LINGUISTIC INTERPRETABILITY AND COMPOSITION OF ABSTRACT MEANING REPRESENTATIONS

A Dissertation
submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy
in Linguistics

By

Austin Blodgett, M.S.

Washington, DC
June 22, 2021
LINGUISTIC INTERPRETABILITY AND COMPOSITION OF ABSTRACT MEANING REPRESENTATIONS

Austin Blodgett, M.S.

Dissertation Advisor: Nathan Schneider, Ph.D.

ABSTRACT

Many Natural Language Processing (NLP) and Natural Language Understanding (NLU) tasks require some, usually implicit, representation of meaning. Abstract Meaning Representation (AMR; Banarescu et al., 2013) aims to be a scalable way of including explicit representations of meaning, in the form of semantic graphs. This work takes on a goal of augmenting AMR semantic graphs to be made linguistically interpretable—increasing the interpretability they add to a model.

I pursue this goal through two avenues of research. First, I improve the analyzability of AMR via a novel, structurally comprehensive and linguistically enriched set of AMR-to-text alignments. I present this new formulation of AMR alignment which addresses a wide variety of linguistic phenomena, as well as a corpus of automatically generated alignments for English sentences, and a probabilistic, structure-aware alignment algorithm which produces alignments without supervision and with higher coverage, accuracy, and variety than alignments from existing AMR aligners.

Second, I improve the compositionality of AMR via an extension of Combinatory Categorial Grammar (CCG) which allows AMR semantics and compositional derivation of AMR graphs. This formulation of AMR as graph semantics in CCG and accompanying combinatorial rules of CCG allow derivation of a full AMR graph in an interpretable way. Lastly, I conduct an empirical analysis of the compatibility and structural similarity/dissimilarity of AMR with automatically generated CCG parse data, and identify linguistic sources of complexity for the benefit of future research.

INDEX WORDS: AMR, semantic graph, CCG, interpretability, explainability
ACKNOWLEDGMENTS

The author would like to thank Dr. Nathan Schneider for being an exceptional adviser and offering continuous guidance and support, the committee members Dr. Amir Zeldes, Dr. Paul Portner, Dr. Adam Lopez, and Dr. Ramón Astudillo for their advice, time, and hard work, and members of the NERT research lab and other friends and colleagues at Georgetown for their enthusiasm and positivity and for being a supportive community throughout my graduate career.
# Table of Contents

## Chapter 1 Introduction
- 1.1 The Purpose of Symbolic Representations ................................. 2
- 1.2 The Relationship between Syntax and Semantics ...................... 5
- 1.3 Publications ............................................................................. 8

## Chapter 2 Background
- 2.1 Form, Meaning, and Composition ........................................... 9
- 2.2 Abstract Meaning Representation ........................................... 12
- 2.3 Combinatory Categorial Grammar ........................................... 20

## Chapter 3 Literature Review
- 3.1 Symbolic Interpretability of Linguistic Structures and Neural Networks 25
- 3.2 Abstract Meaning Representation ........................................... 31
- 3.3 Combinatory Categorial Grammar ........................................... 43
- 3.4 Composition and Graph Grammars ........................................... 46

## Chapter 4 LEAMR: A Novel Dataset of AMR-Text Alignments
- 4.1 Introduction ............................................................................. 50
- 4.2 Related Work ........................................................................... 51
- 4.3 An All-Inclusive Formulation of AMR Alignment ....................... 55
- 4.4 Released Data .......................................................................... 61
- 4.5 Conclusions ............................................................................ 62

## Chapter 5 Probabilistic, Structure-Aware Algorithms for Improved Variety, Accuracy, and Coverage of AMR Alignments
- 5.1 Introduction ............................................................................. 63
- 5.2 Relevant Work .......................................................................... 63
- 5.3 LEAMR Aligner ......................................................................... 64
- 5.4 Experimental Setup ................................................................... 71
- 5.5 Results ..................................................................................... 72
- 5.6 Conclusions ............................................................................ 74

## Chapter 6 An Improved Approach to Semantic Graph Composition with CCG
- 6.1 Introduction ............................................................................. 75
- 6.2 Relevant Literature .................................................................... 77
- 6.3 Graph Semantics ....................................................................... 78
- 6.4 Combinators ............................................................................ 82
- 6.5 Linguistic Examples ................................................................... 85
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6 Discussion</td>
<td>95</td>
</tr>
<tr>
<td>6.7 Conclusions</td>
<td>97</td>
</tr>
<tr>
<td>7 An Analysis of Concordance and Discordance between CCG and AMR</td>
<td>98</td>
</tr>
<tr>
<td>7.1 Introduction</td>
<td>98</td>
</tr>
<tr>
<td>7.2 Relevant Work</td>
<td>99</td>
</tr>
<tr>
<td>7.3 Tools For Analysis</td>
<td>100</td>
</tr>
<tr>
<td>7.4 Data</td>
<td>108</td>
</tr>
<tr>
<td>7.5 Methodology</td>
<td>109</td>
</tr>
<tr>
<td>7.6 Discussion</td>
<td>115</td>
</tr>
<tr>
<td>7.7 Results</td>
<td>118</td>
</tr>
<tr>
<td>7.8 Future Work</td>
<td>123</td>
</tr>
<tr>
<td>7.9 Conclusions</td>
<td>124</td>
</tr>
<tr>
<td>8 Conclusions</td>
<td>125</td>
</tr>
<tr>
<td>8.1 Lessons for Future AMR Research</td>
<td>125</td>
</tr>
<tr>
<td>8.2 Future Work</td>
<td>126</td>
</tr>
<tr>
<td>APPENDIX</td>
<td></td>
</tr>
<tr>
<td>A Alignment Annotation Guidelines</td>
<td>128</td>
</tr>
<tr>
<td>A.1 General Principles</td>
<td>129</td>
</tr>
<tr>
<td>A.2 Spans</td>
<td>131</td>
</tr>
<tr>
<td>A.3 Nodes</td>
<td>132</td>
</tr>
<tr>
<td>A.4 Edges</td>
<td>133</td>
</tr>
<tr>
<td>A.5 Duplicate Subgraphs</td>
<td>133</td>
</tr>
<tr>
<td>A.6 Reentrancies</td>
<td>134</td>
</tr>
<tr>
<td>B Details of Rules and Pre/Post-Processing for AMR Alignment</td>
<td>141</td>
</tr>
<tr>
<td>B.1 Identifying Spans</td>
<td>141</td>
</tr>
<tr>
<td>B.2 Rule-based Subgraph Alignment Preprocessing</td>
<td>142</td>
</tr>
<tr>
<td>B.3 Rule-based Relation Alignment Preprocessing</td>
<td>143</td>
</tr>
<tr>
<td>B.4 Rule-based Reentrancy Alignment Preprocessing</td>
<td>144</td>
</tr>
<tr>
<td>C Addition CCG-AMR Derivations</td>
<td>145</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>149</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

2.1 AMR for the sentence “Most of the students want to visit New York when they
graduate.” ................................................................. 13

2.2 The equivalent logical form for the sentence “Most of the students want to visit
New York when they graduate.” ........................................ 14

2.3 Example of morphological decomposition represented in AMR. ....................... 15

2.4 AMR representation of the named entity “New York”. ..................................... 16

2.5 Reentrancies example for the sentence “Most of the students want to visit New York
when they graduate”. .......................................................... 17

2.6 CCG derivation for the sentence “Most of the students want to visit New York when
they graduate.” ................................................................. 23

4.1 AMR and alignments for the sentence “Most of the students want to visit New York
when they graduate.” ............................................................. 56

6.1 Basic shape of AMR subgraph. ...................................................... 79

6.2 Linguistic examples as AMR subgraphs. ..................................................... 80

6.3 Application and identity shown for the sentence “John likes the cat.” ................. 86

6.4 Composition shown for the sentence “Sarah suddenly entered the room.” ........... 86

6.5 Control and relation-wise application shown for the sentence “Sarah decided to take
the bus.” ........................................................................ 86

6.6 Complex coordination and type raising shown for the sentence “John likes and
Mary hates cats.” ............................................................... 87

6.7 Passive example: “John was eaten by bears.” .................................................. 87

6.8 Wh-question example: “What did you do yesterday?” ..................................... 88

6.9 Object control example: “Mary persuaded John to practice guitar”. .................. 88

vii
6.10 Inverse roles of derived nominals and relative clauses. .......................... 89
6.11 Light verb construction example: “John made a decision on his major.” ........ 91
6.12 CCG derivation for the phrase “most of the students” demonstrating the use of the
dupl() operator. ................................................. 93
6.13 Example of right node raising with shared main verb: “I should and you may eat.” 95
7.1 CCG dependency graph and corresponding parse for the sentence “Most of the
students want to visit New York when they graduate.” ................................. 101
7.2 AMR composition graph for the sentence “Most of the students want to visit New
York when they graduate.” ............................................. 104
7.3 Examples of concordance and discordance. .............................................. 106
A.1 AMR for the sentence “Pierre Vinken, 61 years old, will join the board as a nonex-
etive director Nov. 29.” ......................................................... 128
A.2 AMR for “It would work like medicare . . . ” ............................................. 133
A.3 AMR for “I actually had some other classmates there, and was going to call them” 135
A.4 AMR for “John was afraid to speak up.” .................................................. 137
A.5 AMR for “The key is to be as objective as possible .” ................................. 139
A.6 AMR for “Precursory signs of the earthquake” ........................................ 139
A.7 AMR for “John met up with a friend.” ....................................................... 140
C.1 Raising example: “Mary seems to practice guitar often.” ......................... 145
C.2 Subject control example: “Mary wants to practice guitar.” ....................... 146
C.3 Object control wh-question example: “Who did you persuade to smile?” ...... 146
C.4 Modal auxiliary with preposed adjunct: “Tomorrow, John may eat rice.” ...... 147
C.5 To-purpose example: “Mary bought a ticket to see the movie.” .................. 147
C.6 Coordinated purpose clauses: “John arrived to eat and to party.” ............... 148
# List of Tables

2.1 Released gold AMR data for 5 languages. ........................................... 19
2.2 Common CCG categories/supertags for various types of words and phrases. .... 21
2.3 Syntactic effects of CCG combinators. .................................................. 22
3.1 A comparison of attributes for 6 semantic representations with support for English. 32
4.1 Coverage and types of previous alignment systems. ............................... 53
4.2 Reentrancy types with examples. ........................................................... 60
4.3 Interannotator agreement. ................................................................. 62
5.1 Main results on the test set. ................................................................. 72
5.2 Detailed results for relation alignments and reentrancy alignments. ............ 73
5.3 Subgraph and relation ablation results. ............................................... 74
6.1 Formal semantic rules for AMR combinators. ....................................... 82
6.2 English eventive nouns in light verb and possessive forms. .................... 90
6.3 Refined conjunction combinators. ...................................................... 93
7.1 Number of sentences in each dataset. ................................................ 109
7.2 Definitions and examples for various sources of discordance. ................... 113
7.3 Definitions and examples for various sources of discordance (continued part 1). 114
7.4 Definitions and examples for various sources of discordance (continued part 2). 115
7.5 Concordance recall by dependency. ..................................................... 118
7.6 Discordance details. ............................................................................. 119
7.7 Concordant structure by token distance. .............................................. 120
7.8 Lexical concordance. ........................................................................... 121
7.9 Supertag templates and their prevalence in AMR vs. CCG. ..................... 121
7.10 Concordance for reentrancies. ......................................................... 122
7.11 Compositionality of multi-token spans. ........................................ 123

A.1 Alignments for the sentence “Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.” .................................................... 129

A.2 Alignment annotation for the sentence “Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.” .................................................... 130

A.3 Annotation of duplicate subgraphs in “It would work like medicare…” ................................................................. 134

A.4 Annotation of reentrancy alignments for the sentence “I actually had some other classmates there, and was going to call them.” .................................................... 135

A.5 Explanation of reentrancy types. ............................................................... 136

A.6 Reentrancy alignment for the sentence “John was afraid to speak up.” ............... 138
CHAPTER 1

INTRODUCTION

Many Natural Language Processing (NLP) and Natural Language Understanding (NLU) tasks require some, usually implicit, representation of meaning. Abstract Meaning Representation (Banarescu et al., 2013) aims to be a scalable way of including explicit representations of meaning, in the form of semantic graphs. AMR, and other symbolic meaning representations, can be used to aid a model’s performance in tasks requiring semantic knowledge or simply to make the model’s use of semantic knowledge more interpretable.

This dissertation is written with the aim of answering the following question: Can AMR semantic graph representations be augmented to be made linguistically interpretable and well-founded with respect to linguistic theory—increasing the interpretability they add to a model—without reducing their scalability or usefulness to NLP models? I break this down into two sub-problems:

a) Can AMR be made analyzable, such that every substructure of an AMR is analyzed as being introduced by a part of the sentence it corresponds to?

b) Can AMR be made compositional, such that the derivation of an AMR can be understood as a procedure of composing form-meaning pairs in a linguistically sensible way?

To make AMR analyzable, I adopt a strategy of inferring form-meaning pairs between AMR substructures and text, by means of improved AMR-to-text alignment. To make AMR compositional, I adopt a strategy of incorporating meaningful AMR substructures into an existing grammar with available parsers and datasets. This will allow me to interpret ordered composition of AMR subgraphs as being akin to ordered composition in natural language.

My contributions are as follows:

1
1. A novel formulation of AMR alignment, formulated in terms of mappings between spans and connected subgraphs, which is comprehensive with respect to AMR substructures and whose alignments correspond to wide variety of linguistic phenomena including argument structures, non-core roles, coreference, control, and ellipsis,

2. A corpus of automatically generated alignments for AMR Release 3.0 and *Little Prince* data as well as several hundred manually annotated sentences for tuning and evaluation,

3. A probabilistic, structure-aware alignment algorithm to automatically align English sentences to AMRs without supervision, with higher coverage, accuracy, and variety than alignments from existing AMR aligners,

4. A formulation of AMR as graph semantics in CCG and accompanying combinatorial rules of CCG for deriving a full AMR graph in an interpretable way.

5. An empirical analysis of the compatibility and structural similarity/dissimilarity of AMR with automatically generated CCG parse data.

The remainder of this chapter will motivate my central research question in more detail. Ch. 2 will present background necessary for understanding the remaining chapters, and Ch. 3 will present a review of relevant literature. My research and experiments are presented in Ch. 4 which outlines a novel AMR alignment formulation and dataset, Ch. 5 which presents the associated automatic aligner, Ch. 6 which presents a formalism for incorporating AMR semantics into CCG derivation, and Ch. 7 which presents an empirical analysis of the compatibility and structural similarity/dissimilarity of AMR and CCG. Finally, Ch. 8 will present my conclusions and future work.

### 1.1 The Purpose of Symbolic Representations

Aside from performance, “explainability” is arguably the most important attribute for a neural model. Explainability of a model enables researchers to conduct better error analysis, and is thus important for pace of scientific advancement in NLP and similar fields. There are additionally real-world settings where explainability is crucial—scenarios where it is important to be able to diagnose errors or where transparency of a decision process is necessary. Consider for example,
information extraction and summarization in a biomedical setting, or a legal setting, or any setting where ethics and fairness are of major concern. In each case a user of the NLP system—whether a doctor, lawyer, or otherwise—must not only receive the relevant information being recommended by the system, but also be able to identify and diagnose errors and justify decisions as to why the information was recommended, both of which require that the model’s procedure is explainable. The technical term for this attribute of an NLP or other algorithmic system is interpretability.

Interpretability is today an established area of research within machine learning, driven in large part by the success and simultaneous complexity and ineffability of neural models. In this dissertation, I am interested in a particular form of interpretability which hereafter will be called symbolic interpretability of linguistic structures, more specifically structures representing syntax, semantics, and ordered composition of English sentences. I define symbolic interpretability as follows:

Symbolic Interpretability—Any method of incorporating symbols which are discrete (as opposed to soft alignments or vector representations), explainable, and meaningful at some stage of a model, whether as an input, an intermediate representation, or an additional target output, either in order to add an inductive bias or to improve the explainability of that model.

I will use the term linguistic interpretability for cases where the symbols carry linguistic meaning. My motivation for interest in this area is two-fold. First, when explainable linguistic structures are included with a neural model created for some downstream task, they may add interpretability to the model in a way that is analogous to human processing of language and is thus easier to grasp without special training. Second, there are many cases where incorporating explicit or implicit linguistic structures into a model improves performance. This happens because linguistic structures can provide “scaffolding” for the model’s learning (Smith, 2017), introducing inductive biases that

---

constrain or guide the representations the model learns to those which are linguistically plausible and thus generalize well across a language or to other languages.² Linguistic structure has been shown to allow improvements in machine translation (dependency syntax: Wu et al. (2017), semantic roles: (Marcheggiani et al., 2018)), question answering (dependency syntax: Meng et al. (2017), semantic frames: Alizadeh and Eugenio (2020)), natural language inference (dependency syntax: (He et al., 2020), semantic roles: Cengiz and Yuret (2020)), and information extraction (dependency syntax: Li and Ji (2019), semantic roles: Christensen et al. (2010)) among other areas, and especially helps when resources are not very large. Henderson (2020) argues that much of the success in NLP with neural models is due to linguistic structures being incorporated into models—most often implicitly, by building linguistic biases into a model’s architecture—driving NLP systems to be more linguistically informed and more successful.

Though there is also research on interpretable distributional representations (Baroni and Zamparelli, 2010; Blacoe and Lapata, 2012; Boleda, 2020) and time-intensive investigations of complex models (Rogers et al., 2020), symbolic interetability offers the prospect of grounding complex NLP tasks in linguistic theory and creating a framework for explaining the procedures and errors associated with our models in a way that does not require specialized computer science training.

1.1.1 What Should Linguistic Interpretability Look Like

Application of semantic representations like AMR in downstream applications is still a young area of research in NLP, and it is not yet clear what the best way of incorporating such representations should be. Possible avenues include AMR as multitask learning, as pre-training, as additional features, or as intermediary representations. See Ch. 3 for a review of literature demonstrating these approaches with various linguistic structures. I take the position that an ideal linguistically interpretable model should have model decisions and model subtasks resembling linguistic decisions and linguistic subtasks. This would decouple linguistic complexity from other sources of complexity.

²Inductive biases are called “hard” if they are strict constraints on the model’s behavior or “soft” if the model is allowed some flexibility in how it apply them.
in an NLP model. To accomplish this, we need some notion of ordered composition to break down NLP tasks into linguistically interpretable subtasks.\(^3\)

1.1.2 Why AMR

Much of the work presented in this dissertation can be summarized as taking AMR, a highly scalable data structure for sentence meaning, and adapting it to be more similar to well-founded linguistic theory, namely by allowing analyzability and ordered composition. This might seem to be a strange order of doing things. A linguist would ordinarily start with a linguistic theory and build a data structure capable of encoding the claims of that theory. An astute reader may wonder “why start with the data and move toward the theory instead of the other way around?” The reason is precisely because AMR is scalable. Its ease of annotation allows for wider availability of data and easier adaptation to new domains than (most) other semantic representations while still encoding the core features of meaning. AMR is compared to other semantic representation in §3.2.1, but note here that AMR has an advantage over many of these other representations in that it encompasses important semantic prediction tasks but excludes annotation which might be burdensome for annotators or require extensive linguistic knowledge.\(^4\) So, the central research question of this dissertation could be paraphrased as “can a scalable semantic representation also adhere to linguistic theory?” AMR is introduced in detail in §2.2.

1.2 The Relationship between Syntax and Semantics

The research presented in Ch. 6 and 7 is motivated by a goal of augmenting AMR with syntax in order to implement ordered composition. This raises an important question “how useful is syntactic

\(^3\)One potentially fruitful area of research is first factoring AMR parsing into linguistically meaningful subtasks, such as lexical prediction and ordered composition (see Ch. 6), and using each subtask as an auxiliary task for a multitask learning setup. That approach would create a correspondence between decisions in an NLP subtask and linguistic decisions in an elegant way. Additionally, an advantage of multitask learning is that it trains a model to predict a secondary type of output at train time or run time, which in the case of AMR would give insight into the semantic state of a model at each step of an algorithm.

\(^4\)For example, AMR does not include annotation of scope, aspect, or hand alignment to text. This makes it easier to produce large datasets of AMR annotation.
information for inferring semantics in a computational setting?” I argue that incorporating syntax into AMR or an associated model may be beneficial both in that it will improve interpretability and that it will introduce useful inductive biases. The second part of that claim is worth discussing here. This section illustrates that there is support in linguistic theory for intuition that syntactic information should be useful for inferring semantic information.

1.2.1 Pairing of Syntax with Semantics

It is widely accepted among linguists of various traditions that there is a strong relationship between syntax and semantics. Linguists in the Generative tradition argue that meaning (or LF) and form (or PF) are derived from the same deep structure and can be thought of as transformations from the same underlying representation (see Kratzer and Heim, 1998, p. 185). Steedman (2005) goes further and argues for a transparent syntax-semantics interface at every stage of derivation such that long-distance dependencies are addressed by ordered composition without transformations. Linguists in the Cognitive Grammar tradition posit that syntactic patterns are “motivated” by semantics and can therefore carry meanings themselves (Taylor, 2003, p. 29).

1.2.2 Abstractions, Generalizations, and Inheritance

In addition to the compositional pairing of syntax and semantics, many linguists posit that particular syntactic-semantic patterns are present in the lexicon and generalize across many cases. Proponents of Frame Semantics (Fillmore, 1982) argue that the syntactic structures encoding predicate-argument relationships generalize across different lexical units based on their semantic association with abstract scenes, called frames (Osswald and Van Valin, 2014). Construction grammarians like Goldberg go further, arguing that lexico-syntactic patterns, called constructions, form a network where specific semantic and syntactic information is inherited from more abstract parent constructions (Goldberg, 1995).
Consider as an example the AGENT thematic role. In English and many other languages, the AGENT role is associated with the subject syntactic position, and so the AGENT of a clause can be identified from the syntax.

However the relationship is not an exact correspondence. For example:

1. “The boy opened the door” (AGENT)
2. “The key opened the door” (INSTRUMENT)
3. “The door opened” (THEME)

In the above sentences, subject position is able to denote 3 thematic roles (AGENT, INSTRUMENT, and THEME). See Dowty (1991) for an introduction to this problem of aligning thematic and syntactic roles as well as Grimm (2011) for an analysis of the semantics of case. Also, see Schäfer (2009) for a detailed introduction to this alternation in particular, called the “causative alternation”. So the relationship between syntax and semantics is not a perfect, unambiguous one. Ch. 7 will explore some of the discrepancies which come up between (AMR) semantics and (CCG) syntax.

