prepositions and encode them as nodes, whereas PSD uses strategy (i) and encodes them as edges. See the preposition "in" in Fig. 1, to which the PSD 'LOC' edge corresponds. In the AM trees, these prepositions have the [CONN / CONN / EMPTY] signature. Note that in the PSD graph in this example, the 'LOC' edge corresponds to this preposition instead, which

we find to be a general rule: if a preposition is not a node in the graph, the relation it expresses is instead encoded as an edge. In the original decomposition of L19, the 'LOC' edge here is attached to the lexical as-graph of "house" yielding the MOD pattern, a common cause for the [BASIC / BASIC / MOD] signature. Thus, normalizing the preposition case addresses two of the most common pattern differences at once. Some prepositions such as possessives ("of") are encoded as edges in DM, too, yielding the [EMPTY / CONN / EMPTY] signature.

As a fix in the graphs that encode the preposition as an edge, we use that edge as the lexical as-graph for the preposition, such as the lexical as-graph for "in" in Fig. 4e. This new lexical as-graph consists of only the 'LOC' edge, with the root node at its origin, and sources S and O. At the same time, we remove that edge from the lexical as-graph that contained it before (here "house"). We then update the AM dependency edges accordingly, for example in the transformation from Fig. 4b to Fig. 4e, we remove the MOD_M edge from "playing" to "house" and instead connect the new constant for "in" with an outgoing APP_S edge to "house" and an incoming MOD_O edge from "playing", creating the CONN pattern now in PSD as well.

To find the edge in PSD in the general case, we use a similar strategy as we employed for copula (Section 4.1): we look at the parent of the preposition in PAS ("playing" in the example) as well as the APP-child of the preposition ("house"), and consider the PSD AM tree edge between them (here MOD_M). We then identify the graph edge (here 'LOC') via the source of the AM tree operation: the 'LOC' edge here is the edge from "house" to the M-source. Such a matching PSD AM tree edge exists in 86% of the cases; in the other cases PAS and PSD disagree on the heads of the preposition arguments and we cannot apply an automatic transformation.

Note that this transformation also changes the MOD pattern at "house" to the more consistent BASIC pattern. In total, we have changed a [CONN / CONN / EMPTY] and a [BASIC / BASIC / MOD] signature to a [CONN / CONN / CONN] and a [BASIC / BASIC / BASIC] signature. In the [EMPTY / CONN / EMPTY] case, we employ the same strategy to change the AM trees for both DM and PSD. We restrict our fix to *IN* or *TO* POS tags at the preposition, covering 99% of [CONN / CONN / EMPTY] and 66% of [EMPTY / CONN / EMPTY].

D Hyperparameters

We use the implementation of L19: https://github.com/coli-saar/am-parser. The hyperparameters common to all experiment are collected in Table 5. We make some minor adjustment of the hyperparameters to take into account with 100 training examples only. In particular, we do not use embeddings for lemmas and train longer (100 epochs). We pick the model with the highest performance on the development set after epoch 25 and perform early stopping with patience of 10 epochs.

We use the "large-uncased" BERT model as available through AllenNLP. We follow L19 and learn a weighted average of the layers without finetuning BERT's weights.

Single Task We use a batch size of 16 sentences.

Multi Task In our MTL experiments, we follow the setup of L19: we have one LSTM per graphbank and one that is shared between the graphbanks. When we compute scores for a sentence, we run it through its graphbank-specific LSTM and the shared one. We concatenate the outputs and feed it to graphbank-specific MLPs. We use a separate LSTM for the edge model (input to edge existence and edge label MLP) and the supertagging model. In effect, we have two LSTMs that are shared over the graphbanks: one for the edge model and one for the supertagging model. All LSTMs have the hyperparameters detailed in Table 5. We use a batch size of 16 for the target formalism and a batch size of 64 for all non-target formalisms. Each batch consists of instances from only one formalism.

Activation function in all MLPs	tanh
Optimizer	Adam
Learning rate	0.001
Epochs	**100**
Dim of lemma embeddings	**0**
Dim of POS embeddings	32
Dim of NE embeddings	16
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	0.3
Recurrent dropout in LSTMs	0.4
Input dropout	0.3
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 5: Common hyperparameters used in all experiments. Deviations from L'19 are highlighted with boldface.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria, August. Association for Computational Linguistics.

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, August. Association for Computational Linguistics.

Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge University Press.

Lucia Donatelli, Meaghan Fowlie, Jonas Groschwitz, Alexander Koller, Matthias Lindemann, Mario Mina, and Pia Weißenhorn. 2019. Saarland at MRP 2019: Compositional parsing across all graphbanks. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 66–75.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. *Proceedings of ACL*.

Jonas Groschwitz. 2019. *Methods for taking semantic graphs apart and putting them back together again*. Ph.D. thesis, Macquarie University and Saarland University.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? a contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. *Proceedings of ACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May. European Language Resources Association (ELRA).

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado, June. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3991–3995.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. MRP 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1–27, Hong Kong, November. Association for Computational Linguistics.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proceedings of ACL*, pages 185–196, Melbourne, Australia, July.

Martin van Harmelen and Jonas Groschwitz. 2020. Graphs with Multiple Sources per Vertex. *arXiv e-prints*. https://arxiv.org/abs/2006.11159.