1.2.3 Why Syntax

The main benefit of syntax (in particular CCG) to a semantic representation like AMR is that it provides a framework to relate AMR structure to the form of the corresponding sentence and explain how that AMR graph is derived. This is important for applying AMR as a tool for making an NLP task more interpretable because the relationship between an input sentence and a semantic representation of that sentence is itself a source of complexity for NLP tasks.

1.2.4 Why CCG

In Ch. 6 and 7, I rely on Combinatory Categorial Grammar (CCG; Steedman, 2000b) as the form of syntax and ordered composition for augmenting AMR with compositionality. There are several advantages to using CCG over other grammar formalism with available parsers. First, CCG has

---

5Many languages use morphological markings to mark the subject independent of word order—Japanese and Russian for example. Additionally, there are languages where the AGENT role is not the most salient feature and instead TOPIC and FOCUS take precedence.
robust parsers that perform well on text from many different domains. Second, CCG is designed with a transparent mapping between syntax and semantics at each stage of derivation. This means that at every step of ordered composition, CCG provides a linguistically interpretable explanation of the relationship between syntax and semantics. Third, CCG’s emphasis on functional operations for deriving new syntactic and semantic forms lends to an easier and simpler implementation of ordered composition when working with a novel representation of semantics like AMR. While grammar formalisms differ in terms of their theoretical claims about syntax, I do not concern myself with those differences in the context of this dissertation. I take the perspective of being methodologically agnostic with respect to theoretic claims and focus on what CCG has to offer in terms of interpretability and computational implementation. CCG is introduced in detail in §2.3.

1.3 Publications

The research presented in this dissertation resulted in two publications:

- “An Improved Approach for Semantic Graph Composition with CCG” (Blodgett and Schneider, 2019) was published at IWCS 2019.
- “Probabilistic, Structure-Aware Algorithms for Improved Variety, Accuracy, and Coverage of AMR Alignments” (Blodgett and Schneider, 2021) was published at ACL 2021.
Chapter 2

Background

2.1 Form, Meaning, and Composition

This section presents basic concepts related to meaning and composition which will allow me to motivate implementing analyzability and compositionality of AMR. One goal of this section is to show how notions of analyzability and compositionality can be operationalized from an NLP perspective.

2.1.1 Form-Meaning Pairs

The idea that language is composed of form-meaning pairs goes back to de Saussure (1916). He argues that the most fundamental property of language is its ability to pair symbols (words or morphemes) with concepts in an arbitrary way. The collection of such form-meaning pairs in a language is called its lexicon. The task of building an annotated lexicon is well studied by NLP researchers (Kipper et al., 2000; Kingsbury and Palmer, 2002; Baker et al., 1998; Miller, 1994).

2.1.2 Analyzability

We can define the analyzability of a semantic representation to mean our ability to infer form-meaning pairs given the representation and the sentence it represents. This term is closely related a classic principle in linguistics called Frege’s principle.

Frege’s Principle: The meaning of a compound expression is a function of the meanings of its parts and of the way the parts are combined.
Incorporating this well established linguistic property into AMR is one of the goals of this work. We can interpret *analyzability* as the first part of this principle—that the meaning of a compound expression is related to “the meanings of its parts.” *Compositionality* is then the process by which the meaning of a compound expression is derived. To illustrate how this principle can be operationalized, I will rely on the following scale to describe degrees of analyzability for a given semantic representation.\(^1\)

- **Analyzability 0**: No mapping of meanings to forms except at the sentence level (e.g., unaligned AMR, DRS (Basile et al., 2012))
- **Analyzability 1**: Many-to-many mappings between meanings and forms, mappings may be nested or overlap and are not reducible to a mutually exclusive collectively exhaustive subset (e.g., Szubert alignments (Szubert et al., 2018), EDS (Oepen and Lønning, 2006))
- **Analyzability 2**: A bijective mapping where the domain is a set of mutually exclusive collectively exhaustive subsets of the meaning representation and the codomain is a set of mutually exclusive (but non-exhaustive) subsets of the sentence.
- **Analyzability 2b**: A mapping with the properties of Analyzability 2 with the exception that a subset of the meaning representation is excluded from the mappings domain and identified as pragmatic phenomena (AMR alignments demonstrated in Ch. 4)

I will call any ordered pair associated with such a mapping an *alignment*. Analyzability 2b is necessary to account for parts of an AMR, or other representation, which are contributed by pragmatics and not easily explained in terms of Frege’s principle (for example, the duplicate subgraph alignments of Ch. 4).

### 2.1.3 Ordered Composition

Likewise, I will rely on a scale to describe degrees of compositionality for a given semantic graph representation. Composition firstly requires the existence of a set of form-meaning pairs, which can be constructed by taking the alignments (or ordered pairs) given an analyzable mapping of the

---

\(^1\)This scale is adapted from the Oepen et al. (2020) where the term anchoring “flavors” is used to categorize different levels of analyzability.
representation. I will also describe a graph function as “local” if it only operates on graphs whose associated forms are next to each other in the sentence.

- **Compositionality 0**: There exists some graph function which derives a complete graph from its analyzable parts, though the the implementation of this function may be trivial, such as memorizing the correct output for a given input.

- **Compositionality 1**: There exists a composition of local binary and unary functions and permutation functions that derive the correct graph (e.g., SWAP actions in a transition-based parser (Ballesteros and Al-Onaizan, 2017; Fernandez Astudillo et al., 2020) or transformations in a transformational grammar).

- **Compositionality 2**: There exists a composition of local binary and unary functions with an associated syntactic interpretation that derive the correct graph (CCG-AMR derivation presented in Ch. 6).

### 2.1.4 Multiword Expressions and Non-Compositionality

In some cases, form-meaning pairs which are larger than a word will have a unique meaning which is not derivable from its parts. These are called multiword expressions and include names (e.g., “John Smith”), idioms (e.g., “kick the bucket”), phrasal verbs (e.g., “pay attention”), and other expressions. Multiword expressions can in some cases be discontiguous meaning that they are composed of parts which are some distance apart from each other in a sentence (e.g., “made . . up” is a discontiguous multiword expression in the sentence “He made the story up”). A detailed introduction to multiword expressions is given in Baldwin and Kim (2010). Sag et al. (2002) discuss challenges that multiword expressions pose for NLP.
2.2 Abstract Meaning Representation

Abstract Meaning Representation² (AMR; Banarescu et al., 2013) is a graph representation of sentence meaning. An AMR³ represents the people, places, things, events and other concepts evoked by a sentence as well as the semantic relationships between them. Succinctly, an AMR represents who did what to whom for a given sentence. At the same time, AMR abstracts away from syntax, inflectional morphology, and part-of-speech, such that two sentences with essentially the same meaning will correspond to the same AMR.⁴ For example, the phrases

1. “The dog decided to chase the cat”
2. “The dog made a decision to chase the cat”
3. “It was decided by the dog to chase the cat”
4. “the dog’s decision to chase the cat”

are all represented by the same AMR (d/decide-01 :ARG0 d2/dog :ARG1 (c/chase-01 :ARG0 d2 :ARG1 c2/cat)), regardless of the morphological form of the lexeme decide or the surrounding syntax and phrasal structure.

2.2.1 Graph Structure

An AMR is represented as a rooted directed acyclic graph⁵ with nodes (also called concepts) representing events, entities, and other descriptive information evoked by the sentence and edges (also called relations) representing semantic roles. Figure 2.1 shows an AMR both as a graph and in the more common PENMAN notation.

---

²See the full guidelines here: https://github.com/amrisi/amr-guidelines/blob/master/amr.md
³The term AMR may be used to refer to either the representation scheme as a whole or to a particular semantic graph.
⁴As a practical matter, AMRs do exhibit differences when created by two different annotators. Inter-annotator agreement for AMRs is around 80% (Banarescu et al., 2013). This can happen for many reasons but most notable are: ambiguity of the guidelines, annotator error or bias, and differing judgements about how much pragmatic information to include in the AMR.
⁵Though cycles do exist in AMRs in the wild. According to Banarescu et al. (2013), about 0.3% of AMRs contain cycles. This suggests that, for a very small number of sentences, the DAG structure of AMRs may be too restrictive.
Figure 2.1: AMR for the sentence “Most of the students want to visit New York when they graduate.” The AMR is displayed as a graph (left) and in PENMAN notation (right).

2.2.2 LOGICAL STRUCTURE

It is worth comparing an AMR graph to the corresponding logical representation of sentence semantics, since this will allow readers to better understand the AMR notation. Specifically, AMR corresponds to a neo-Davidsonian (Parsons, 1990) representation of semantics, where entities and event instances are represented by variables and semantic roles are represented by relations. Figure 2.2 shows the neo-Davidsonian logical form for the same sentence depicted as an AMR in Figure 2.1. The logical form may be divided into three parts: the existence of variables (gray), descriptions of each variable as an instance of some entity or event (blue), and semantic role relations (red). In an AMR, the existence of variables (e.g. $\exists x, y, z, \ldots$) is always assumed for any variable in the graph, so it is not represented directly. “Instance” relations correspond to nodes in the graph, and semantic roles correspond to edges in the graph. Thus, each node in an AMR represents a person, event, or entity, and edges describe how those nodes are related. The relations :ARG0, :ARG1, :ARG2, etc. denote the 1st, 2nd, 3rd, etc. core arguments of an event, and non-core semantic roles are represented by edges labeled :time, :location, etc.
variables
\[ \exists w, p, p2, s, s2, i, m, v, c, n, g: \]
\[ \text{instance}(w, \text{want-01}) \land \text{instance}(p, \text{person}) \land \text{instance}(s2, \text{study-01}) \]
\[ \land \text{instance}(i, \text{include-91}) \land \text{instance}(p2, \text{person}) \land \text{instance}(s2, \text{study-01}) \]
\[ \land \text{instance}(m, \text{most}) \land \text{instance}(v, \text{visit}) \land \text{instance}(c, \text{city}) \land \text{instance}(n, \text{name}) \]
\[ \land \text{instance}(g, \text{graduate}) \]

concepts
\[ \land \text{arg0}(w, p) \land \text{arg0}(s, p) \land \text{arg1}(i, p) \land \text{arg2}(i, p2) \land \text{arg0}(s2, p2) \land \text{arg3}(i, m) \]

semantic roles
\[ \land \text{arg1}(w, v) \land \text{arg0}(v, p) \land \text{arg1}(v, c) \land \text{name}(c, n) \land \text{op1}(n, \text{“New”}) \land \text{op2}(n, \text{“York”}) \]
\[ \land \text{time}(v, g) \land \text{arg0}(g, p) \]

Figure 2.2: The equivalent logical form for the sentence “Most of the students want to visit New York when they graduate.” The formula is divided into 3 parts—variables (gray): the existence of variables, concepts (blue): descriptions of each variable as an instance of some entity or event, and semantic roles (red): relations describing the role of each argument or other semantic relationship. Note that instance is an AMR role and instance(x, f) is equivalent to f(x) in typical predicate calculus notation.

2.2.3 PropBank

The inventory of event types (also called frames) used in AMR is derived from Proposition Bank (PropBank; Kingsbury and Palmer, 2002; Palmer et al., 2005). PropBank is a corpus and inventory of 8,733 semantic frames for particular senses of a word—including verbs like destroy (destroy-01) and eventive nouns and adjectives like destruction (destroy-01) and afraid (fear-01)—and a list of associated word-specific arguments accepted by that sense. Each frame is a predicate with one or more semantic roles, corresponds to a particular word sense, and was attested in PropBank’s annotated data. For example, PropBank includes a frame want-01 which has semantic roles ARG0: wanter, ARG1: thing wanted, ARG2: beneficiary, ARG3: in-exchange-for, and ARG4: from. The interpretation of each core argument depends on the frame it is taken as an argument of, however, ARG0 generally refers to a proto-agent and ARG1 to a proto-theme. See Dowty (1991). PropBank’s inventory of frames allows us to give detailed descriptions of events and the participants they take as arguments in a succinct way, such that the meaning can be easily looked up in PropBank’s inventory.
2.2.4 Morphology

AMR includes some morphological separation when interpreting the meaning of some words. When the meaning of a word is clearly compositional with respect to its (non-inflectional) morphemes and each morpheme contributes some part of the meaning, AMR decomposes the word into separate nodes each representing one part of the evoked meaning. For example *teacher* is represented as 

(person :ARG0-of teach-01) (a person who teaches).

Examples of words broken apart based on their morphology include agentive nouns (e.g., *teacher* as (person :ARG0-of teach-01)), negation (e.g., *unimportant* as important-01 :polarity -), and comparatives/superlatives (e.g., *bigger* as big :degree more). See Figure 2.3. In other cases where the semantics of a word can be analyzed in terms of events and participants, AMR decomposes these regardless of their morphology: *students* becomes (person :ARG0-of study-01), *advice* becomes (thing :ARG1-of advise-01).

2.2.5 Named Entities

In order to incorporate named entity recognition into AMR’s representations, AMR uses a special structure to represent named entities, such as “New York” in Figure 2.1 above. Each named entity is assigned an entity type such as person, country, city, government-organization, animal, vehicle, and so on. In total, AMR uses 110 named entity types, with an additional 18 used in biomedical corpora. In addition to the entity type, each named entity is given a name property
which takes arguments using the relations :op1, :op2, etc. for each token in the entity name. See Figure 2.4.

2.2.6 **Reentrancies Represent Control, Coreference, etc**

AMR allows a node to take multiple parents. When there are multiple edges pointing to the same node, we call these edges *reentrancies*. In AMR, reentrancies are used in any case where a concept functions as the semantic argument of multiple other concepts, which can happen because of control, pronominal coreference, and a number of other phenomena. Figure 2.5 shows the reentrancies present in our example sentence: “Most of the students want to visit New York when they graduate.” In the sentence, *students* is taken as a semantic argument of three events: a *wanting* event, a *visiting* event, and a *graduating* event. Each semantic argument that *students* occupies corresponds to an edge in the graph, and each edge can be explained by some linguistic phenomenon: the edge (*want*-01 :ARG0 person) is present because *students* is the syntactic subject of *want*, the edge (*visit*-01 :ARG0 person) is present because *students* is the semantic subject of *visit*, etc.

---

6Typically researchers use the term *reentrancy* to refer only to the “extra” or re-entrant edges, implying that one of the edges is primary and not a reentrancy. However, since identifying the primary edge is not always trivial in an automatic setting, we opt to refer to complete set of edges pointing to the same target (primary and re-entrant) as reentrancies anytime there is more than one parent.
Figure 2.5: Reentrancies example for the sentence “Most of the students want to visit New York when they graduate”. This figure displays a subgraph of the AMR for the sentence “Most of the students want to visit New York when they graduate” with relevant words in bold. This figure shows the multiple edges (called reentrancies) pointing to person in the graph. Reentrancies are dashed and highlighted in red.

(:ARG0 person) is present because of the control verb want passing its semantic argument to visit, and the edge (graduate-01 :ARG0 person) is present because the pronoun they is coreferent with students. In Ch. 4, we show how reentrancies can be classified and aligned to spans in a sentence which trigger them.

2.2.7 Special AMR Frames

In its goal of large-scale semantic coverage, AMR includes a number of special frames for handling situations that are not covered by PropBank alone. For example have-degree-91 can be used to express complex comparatives like “I am taller than you”, include-91 can be used to express subset constructions like “most of the students”, etc. An inventory of special frames and their use cases can be found at the AMR Annotation Dictionary.\(^7\)

\(^7\)https://www.isi.edu/~ulf/amr/lib/amr-dict.html
Implicit Semantics

AMR includes some pragmatic and morphologically implicit sentence meaning. For example, AMR uses the node say-01 as the root node of vocative expressions. In the case of imperatives, AMR depicts the imperative mood and 2nd person subject explicitly: “Eat your vegetables” becomes (e/eat-01 :mode imperative :ARG0 y/you :ARG1 (v/vegetable :poss y)). This means that in practice, there are nodes depicted in AMR which do not correspond to any pronounced part of the sentence.

Scope of AMR

For simplicity, AMR does not include representation of definiteness, tense, aspect, or quantifier scope. As a consequence, some semantic information is lost when converting a sentence to AMR. However, many extensions to AMR have been proposed to incorporate linguistic information including constructions\(^8\) (Bonial et al., 2018), tense and aspect (Donatelli et al., 2018), quantifier and negation scope (Pustejovsky et al., 2019; Lai et al., 2020; Bos, 2020), spatial relations (Dan et al., 2020; Bonn et al., 2020), multi-sentence semantics (O’Gorman et al., 2018), and illocutionary force (Bonial et al., 2020a). Additionally, Uniform Meaning Representation (UMR; Van Gysel et al., 2021) is a recent extension of AMR which adds optional features for tense, aspect, scope, and modality, and revises AMR to make it more typologically plausible and more adaptable for annotation in additional languages, especially in low-resource settings.

Data

A number of corpora of gold AMR annotations have been built in several languages and domains. AMR annotation has been done on English, Mandarin Chinese (Li et al., 2016), Brazilian Portuguese (Anchiêta and Pardo, 2018; Sobrevilla Cabezudo and Pardo, 2019), Korean (Choe et al., 2020), Spanish (Migueles-Abraira et al., 2018), Vietnamese (Linh and Nguyen, 2019), Indonesian (Ilmy and Khodra, 2021), and Czech (Xue et al., 2014a). Table 2.1 shows the sizes of released AMR

\(^8\)Annotation of constructions is included in AMR Release 3.0.
Table 2.1: Released gold AMR data for 5 languages.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total Sentences</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMR Release 3.0</td>
<td>60k</td>
<td>LDC2020T02</td>
</tr>
<tr>
<td>Little Prince</td>
<td>1.5k</td>
<td>ISI Downloads</td>
</tr>
<tr>
<td><strong>English (Biomedical)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bio AMR Corpus</td>
<td>7k</td>
<td>ISI Downloads</td>
</tr>
<tr>
<td><strong>Mandarin Chinese</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese AMR 1.0</td>
<td>10k</td>
<td>LDC2019T07</td>
</tr>
<tr>
<td>Little Prince (CAMR)</td>
<td>1.5k</td>
<td>CAMR</td>
</tr>
<tr>
<td><strong>Brazilian Portuguese</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anchiêta and Pardo (2018)</td>
<td>1.5k</td>
<td>GitHub</td>
</tr>
<tr>
<td><strong>Korean</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choe et al. (2020)</td>
<td>1.2k</td>
<td>GitHub</td>
</tr>
<tr>
<td><strong>Spanish</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migueles-Abraira et al. (2018)</td>
<td>50</td>
<td>GitHub</td>
</tr>
</tbody>
</table>

corpora in various languages for which data is available online. Some researchers have also used automatically generated AMR data for parsing experiments in languages that are not yet supported by a gold corpus (Damonte and Cohen, 2018). Most of the available AMR data is based on news data (including the Wall Street Journal corpus) and the public domain translations of *The Little Prince* (Le Petite Prince; de Saint-Exupéry, 1943/1986), however AMR has also been applied in biomedical (Rao et al., 2017), legal (Viet et al., 2017), and robot-dialogue (Bonial et al., 2019) settings.

Additionally, Vigus et al. (2020) and Van Gysel et al. (2021) provide resources for annotators wishing to create AMR corpora in low-resource settings, and Xue et al. (2014a) conduct experiments to investigate sources of difficulty in extending AMR to new languages.
2.2.11 Tools and Resources

A few resources have been made available for learning, annotating, and using AMR. In particular, the AMR Annotation Guidelines\textsuperscript{9} and the AMR dictionary\textsuperscript{10} both provide useful instruction about how AMR is annotated and what various constructions mean, a GUI AMR editor\textsuperscript{11} is available for creating AMR annotations in an intuitive and efficient interface, and the AMR Bibliography\textsuperscript{12} provides a sortable, topic-annotated display of the currently 191\textsuperscript{13} papers and works on research related to AMR.

Useful python libraries include PENMAN\textsuperscript{14} (Goodman, 2020) for reading and writing AMR data and AMR-utils \textsuperscript{15} for useful operations and visualization of AMRs. Evaluation tools include SMATCH\textsuperscript{16} (Cai and Knight, 2013), a fine-grained version of SMATCH for various linguistic categories\textsuperscript{17} (Damonte et al., 2017), and S\textsuperscript{2}MATCH\textsuperscript{18} (Opitz et al., 2020), an extension of SMATCH which incorporates semantic similarity scores between nodes.

2.3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG; Steedman, 2000b) is a lexicalized\textsuperscript{19} grammar formalism for representing both syntax and semantics in a way that maintains a transparent mapping between the two. Objects in CCG are constituents, which can be either a word or a phrase and which are accompanied by a syntactic label (a supertag) and a semantics. Operations in CCG are called combinators, and these are used to combine or alter constituents in a conventional way.

\textsuperscript{9}https://github.com/amrisi/amr-guidelines/blob/master/amr.md
\textsuperscript{10}https://www.isi.edu/~ulf/amr/lib/amr-dict.html
\textsuperscript{11}https://amr.isi.edu/editor.html
\textsuperscript{12}https://nert-nlp.github.io/AMR-Bibliography/
\textsuperscript{13}at the time of writing, August 19, 2021
\textsuperscript{14}https://github.com/goodmami/penman
\textsuperscript{15}https://github.com/ablodge/amr-utils
\textsuperscript{16}https://github.com/snowblink14/smatch
\textsuperscript{17}https://github.com/mdtux89/amr-evaluation
\textsuperscript{18}https://github.com/flipz357/amr-metric-suite
\textsuperscript{19}A lexicalized grammar is a grammar where much of the mechanics is determined by lexical entries rather than generally applicable rules. CCG, HPSG (Pollard and Sag, 1994), and LFG (Dalrymple, 2001) are considered of lexicalized grammars.
Table 2.2: Common CCG categories/supertags for various types of words and phrases.

<table>
<thead>
<tr>
<th>Part-of-speech</th>
<th>Category/Supertag</th>
<th>Part-of-speech</th>
<th>Category/Supertag</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>N</td>
<td>sentence, complete clause</td>
<td>S</td>
</tr>
<tr>
<td>noun phrase</td>
<td>NP</td>
<td>verb phrase, intransitive verb</td>
<td>S\NP</td>
</tr>
<tr>
<td>determiner</td>
<td>NP/N</td>
<td>transitive verb</td>
<td>(S\NP)/NP</td>
</tr>
<tr>
<td>adjective</td>
<td>N/N</td>
<td>adverb (adjective modifying)</td>
<td>(N/N)/(N/N)</td>
</tr>
<tr>
<td>prepositional phrase (core)</td>
<td>PP</td>
<td>adverb (verb modifying)</td>
<td>(S\NP)/(S\NP)</td>
</tr>
<tr>
<td>preposition (core)</td>
<td>PP/NP</td>
<td>preposition (adjunct)</td>
<td>(S\NP)/(S\NP)/NP</td>
</tr>
</tbody>
</table>

2.3.1 Syntactic Supertags

For a given token or phrase, the representation of the syntactic type is called its category in theoretical linguistics or it’s supertag among parsing researchers. A supertag is a succinct representation of the syntactic properties of a phrase, describing the types of phrase’s it is able to combine with. For a given phrase, a supertag is paired with a semantics to serve as the representation for that phrase. Supertags are built up from atomic types, including N (noun), S (sentence), etc., and slashes. CCG uses forward and backward slashes to represent a phrase’s ability to take arguments. A forward slash (/) indicates that a phrase expects an argument on the right. A backward slash (\) indicates that a phrase expects an argument on the left. Supertags are then defined recursively as follows:

1. Atomic types are supertags: N, NP, S, PP, etc.
2. If A and B are supertags, A/B is a supertag which combines with a phrase of type B on the right and becomes a phrase of type A.
3. If A and B are supertags, A\B is a supertag which combines with a phrase of type B on the left and becomes a phrase of type A.

Additionally, some atomic types can be featurized to further constrain the syntactic operations they can participate in: S[b], S[to], S[q], etc. Examples of common supertags are given in Table 2.2. Note that because of the recursive definition of supertags, possible supertags have in principle an unlimited number, and many rare supertags appear in real data.\footnote{See for example Prange et al. (2021) who address the problem of predicting rare supertags that appear in CCG data by modelling the internal structure of each supertag.}
The mechanics of CCG work because it assumes that syntactic categories correspond to mathematical functions. A phrase with a category \(X/Y\) must have a semantic representation that looks like \(\lambda y, f(y)\) which must be a function of one variable. Any operation that acts on this phrase to fill its syntactic argument \(Y\) must also fill its semantic argument \(y\). Thus CCG maintain a correspondence between syntactic arguments and semantic arguments at each stage of derivation. It is CCG’s emphasis on functional semantics and a correspondence between syntactic and semantic arguments that allows its transparent syntax-semantics interface. This assures that the relationship between syntax and semantics is clear at every step of a derivation.

2.3.2 COMBINATORS

The syntactic operators that CCG uses to combine phrases are called combinators. Combinators describe both how the syntactic types of phrases change after they are combined as well as how the semantics is updated. They also serve as the primary way of constraining what phrases are grammatical and how phrases can be combined. The descriptions and effects of several combinators

<table>
<thead>
<tr>
<th>Table 2.3: Syntactic effects of CCG Combinators.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COMBINATOR</strong></td>
</tr>
<tr>
<td>forward application</td>
</tr>
<tr>
<td>backward application</td>
</tr>
<tr>
<td>forward composition</td>
</tr>
<tr>
<td>backward composition</td>
</tr>
<tr>
<td>(crossing) composition</td>
</tr>
<tr>
<td>(2(^{nd}) order) composition</td>
</tr>
</tbody>
</table>

| **SYMBOL** | **LEFT** | **RIGHT** | **RESULT** |
|-------------------|-------------------|-------------------|
| forward type raising | >T[Y]   | X \(c\) | Y/(Y/X) \(\lambda f.f(c)\) |
| backward type raising    | <T[Y]   | X \(c\) | Y/(Y/X) \(\lambda f.f(c)\) |

| **SYMBOL** | **LEFT** | **RIGHT** | **RESULT** |
|-------------------|-------------------|-------------------|
| coordination      | \&                | X... \(\lambda y... f(y)\) conj X... \(\lambda z... g(z)\)... | X... \(\lambda y... f(y)... \land g(y)...\) |

Each binary combinator applies to two constituents with supertags of the form shown and produces the result shown, while each unary combinator applies to one constituent and produces the result shown. The corresponding semantics of each constituent is shown in gray, where \(c\) is a constant, \(w\), \(y\), and \(z\) are variables, and \(f\) and \(g\) are functions. See Hockenmaier and Steedman (2005) for more details.
Most of the students want to visit New York when they graduate.

Figure 2.6: CCG derivation for the sentence “Most of the students want to visit New York when they graduate.”

are summarized in Table 2.3. The simplest combinators are forward and backward application, which can be defined as follows. Forward application (>) combines a left phrase of type X/Y (for arbitrary sub-tags X and Y) with semantics $\lambda y.f(y)$ and a right phrase of type Y with semantics $c$ where $c$ is a constant. Forward application treats the right phrase as a syntactic and semantic argument of the left phrase resulting a new phrase of type X with semantics $f(c)$. Backwards application similarly combines phrases Y : $c$ and X \ Y : $\lambda y.f(y)$ to form a phrase X : $f(c)$. Other combinators allow phrases to combine in more complex ways. Application can be thought of as filling a function with its argument. Forward and backward composition can be thought of as composing two functions. Type raising can be thought of as changing the syntactic type of a phrase to turn it into a higher order function. Type raising and composition are most often used to change the order of composition, due to some grammatical constraint, while still producing the correct semantics. Various extensions of composition exist which allow parsing in a less constrained way. Following CCGBank, we additionally allow a combinator lex for certain unary, type changing rules such as converting an N directly to an NP in the case of plural nouns and mass nouns in English.
A CCG derivation used to build the syntax and semantics of a sentence can be seen in Figure 2.6. Each horizontal line indicates which phrases are being combined and which combinator is being applied. For example, *the* (NP/N) and *student* (N) combine with forward application to form an NP. For a given phrase, the supertags of neighboring phrases constrain what it is able to combine with and in what order. The complete derivation of the sentence allows us to build the full semantics of the sentence by means of compositionality.

2.3.3 Data

CCGBank (Hockenmaier and Steedman, 2007) is a corpus of annotated CCG parses for 41k English sentences based on the Penn Treebank. A later version of the corpus, Rebanked CCGBank (Honnibal et al., 2010), modifies the CCGBank corpus to add improved interpretation of deep structure and better dependencies for representing semantic arguments. For a guide on the structure of CCGBank parses, see Hockenmaier and Steedman (2005).
CHAPTER 3

LITERATURE REVIEW

3.1 SYMBOLIC INTERPRETABILITY OF LINGUISTIC STRUCTURES AND NEURAL NETWORKS

This section presents a broad overview of several avenues of research where linguistic interpretability is incorporated into neural models.

3.1.1 LINGUISTIC SYMBOLS AS FEATURES

The simplest and by far the most widely used approach for incorporating linguistic structures into a neural model is to use them as additional input features. Linguistic symbols as features is far too ubiquitous in natural language processing to be discussed comprehensively here, but some illustrative examples include use of AMR features for machine translation (Song et al., 2019) improving over an attention-based sequence-to-sequence baseline, summarization (Hardy and Vlachos, 2018), information extraction (Wang et al., 2017) improving over word and dependency embeddings alone, question answering (Mitra and Baral, 2016), and common sense reasoning (Lim et al., 2020) improving over knowledge graphs alone; SRL features have been applied in machine translation (Marcheggiani et al., 2018), information extraction (Christensen et al., 2010), and question answering (Shen and Lapata, 2007); syntactic dependency features have been applied in machine translation (Zhang et al., 2019a), natural language inference (He et al., 2020), question answering (Meng et al., 2017), among others.

3.1.2 LINGUISTICALLY MOTIVATED NEURAL ARCHITECTURES

One perpetual area of interest for computational linguists is to devise model architectures which are motivated by and reflect linguistic structures or linguistic processes. A simple example is recursive
neural networks discussed below, which are designed to mimic syntactic phrase structure—recursive neural networks have a tree-structured architecture where word embeddings are used as terminals in the tree and each non-terminal represents a constituent span. Linguistically motivated neural architectures are not strictly symbolically interpretable in the sense defined in Ch. 1, but they are included for discussion here because they allow neural models to be analogous to linguistic structures. They add inductive biases to a model by constraining the types of representations which are learned to be linguistically structured and add interpretability by allowing users to interpret parts of the architecture as corresponding to linguistic objects such as phrases or syntactic/semantic relations.

**Recursive Neural Networks.** A recursive neural network (Socher et al., 2012) builds a compositional representation of a sequence of words in a tree-structured way. Recursive neural networks follow the phrase structure of a syntactic parse to determine their shape. Each token in the input is assigned a word embedding which is taken as an input, and for each phrase in the tree, a vector representation for that phrase is computed as a function of the vector representations of its children, where the composition function is some linear function with weight-sharing across different phrases followed by some activation function. The top node in a recursive neural net is then a compositional representation of the input text. Socher et al. (2013) use a recursive neural network to perform sentiment analysis, while preserving the phrase structure of the sentence. Sentiment analysis relies on phrase structure because the scope of negation and negative-sentiment modifiers can have a large impact on how the sentiment of an input text should be interpreted.

**Self-Attention.** Vaswani et al. (2017) introduce self-attention as a part of the transformer architecture. Self-attention allows the computation of contextualized word representations where each word in an input text is represented as a weighted average of word embeddings from surrounding words. Further, an attention matrix is used to determine how important word $a$ is for building the representation of word $b$. So, in calculating an attention matrix, a model with self-attention learns to assess which words in an input text are related and how strongly related they are. Henderson
(2020) argues that self-attention is an induced-structure model rather than a sequence model, and therefore is analogous to the dependency structure of dependency syntax trees or semantic roles. This is especially true when the self-attention head is trained to represent dependency structure directly, as is done by Strubell et al. (2018).

**Stack-LSTMs.** Dyer et al. (2015) propose a model for dependency parsing which represents phrases compositionally, similar to Socher et al. (2012). The algorithm relies on a stack-LSTM, which is a sequence model designed to efficiently push or pop embeddings at the top of the sequence. Dyer et al. (2015) use stack-LSTMs to represent the traditional stack and buffer of shift-reduce dependency parsing (Aho and Ullman, 1972), with the addition that each time a dependency is added, the head and dependent are combined to form a new vector representation. Items in the stack thus represent phrases\(^1\) whose representation is built compositionally from the representations of the contained words.

**Graph Convolutional Networks.** Convolutional neural networks are a type of feed forward neural model that place focus on local information by, for each input feature, coalescing information from that input feature and its neighbors and using that combined information to produce an output.\(^2\) For example, convolutional neural networks can learn patterns from the relationship between neighboring pixels in an image or between neighboring words in a sentence. Graph convolutional networks (Kipf and Welling, 2017) are an extension of convolutional neural networks where the neighboring input features are nodes in a graph structure. Graph convolutional networks are useful in a number of contexts where linguistic information is best captured by a tree or graph structure. Marcheggiani and Titov (2017), for example, apply a graph convolutional network over dependency parse inputs and obtain improved performance in semantic role labeling.

---

\(^1\)or possibly discontinuous “phrases” if an action SWAP is included and non-projective parses are allowed

\(^2\)Specifically, convolutional neural networks use a “kernel” of shared weights to coalesce information from a span of neighboring features to produce an output. They do that for every span of a certain size and shape (possibly 1-, 2-, or 3-dimensional) in the input and pass all the computed outputs to the next layer.
3.1.3 Linguistic Symbols as Intermediary Representations

Some algorithms use a linguistic structure as intermediary representations, first converting some input to that linguistic structure and then using that new representation for the remainder of the task. AMR has been used as an intermediary representation for a variety of tasks: human-robot dialogue (Bonial et al., 2021; Abrams et al., 2020; Bonial et al., 2019), summarization (Liao et al., 2018; Liu et al., 2015; Dohare and Karnick, 2017), machine translation (Jones et al., 2012), question answering (Khashabi et al., 2018), information extraction (Rao et al., 2017), semantic search (Bonial et al., 2020b), and entity linking (Pan et al., 2015). Some of these are novel works pioneer novel tasks (Bonial et al., 2021; Abrams et al., 2020; Bonial et al., 2019), while others demonstrate promising improvements over standard benchmarks (Liao et al., 2018; Rao et al., 2017; Pan et al., 2015).

3.1.4 Linguistic Symbols as an Auxiliary Task

Alternatively, linguistic structure can be incorporated into a neural model by training it to predict that structure as an auxiliary task, which may constitute transfer learning or multitask learning.

**Transfer Learning.** Transfer learning is an approach where a model is pre-trained on some auxiliary task, where knowledge learned for the auxiliary task is believed to generalize well to many other tasks. The pre-trained model is then made available to be used in other tasks and experiments. In NLP, pre-trained models are usually associated with word embeddings such as GloVe (Pennington et al., 2014) or language models such as Elmo (Peters et al., 2017) or BERT (Devlin et al., 2019) and its extensions, which learn a representation from raw text rather than explicit linguistic structure. Zhou et al. (2020) design a linguistically pre-trained language model called LIMIT-BERT, which is built on top of BERT and fine-tuned on 5 auxiliary tasks (part-of-speech tagging, dependency parsing, constituency parsing, span semantic roles, and dependency semantic roles). LIMIT-BERT was evaluated for syntactic parsing, semantic parsing, natural language inference, and the multi-task

---

3RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2020), LIMIT-BERT (Zhou et al., 2020), TinyBERT (Jiao et al., 2020), SciBERT (Beltagy et al., 2019), Sense-BERT (Levine et al., 2020), BART (Lewis et al., 2020), etc.
benchmark GLUE (Wang et al., 2018a), and was found to improve performance on syntactic parsing, natural language inference, 1 out of 2 semantic parsing settings, and 5 out of 8 GLUE benchmark natural language understanding tasks relative to a BERT baseline.

**Multitask Learning.** Multitask learning (Caruana, 1997) is the approach of having a model perform its main task jointly with an auxiliary task during training, where knowledge learned for the auxiliary task is believed to be beneficial for the model’s performance in the main task. Strubell et al. (2018) perform multitask learning of semantic role labels with auxiliary tasks of dependency parsing, part-of-speech tagging, and predicate detection, in a transformer architecture. Further, they train the self-attention heads to attend to the syntactic parent of every token, resulting in self-attention heads which represent the dependency syntax for the sentence. Martínez Alonso and Plank (2017) experiment with multitask learning with 5 main tasks (frame semantics, WordNet supersenses, named entity recognition, EuroWordNet coarse-grained semantics, and multi-perspective question answering) and 4 auxiliary tasks (VP and NP chunking, universal dependency labels, part-of-speech tagging, and word log-frequency prediction) in a bidirectional LSTM setting. They conclude that auxiliary tasks with a label set that is compact and has a uniform frequency distribution are most beneficial. Staliūnaitė and Iacobacci (2020) experiment with multitask learning for conversational question answering with 4 auxiliary tasks (negation scope, position relative to the answer, sentiment, and semantic role labels) and perform error analyses on RoBERTa, BERT, and DistilBERT to see when they may be improved by an auxiliary task. They find that an ensemble of models based on each auxiliary task performs well. Multitask learning of linguistic structure has also been shown to improve the performance of machine translation (Nädejde et al., 2017; Eriguchi et al., 2017; Wu et al., 2017), natural language inference (Cengiz and Yuret, 2020), and question answering (Alizadeh and Eugenio, 2020).

Additionally, multitask learning of linguistic structures has been shown to be useful in cases where both the main and auxiliary tasks have linguistic structure outputs. Swayamdipta et al. (2018) use multitask learning of a shallow set of syntactic features and improve performance in 3 semantic
prediction tasks: predicting PropBank (Kingsbury and Palmer, 2002) semantics and roles, predicting FrameNet (Baker et al., 1998) semantics and roles, and coreference resolution. The particular relationship between syntactic and prediction of semantics is discussed further in §3.1.5.

Finally, Hershcovich et al. (2018) experiment with cross-framework multitask learning where several semantic parsing tasks are learned jointly allowing for knowledge gained in one task to benefit performance in the others.

3.1.5 The Relationship between Syntax in NLP

A number of works in NLP have focused on the question of whether and when syntactic information is useful for inferring semantic information. For example, Marcheggiani and Titov (2017) apply a graph convolutional network to use dependency parses as features for better prediction of semantic roles, Strubell et al. (2018) use dependency parsing as an auxiliary task to train self-attention heads for better prediction of semantic roles, Lewis et al. (2015) jointly predict CCG syntax and semantic roles using a compositional framework, and Swayamdipta et al. (2016) and Lluís et al. (2013) jointly parse syntax and semantics using transition-based and arc-factored methods respectively.

Another strategy is to develop simple or shallow syntactic features that are easier to incorporate into a complex parser. One advantage of this approach is that it can reduce the noise that comes with more complex semantic schemas. As noted above, Swayamdipta et al. (2018) use “syntactic scaffolds”, which are a set of shallow syntactic features, as an auxiliary task to improve semantic prediction of PropBank (Kingsbury and Palmer, 2002) frames and roles, FrameNet (Baker et al., 1998) frames and roles, and coreference.
3.2 Abstract Meaning Representation

This section reviews research related to AMR (Banarescu et al., 2013).

3.2.1 Related Semantic Representations

A number of semantic representations and annotated English corpora have been developed that are worth comparing to AMR. All of these representations encode semantics for the entire sentence and have English datasets available. Each of them carries different design decisions that affect how much information they convey, how easy or difficult they are to annotate, and how useful they are to an NLP application. Most semantic representations discussed in this section are designed in principle to have broad language support, though they may only have available data for particular languages.

**Elementary Dependency Structures (EDS).** EDS (Oepen and Lønning, 2006) is a dependency semantic representation based on Minimal Recursion Semantics (Copestake et al., 2005) and with a grammatical basis in Head-driven Phrase Structure Grammar (Pollard and Sag, 1994). EDS was designed to be equivalent to logical form including scope, while allowing underspecification of ambiguous scope.

**Discourse Representation Structures (DRS).** DRS (Basile et al., 2012; Kamp and Reyle, 1993) is a schema based on Discourse Representation Theory (Kamp and Reyle, 1993) and associated with the Parallel Meaning Bank (Abzianidze et al., 2017; Bos et al., 2017). DRS includes annotation of discourse relations, quantifier and negation scope, and other logical and pragmatic features.

**Prague Dependency Treebank (PDT).** The Prague Dependency Treebank (Hajič et al., 2020) is a multi-layer corpus including morphological, syntactic, semantic, and pragmatic features including dependency syntax, thematic roles, coreference, ellipsis, topic and focus, discourse relations, and others. PDT is based on a grammar formalism called Functional Generative Description (Sgall, 1967).
Table 3.1: A comparison of attributes for 6 semantic representations with support for English.

<table>
<thead>
<tr>
<th></th>
<th>Lexicon</th>
<th>Languages</th>
<th>Anchored</th>
<th>Word senses</th>
<th>Semantic roles</th>
<th>Thematic roles</th>
<th>Coreference</th>
<th>Entity types</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMR</td>
<td>PropBank</td>
<td>en, es, ko, pt-br, zh</td>
<td>✗</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>DRS</td>
<td>WordNet, VerbNet</td>
<td>de, en, it, nl</td>
<td>✗</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>EDS</td>
<td>in house</td>
<td>de, en, es, ja</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>PDT</td>
<td>in house</td>
<td>cs, en</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>UCCA</td>
<td>–</td>
<td>de, en, fr</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>UDS</td>
<td>WordNet</td>
<td>en</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>

1 EDS allows overlapping anchors.
2 PTG allows discontiguous anchors.
3 UDS uses proto-roles based on Dowty (1991) instead of simple thematic roles.

For a comparison of semantic representations across languages, see (Žabokrtský et al., 2020).

**Universal Conceptual Cognitive Annotation (UCCA).** UCCA (Abend and Rappoport, 2013) is a dependency graph annotation schema based in cognitive linguistics designed to be as intuitive to non-linguists as possible and to exemplify the same or similar structure across languages. It does not, however, disambiguate word senses or include semantic roles.

**Universal Decompositional Semantics (UDS).** UDS (White et al., 2016) uses featurized dependency graphs using semantic proto-roles, feature-based word sense decomposition, event properties based on Dowty (1991) and with an emphasis on crowd-sourced annotation based on carefully crafted questions.

Table 3.1 compares properties of these representations along with AMR. Word sense disambiguation and semantic role labelling require a lexicon to catalogue possible word senses and argument structures. All of these semantic representations can be converted to a featurized graph form. See for example Abzianidze et al. (2020) on converting DRS to a graph format and Zeman and Hajic (2020) on converting PDT to a graph format. Oepen et al. (2020) and Oepen et al. (2019)
organized shared tasks on multi-framework parsing where participants designed systems that were capable of predicting multiple type of these semantic representation formats as semantic graphs.

A related area of research to semantic representations is the development of computational lexicons (or *lexica*), such as PropBank (*Kingsbury and Palmer, 2002; Palmer et al., 2005*), WordNet (*Miller, 1994; Fellbaum, 1998*), VerbNet (*Kipper et al., 2000; Kipper-Schuler, 2005*), and FrameNet (*Baker et al., 1998*). Some semantic representations rely on a pre-existing lexicon, like those described above, and some use on an in-house lexicon. Computational lexicons define and allow us to disambiguate word senses. Different lexicons also offer different tools for semantic annotation. WordNet describes ontological relationships between word senses including synonymy, antonymy, and hyponymy. PropBank and VerbNet define the semantic roles associated with each word sense. FrameNet\(^4\) defines a network of lexical items and constructions (called *frames*) based on shared properties and participants that frames have in common.

AMR stands out from these frameworks for its emphasis on predicate-argument structure, its relative simplicity and ease of annotation, and its not being anchored to tokens during annotation in English. Being an unanchored representation can be understood as a strength in that it allows easy and scalable annotation and easier abstraction away from syntax, but it also means that AMR annotation is missing a vital feature of linguistic interpretability, making AMR alignment a necessary area of research. Ch. 4 and 5 demonstrate a structurally comprehensive alignments dataset and aligner.

### 3.2.2 Applications

While the problem of using AMR effectively and efficiently in downstream tasks is still very much an open area of research, AMR has been used in applications including machine translation (*Wu et al., 2017; Jones et al., 2012; Song et al., 2019*), robot-human dialogue (*Abrams et al., 2020; Bonial et al., 2019*), summarization (*Liu et al., 2015; Dohare and Karnick, 2017; Liao et al., 2018; Hardy and Vlachos, 2018*), information extraction (*Rao et al., 2017; Garg et al., 2016; Wang et al., 2016*), and...
2017; Zhang and Ji, 2021), question answering (Mitra and Baral, 2016), entity linking (Pan et al., 2015), machine reading comprehension (Sachan and Xing, 2016), paraphrase detection (Issa et al., 2018), common sense reasoning (Lim et al., 2020), event detection (Li et al., 2015), and semantic search (Bonial et al., 2020b). Most of the above applications use AMR as an additional feature input, but some operate on AMR directly, using it as an intermediary representation (Liao et al., 2018; Pan et al., 2015; Liu et al., 2015; Rao et al., 2017; Dohare and Karnick, 2017; Bonial et al., 2020b).

3.2.3 Alignments

A detailed introduction to AMR alignment is given in §4.2.

3.2.4 Evaluation

S\text{MATCH} (Cai and Knight, 2013) is a well established evaluation metric for measuring AMR quality. S\text{MATCH} works by comparing a parsed (or annotated) AMR to a gold-standard AMR, by first aligning the two graphs using heuristic-driven semi-random initialization and hill climbing. S\text{MATCH} then calculates an F1 score of matched or mismatched concepts and relation triples for the two graphs.

Several works have suggested improvements to S\text{MATCH}. S^{2}\text{MATCH} (Opitz et al., 2020) extends S\text{MATCH} by incorporating semantic similarity scores to assign partial credit to differently labeled but semantically similar nodes. They use cosine distance between GloVe vectors (Pennington et al., 2014) to calculate the score. Damonte et al. (2017) suggests a fine-grained evaluation which calculates S\text{MATCH} for several linguistically defined subcategories. Goodman (2019) normalizes several conceptually equivalent aspects of AMR graphs, such as relation reification, before running S\text{MATCH}. Opitz and Frank (2019) suggests a set of internal and external evaluation metrics for rating the quality of AMR parsers in the absence of gold data. S\text{EMBLEU} (Song and Gildea, 2019) suggest an evaluation based on BLEU (Papineni et al., 2002) which converts each graph to n-grams and uses a BLEU score to compare the n-grams. S\text{EMBLEU} tends to prioritize information content over graphical structure and by some measures is more consistent with human judgements.
AMR parsers rely on a wide variety of architectures and strategies, but can generally be separated into 4 types—**encoder-decoder** (Bevilacqua et al., 2021; Xu et al., 2020; Zhang et al., 2019b,c), **transition-based** (Zhou et al., 2021; Fernandez Astudillo et al., 2020; Naseem et al., 2019; Ballesteros and Al-Onaizan, 2017; Liu et al., 2018; Wang et al., 2016), **factorization-based** (Cai and Lam, 2020a; Flanigan et al., 2014; Lyu and Titov, 2018), and **composition-based** (Beschke, 2019; Beschke and Menzel, 2018; Misra and Artzi, 2016; Artzi et al., 2015; Peng et al., 2015). It is also worth discussing several characteristics of AMR that make AMR parsing particularly difficult as an NLP task. First, AMRs are unanchored to the sentence they represent, and so there is no gold mapping between tokens and AMR components as a part of the representation, necessitating (typically unsupervised) alignments for training data. Second, AMR graphs are structurally more complex than other NLP representations such as dependency trees, in that AMRs are directed acyclic graphs, allowing nodes to have multiple parents, necessitating more complex methods than those used in syntactic parsing. Third, there is in general a many-to-many relationship between AMR nodes and tokens with some node-token mappings being 1-to-1, 1-to-many, many-to-1, and many-to-many, and with some tokens with no nodes aligned to them. Fourth, under most methods of alignment, some AMR nodes are unaligned and cannot be predicted from any particular token in the input. Fifth, AMR prediction is implicitly a mixture of several complex subtasks—i.e., word sense disambiguation, named entity recognition, semantic role labelling, and coreference resolution.

**Encoder-Decoder.** This category includes approaches that rely on an encoder-decoder architecture where the decoder output is a linearized AMR or AMR graph. These approaches include sequence-to-sequence models (Xu et al., 2020; Bevilacqua et al., 2021) and sequence-to-graph models (Zhang et al., 2019b,c). Approaches based on a sequence-to-sequence architecture first linearize AMRs

---

5 This taxonomy is adapted from Oepen et al. (2019).

6 While many approaches to AMR parsing might use an encoder decoder architecture, they might not output the graph itself. For example, transition-based parsers might use an encoder-decoder to predict a sequence of actions, which can be used to produce an AMR.
for training, predict linearized AMRs using a sequence-to-sequence model such as an encoder-decoder BiLSTM with attention, and then de-linearize the predicted AMR as a post-processing step. Sequence-to-graph models attempt to directly predict a graph on the decoder side using a sequence encoder. The state-of-the-art in AMR parsing is demonstrated by Bevilacqua et al. (2021) who use a sequence-to-sequence transformer model which they train symmetrically to perform both AMR parsing and AMR-to-text generation. They also leverage transfer learning from BART (Lewis et al., 2020) which is a recent pre-trained language model for sequence-to-sequence models.

**Transition-based.** In transition-based approaches to AMR parsing, the aim is—instead of directly predicting a graph—to predict a sequence of actions where each action represents an operation on either the graph or the state of the parser. Transition-based parsers can generally incorporate more information into a single decision, including their own history of actions, but they are also more sensitive to search errors. Strategies for improving performance in transition-based parsers include refining the search space (Guo and Lu, 2018), transfer learning (Naseem et al., 2019), and adaptation of transformers (Fernandez Astudillo et al., 2020).

**Factorization-based.** Also called graph-based, factorization-based architectures aim to predict the best scoring graph while treating an AMR graph’s score as the sum of scores of subgraphs. In the simplest case, the score for an AMR would be the sum of scores of its nodes and its edges. In the more common case, it would be the sum of scores of AMR components predicted for each token and the scores for edges connecting them. Factorization-based methods rely on graph search algorithms to identify the best scoring graph while typically assuming independence of components of the graph that are factored for calculating the score. Factorization methods can be less prone to search error than transition-based or composition-based methods since they consider the entire graph at once, but cannot incorporate as much information into a single decision.

**Composition-based.** Approaches to semantic parsing that rely on a grammar or are inspired by the linguistic notion of semantically composing language units are called composition-based archi-
tectures. Composition-based architecture can use a grammar to constrain what types of operations are allowed, thus reducing the search space. The size of the search space for these approaches depends on the power of the grammar used as well as how robust it is to new data. Composition-based approaches include relying on CCG (Beschke, 2019; Beschke and Menzel, 2018; Misra and Artzi, 2016; Artzi et al., 2015), hyperedge replacement grammars (Peng et al., 2015; Chen et al., 2018), and AM algebra (Groschwitz, 2019; Groschwitz et al., 2018, 2017).

In addition to the four classes of approaches discussed above, there are a number of smaller tasks that are commonly used in various systems to improve performance:

**Transfer Learning.** Transfer learning from pre-trained language models like BERT (Devlin et al., 2019)—like in many other NLP tasks—has been shown to improve semantic parsing. For example, Naseem et al. (2019) show that BERT as an input yields better performance than part-of-speech tags, universal dependency parses, and named entity tags combined. Additionally, Xu et al. (2020) demonstrate that pre-training on syntactic and semantic auxiliary tasks improves performance. The state-of-the-art in AMR parsing is demonstrated by Bevilacqua et al. (2021) who use a sequence-to-sequence transformer model and leverage transfer learning from BART (Lewis et al., 2020) which is a pre-trained transformer-based language model for sequence-to-sequence models. BART is trained to reconstruct text that has been corrupted my random noise and has demonstrated state-of-the-art results on 3 NLI tasks: abstractive dialogue, question answering, and summarization.

**Improved Alignments.** Most AMR parsers are dependent (implicitly or explicitly) on alignments over the training data. There are no gold alignments for most of the gold AMRs commonly available (hand aligned sentences number in the hundreds and are used in evaluating alignment systems only), and so most parsers rely on the output of automatic aligners for training. Automatic alignments is discussed further in Ch. 4. Since automatic alignments do not have full coverage (leaving some AMR nodes unaligned) or gold quality, AMR parsers often see significant gains in performance.
from methods that improve these alignments (Naseem et al., 2019; Liu et al., 2018; Chen and Palmer, 2017).

**Latent Alignment.** Some parsers learn latent alignments rather than relying on explicitly aligned inputs. One advantage of latent alignments is the option of soft alignment where mappings between tokens and nodes are probabilistic rather than discrete (Lyu and Titov, 2018). Lyu et al. (2020) additionally show that soft segmentation of the AMR graph can improve performance.

**Linearization.** Sequence-to-sequence models of AMR parsing require a step of linearizing each AMR, by converting it to a sequence of tokens. While various strategies of linearization are possible, van Noord and Bos (2017) show that removing node ids and reordering branches of the AMR to match the sentence order improves training.

**Graph Recategorization and Graph Templates.** Many alignments represent tokens mapping to many nodes in systematic but individually rare patterns. This happens most often with named entities and date expressions, but it can also happen with token spans whose AMR representation includes multiple nodes and edges by convention. Many parsers take advantage of rule-based functions or templates to map these portions of the AMRs correctly (Naseem et al., 2019; Lyu and Titov, 2018; Ballesteros and Al-Onaizan, 2017; Wang and Xue, 2017). These can include templates for handling date expressions, names of countries, etc. and can be written by hand or inferred from data. Some AMR parsing approaches involve a pre-processing step, called graph recategorization, of converting AMR graphs to a simplified form where complex subgraphs are mapped to single nodes (Peng et al., 2017).

**Iterative Learning.** Cai and Lam (2020a) argue that AMR node prediction and relation prediction are two interdependent tasks each requiring knowledge from the other. To leverage that interdependence, they implement a novel architecture which iteratively predicts node then relations in a loop with each subtask (nodes or relations) relying on a hidden state representation used in prediction
of the previous subtask (relations or nodes), such that node prediction relies on knowledge about relations and vice versa and that each iteration improves performance in each subtask. By doing this iterative procedure, they show that their parser converges to a better AMR graph prediction.

**Partial Graph Embeddings.** Since transition-based and composition-based parsers both build up an AMR in a sequence of steps, it is useful to construct a new embedding at each step using the combined components as inputs (Naseem et al., 2019; Ballesteros and Al-Onaizan, 2017), thus storing an embedding representation of the partially built graph.

**Silver/Unlabeled Data.** Given the limited resources of annotated AMR data, some researchers have experimented with improving parsing with non-gold data. Xu et al. (2020) pre-train their parser on silver AMR data over a large corpus. Lee et al. (2020) train using silver paraphrases of each input sentence and silver AMRs, created using a pre-trained AMR-to-text generator and an AMR parser respectively. Konstas et al. (2017) create silver AMR data and sentences through a paired training procedure with both a parser and generator.

**Reinforcement Learning.** Naseem et al. (2019) demonstrate improved performance with a reinforcement learning approach by rewarding the model based on the SMATCH scores of predicted graphs. They found that this strategy allows their parser to better explore its prediction space and reduce the dependency on alignments.

**Handling Reentrancies.** Because reentrancies are a source of difficulty in AMR parsing, different works use a number of strategies for predicting them. Zhang et al. (2019b,c) create duplicate, indexed nodes whenever there is a reentrancy, converting the AMR graph into a tree structure. Some transition-based approaches design actions that allow caching a node until it can be assigned all of its parents (Ballesteros and Al-Onaizan, 2017; Naseem et al., 2019). Anikina et al. (2020) incorporate a coreference resolution system into AMR parsing. Additionally, Szubert et al. (2020) categories AMR reentrancies into types and discuss their complexity.
Additionally, a number of works aim to adapt AMR parsing to cross-framework or multilingual settings:

**Cross-Framework and Multilingual Parsing.** Oepen et al. (2020) present a shared task of semantic graph parsing, where participants predicted AMR, UCCA (Abend and Rappoport, 2013), EDS (Oepen and Lønning, 2006), PDT (Sgall, 1992; Hajič et al., 2020), and DRS (Basile et al., 2012; Kamp and Reyle, 1993) structures using the same model and for multiple languages. Participants included Ozaki et al. (2020), Samuel and Straka (2020), Dou et al. (2020), and Arviv et al. (2020). Ozaki et al. (2020) achieved the highest performance in both cross-framework parsing and multilingual parsing using a transformer encoder-decoder trained to predict a plain graph notation which they devised. Additional work on multilingual AMR parsing includes Damonte and Cohen (2018) who trained an AMR parser to build AMRs for Italian, Spanish, German and Chinese based on silver training data.

### 3.2.6 Generation

Text generation is an important subtask for many areas of study in NLP such as automatic dialogue, machine translation, summarization, and question answering. Text generation from an AMR input has been done using a number of approaches, most of which rely on an encoder-decoder architecture with wide variation in the types of encoders used. For a more extensive review of AMR text generation, see Manning (2021).

**Seq2seq.** Sequence-to-sequence models for AMR-to-text generation require first converting an AMR to a sequence through linearization. Given the success of seq2seq models in NLP, this approach to generation can be effective, even though structural information is lost when simplifying an AMR from a graph to a sequence. Some generation systems using this approach also benefit from transfer learning (Bevilacqua et al., 2021; Mager et al., 2020; Fan and Gardent, 2020) from pretrained embeddings or language models. Other works try to encode some of the graph information into the sequence model (Hoyle et al., 2020; Ribeiro et al., 2021).
Graph2seq. Another approach is to use an encoder-decoder architecture with an encoder designed to take graphs as inputs. A number of works propose or adapt graph encoder architectures to use for this task including graph transformers (Jin and Gildea, 2020; Bai et al., 2020; Wang et al., 2020b,a; Cai and Lam, 2020b; Zhu et al., 2019), graph convolutional networks (Zhang et al., 2020), gated graph neural networks (Beck et al., 2018), graph attention networks (Zhao et al., 2020), and graph LSTMs (Song et al., 2018). Damonte and Cohen (2019) experiment with three encoders—an LSTM, a tree LSTM, and a graph convolutional network—to test the effectiveness of the graph2seq approach for this task.

Additional Strategies. Other strategies that have been tried include machine-translation-based methods (Castro Ferreira et al., 2017; Pourdamghani et al., 2016) and hybrid rule-based methods (Manning, 2019). Jin and Gildea (2019) use a transition-based model to traverse the AMR graph while generating instead of using a simple encoder. Cao and Clark (2019) first generate syntactic structure from the AMR and then generate words, which they argue is a useful step because it adds phrase structure to a syntactically neutral AMR. Several works use particular strategies to improve the encoder-decoder model. Bai et al. (2020) and Sobrevilla Cabezudo et al. (2019) use an auxiliary task called back-translation to train the decoder to also predict the original input from a hidden state representation. They find that this constrains the model to preserve more meaning of the input in the generated text. Ribeiro et al. (2019) build a graph encoder with parallel top-down and bottom-up perspectives of the input graph structure.

Evaluation. It is common to use BLEU score (Papineni et al., 2002) between the generated text and the sentence from which an AMR was annotated when evaluating AMR-to-text generation. However, several researchers argue that BLEU score is a poor evaluation metric for this task. Manning (2019) and Manning et al. (2020) compare a number of metrics including BLEU and more semantically motivated metrics against human judgements and find that while metrics METEOR and TER are better able to fit human judgements better metrics in general are needed.
Frank (2021) advocate for decomposing a generation score into two metrics which evaluate form and meaning respectively.

3.2.7 Syntax and Abstract Meaning Representation

Syntax has been used in AMR tasks in a number of contexts.

Parsing. Composition-based based parsers can use a grammar formalism, such as CCG, to derive an AMR parse (Artzi et al., 2015; Misra and Artzi, 2016; Beschke and Menzel, 2018; Beschke, 2019). Chen (2015) converts Universal Dependency parses to AMR using a rule-based system. Xu et al. (2020) and Ge et al. (2019) both incorporate syntactic information into a sequence-to-sequence AMR parser. Ge et al. (2019) experiment with syntactic features in the input and encoding dependency structure in a self-attention head. Xu et al. (2020) pre-train an encoder-decoder on a syntactic parsing auxiliary task.


Generation. Cao and Clark (2019) use generation of syntactic structure as an intermediary step for AMR-to-text generation. They argue that this helps generation performance because it adds phrase structure to a syntactically neutral AMR.
This section presents relevant literature pertaining to CCG (Steedman, 2000b).

### 3.3.1 Related Grammar Formalisms

CCG is one of several grammar formalisms with implemented grammars and annotated corpora. It is worth comparing CCG and to several other formalisms here, to establish how it differs from other available grammars. For a thorough introduction to grammar formalisms like those discussed below, see Müller (2020).

Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994) is a highly expressive constraint-based grammar which is represented with feature-value structures. Rules in HPSG can be applied at any phrase level, making it a popular formalism for construction grammarians. Lexical Functional Grammar (LFG; Dalrymple, 2001) is a lexicalized grammar divided into c-structure, or constituent structure, for representing constituents, and f-structure, or functional structure, for representing functional relations between words or phrases. LFG places an emphasis on functional roles like subject and object as well as phrase structure. Tree Adjoining Grammar (TAG Joshi, 1987) is a transformational grammar where rules and lexical entries correspond to tree fragments, called elementary trees, of a constituency structure. In the derivation of a TAG parse, tree fragments are merged together, while also combining their semantics. A lexicalized variant called LTAG (Abeille et al., 1998) allows rules which are lexically specified. TAG is a mildly context sensitive grammar and has been shown to be weakly equivalent in expressiveness to CCG (Vijay-Shanker and Weir, 1995).

CCG stands out from these other prominent grammar formalisms for its availability of robust parsers and its transparent modelling of the syntax-semantics interface, which it implements by tightly pairing syntactic supertags and semantic forms at every stage of derivation.
3.3.2 Supertagging

Predicting token supertags for a sentence is most often treated as a sequence tagging task. Supertagging has been called “almost parsing” (Bangalore and Joshi, 1999) because so much of the knowledge relevant for deriving a CCG parse is present in the supertags. The most successful supertaggers seem to all rely on LSTM sequence models (Clark et al., 2018; Vaswani et al., 2016; Lewis et al., 2016; Xu et al., 2015), but this trend has not been tested in several years.

The state-of-the-art in CCG supertag is established by Prange et al. (2021), who use a constructive model, rather than a sequence model to build supertags out of atomic types. Their decoding is tree-structured for each supertag allowing the model to learn generalizations based on the internal structure of supertags and better handle the long tail of rare supertags that occur in the data.

Because CCG supertags are themselves composed of parts, there is research on predicting each supertag as a structured output. Bhargava and Penn (2020) and Prange et al. (2021) both experiment with structured prediction for supertags, with Bhargava and Penn treating supertags as a sequence and with Prange et al. treating them as tree structured with slashes (/) and (\) corresponding to binary branches. Other notable experiments include Tian et al. (2020) which performs supertagging using a graph convolutional network with attention over graphs induced on the tokens based loosely on pointwise mutual information between tokens, and Kummerfeld et al. (2010) which prunes supertag predictions by jointly training a supertagger to predict the same outputs that are selected by a dynamic parser in order to speed up parsing. Clark et al. (2018) use a BiLSTM architecture with “cross-view training” involving a combination of supervised learning and masked auxiliary tasks where the tagger is trained to make the same prediction given a masked input as it would predict with the full input. Clark et al. find that this multitask procedure improves performance in several sequence tagging tasks.

---

7 at the time of writing, August 19, 2021
3.3.3 PARSING

The parsing subtask has been shown to be much easier given supertags (Bangalore and Joshi, 1999). The process of deriving a full CCG parse given the supertags may rely on transition-based parsing (Misra and Artzi, 2016; Xu, 2016; Zhang and Clark, 2011), various search algorithms such as A* (Lewis and Steedman, 2014), or may be deterministic (Lewis et al., 2016). Additionally, there has been work on incremental CCG parsing where parsing is done in an efficient way from left to right. Stanojević and Steedman (2019) use tree rotation to parse incrementally, essentially altering right-branching trees to be left branching whenever one is encountered.

3.3.4 SEMANTICS

CCG traditionally uses predicate and lambda calculus in its semantic derivations, but CCG is flexible enough that this can be replaced by most other semantic representations. CCG has been used to derive Discourse Representation Structure (Bos, 2015), hybrid logic dependency semantics (Baldridge and Kruijff, 2002), and information structure (Steedman, 2000a). A number of works use AMR (or AMR-like) semantics in CCG, with some converting AMR to a lambda calculus (Artzi et al., 2015; Misra and Artzi, 2016) and others using composition of the graphs themselves (Beschke and Menzel, 2018; Beschke, 2019).
3.4 COMPOSITION AND GRAPH GRAMMARS

A graph grammar (Rozenberg, 1997) is a formal grammar defined in terms of graphs, rather than strings or trees. Graph grammars occupy an area of mathematics related to formal languages, and have been applied in research related to semantic graph representations as a framework for composing semantic graphs. Readers can think of graph grammars as generalizations of formal grammars in the Chomsky hierarchy (Chomsky, 1956) to allow graph languages instead of string languages.

3.4.1 HYPEREDGE REPLACEMENT GRAMMARS

Hyperedge replacement grammars (HRGs) (Drewes et al., 1997) are closely related to context free grammars in that they generalize context free grammars to languages over graphs instead of strings. A hyperedge replacement grammar can be used to generate a graph in a top-down way by replacing non-terminal subgraphs called “hyperedges” with new subgraphs based on a fixed set of rules until the graph is complete. A hyperedge connecting \( n \) nodes is said to be of rank \( n \). Each rule in the grammar merges the \( n \) nodes in a hyperedge with \( n \) nodes of a replacement subgraph. Hyperedge replacement grammars are quite general, and many other graph grammars used in practice can be defined as special cases of a hyperedge replacement grammar. Hyperedge replacement grammars have been applied in the AMR parsing literature (Jones et al., 2012; Chiang et al., 2013; Peng et al., 2015; Peng and Gildea, 2016; Björklund et al., 2016; Groschwitz et al., 2018). Readers may assume that the graph formalism described in Ch. 6 is a simplified hyperedge replacement grammar which only allows hyperedges of rank 1 and rank 2 for relation-wise operations.

---

8A hyperedge is a connection which links \( n \) nodes where \( n \) may be \( \geq 1 \), and may be thought of as an edge connecting \( n \) nodes instead of 2.
3.4.2 S-GRAPH ALGEBRAS

Equivalently, a graph grammar can be formulated in terms of s-graphs\(^9\) (Courcelle, 1993). Algebras over s-graphs are equivalent in expressiveness to HRGs (Courcelle and Engelfriet, 2012, Prop. 4.27), and so they provide an additional point of view with which we can view composition of graphs. S-graphs (or source-graphs) are graphs with a function that assigns public names to each nodes, and are equipped with 3 operations:

1. \textit{merge} which combines two s-graphs and merges any nodes with the same name,
2. \textit{rename} which assigns a new name to a node or nodes, and
3. \textit{forget} which reduces the naming function by removing a node from its domain, preventing that node from being merged again.

S-graphs thus provide a perspective of graph composition which is defined as an algebra rather than a generative process. A merge operation of two s-graphs with \(n\) matching nodes can be thought of as being like a hyperedge replacement of rank \(n\). S-graphs have been applied as a framework for AMR parsing (Koller, 2015; Beschke and Menzel, 2018; Groschwitz et al., 2015).

3.4.3 REGULAR GRAPH GRAMMARS

Regular graph grammars (Courcelle, 1991) are a subset of hyperedge replacement grammars which may be thought of as the natural generalization of regular languages to graphs instead of strings. See Gilroy et al. (2017b) for a mathematical introduction. Gilroy et al. (2017a) argue that a proper representation of language which uses semantic graphs should have an expressiveness between that of regular graph grammars and hyperedge replacement grammars.

3.4.4 AS-GRAPH ALGEBRAS

AS-graphs are a variation of s-graphs developed by Groschwitz et al. (2017) for use in linguistic composition of semantic graphs. An as-graph (or annotated s-graph) is an s-graph with a special

\(^9\)A commonly used term for an algebra over s-graphs is an HR algebra (See Courcelle and Engelfriet, 2012), which is a term dis-preferred here in order to avoid confusion with HR grammars.
root node and a graph type to describe what kinds of as-graphs it should merge with. The premise of an as-graph is that they behave like typed lambda terms, where operators can combine two as-graphs, using one as an argument of the other and merging the argument as-graph’s root to the function as-graph’s argument position. This mindset of treating graphs as lambda terms is similar to my approach given in Ch. 6. Readers may think of subgraphs with free variables in Ch. 6 as being like s-graphs but with a different set of operations defined. Once as-graphs were defined, Groschwitz et al. (2017) went on to develop AM algebra with operators on as-graphs.

3.4.5 AM Algebra

An AM Algebra is an adaptation of s-graph grammars using as-graphs that allows two compositional operations:

1. Apply (A): Given two as-graphs $G_1$ and $G_2$, $\text{APP}_\alpha(G_1, G_2)$ renames and merges the root node of $G_2$ with a node in $G_1$ specified by the graph type $\alpha$, merges any additional nodes specified by $\alpha$, and leaves the root node of $G_1$ as the root node of the result.

2. Modify (M): Given two as-graphs $G_1$ and $G_2$, $\text{MOD}_\alpha(G_1, G_2)$ renames and merges the root node of $G_2$ with a node in $G_1$ specified by the graph type $\alpha$, merges any additional nodes specified by $\alpha$, and assigns the root node of $G_2$ as the root node of the result.

Apply and modify operations provide a way to implement application and modification which are ubiquitous in linguistics. Apply and modify differ only in terms of what is treated as the new semantic head of the resulting expression. AM algebra can also be thought of as a way to convert graph composition to a dependency parsing problem where the derivation tree of apply and modify operations is the structure being parsed instead of the semantic graph itself. AM algebra has been applied to AMR parsing in several works (Lindemann et al., 2020; Groschwitz, 2019; Groschwitz et al., 2018).
Beschke and Menzel (2018) develop another variant of s-graphs that we will call s-graphs with placeholders. S-graphs with placeholders allow for the treatment of semantic subgraphs as lambda terms with an ordered list of bound variables each of which corresponds to a node in the subgraph. Thus when an operator is applied, an s-graph may act as a function taking an argument: its outermost place holder is replaced with the root of the s-graph which is acting as an argument. Beschke (2019) and Beschke and Menzel (2018) use this formulation of s-graphs as a basis for parsing AMRs using CCG. Their approach is the most similar to my work described in Ch. 6 out of the graph formalisms described in this section. The research presented in Ch. 6 extends their approach by allowing for a number of other CCG combinators, rank-2 (relation-wise) graph operations for control and other phenomena, and to better and more explicitly handle other linguistic structures.
4.1 INTRODUCTION

Research with the Abstract Meaning Representation must contend with the rarity of gold-standard alignments between words and semantic units in the English data. A variety of rule-based and statistical algorithms have sought to fill this void, with improvements in alignment accuracy often translating into improvements in AMR parsing accuracy (Pourdamghani et al., 2014; Naseem et al., 2019; Liu et al., 2018). Yet current alignment algorithms still suffer from limited coverage and less-than-ideal accuracy, constraining the design and accuracy of parsing algorithms. Moreover, AMR-to-text generation research and applications using AMR stand to benefit from from accurate, human-interpretable alignments.

We present Linguistically Enriched AMR (LEAMR, pronounced lemur) alignment, which achieves full graph coverage via four distinct types of aligned structures: subgraphs, relations, reentrancies, and duplicate subgraphs arising from ellipsis. This formulation lends itself to unsupervised learning of alignment models. Advantages of our algorithm and released alignments include: (1) much improved coverage over previous datasets, (2) increased variety of the substructures aligned, including alignments for all relations, and alignments for diagnosing reentrancies, (3) alignments are made between spans and connected substructures of an AMR, (4) broader identification of spans including named entities and verbal and prepositional multiword expressions.

Contributions are as follows:

- A novel all-inclusive formulation of AMR alignment in terms of mappings between spans and connected subgraphs, including spans aligned to multiple subgraphs; mappings between spans

---

1This chapter is based on research published at ACL 2021 as Blodgett and Schneider (2021).
and inter-subgraph edges; and characterization of reentrancies. Together these alignments cover all of the AMR graph structure (§4.3).

- A corpus with automatic alignments for LDC2020 and *Little Prince* data as well as a few hundred manually annotated sentences for tuning and evaluation (§4.4).

We release this dataset of alignments for over 60,000 English sentences along with our aligner code to facilitate more accurate models and greater interpretability in future AMR research. Ch. 5 will present a probabilistic, structure-aware algorithm for deriving these alignments.

4.2 RELATED WORK

The main difficulty presented by AMR alignment is that it is a many-to-many mapping problem, with gold alignments often mapping multiple tokens to multiple nodes while preserving AMR structure. Previous systems use various strategies for aligning. They also have differing approaches to what types of substructures of AMR are aligned—whether they are nodes, subgraphs, or relations—and what they are aligned to—whether individual tokens, token spans, or syntactic parses. Two main alignment strategies remain dominant, though they may be combined or extended in various ways: rule-based strategies as in Flanigan et al. (2014), Flanigan et al. (2016), Liu et al. (2018), and Szubert et al. (2018) and statistical strategies using Expectation-Maximization as in Pourdamghani et al. (2014).

4.2.1 JAMR

The JAMR system (Flanigan et al., 2014, 2016) aligns token spans to subgraphs using iterative application of an ordered list of 14 rules which include exact and fuzzy matching. JAMR alignments form a connected subgraph of the AMR by the nature of the rules being applied. A disadvantage of JAMR is that it lacks a method for resolving ambiguities, such as repeated tokens, or of learning novel alignment patterns.
4.2.2 ISI

The ISI system (Pourdamghani et al., 2014) produces alignments between tokens and nodes and between tokens and relations via an Expectation-Maximization (EM) algorithm in the style of IBM Model 2 (Brown et al., 1988). First, the AMR is linearized, then EM is applied using a symmetrized scoring function of the form $P(a \mid t) + P(t \mid a)$ where $a$ is any node or edge in the linearized AMR and $t$ is any token in the sentence. ISI alignments do not form connected substructures of the AMR as a token can be aligned to multiple nodes or edges that need not be connected. ISI alignments do include more novel patterns of alignment, but also struggle with rare strings such as dates and names, where a rule-based approach is more appropriate.

4.2.3 Extensions and combinations

TAMR (Tuned Abstract Meaning Representation; Liu et al., 2018) uses the JAMR alignment rules, along with two others, to produce a set of candidate alignments for the sentence. Then, the alignments are “tuned” with a parser oracle to select the candidates that correspond to the oracle parse that is most similar to the gold AMR.

Some AMR parsers (Naseem et al., 2019; Fernandez Astudillo et al., 2020) use alignments which are a union of alignments produced by the JAMR and ISI systems. The unioned alignments achieve greater coverage, improving parser performance.

4.2.4 Syntax-based

Table 4.1: Coverage and types of previous alignment systems.

<table>
<thead>
<tr>
<th></th>
<th>nodes</th>
<th>edges</th>
<th>reentrancies</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAMR</td>
<td>91.1</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ISI</td>
<td>78.7</td>
<td>9.8</td>
<td>✗</td>
</tr>
<tr>
<td>TAMR*</td>
<td>94.9</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Scores are evaluated on ≈200 gold test sentences. *TAMR is evaluated on a subset of 91 sentences.

4.2.5 Word Embeddings

Additionally, Anchiêta and Pardo (2020) use an alignment method designed to work well in low-resource settings using pretrained word embeddings for tokens and nodes.

4.2.6 Graph Distance

Wang and Xue (2017) use an HMM-based aligner to align tokens and nodes. They include in their aligner a calculation of graph distance as a locality constraint on predicted alignments. This is similar to our use of projection distance as described in §5.3.

4.2.7 Drawbacks of Current Alignments

Alignment methods vary in terms of components of the AMR that are candidates for alignment. Most systems either align nodes (e.g., ISI) or connected subgraphs (e.g., JAMR), with incomplete coverage. Most current systems do not align relations to tokens or spans, and those that do (such as ISI) do so with low coverage and performance, though some aligners that use syntax can align in a way that captures the same information (Szubert et al., 2018). None of the current systems align reentrancies, although Szubert et al. (2020) developed a rule-based set of heuristics for identifying reentrancy types. Table 4.1 summarizes the coverage and variety of prominent alignment systems.

Coverage. No alignment dataset aligns every AMR node, meaning that for most parsers, there is no reasonable way to predict certain nodes in the gold AMRs. Some papers (Ballesteros and Al-Onaizan, 2017) address this problem by taking the union of alignment datasets to provide as
much node coverage as possible, but while still not reaching 100%. Fernandez Astudillo et al. (2020) addressed this problem by arbitrarily aligning any unaligned nodes to nearby tokens, and demonstrated an improvement in 1 point SMATCH performance.\(^2\)

**Alignment Errors.** There are two notable types of alignment errors: non-synonymous and synonymous errors. A non-synonymous error is an alignment between a token and an AMR component whose semantics is not signalled by that token (e.g., cars \(\rightarrow\) drive-01). A synonymous error happens when a token is aligned to an AMR component with the correct meaning but in the wrong position of the AMR (e.g., cars \(\rightarrow\) c1/car instead of c2/car). Conditional entropy indirectly measures non-synonymous errors, but not synonymous errors. Non-synonymous errors can in particular be a problem in that they encourage the parser during training to operate on token spans that are very far apart from each other when the correct span is much closer.

**Lack of Relation Alignments.** Most current alignment systems do not align relations, and those that do (Pourdamghani et al., 2014) do so with low coverage and performance. Certain English function words signal a semantics corresponding to AMR relations, and these include prepositions, relativizers, and subordinators—words often used to denote relationships between two semantic arguments whether a noun phrase or clause. For example, prepositions can signal :time (the meeting at 3), :location (born in New York), and :ARG0 (killed by a lion). The semantics contributed by these tokens is not in general easy to infer without relation alignments.

**Lack of Reentrancy Alignments.** AMR allows reentrancies, which are additional edges pointing to a single node such that AMR nodes can have multiple parents. Reentrancies are associated with control structures and coreference and are one the features that make AMR both more complex and more difficult to parse than Universal Dependencies and other tree-shaped data structures. Reentrancies from coreference may be a problem for parsers because they can result in an edge

\(^2\)An argument could be made that some AMR concepts should not be aligned to any token. These might include multi-sentence which is an AMR notation only and you in an English imperative sentence where no corresponding token is pronounced. Though aligning this can still result in improved parser performance.
between nodes whose alignments are far away from each other in the sentence. Conceptually, these edges follow vastly different behaviors than other AMR edges the parser trains on—they can be between any two positions in a sentence and are signalled by different linguistic structures than other semantic roles. Ideally, they would be predicted using a different strategy than other edges, but currently coreference is not treated any differently or even identified in any alignment datasets.\footnote{Zhang et al. (2019b) addresses this issue by duplicating AMR nodes with reentrancies, essentially turning AMRs into trees.}

**Lack of Discontiguous Alignments.** Some tokens are useful for disambiguating the sense of a word, but do not form a contiguous span with that word. In the phrase “make the test up”. *up* does not correspond to an AMR concept, but it does help disambiguate the correct sense of *make* as *make-up-01* instead of *make-01*. Particles, prepositions, and light verbs can be useful signals for sense disambiguation and for predicting semantic roles, but only if they are known to be associated with a particular semantic head. One approach for dealing with these cases is to treat them as discontiguous multiword expressions (e.g., “make . . . up”), which would require them to be identified with a list of spans, each of which would correspond to a sub-part of the multiword expression. Yet these sub-parts can be an arbitrary distance apart from each other, and considering combinations of spans as candidates for alignment drastically increases the complexity of the alignment task.

### 4.3 An All-Inclusive Formulation of AMR Alignment

Aligning AMRs to English sentences is a vexing problem not only because the English training data lacks gold alignments, but also because AMRs—unlike many semantic representations—are not *designed* with a derivational process of form–function sub-units in mind. Rather, each AMR graph represents the full-sentence meaning, and AMR annotation conventions can be opaque with respect to the words or surface structure of the sentence, e.g., by unifying coreferent mentions and making explicit certain elided or pragmatically inferable concepts and relations. Previous efforts toward general tools for AMR alignment have considered mapping tokens, spans, or syntactic units
Figure 4.1: AMR and alignments for the sentence “Most of the students want to visit New York when they graduate.” Alignments are differentiated by colors: blue (subgraphs), green (duplicate subgraphs), and orange (relations). Relations that also participate in reentrancy alignments are bolded.

4.3.1 Definitions

Given a tokenized sentence $w$ and its corresponding AMR graph $G$, a complete alignment assumes a segmentation of $w$ into spans $s$, each containing one or more contiguous tokens; and puts each of the nodes and edges of $G$ in correspondence with some span in $s$. A span may be aligned to one or more parts of the AMR, or else is null-aligned. Individual alignments for a sentence are grouped

---

56
into four layers: subgraph alignments, duplicate subgraph alignments, relation alignments, and reentrancy alignments. These are given for an example in figure 4.1.

All alignments are between a single span and a substructure of the AMR. A span may be aligned in multiple layers which are designed to capture different information. Within the subgraph layer, alignments are mutually exclusive with respect to both spans and AMR components. The same holds true within the relation layer. Every node will be aligned exactly once between the subgraph and duplicate subgraph layers. Every edge will be aligned exactly once between the subgraph and relation layers, and may additionally have a secondary alignment in the reentrancy layer.

4.3.2 Caveats of This Formulation

There are several limitations to this and other formulations of AMR alignments that are worth discussing. AMR annotations sometimes go beyond the compositional ‘sentence meaning’ and incorporate elements of ‘speaker meaning,’ informed by pragmatics rather than compositional semantics. As a consequence, there are some AMR concepts that arguably should not be aligned but should instead be labelled as contributed by pragmatics. For example, for vocatives, a concept say-01 appears in the AMR which does not obviously align to any pronounced part of the sentence. Our aligner is able to handle some concepts contributed by pragmatics, thanks to our duplicate subgraph layer. For other such concepts, we align them in our subgraph layer, typically to whichever token most is the most reliable signal for predicting them. Another limitation is our use of contiguous token spans. As discussed in §2.1, some multiword expressions are discontiguous such as “made . . . up” in the sentence “He made the story up.” Ideally, an alignment of this expression would align to both tokens made and up despite the tokens not forming a contiguous span. However, identifying discontiguous multiword expressions is difficult enough that we opted to align only to the first token of these expressions. This creates some issues in our gold data where we find that for some expressions, the first token is the least salient part of the expression. For example take in “Please take my advice” is a part of the multiword expression but is the least salient and least informative part. An alternative approach might be to identify discontiguous expressions as left centric or right...
centric following Wulff (2008) and aligning to the most salient part of the expression, but we consider that outside the scope of this chapter.

4.3.3 Subgraph Layer

Alignments in this layer generally reflect the lexical semantic content of words in terms of connected, directed acyclic subgraphs of the corresponding AMR. Alignments are mutually exclusive (disjoint) on both the form and meaning sides.

4.3.4 Duplicate Subgraph Layer

A span may be aligned to multiple subgraphs if one is a duplicate of the others, with a matching concept. This is often necessary when dealing with ellipsis constructions, where there is more semantic content in the AMR than is pronounced in the sentence and thus several identical parts of the AMR must be aligned to the same span. In this case, a single subgraph is chosen as the primary alignment (whichever is first based on depth-first order) and is aligned in the subgraph alignment layer, and any others are represented in the duplicates alignment layer.

- Most of the students (figure 4.1): AMR conventions treat this as a subset-superset structure with include, where the subset and superset correspond to separate nodes. Because the word student is represented in AMR like person who studies there are two 2-node subgraphs aligned to student, one with the variables p & s, and the duplicate with p2 & s2.
- Verb phrase ellipsis, as in I swim and so do you, would involve duplication of the predicate swim, with distinct :ARG0s.
- AMR annotators might express pragmatic inferences as concept duplication. For instance, the second part of She wanted to swim but I was undecided might be interpreted as ‘I was undecided about swimming’, requiring a second AMR predicate aligned to swim.

---

4Nodes aligned to a span must form a connected subgraph with two exceptions: (1) duplicate alignments are allowed and are separated into subgraph and duplicate layers; (2) a span may be aligned to two terminal nodes that have the same parent. For example, never aligns to :polarity :time ever, two nodes and two edges which share the same parent.
The difficulty that duplicate subgraphs pose for parsing and generation makes it convenient to put these alignments in a separate layer.

A similar type of analysis exists in research on Universal Dependencies where handling ellipsis is a topic of interest. Schuster et al. (2017) propose enhanced dependency parses as a way of including elided tokens in each dependency parse along with associated dependencies, and Schuster et al. (2018) demonstrate methods for parsing enhanced dependency parses that include elided tokens.

4.3.5 Relation Layer

This layer includes alignments between a span and a single relation—such as when → :time—and alignments mapping a span to its argument structure—such as give → :ARG0 :ARG1 :ARG2. All edges in an AMR that are not contained in a subgraph fit into one of these two categories.

English function words such as prepositions and subordinators typically function as connectives between two semantically related words or phrases, and can often be identified with the semantics of AMR relations. But many of these function words are highly ambiguous. Relation alignments make their contribution explicit. For example, when in figure 4.1 aligns to a :time relation.

For spans that are aligned to a subgraph, incoming or outgoing edges attached to that subgraph may also be aligned to the span in the relation layer. These can include core or non-core roles as long as they are evoked by the token span. For example, figure 4.1 contains visit → :ARG0 :ARG1.

4.3.6 Reentrancy Layer

A reentrant node is one with multiple incoming edges. In figure 4.1, for example, p appears three times: once as the ARG0 of w (the wanter), once as the ARG0 of v (the visitor), and once as the ARG0 of g (the graduate). The p node is labeled with the concept person—in the PENMAN notation used by annotators, each variable’s concept is only designated on one occurrence of the variable, the choice of occurrence being, in principle, arbitrary. These three ARG0 relations are aligned to their respective predicates in the relation layer. But there are many different causes of reentrancy, and AMR parsers
Table 4.2: Reentrancy types with examples.

<table>
<thead>
<tr>
<th>Type</th>
<th>Triggered by</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>COREF</td>
<td>a pronoun (including possessive or reflexive) (anaphora)</td>
<td><em>I love my house</em></td>
</tr>
<tr>
<td>REPETITION</td>
<td>a repeated name or non-pronominal phrase (non-anaphoric coreference)</td>
<td><em>The U.S. promotes American goods</em></td>
</tr>
<tr>
<td>COORDINATION</td>
<td>coordination of two or more phrases sharing an argument</td>
<td><em>They cheered and celebrated</em></td>
</tr>
<tr>
<td>CONTROL</td>
<td>control verbs, control nouns, or control adjectives</td>
<td><em>I was afraid to speak up</em></td>
</tr>
<tr>
<td>ADJUNCT CONTROL</td>
<td>control within an adjunct phrase</td>
<td><em>I left to buy some milk; Mary cooked while listening to music</em></td>
</tr>
<tr>
<td>UNMARKED ADJUNCT CONTROL</td>
<td>control within an adjunct phrase with only a bare verb and no subordinating conjunction</td>
<td><em>Mary did her homework listening to music</em></td>
</tr>
<tr>
<td>COMPARATIVE CONTROL</td>
<td>a comparative construction</td>
<td><em>Be as objective as possible</em></td>
</tr>
<tr>
<td>PRAGMATIC</td>
<td>Reentrancies that must be resolved using context</td>
<td><em>John met up with a friend</em></td>
</tr>
</tbody>
</table>

For each reentrant node, one of its incoming edges is labeled **PRIMARY** and the others are labeled with one of the above reentrancy types. In the examples, the word aligned to an edge labeled with the specified type is underlined, and the word aligned to the parent of that edge is bolded.

stand to benefit from additional information about the nature of each reentrant edge, such as the fact that the pronoun *they* is associated with one of the **ARG0** relations.

The reentrancy layer “explains” the cause of each reentrancy as follows: for the incoming edges of a reentrant node, one of these edges is designated as **PRIMARY**—this is usually the first mention of the entity in a local surface syntactic attachment, e.g. the argument of a control predicate like *want* doubles as an argument of an embedded clause predicate. The remaining incoming edges to a reentrant node are aligned to a **reentrancy trigger** and labeled with one of 8 **reentrancy types**: coref, repetition, coordination, control, adjunct control, unmarked adjunct control, comparative control, and pragmatic. These are illustrated in table 4.2. These types, adapted from Szubert et al.’s (2020) classification, correspond to different linguistic phenomena leading to AMR reentrancies— anaphoric and non-anaphoric coreference, coordination, control, etc. The trigger is the word that most directly signals the reentrancy phenomenon in question. For the example in figure 4.1, the control verb *want* is aligned to the embedded predicate–argument relation and typed as CONTROL, while the pronoun *they* serves as the trigger for the third instance of *p* in *when they graduate.*
4.3.7 Validation

To validate the annotation scheme we elicited two gold-standard annotations for 40 of the test sentences described in §4.4 and measured interannotator agreement.\(^5\) Interannotator exact-match F1 scores were 94.54 for subgraphs, 90.73 for relations, 76.92 for reentrancies, and 66.67 for duplicate subgraphs (details in Table 4.3).

Alignment of reentrancies and duplicate subgraphs resulted in noticeably lower interannotator agreement. Both reentrancies and duplicate subgraphs are significantly less more rare than subgraph and relation alignments, and this lack of data made it more difficult to develop concrete guidelines and identify sources of confusion. One source of disagreement was identification of spans. For example, one annotator aligned “at no time” to :polarity -, treating it as a multiword expression (which then had a duplicate subgraph associated with it), while the other annotator aligned no to :polarity - (along with its duplicate subgraph). Another source of disagreement was from identifying reentrancy types where more than one type seems to fit. For example in the sentence “The military is against things that make it run less efficiently,” there is a reentrancy indicating that military is an argument of run. One annotator aligned this to it and labelled it as COREF. The other aligned it to make and labelled it as CONTROL. The difference comes down to whether annotators consider “it run” to be a small clause or whether it is a syntactic argument of an object control verb make.

4.4 Released Data

We release a dataset\(^6\) of the four alignment layers reflecting correspondence between English text and various linguistic phenomena in gold AMR graphs—subgraphs, relations (including argument structures), reentrancies (including coreference, control, etc.), and duplicate subgraphs.

**Automatic alignments** cover the \(\approx 60,000\) sentences of the LDC2020T02 dataset (Knight et al., 2020) and \(\approx 1,500\) sentences of The Little Prince.

---

\(^5\)Both annotators are Ph.D. students with backgrounds in linguistics. One annotator aligned all development and test sentences; the other aligned a subset of 40 test sentences.

\(^6\)https://github.com/ablodge/leamr
Table 4.3: Interannotator agreement.

<table>
<thead>
<tr>
<th>IAA</th>
<th>Exact Align</th>
<th></th>
<th>Partial Align</th>
<th></th>
<th>Spans</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>Subgraphs (366)</td>
<td>94.54</td>
<td>94.54</td>
<td>94.54</td>
<td>95.56</td>
<td>95.56</td>
<td>95.56</td>
</tr>
<tr>
<td>Relations (260)</td>
<td>91.09</td>
<td>90.38</td>
<td>90.73</td>
<td>93.38</td>
<td>92.66</td>
<td>93.02</td>
</tr>
<tr>
<td>Reentrancies (65)</td>
<td>76.92</td>
<td>76.92</td>
<td>76.92</td>
<td>90.00</td>
<td>90.00</td>
<td>90.00</td>
</tr>
<tr>
<td>Duplicates (5)</td>
<td>75.00</td>
<td>60.00</td>
<td>66.67</td>
<td>79.17</td>
<td>63.33</td>
<td>70.37</td>
</tr>
</tbody>
</table>

Interannotator agreement for subgraph, relation, reentrancy, and duplicate subgraph layers of alignment scored on a sample of 40 sentences of the gold test data.

We manually created gold alignments for evaluating our automatic aligner, split into a development set (150 sentences) and a test set (200 sentences). The test sentences were annotated from scratch; the development sentences were first automatically aligned and then hand-corrected. We stress that no preprocessing apart from tokenization is required to prepare the test sentences and AMRs for human annotation. We also release our annotation guidelines as a part of our data release.

4.5 Conclusions

We demonstrate an all-inclusive formulation of AMR alignments, improving on the coverage and variety of linguistic phenomena aligned by previous systems. We additionally release a dataset of automatic alignments for over 60,000 English sentences as well as gold annotated alignments for 350 sentences. Ch. 5 will present our algorithms for automatically producing these alignments.

---

7Our test set consists of sentences from the test set of Szubert et al. (2018) but with AMRs updated to the latest release version. This test set contains a mix of English sentences drawn from the LDC data and The Little Prince—some sampled randomly, others hand-selected—as well as several sentences constructed to illustrate particular phenomena.
CHAPTER 5

PROBABILISTIC, STRUCTURE-AWARE ALGORITHMS FOR IMPROVED VARIETY, ACCURACY, AND COVERAGE OF AMR ALIGNMENTS

5.1 INTRODUCTION

As discussed in Ch. 4, current alignment algorithms still suffer from limited coverage and less-than-ideal accuracy, constraining the design and accuracy of parsing algorithms. This chapter presents automatic alignment algorithms used to produce our LEAMR alignments data. Advantages of our algorithm and released alignments include: (1) much improved coverage over previous datasets, (2) increased variety of the substructures aligned, including alignments for all relations, and alignments for diagnosing reentrancies, (3) alignments are made between spans and connected substructures of an AMR, (4) broader identification of spans including named entities and verbal and prepositional multiword expressions. This chapter presents an algorithm combining rules and EM to automatically align English sentences to AMRs without supervision (§5.3), with higher coverage and quality than existing AMR aligners (§5.5).

We release our aligner code to facilitate more accurate models and greater interpretability in future AMR research.

5.2 RELEVANT WORK

A detailed literature review of AMR alignments and aligners is presented in Ch. 4.

\footnote{This chapter is based on research published at ACL 2021 as Blodgett and Schneider (2021).}
5.3 LEAMR Aligner

We formulate statistical models for the alignment layers described above—subgraphs, duplicate subgraphs, relations, and reentrancies—and use the Expectation-Maximization (EM) algorithm to estimate probability distributions without supervision, with a decoding procedure that constrains aligned units to obey structural requirements. In line with Flanigan et al. (2014, 2016), we use rule-based preprocessing to align some substructures using string-matching, morphological features, etc.

5.3.1 LEAMR Alignment Characteristics

Before delving into the models and algorithm, we motivate two important characteristics:

**Structure-Preserving.** Constraints on legal candidates during alignment ensure that at any point only connected substructures may be aligned to a span. Thus, while our aligner is probabilistic like the ISI aligner, it has the advantage of preserving the AMR graph structure.

**Projection Distance.** The scores calculated for an alignment take into account a distance metric designed to encourage locality—tokens that are close together in a sentence are aligned to substructures that are close together in the AMR graph. We define the projection distance $\text{dist}(n_1, n_2)$ between two neighboring nodes $n_1$ and $n_2$ to be the signed distance in the corresponding sentence between the span aligned to $n_1$ and the span aligned to $n_2$. This motivates the model to prefer alignments whose spans are close together when aligning nodes which are close together—particularly useful when a word occurs twice with identical subgraphs. Thus, our aligner relies on more information from the AMR graph structure than other aligners (note that the ISI system linearizes the graph). Further details are given in §5.3.3.
5.3.2 Aligner Overview

Algorithm 1 illustrates our base algorithm in pseudocode. The likelihood for a sentence can be expressed as a sum of per-span alignment scores: we write the score of a full set of a sentence’s subgraph alignments $\mathcal{A}$ as

$$
Score(\mathcal{A} \mid \mathcal{G}, \mathbf{w}) = \prod_{i=1}^{N} score(\langle g_i, s_i \rangle \mid \mathcal{G}, \mathbf{w})
$$

(5.1)

where $s$ are $N$ aligned spans in the sentence $\mathbf{w}$, and $g$ are sets of subgraphs of the AMR graph $\mathcal{G}$ aligned to each span. For relations model and the reentrancies model, each $g_i$ consists of relations rather than subgraphs. Henceforth we assume all alignment scores are conditioned on the sentence and graph and omit $\mathbf{w}$ and $\mathcal{G}$ for brevity. The $score(\cdot)$ component of eq. (5.1) is calculated differently for each of the three models detailed below.

**Algorithm 1** Procedure for greedily aligning all nodes to spans using a scoring function that decomposes over (span, subgraph) pairs. (Scores are expressed in real space but the implementation is in log space.)

```
1: function ALIGNSUBGRAPHS(spans, amr)
2:     alignments ← dict()  ▶ map from span to an ordered list of aligned subgraphs
3:     unaligned_nodes ← get_unaligned_nodes(amr, alignments)
4:     while |unaligned_nodes| > 0 do
5:         ∆scores ← []
6:         candidate_s_g_pairs ← []
7:         for n ∈ unaligned_nodes do
8:             candidate_spans ← get_legal_alignments(n, alignments)
9:             for span, i_subgraph ∈ candidate_spans do ▶ either there is an edge between n and the indicated subgraph already aligned to span, or i_subgraph would be a new subgraph consisting of n
10:                current_aligned_nodes ← alignments[span][i_subgraph] ▶ ⊖ if this would be a new subgraph
11:                new_aligned_nodes ← current_aligned_nodes ∪ {n}
12:                ∆score ← get_score(span, new_aligned_nodes, alignments)
13:                − get_score(span, current_aligned_nodes, alignments) ▶ change from adding $n$
14:                ∆scores.add(∆score)
15:                candidate_s_g_pairs.add((span, new_aligned_nodes, i_subgraph))
16:                span*, subgraph*, i_subgraph* ← candidate_s_g_pairs[argmax(∆scores)] ▶ update having the best impact on score (equivalently, maximizing sum of scores across individual aligned spans)
17:     alignments[span*][i_subgraph*] ← subgraph*
18:     unaligned_nodes ← get_unaligned_nodes(amr, alignments)
19:     return alignments
```
Alignment Pipeline. Alignment proceeds in the following phases, with each phase depending on the output of the previous phase:

1. Preprocessing: Using external tools we extract lemmas, parts of speech, and coreference.
2. Span Segmentation: Tokens are grouped into spans using a rule-based procedure (§B.1).
3. Align Subgraphs & Duplicate Subgraphs: We greedily identify subgraph and duplicate subgraph alignments in the same alignment phase (§5.3.3).
4. Align Relations: Relations not belonging to a subgraph are greedily aligned in this phase, using POS criteria to identify legal candidates (§5.3.4).
5. Align Reentrancies: Reentrancies are aligned in this phase, using POS and coreference in criteria for identifying legal candidates (§5.3.5).

The three main alignment phases use different models with different parameters; they also have their own preprocessing rules used to identify some alignments heuristically (§§B.2–B.4). In training, parameters for each phase are iteratively learned and used to align the entire training set by running EM to convergence before moving on to the next phase. At test time, the pipeline can be run sentence-by-sentence.

Decoding. The three main alignment phases all use essentially the same greedy, substructure-aware search procedure. This searches over node–span candidate pairs based on the scoring function modeling the compatibility between a subgraph (or relation) $g$ and span $s$, which we denote $score((g,s))$. For each unaligned node (or edge), we identify a set of legal candidate alignments using phase-specific criteria. The incremental score improvement of adding each candidate—either extending a subgraph/set of relations already aligned to the span, or adding a completely new alignment—is calculated as $\Delta score = score((g_0 \cup \{n\},s)) - score((g_0,s))$, where $g_0$ is the current aligned subgraph, $s$ is the span, and $n$ is an AMR component being considered. Of the candidates for all unaligned nodes, the node–span pair giving the best score improvement is then greedily selected to add to the alignment. This is repeated until all nodes have been aligned (even if the last

---

279% of nodes and 89% of edges are aligned by rules. We believe this is why in practice, EM performs well without random restarts.
ones decrease the score). The procedure is detailed in algorithm 1 for subgraphs; the relations phase and the reentrancies phase use different candidates (respectively: unaligned edges; reentrant edges), different criteria for legal candidates, and different scoring functions.

**Expectation-Maximization Algorithm.** An expectation-maximization algorithm involves two steps, repeated iteratively. The *expectation* step involves computing the log-likelihood function (and the most likely latent variables based on this function) for a given set of parameters. The *maximization* step involves identifying a new set of parameters that maximize the log-likelihood given the latent variables predicted in the previous step. The expectation-maximization algorithm for LEAMR is distinguished by three features. First, the likelihood function is defined by a *score* which, generally, is a product of alignment and distance probability terms, such as \( P_{\text{align}}(g \mid s; \theta_1) \) (a categorical distribution) and \( P_{\text{dist}}(d_i; \theta_2) \) (a Skellam distribution).\(^3\) Each probability has an associated maximum likelihood estimator based on its distribution type. Second, we constrain our latent variables to only allow certain values for consideration, and other values are ignored in the expectation step. Lastly, instead of random initialization, we use a smart initialization based on our aligner’s rules. This allows us to incorporate the advantages of both rule-based and EM-based alignment.

### 5.3.3 Aligning Subgraphs

The score assigned to an alignment between a span and subgraph is calculated as

\[
\text{score}((g, s)) = P_{\text{align}}(g \mid s; \theta_1) \cdot \prod_{d_i \in D} P_{\text{dist}}(d_i; \theta_2)^{k_i} \cdot IB(g, s)
\]  

(5.2)

where \( g \) is a subgraph, \( s \) is a span, \( d_i \) is the projection distance of \( g \) with its \( i \)th neighboring node, and \( \theta_1 \) and \( \theta_2 \) are model parameters which are updated after each iteration. The subgraph \( g \) is represented in the model as a bag of concept labels and (parent concept, relation, child concept) triples.

---

\(^3\)As explained below, we also experimented with a regularization term \( IB(g, s) \) with underwhelming results.
The distributions $P_{\text{align}}$ and $P_{\text{dist}}$ are inspired by IBM Model 2 (Brown et al., 1988), and can be thought of as graph-theoretic extensions of translation (align) and alignment (dist) probabilities. IB stands for inductive bias, explained below.

**Legal Candidates.** For each unaligned node $n$, the model calculates a score for spans of three possible categories: 1) unaligned spans; 2) spans aligned to a neighboring node (in this case, the aligner considers adding $n$ to an existing subgraph if the resulting subgraph would be connected); 3) spans aligned to a node with the same concept as $n$ (this allows the aligner to identify duplicate subgraphs—candidates in this category receive a score penalty because duplicates are quite rare, so they are generally the option of last resort).

Limiting the candidate spans in this way ensures only connected, plausible substructures of the AMR are aligned. To form a multinode subgraph alignment $t_1 \rightarrow n_1 : \text{rel} n_2$, the aligner could first align $n_1$ to an unaligned span $t_1$, then add $n_2$, which is a legal candidate because $t_1$ is aligned to a neighboring node of $n_2$ (ensuring a connected subgraph).

**Distance.** We model the probability of the projection distance $P_{\text{dist}}(d; \theta_2)$ using a Skellam distribution, which is the difference of two Poisson distributed random variables $D = N_1 - N_2$ and can be positive or negative valued. Parameters are updated based on alignments in the previous iteration. For each aligned neighbor $n_i$ of a subgraph $g$, we calculate $P_{\text{dist}}(dist(g,n_i); \theta_2)$ and take the geometric mean of probabilities as $P_{\text{dist}}$.

**Null Alignment.** The aligner models the possibility of a span being unaligned using a fixed heuristic:

$$P_{\text{align}}(\emptyset | s) = \max\{\text{rank}(s)^{-\frac{1}{2}}, 0.01\} \quad (5.3)$$

where \text{rank} assigns 1 to the most frequent word, 2 to the 2nd most frequent, etc. Thus, the model expects that very common words are more likely to be null-aligned and rare words should almost
always be aligned. 4 Note that the idea that common words should be unaligned is a theoretical assumption that affects the types of alignments produced by our system. An alternative set of assumptions that we considered during our research would be that common function words (e.g. infinitive to, complementizer that, prepositions for core roles, etc.) should align to semantic roles like :ARG1, :ARG2, etc. We opted not to pursue that approach for design simplicity, and we instead prefer to make function words null aligned.

**Factorized Backoff.** So that the aligner generalizes to unseen subgraph–span pairs, where \( P_{\text{align}}(g \mid s) = 0 \), we use a backoff factorization into components of the subgraph. In particular, the factors are empirical probabilities of (i) an AMR concept given a span string in the sentence, and (ii) a relation and child node concept given the parent node concept and span string. These co-occurrence probabilities \( \hat{p} \) are estimated directly from the training sentence/AMR pairs (irrespective of latent alignments). The product is scaled by a factor \( \lambda \). E.g., for a subgraph \( n_1 : \text{rel1} n_2 : \text{rel2} n_3 \), where \( c_n \) is the concept of node \( n \), we have

\[
P_{\text{factorized}}(g \mid s) = \lambda \cdot \hat{p}(c_{n1} \mid s) \cdot \hat{p}(\text{:rel1} c_{n2} \mid c_{n1}, s) \cdot \hat{p}(\text{:rel2} c_{n3} \mid c_{n1}, s)
\] (5.4)

**Inductive Bias.** Lastly, to encourage good initialization, the score function includes an inductive bias which does not depend on EM-trained parameters. This inductive bias is based on the empirical probability of a node occurring in the same AMR with a span in the training data. We calculate inductive bias as an average of exponentiated PMIs \( \frac{1}{N} \sum_i \exp(\text{PMI}(n_i, s)) \), where \( N \) is the number of nodes in \( g \), \( n_i \) is the \( i \)th node contained in the subgraph, and \( \text{PMI} \) is the PMI of \( n_i \) and \( s \).

**Aligning Duplicate Subgraphs.** On rare occasion a span should be aligned to multiple subgraphs (§4.3.4). To encourage the model to align a different span where possible, there is a constant penalty \( \lambda_{\text{dup}} \) for each additional subgraph aligned to a span beyond the first. Thus the score for a span and

---

4We allow several exceptions. For punctuation, words in parentheses, and spans that are coreferent to another span, the probability is 0.5. For repeated spans, the probability is 0.1.
its subgraphs is computed as:

\[
score((g, s)) = \lambda_{\text{dup}} \prod_{g \in g} score((g, s)) \tag{5.5}
\]

### 5.3.4 Aligning Relations

For a given relation alignment between a span and a collection of edges, we calculate a score as follows:

\[
score((a, s)) = P_{\text{align}}(a \mid s; \theta_3) \cdot \prod_{d_i \in D_1} P_{\text{dist}}(d_i; \theta_4)^{|D_1|} \cdot \prod_{d_j \in D_2} P_{\text{dist}}(d_j; \theta_5)^{|D_2|} \tag{5.6}
\]

where \(a\) is the argument structure (the collection of aligned edges), \(s\) is a span, \(D_1\) is the projection distances of each edge and its parent, and \(D_2\) is the projection distances of each edge and its child. The collection of edges \(a\) is given a normalized label which represents the relations contained in the alignment (distinguishing incoming versus outgoing relations, and normalizing inverse edges).

#### Legal Candidates.

There are two kinds of candidate spans for relation alignment. First, previously unaligned spans\(^5\) (with no relation or subgraph alignments), e.g. prepositions and subordinating conjunctions such as \(\text{in} \to \text{location}\) or \(\text{when} \to \text{time}\). Second, any spans aligned to the relation’s parent or child in the subgraph layer: this facilitates alignment of argument structures such as \(\text{give} \to \text{ARG0 :ARG1 :ARG2}\). Additionally, we constrain certain types of edges to only align with the parent and others to only align with the child.

#### Distance.

For relations there are potentially two distances of interest—the projected distance of the relation from its parent and the projected distance of the relation from its child. We model these separately as \textit{parent distance} and \textit{child distance} with distinct parameters. To see why this is useful, consider the sentence “Should we meet at the restaurant or at the office?”, where each \textit{at} token

\(^5\)We constrain these to particular parts of speech: prepositions (IN), infinitival to (TO), possessives (POS), and possessive pronouns (PRP$). Additionally, only spans that are between the spans aligned to the parent and any descendent of child nodes of the relation (and are not between the child’s aligned span and any of its descendants’ spans) are allowed. This works well in practice for English.
should be aligned to a :location edge. In English, prepositions like at precede an object and follow a governor. Thus parent distance tends to be to the left (negative valued) while child distance tends to be to the right (positive valued).

5.3.5 Aligning Reentrancies

The probability of a reentrancy alignment is similar to eq. (5.6), but with an extra variable for the reentrancy type:

\[
\text{score}(\langle r, s, \text{type} \rangle) = P_{\text{align}}(r, \text{type} \mid s; \theta_6) \cdot P_{\text{dist}}(d_1; \theta_7) \cdot P_{\text{dist}}(d_2; \theta_8)
\]  

(5.7)

where \( r \) is the role label of the reentrant edge.

Legal Candidates. There are 8 reentrancy types (§4.3.6). For each type, a rule-based test determines if a span and edge are permitted to be aligned. The 8 tests use part of speech, the structure of the AMR, and subgraph and relation alignments. A span may be aligned (rarely) to multiple reentrancies, but these alignments are scored separately.

5.4 Experimental Setup

Sentences are preprocessed with the Stanza library (Qi et al., 2020) to obtain lemmas, part-of-speech tags, and named entities. We identify token spans using a combination of named entities and a fixed list of multiword expressions (details are given in §B.1). Coreference information, which is used to identify legal candidates in the reentrancy alignment phase, is obtained using NeuralCoref.\(^6\)

Lemmas are used in each alignment phase to normalize representation of spans, while parts of speech and coreference are used to restrict legal candidates in the relation and reentrancy alignment phases. We tune hyperparameters, including penalties for duplicate alignments and our factorized backoff probability, on the development set.

\(^6\)https://github.com/huggingface/neuralcoref
Table 5.1: Main results on the test set.

<table>
<thead>
<tr>
<th></th>
<th>Exact Align</th>
<th></th>
<th>Partial Align</th>
<th></th>
<th>Spans</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td><strong>Subgraph Alignments</strong> ((N = 1707))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our system</td>
<td>93.91</td>
<td>94.02</td>
<td>93.97</td>
<td>95.69</td>
<td>95.81</td>
<td>95.75</td>
</tr>
<tr>
<td>JAMR</td>
<td>87.21</td>
<td>83.06</td>
<td>85.09</td>
<td>90.29</td>
<td>85.99</td>
<td>88.09</td>
</tr>
<tr>
<td>ISI</td>
<td>71.56</td>
<td>68.24</td>
<td>69.86</td>
<td>78.03</td>
<td>74.54</td>
<td>76.24</td>
</tr>
<tr>
<td>TAMR (91 sentences)</td>
<td>85.68</td>
<td>83.38</td>
<td>84.51</td>
<td>88.62</td>
<td>86.24</td>
<td>87.41</td>
</tr>
<tr>
<td><strong>Relation Alignments</strong> ((N = 1263))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our system</td>
<td>85.67</td>
<td>85.37</td>
<td>85.52</td>
<td>88.74</td>
<td>88.44</td>
<td>88.59</td>
</tr>
<tr>
<td>ISI</td>
<td>59.28</td>
<td>8.51</td>
<td>14.89</td>
<td>66.32</td>
<td>9.52</td>
<td>16.65</td>
</tr>
<tr>
<td><strong>Reentrancy Alignments</strong> ((N = 293))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours (labeled)</td>
<td>55.75</td>
<td>54.61</td>
<td>55.17</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours (unlabeled)</td>
<td>62.72</td>
<td>61.43</td>
<td>62.07</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Duplicate Subgraph Alignments</strong> ((N = 17))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our system</td>
<td>66.67</td>
<td>58.82</td>
<td>62.50</td>
<td>70.00</td>
<td>61.76</td>
<td>65.62</td>
</tr>
</tbody>
</table>

\(N\) represents the denominator of exact alignment recall. There are 2860 gold spans in total, 41% of which are null-aligned and 0.6% of which are aligned to multiple subgraphs. 95% of the spans consist of a single token, and 49% of spans are aligned to a single subgraph consisting of a single node.

5.5 Results

Table 5.1 describes our main results on the 200-sentence test set (§4.4), reporting exact-match and partial-match alignment scores as well as span identification F1 and coverage. The partial alignment evaluation metric is designed to be more forgiving of arbitrary or slight differences between alignment systems. We argue that this metric is more comparable across alignment systems. It assigns partial credit equal to the product of Jaccard indices \(\frac{|N_1 \cap N_2|}{|N_1 \cup N_2|} \times \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}\) for nodes (or edges) and tokens respectively. This partial credit is calculated for each gold alignment and the closest matching predicted alignment with nodes (or edges) \(N_1\) and \(N_2\) and tokens \(T_1\) and \(T_2\). Coverage is the percentage of relevant AMR components that are aligned.

Our aligner shows improvements over previous aligners in terms of coverage and accuracy even when using a partial credit metric for evaluation. We demonstrate greater coverage, including coverage of phenomena not aligned by previous systems.

\(^7\)A previous draft of this work reported lower scores on relations before a constraint was added to improve the legal candidates for relation alignment.
Table 5.2: Detailed results for relation alignments and reentrancy alignments.

<table>
<thead>
<tr>
<th>Relation Alignments Breakdown</th>
<th>Exact Align</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system: all (1163)</td>
<td>85.67</td>
<td>85.37</td>
<td>85.52</td>
</tr>
<tr>
<td>... single relations (121)</td>
<td>53.49</td>
<td>56.56</td>
<td>54.98</td>
</tr>
<tr>
<td>... argument structures (1042)</td>
<td>89.67</td>
<td>88.73</td>
<td>89.20</td>
</tr>
<tr>
<td>ISI: all (1163)</td>
<td>59.28</td>
<td>8.51</td>
<td>14.89</td>
</tr>
<tr>
<td>... single relations (121)</td>
<td>82.89</td>
<td>52.07</td>
<td>63.96</td>
</tr>
<tr>
<td>... argument structures (1042)</td>
<td>39.56</td>
<td>3.45</td>
<td>6.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reentrancy Alignments Breakdown</th>
<th>Exact Align</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system: all (293)</td>
<td>62.37</td>
<td>61.09</td>
<td>61.72</td>
</tr>
<tr>
<td>... primary (128)</td>
<td>79.37</td>
<td>78.12</td>
<td>78.74</td>
</tr>
<tr>
<td>... coref (41)</td>
<td>57.14</td>
<td>58.54</td>
<td>57.83</td>
</tr>
<tr>
<td>... control (36)</td>
<td>73.08</td>
<td>52.78</td>
<td>61.29</td>
</tr>
<tr>
<td>... coordination (29)</td>
<td>57.14</td>
<td>58.54</td>
<td>57.83</td>
</tr>
<tr>
<td>... pragmatic (25)</td>
<td>20.93</td>
<td>36.00</td>
<td>26.47</td>
</tr>
<tr>
<td>... adjunct control (15)</td>
<td>100.00</td>
<td>6.67</td>
<td>12.50</td>
</tr>
<tr>
<td>... repetition (13)</td>
<td>60.00</td>
<td>46.15</td>
<td>52.17</td>
</tr>
<tr>
<td>... comparative control (5)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>... unmarked adjunct control (1)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 5.2 shows detailed results for relation subtypes and reentrancy subtypes. Here, we see room for improvement. In particular, ISI outperforms our system at aligning single relations. Our reentrancy aligner lacks a baseline to compare to, but the breakdown of results by type suggest there are several categories of reentrancies where scores could be improved.

5.5.1 Qualitative Analysis

A number of errors from our subgraph aligner resulted from unseen multiword expressions in our test data that our span preprocessing failed to recognize and our aligner failed to align. For example, the expression “on the one hand” appears in test and should be aligned to contrast-01. The JAMR aligner suffers without a locality bias; we notice several cases where it misaligns words that are repeated in the sentence. The ISI aligner generally does not align very frequent nodes such as person, thing, country, or name, resulting in generally lower coverage. It also frequently aligns disconnected nodes with the same concept to one token instead of separate tokens. While our
Table 5.3: Subgraph and relation ablation results.

<table>
<thead>
<tr>
<th>Ablations</th>
<th>Exact Align</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>Subgraphs</td>
<td>93.91</td>
<td>94.02</td>
<td>93.97</td>
</tr>
<tr>
<td>Subgraphs (−distance)</td>
<td>92.69</td>
<td>92.85</td>
<td>92.77</td>
</tr>
<tr>
<td>Subgraphs (−inductive bias)</td>
<td>93.88</td>
<td>93.44</td>
<td>93.66</td>
</tr>
<tr>
<td>Relations</td>
<td>85.67</td>
<td>85.37</td>
<td>85.52</td>
</tr>
<tr>
<td>Relations (−distance)</td>
<td>85.14</td>
<td>84.77</td>
<td>84.95</td>
</tr>
<tr>
<td>Relations (gold subgraphs)</td>
<td>91.21</td>
<td>90.59</td>
<td>90.90</td>
</tr>
</tbody>
</table>

This table shows results of when the aligner is trained without projection distance probabilities (−distance) and without the subgraph inductive bias (−inductive bias), as well as a relation aligner with access to gold (instead of trained) subgraphs.

relation aligner yields significantly higher coverage, we do observe that the model is overeager to align relations to extremely frequent prepositions (such as to and of), resulting in lower precision of single relations in particular.

5.5.2 ABLATIONS

Table 5.3 shows that projection distance is valuable, adding 1.20 points (exact align F1) for subgraph alignment and 0.57 points for relation alignment. Despite showing anecdotal benefits in early experiments, the inductive bias does not aid the model in a statistically significant way. Using gold subgraphs for relation alignment produces an improvement of over 5 points, indicating the scope of error propagation for the relation aligner.

5.6 CONCLUSIONS

We demonstrate structure-aware AMR aligners that combine the best parts of rule-based and statistical methods for AMR alignment. We improve on previous systems in terms of accuracy and particularly in terms of alignment coverage and variety of AMR components to be aligned.
6.1 Introduction

At the heart of semantic parsing are two goals: the disambiguation of linguistic forms that can have multiple meanings, and the normalization of morphological and syntactic variation of linguistic forms with the same semantics. Among many techniques for semantic parsing, one profitable direction exploits computational linguistic grammar formalisms that make explicit the correspondence between the linguistic form of a sentence and the semantics (e.g., broad-coverage logical forms, or database queries in a domain-specific query language). In particular, English semantic parsers using Combinatory Categorial Grammar (CCG; Steedman, 2000b) have been quite successful thanks to the CCGBank resource (Hockenmaier and Steedman, 2007; Honnibal et al., 2010) and the broad-coverage statistical parsing models trained on it (e.g., Clark and Curran, 2004; Lewis et al., 2016; Clark et al., 2018).

The CCG formalism assumes that all language-specific grammatical information is stored in a lexicon: each word in the lexicon is associated with a structured syntactic category and a semantic form, such that the compositional potentials of the category and the semantics are isomorphic. A small universal set of combiners are responsible for assembling constituents into a full syntactic derivation; each combinator operates on adjacent constituents with appropriate categories to produce a new constituent and its compositional semantics, subject to constraints. A full grammar thus allows well-formed sentences to be transduced into semantic structures. The categories and combinators cooperate to license productive syntactic constructions like control and

---

1The research in this chapter was published at IWCS as Blodgett and Schneider (2019). §6.5.4 introduces novel analysis for coordination and ellipsis.
wh-questions, requiring the correct word order and producing the correct semantic dependencies. For example, consider the sentence “Who did John seem to forget to invite to attend?”: the correct logical form—in predicate logic, something like \( \text{seem}(\text{forget}(\text{John}_{i}, \text{invite}(\text{John}_{j}, \text{who}_{j}, \text{attend}(\text{who}_{j})))) \)—is nontrivial, requiring a precise account of several constructions that together produce long-range dependencies.

Whereas CCG traditionally uses some version of lambda calculus for its semantics, there has also been initial work using CCG to build parsers for Abstract Meaning Representation (AMR; Banarescu et al., 2013), a standard with which a large “sembank” of English sentences\(^2\) has been manually annotated.\(^3\) To date, dozens of publications\(^4\) have used the corpus to train and evaluate semantic parsers—most using graph-based or transition-based parsing methods (e.g., Flanigan et al., 2014; Wang et al., 2016; Lyu and Titov, 2018) to transform the sentence string or syntactic parse into a semantic graph via a learned statistical model, without any explicit characterization of the syntax-semantics interface. There is good reason to apply CCG to the AMR parsing task: apart from transparency of the syntax-semantics interface, state-of-the-art AMR parsers are known to be weak at reentrancy (e.g., Lyu and Titov, 2018), which presumably can be partially attributed to syntactic reentrancy in control constructions, for example. Most AMR parsers do not have specific strategies for dealing with different types of reentrancies. CCG offers a way to identify and derive reentrancies due to compositional structures like control and coordination. Prior work applying CCG to AMR parsing has begun to address this, but some of the important mechanisms that make CCG a linguistically powerful and robust theory have yet to be incorporated into these approaches.

In this chapter, we build on a core insight of previous work (e.g., Artzi et al., 2015; Beschke and Menzel, 2018) that AMR fragments can be directly represented as the semantics of CCG lexical entries. With appropriate definitions of the lexical items and combinatorial rules of CCG, the

---

\(^2\)See [https://amr.isi.edu/download.html](https://amr.isi.edu/download.html)

\(^3\)As originally defined, AMR is English-specific. However, a companion annotation standard, corpus, and parsers exist for Chinese (Xue et al., 2014b; Li et al., 2016; Wang et al., 2018b), and initial investigations have been made toward adapting AMR to several other languages (Xue et al., 2014b; Migueles-Abraira et al., 2018; Anchiêta and Pardo, 2018).

\(^4\)[https://nert-nlp.github.io/AMR-Bibliography/](https://nert-nlp.github.io/AMR-Bibliography/) is a categorized list of publications about or using AMR.
compositionality of CCG gives a derivation of a full AMR “for free”. In other words, AMR parsing can be reduced to CCG parsing (plus some additional semantic disambiguation and postprocessing). On a practical level, this should allow us to take advantage of existing CCG datasets and parsing methods for AMR parsing. In addition, explicitly storing AMR fragments in the CCG lexicon would provide a level of interpretability not seen in most statistical AMR parsers: the transparent syntax-semantics interface would decouple errors in the grammar from errors in the parsing model.

As a prerequisite for building a CCG-based AMR parser, or inducing a broad-coverage grammar (CCG lexicon) from data, we consider in this paper the formal mechanisms that would be necessary to derive AMRs with linguistic robustness. In particular, we address a variety of challenging syntactic phenomena with respect to AMR, showing the semantic fragments, associated syntactic categories, and combinators that will facilitate parsing of constructions including control, wh-questions, relative clauses, case marking, non-constituent coordination, eventive nouns, and light verbs. In so doing, we offer new semantics of combinators for semantic graphs beyond the proposals of previous work.

After an overview of related work (§3.3), we introduce our formalism for AMR graph semantics in CCG (§6.3). §6.5 gives example derivations for well-known linguistic phenomena including control, complex coordination, and eventive nouns. §6.6 discusses some implications of our approach.

6.2 Relevant Literature

A limited amount of prior research has combined CCG and AMR. Artzi et al. (2015) and Misra and Artzi (2016) develop an AMR parser using CCG by reformulating AMR graphs as logical forms in lambda calculus. We opt here for an approach similar to that of Beschke and Menzel (2018), where AMR subgraphs with free variables are treated as the semantics in the CCG lexicon. This requires definitions of the combinators that operate directly on AMR subgraphs rather than lambda calculus expressions.

Beschke and Menzel (2018) situate their formalization within the literature on graph grammars. They formulate their approach in terms of the HR algebra (Courcelle and Engelfriet, 2012), which

\footnote{Due to space constraints, we assume the reader is familiar with the basics of both CCG and AMR.}
Koller (2015) had applied to AMR graphs (but not with CCG). In this formalism, graph fragments called s-graphs are assembled to derive full graphs. S-graphs are equivalent to the AMR subgraphs described in this paper.

In particular, Beschke and Menzel define the semantics of CCG combinators in terms of HR-algebraic operations on s-graphs. They discuss a small set of combinators from Lewis and Steedman (2014) that includes forward and backward application and forward, backward, crossed, and generalized variants of composition. Readers may think of the research in Beschke and Menzel (2018) as having similar goals to this chapter with more mathematical and less linguistic foundation. We introduce equivalent semantics for application and composition (§6.4.1), avoiding the conceptually heavy notation and formalism from the HR algebra. They also specify Conjunction and Identity combinators, which we adapt slightly to suit our needs, and a Punctuation combinator. More significantly, they treat unary operators such as type raising to have no effect on the semantics, whereas we will take another route for type raising (§6.4.3), and will introduce new, relation-wise versions of application and composition (§6.4.2). Finally, whereas Beschke and Menzel devote most of their paper to a lexicon induction algorithm and experiments, we focus on the linguistic motivation for our definition of the combinators, and leave the development of suitable lexicon induction techniques to future work.

6.3 Graph Semantics

AMR is designed to represent semantics at the sentence level. For CCG lexical entries and combinators to parse AMR semantics, we need to formalize how AMR subgraphs can represent the semantics of individual words, and how combinators combine subgraphs to derive a full AMR. This section will formalize AMR subgraph semantics and CCG combinators for function application, composition, and type raising. Additionally, we propose new relation-wise variants of application and composition which are unique to graph semantics.

Each AMR subgraph contains nodes and edges from the resulting AMR as well as some nodes which correspond to free variables. The basic shape of an AMR subgraph appears in Figure 6.1.
Figure 6.1: Basic shape of AMR subgraph. Free variables (square, blue) are represented with \( x, y, z \), etc. AMR nodes (round, red) are represented with \( a, b, c \), etc. Dots indicate that part of the graph may be present or not. In this diagram, \( G \) is the entire graph, \( R \) is the root node labeled \( p \), and \( FV \) is an ordered list of free variables including \( x \) and \( y \).

Formally, an AMR subgraph is a tuple \( \langle G, R, FV \rangle \), where \( G \) is a connected, labeled, directed acyclic graph; \( R \) is the root node in \( G \); and \( FV \) is an ordered list of the nodes of \( G \) which are free and must be substituted by the end of the derivation. Though not shown in Figure 6.1, the root of an AMR subgraph may be a free variable. Intuitively, a subgraph with at least one free variable corresponds to a function, and a subgraph with no free variables corresponds to a formula.

6.3.1 Textual Notation

Taking inspiration from the PENMAN notation used for AMR, we use a notation for denoting AMR subgraphs in text form. First, any AMR graph can be written out in PENMAN notation, where parentheses describe the graph’s structure. For example, the AMR for “he wants to leave” can be written in one line as \((w/want-01 :ARG0 h/he :ARG1 (l/leave-01 :ARG0 h))\). Second, we use boxed number values to represent free variables, where \( \text{[1]} \), \( \text{[2]} \), etc. represent the 1st, 2nd, and so on free variables of the AMR subgraph. The textual notation for \textit{want} can be represented as \((w/want-01 :ARG0 \text{[2]} :ARG1 (\text{[1]} :ARG0 \text{[2]})\), where the notation describes both the graph structure and the location and order of free variables. Note that there is an equivalence between free variables and lambda notation such that \((a : \text{rel} \text{[1]})\) is equivalent to \( \lambda x.\text{rel}(a,x) \).
6.3.2 Examples of Semantic Graphs

Transitive and Intransitive Verbs. Figure 6.2a shows the semantics for a transitive verb. Since “read” has more than one semantic argument, the order of free variables matters: \( x \), the first free variable, must correspond to NP\(_1\), the rightmost syntactic argument in the category.

Adjectives. Figure 6.2b shows the semantics for an adjective. Note that, unlike in the examples above, the root of this subgraph is a free variable, since the root of this subgraph is what will be filled in. Ordinary adverbs have similar semantics.

Prepositional Phrases (Adjunct). Figure 6.2c shows semantics for the locative preposition “at”. To derive a prepositional phrase, assume available constituents “at”: \( \text{[\text{location}] } \) and “the library”: \( \text{[l/library]} \), which may be combined by application.

Null Semantics: Articles, etc. Some linguistic features, including tense and definite/indefinite articles, are not represented in AMR. For CCG derivations to deal with these elements, there will need to be a semantic representation which allows them to be “syntactic sugar”, affecting the syntactic category but adding nothing to the semantics in the derivation. We call this the identity function, following Beschke and Menzel (2018), and notate it as id. More precisely, if a constituent \( a \) has \( id \) as its semantics, then \( a \), when combined with another constituent \( b \) via application or composition (either as the function or as the argument), will produce \( b \)'s semantics for the resulting constituent.
6.3.3 Syntax-Semantics Isomorphism

A core property of CCG is that it provides transparency in the syntax-semantics interface: both syntactic categories and semantic forms are defined as functions permitting a compositional derivation of the sentence. The syntactic category determines which constituents may be constructed and in what word order. In the semantics, the word order (direction of the slashes) is irrelevant, but the functional structure—the arity and the order in which arguments are to be applied—must match in order for the semantics to remain well-formed as the sentence is derived based on the syntactic categories and combinatorial rules.

In other words, the functional structure of the category must be isomorphic to the functional structure of the semantics. For example, a hypothetical CCG category V/W/X/(Y/Z) would naturally correspond to a ternary function whose first argument, Y/Z, is itself a unary function.

This brings us to the following principle:

**Principle of Functional Isomorphism.** The semantics of a word or constituent cannot have higher\(^6\) arity than the CCG category calls for, and every functional category must take at least one semantic argument. For instance, a word or constituent with category PP/NP must have exactly 1 semantic argument; and the VP adjunct category (S/NP)/(S/NP) a.k.a. S/NP/(S/NP) can be interpreted as having 1 or 2 semantic arguments.

Without proving it formally, we remark that this helps ensure that syntactic well-formedness according to the categories will guarantee semantic well-formedness, with no attempt to apply something that is not expecting any arguments, and no free variables remaining in the semantics at the end of a sentence derivation. (An edge case where this guarantee might not hold is noted in fn. 9.)

---

\(^6\)We find that there are legitimate cases where the AMR arity should be *lower* than the CCG arity. FOR EXAMPLE, adverbs modifying a VP have the type (S/NP)/(S/NP) and have two syntactic arguments: NP and (S/NP). The syntactic argument NP is not a semantic argument of the adverb but is included in the type to specify that the adverb combines with a VP to produce a VP.
Table 6.1: Formal semantic rules for AMR combinators.

<table>
<thead>
<tr>
<th>combinator</th>
<th>function (left/right)</th>
<th>arg. (right/left)</th>
<th>result</th>
<th>FV ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>...1 □ ...2</td>
<td>a ...3</td>
<td>...1 a ...2 ...3</td>
<td>□ ...□ ...□</td>
</tr>
<tr>
<td>Composition (B, B^2)</td>
<td>...1 □ ...2</td>
<td>a ...3</td>
<td>...1 a ...2 ...3</td>
<td>□ ...□ ...□</td>
</tr>
<tr>
<td>Relation-wise Application (R)</td>
<td>...1 □ :rel_b ...2</td>
<td>a :rel_b □ ...3</td>
<td>...1 a :rel_b ...2 ...3</td>
<td>□ ...□ ...□</td>
</tr>
<tr>
<td>Relation-wise Composition (RB)</td>
<td>...1 □ :rel_b ...2</td>
<td>a :rel_b □ ...3</td>
<td>...1 a :rel_b ...2 ...3</td>
<td>□ □ □ ...□</td>
</tr>
<tr>
<td>...Second-order (RB^2)</td>
<td>...1 □ :rel_b ...2</td>
<td>a :rel_b □ ...3</td>
<td>...1 a :rel_b ...2 ...3</td>
<td>□ □ □ ...□</td>
</tr>
<tr>
<td><strong>Unary</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type Raising (T)</td>
<td>a ...1</td>
<td>□ :? a ...1</td>
<td>□ □ ...</td>
<td></td>
</tr>
<tr>
<td>Conjunction (&amp;)</td>
<td>x</td>
<td>a ...1, b ...2,...</td>
<td>x :op1 a ...1 :op2 b ...2...</td>
<td>□</td>
</tr>
</tbody>
</table>

Boxed numbers stand for free variables (FVs) in the semantics of each of the constituents being combined: □ stands for the lowest indexed FV in the function (head) constituent, and □ for the lowest indexed FV in the argument constituent, if any. Ellipses ... denote optional dominating structure (if preceding) and optional dominated structure (if following). Any FVs in these optional structures are preserved in the result, in the order given in the last column. For relation-wise combinators, the function constituent’s relation may also be :?. Crossing composition (B_x) and its variants behave semantically like their non-crossing counterparts. Not shown: exceptions to application and composition for the identity function (id), discussed in §6.3.2.

6.4 COMBINATORS

Table 6.1 shows the formulation of graph semantics for all the combinators described below. The formulas are schematic with attention paid to the resulting order of free variables, which semantically distinguishes application from composition. Another combinator in CCG, crossing composition, has the same semantics as regular composition. Semantics for the substitution combinator is left to future work.

6.4.1 FUNCTION APPLICATION AND COMPOSITION

In Function Application of AMR subgraphs, a free variable (blue) can be filled by the root of another AMR subgraph. Function application can only substitute the first free variable in FV corresponding to the rightmost argument.

Composition differs from function application in terms of the resulting order of free variables. While application and composition always differ syntactically, from a graph semantics point of
view, composition turns out to be the same as function application,\textsuperscript{7} where the root of one subgraph is substituted for a free variable in another subgraph. The difference between application and composition is captured in the resulting order of free variables. In the case of composition, the argument’s free variables are placed first on the free variable stack followed by the function’s free variables. This allows free variables in the AMR subgraph to consistently match syntactic arguments in the CCG category. This is a difference between composition in this work and in Beschke and Menzel’s (2018) work, where the semantics of application and composition is the same.

6.4.2 Relation-wise Application and Composition

When deriving a constituent, there are situations where it is desirable to have a semantic edge that is shared between the two constituents being combined. This corresponds to situations in lambda calculus where a function is taken as an argument and applied to some variable. For example, we specify the following lexical entry for the control verb “decide”, indexing arguments in the category with subscripts for clarity: \( S_b/\text{NP}_2/(S_{to}/\text{NP})_1 \): decide-\text{01} :ARG\text{0} :\text{ARG}1 (\text{\[\text{ARG}0\] }). Unlike a simple verb, “decide” selects for an embedded clause and controls its subject, coindexing it with the matrix subject. This is indicated in the semantics with the bolded :ARG\text{0} edge, which needs to unify with the :ARG\text{0} edge of the embedded predicate. Thus the sentence “Sarah decided to take the bus” in Figure 6.5 is formed by merging the :ARG\text{0} edge expected by “decide” and the :ARG\text{0} edge expected by “take” so that they may later be filled by the same argument, Sarah. Note that the number of semantic free variables respects the functional structure of the category (§6.3.3). To facilitate this, we define novel relation-wise variants of the application and composition combinators that expect an edge in common (call it the shared edge). Apart from control, relation-wise combinators are also useful for derivations with type raising and various interesting syntactic constructions.

\textsuperscript{7}Note that in some cases, function application can be used even when an argument has one or more free variables. Consider adverbs modifying a VP for example. The type of an adverb modifying a VP is (S\NP)/(S\NP) and can take an argument of type (S\NP) as an argument using function application, but it’s argument will have one free variable slot for its subject.
The full definition for relation-wise combinators is given in Table 6.1. Notably, the function constituent has its lowest-indexed free variable at the source of the shared edge, and the argument constituent has a free variable at the target of the shared edge (the variable’s index depending on the kind of application or composition). In the result, each free variable unifies with the node or variable at the same side of the edge in the other constituent. Other material attached to the shared edge in either constituent will be preserved in the result. Note that because relation-wise combinators act on the lowest-indexed free variable, it is possible to use them for any semantic role, which is what makes these combinators useful for both subject and object control structures.

The regular vs. relation-wise distinction applies only to the semantics; syntactically, relation-wise application (composition) is just like regular application (composition). We use the symbol R for Relation-wise Application and RB for Relation-wise Composition. During parsing, relation-wise combinators apply if and only if the two constituents being combined share a common relation with the appropriate free variables; otherwise, the non–relation-wise version of the combinator is used.

Relation-wise composition differs from relation-wise application in the index of the argument's free variable being unified and in the resulting order of free variables. Just as regular composition can be used to adjust the order that constituents are normally combined and “save an argument for later”, relation-wise composition does this with respect to a common edge. Examples of relation-wise application and relation-wise composition appear in Figure 6.8.

6.4.3 Type Raising

In CCG, Type Raising (T) converts an argument into a function. For example, the nominative case of the pronoun I can be coded in the syntactic category by making it a function that expects a verb phrase on the right and returns a sentence, thus preventing I from serving as an object. For our framework to support type raising, we need an appropriate semantic conversion that respects the functional structure of the category—thus, the type-raised semantics must take an argument. However, as type raising can be applied to different types of arguments, we do not know a priori which relation label to produce. Therefore, we introduce the notion of an underspecified edge.
denoted :?. The type-raised structure has a free variable at the source of the underspecified edge, with the original subgraph at the target. For example, see “John” and “Mary” in Figure 6.6, where type raising is necessary to support subject+verb constituents for coordination. The type-raised constituent must eventually be the input to a relation-wise combinator, which will specify the label on the edge.

The intuition behind type raising is that the resulting object should look for a function of the initial object and return that function’s output. Type raising allows the resulting subgraph to accept an object which needs to fill the target of some relation rel and allows the pre-type-raised object to fill that slot. In graph semantics, to make a non-function into a function, at least one free variable and relation must be added. The type-raised graph thus has a free-variable root, a relation :?, and its original node as a target of :?.

Note that in this strategy of representing type raising, the isomorphism between functions in semantics and syntactic category is maintained. This fits with CCG’s philosophy of a transparent syntax-semantics interface (§6.3.3). By contrast, Beschke and Menzel’s (2018) strategy was to leave the result of type raising semantically unchanged, creating a mismatch between the syntax and the semantics.

6.5 Linguistic Examples

This section explains the use of the combinators discussed in §6.3 for particular linguistic constructions.

Figure 6.3 shows the use of application (and identity application) combinators to derive a simple sentence. Composition is demonstrated in 6.4. Relation-wise application is used in Figure 6.5 to derive the semantics for control verb.
John likes the cat

Figure 6.3: Application and identity shown for the sentence “John likes the cat.”

Sarah suddenly entered the room

Figure 6.4: Composition shown for the sentence “Sarah suddenly entered the room.” A more complex example of composition, where the order of free variables matters, is given in Figure 6.6.

Sarah decided to take the bus

Figure 6.5: Control and relation-wise application shown for the sentence “Sarah decided to take the bus.”
Finally, Figure 6.6 demonstrates type raising, relation-wise composition, and conjunction as tools to derive a sentence with complex coordination.

John likes and Mary hates cats.

Figure 6.6: Complex coordination and type raising shown for the sentence “John likes and Mary hates cats.”

6.5.1 Passives, Wh-Questions, and Object Control

Figures 6.7 to 6.9 show CCG derivations with AMR semantics for three well-known linguistic phenomena in English: passives, wh-questions, and control. In a passive construction, a semantically core argument may be added by a syntactically optional adjunct phrase as in Figure 6.7. Note that in this semantic representation, only syntactically required arguments are represented in a predicate’s semantics, and so the passive verb *eaten* does not include an :ARG0 edge.
Figure 6.8 shows wh-question formation. Wh-questions are one complex and difficult phenomenon that is handled by CCG derivation. Additionally, Figure 6.8 gives examples of both types of relation-wise application and relation-wise composition.

Figure 6.9 shows an example of object control. Control is an important problem for graph semantics as it requires representing the subject (here Mary) as an argument of two predicates (see §6.4.2). Figure 6.9 demonstrates how our relation-wise combinators are able to handle complex control structures.

Figure 6.8: Wh-question example: “What did you do yesterday?” \(B_x\) stands for crossing composition, which has the same semantics as regular composition.

Figure 6.9: Object control example: “Mary persuaded John to practice guitar”. Note that the PropBank predicate persuade-01 specifies \(\text{ARG0}\) for the persuader, \(\text{ARG1}\) for the persuadee, and \(\text{ARG2}\) for the impelled action.
AMR provides notation for *inverse roles* that reverse the usual ordering of a relation. These are indicated with the -of suffix: \((a : \text{rel-of} \ b)\) is equivalent to \((b : \text{rel} \ a)\). This ensures that the graph can be constructed with a single root, and provides a convenient mechanism for expressing derived nominals and relative clauses. For instance, the noun phrases “teacher” and “a person who teaches” both receive the AMR \((\text{person} : \text{ARG0-of} \ \text{teach-01})\). If the subject matter is expressed, that is slotted into the :ARG1 of teach-01. This can be handled by treating “teachers” as a predicate of sorts, as seen in the derivation below.

![Figure 6.10: Inverse roles of derived nominals and relative clauses.](image)

In the example, *math* is taken as an argument of *teachers*. Note that some CCG derivations analyze noun-noun compounds as N/N N, but those categories fail to produce the correct semantics, since *math* is a semantic argument of a teaching event rather than the other way around. In the second example, relation-wise application is used to derive the same semantics for the larger expression “people who teach math.” Note that the :ARG0 edge under teach-01 is merged with the :ARG0-of edge using relation-wise application.

Also illustrated is the relative clause paraphrase, “people who teach math”. Here, the relativizer “who” needs to fill the appropriate role of the verbal predicate with its noun head “people”. An inverse role is produced so that *person*, rather than *teach-01*, will be the root of the resulting subgraph. The relation-wise application combinator must therefore be aware of inverses: it must match the :ARG0-of with the :ARG0 edge in the operand and effectively merge the two relations. Alternatively, the phrase could be parsed by first relation-wise composing “who” with “teach”, which requires similar handling of the inverse role, and then attaching “math” by application.
6.5.3 Eventive Nouns and PP Complements

This section will describe an approach to the semantics of eventive nouns like “decision”, and in the process will illustrate our treatment of prepositional phrase complements (as opposed to adjuncts: beginning of §6.3.2), which in CCG are traditionally given the category PP.

In English, many eventive nouns can be linked to semantic arguments via prepositional phrases, possessives, and light verb constructions, as shown in Table 6.2. AMR uses a canonical form with a predicate (typically based on a verbal paraphrase), treating *John decided, John’s decision, and John made a/his decision* as semantically equivalent. Despite some work on integrating event nominals and multiword expressions into CCG (Constable and Curran, 2009; Honnibal et al., 2010; de Lhoneux, 2014), we are not aware of any CCG analyses of light verb constructions, which have been studied computationally in other frameworks (e.g., Baldwin and Kim, 2010; Bonial et al., 2014; Ramisch et al., 2018), that gives them semantics equivalent to a content verb paraphrase. We offer such an analysis based on three principles:

1. The event frame is in the semantics of the eventive noun or verb.
2. For any syntactic argument of a noun or verb, the corresponding edge (and free variable) is in the semantics of the noun or verb.
3. Any function word (light verb, ’s, preposition, or infinitival to) that links the eventive noun to its semantic argument has an associated edge (and free variables) in its semantics.

<table>
<thead>
<tr>
<th>light verb construction</th>
<th>possessive form</th>
<th>AMR predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>make a decision [about/on]</td>
<td>my decision [about/on]</td>
<td>decide-01</td>
</tr>
<tr>
<td>pay attention [to]</td>
<td>my attention [to]</td>
<td>attend-02</td>
</tr>
<tr>
<td>make an attempt [to]</td>
<td>my attempt [to]</td>
<td>attempt-01</td>
</tr>
<tr>
<td>take a nap</td>
<td>my nap</td>
<td>nap-01</td>
</tr>
<tr>
<td>take a picture [of]</td>
<td>— (“my picture” is not eventive)</td>
<td>photograph-01 (suggested)</td>
</tr>
</tbody>
</table>

English eventive nouns shown with a light verb or possessive; words in square brackets mark additional semantic arguments. (In the AMR corpus, “take pictures” is actually treated superficially with take-01 :ARG1 picture, but we suggest photograph-01 instead.)
John made a decision on his major.

Figure 6.11: Light verb construction example: “John made a decision on his major.”

Note that when a verb or noun takes a PP complement, principles 2 and 3 force both the verb or noun and the preposition to hold the same edge in their semantics. This is compatible with relation-wise combinators as described in §6.4.2. The result is a nice analysis where both the eventive noun or verb and its complement preposition signal patientness.

With this analysis, the associated light verbs given in Table 6.2 (‘make’, ‘pay’, etc.) as well as possessive ’s take the semantics $\square : A R G 0$ and associated prepositions take the semantics $\square : A R G 1$. In other words, for each eventive noun, either a special light verb or a possessive contributes the agentive semantic relation—and (if present) a special preposition or infinitive to may contribute the patient semantic relation—thus allowing derivation of the same AMR regardless of form.

Figure 6.11 shows the derivation for “decision” in its light verb construction form. The preposition “on” redundantly represents the $: A R G 1$ edge, and is merged with “decision” by relation-wise application.\(^8\) The light verb “made” specifies the $: A R G 0$ edge.

\(^8\)The category $N / N P \langle N / P P_{\text{on}} \rangle$ for “on” is suggested by Mark Steedman’s analysis of English prepositions as particles (personal communication) and also maintains the Principle of Functional Isomorphism of §6.3.3.
6.5.4 Ellipsis and Coordination

As discussed in Ch. 4, ellipsis and other phenomena will often result in duplicate subgraphs in an AMR. This happens because some concepts in the semantic representation are unpronounced in the corresponding sentence and instead must be inferred. Duplicate subgraphs may be triggered by coordination or by certain expressions. An example related to ellipsis is “John likes coffee and Mary tea,” where there are two like events but only one is pronounced. A non-ellipsis example is the subset construction, such as in the phrase “most of the students”. This phrase evokes two different groups of students: first “the students” which is the larger group, second “most of the students” which is a subset of that larger group. This construction corresponds to duplication of the subgraph \([\text{person} : \text{ARG0-of study-01}]\) in the AMR. This section will explain how we derive the proper semantics for expressions like these using a special operator \(\text{dupl}()\) and additional conjunction combinator for dealing with coordination and ellipsis.

**Duplicate Operator.** To represent duplication in a derivational way, we introduce the operator \(\text{dupl}()\) which acts on a free variable and changes the free variable’s behavior. (This operator is only well defined when an additional is present in the subgraph.) When an AMR subgraph with free variable slots \(\text{[...]}\) and \(\text{dupl}()\) accepts its \(n\)th argument, the subgraph representing that argument is duplicated to fill both slots rather than creating a reentrancy. So for an arbitrary argument \((x/\text{concept}...3)\), an AMR subgraph \((\ldots\text{[...]}\ldots\text{dupl}())\) combines with the argument to form \((\ldots1(x/\text{concept}...3)\ldots2(x2/\text{concept}...3))\), where \(\text{dupl()}\) has been replaced by a new subgraph \((x2/\text{concept}...3)\). Figure 6.12 illustrates the CCG derivation of “most of the students” using this operator. Note that a node \(\text{dupl()}\) is present in the semantics for \(\text{of}\). When the free variable \(\text{[...]}\) is filled, rather than producing a reentrancy, \(\text{dupl}()\) results in a duplicate of whichever subgraph fills \(\text{[...]}\). Thus the final expression has two subgraphs \((\text{person} : \text{ARG0-of study-01})\) even though only one is pronounced.
most of the students

\[
\begin{array}{c|c|c}
\text{NP} & \text{NP} & \text{NP} \\
\text{most} & (\text{include-91}) & :\text{dupl}() \\
\text{id} & \text{person} & :\text{study-01} \\
\text{NP} & \text{N} & \text{N} \\
\end{array}
\]

Figure 6.12: CCG derivation for the phrase “most of the students” demonstrating the use of the \textit{dupl()} operator. Duplicate subgraphs are bolded.

Given this operator, we can now expand our definition of conjunction to handle a wider variety of phenomena. Table 6.3 shows the definitions of 3 types of conjunction combinators (\&_1, \&_2, and \&_3) for handling 3 cases of phenomena in AMR.

**Conjunction type 1** (\&_1) is identical to the definition of conjunction given in Table 6.1. Conjunction type 1 allows for simple coordination of arguments where arguments with unfilled free variables have their free variables merged resulting in a reentrancy. For example, the sentence “John

\[
\begin{array}{c|c|c|c|c|c}
\text{combinator} & \text{function} & \text{left arg.} & \text{right arg.} & \text{result} & \text{FV ordering} \\
\hline
\text{Conjunction type 1 (\&_1)} & \text{x} & \text{a} \ldots & \text{b} \ldots & \text{x :op1 a \ldots :op2 b \ldots} & \text{[]} \\
\text{Conjunction type 2 (\&_2)} & \text{x} & \text{rel1} \ldots & \text{rel2} \ldots & \text{rel1 \ldots :rel2} & \text{[]} \\
\text{Conjunction type 3 (\&_3)} & \text{x} & \text{a} \ldots & \text{b} \ldots & \text{x :op1 a \ldots :op2 b \ldots dupl(\text{[]})} & \text{[]} \\
\end{array}
\]

A table of 3 combinators representing conjunction to handle 3 different AMR phenomena: (1) simple coordination where coordinated phrases with unfilled arguments have their arguments merged resulting in a reentrancy, (2) coordination of modifiers where the resulting AMR expression simply joins the modifiers together at the same root, and (3) coordination with duplication where coordinated phrases with unfilled arguments result in the use of a \textit{dupl()} operator resulting in 2 or more copies of the argument when it is filled. The definition of conjunction given in Table 6.1 corresponds to Conjunction type 1 (\&_1). Conjunction type 2 and type 3 extend type 1 for better handling of modifiers and ellipsis respectively.
cooked and ate his food” coordinates two verbs which share a semantic agent and theme. So the result of coordination of *cooked* (cook-01 :ARG0 1 :ARG1 2) and *ate* (eat-01 :ARG0 1 :ARG1 2) should result in the 1s being being co-indexed and the 2s being co-indexed in the composite phrase. Conjunction type 1 (&1) allows us to derive the correct result (and :op1 (cook-01 :ARG0 1 :ARG1 2) :op2 (eat-01 :ARG0 1 :ARG1 2)).

Conjunction type 2 (&2) is used to combine modifiers to yield the correct AMR representation. For example, the AMR for the phrase “the big and yellow car” does not have an and node but rather attaches both modifiers :mod big and :mod yellow directly to their argument. Conjunction type 2 (&2) allows the correct derivation for this case, such that the semantics of the phrase “big and yellow” becomes 1 :mod big :mod yellow.

Finally, Conjunction type 3 (&3) is used in cases where coordination triggers duplication of one of the arguments in the resulting AMR. Consider the sentence “I should and you may eat.” The phrases being coordinated are *should* and *may* which have a shared object *eat*. However, the sentences actually evokes two separate eating events, where the first eating event takes the speaker as an agent and the second takes the addressee as an agent. To derive the correct semantics, the conjunction combinator has to create free variable that will duplicate the subgraph representing *eat*. conjunction type 3 (&3) allows us to derive the correct result. A complete derivation for the sentence “I should and you may eat” with conjunction type 3 is given in Figure 6.13.
I should and you may eat.

Figure 6.13: Example of right node raising with shared main verb: “I should and you may eat.” The sentence is derived using the combinator conjunction type 3 with duplication, rather than merging, semantic arguments of each coordinated phrase. Duplicate subgraphs are bolded.

6.6 DISCUSSION

Unlike many semantic formalisms, AMR does not specify a ‘compositional story’: annotations do not include any sort of syntactic derivation, or even gold alignments between semantic units and words in the sentence. This presents a challenge for AMR parsing, which in practice relies on various forms of automatic or latent alignments (see Szubert et al., 2018). Above, we have presented an analysis that lays the foundation for a linguistically principled treatment of CCG-to-AMR parsing that meets a variety of challenges in the syntax-semantics interface, and does so in a transparent way so that parsing errors can be diagnosed. We believe the approach is reasonably intuitive, flowing naturally from CCG syntax, AMR semantics, and the notion of free variables in subgraphs, without the additional need for complicated lambda calculus notation or a highly general graph grammar formalism.

To realize this vision in practice, an approach is needed to build a CCG parser enriched with graph semantics for deriving AMRs. We anticipate that existing CCG parsing frameworks can be adapted—for example, by developing an alignment algorithm to induce the semantics for lexical
entries from the AMR corpus, and running an off-the-shelf parser like EasySRL (Lewis et al., 2015) at training and test time for the syntactic side of the derivation. This approach would take advantage of the fact that our analysis assumes the ordinary CCG syntax for obtaining the compositional structure of the derivation. The only additional steps would be a) disambiguating the semantics of lexical entries in the derivation, and b) applying the semantics of the combinators as specified in Table 6.1. For each use of application or composition, the semantic parser would check whether the conditions for relation-wise combination hold, and otherwise apply the ordinary version of the combinator.⁹

Because AMRs are annotated by humans for raw sentences, rather than on top of a syntactic parse, we cannot expect a parser to elegantly handle the full construction of all AMRs according to compositional rules. Several components of AMR parsing are not part of CCG parsing and will have to be performed as postprocessing steps. These components include named entity recognition, time expression parsing, coreference resolution, and wikification, all of which need to be performed after (or before) CCG parsing. Additionally, there is a risk that a CCG lexicon may ‘overgenerate’, producing invalid parses, and additional checking—either in the combinators, or as postprocessing or reranking—may be warranted.

We are aware of certain phenomena where the approach described above would be unable to fully match the conventions of AMR in the CCG-derived semantics. The treatment of modal

⁹We have considered an alternative analysis where underspecified :? edges would be used not only for type raising, but for all case-marked pronouns, prepositions marking syntactic arguments, and other constructions where a word’s syntactic category involves an argument to a separate predicate. Thus, only the predicate would be allowed to specify semantic roles for its syntactic arguments. Relation-wise combinators would then require that the shared edge would be underspecified in the function constituent. The rationale would be that this avoids redundant specification of core roles like :ARG0 and :ARG1 in the lexical entries—e.g. in Figure 6.8, the :ARG1 for “What”, the :ARG0 for “did” would both be replaced with :?. After all, constructions like wh-questions, control, and case target syntactic relations (subject/object), which are merely correlated with semantic roles. And as pointed out by a reviewer, under the current approach, a wrong choice of semantic role for a cased pronoun’s semantics could result in the use of a regular combinator rather than a relation-wise combinator, leaving a free variable in the predicate unsatisfied and essentially breaking the syntax-semantics isomorphism. An argument in favor of the current policy is that prepositions can contain information about roles to a certain extent, and redundant specification of semantic roles may actually be helpful when confronted with a noisy parser and lexicon. We leave this open as an empirical question for parsing research.
auxiliaries, for example, as above the main event predicate in the AMR will be problematic for the CCG derivation when there is a preposed adjunct (as in “Tomorrow, John may eat rice”) because the modifier will semantically attach under the root of the semantics of the rest of the clause (possible-01 from “may”) rather than the main event predicate eat-01. Full derivations for these problem cases, as well as examples of purpose clauses, raising, and subject and object control, are given in Ch. C. We will leave it to future work to explore whether such limitations can be addressed via postprocessing of the parse, or whether additional expressive power in the combinators is necessary.

Finally, as pointed out by Bender et al. (2015), AMR annotations sometimes go beyond the compositional ‘sentence meaning’ and incorporate elements of ‘speaker meaning’, though an empirical study of AMR data found the rate of noncompositional structures to be relatively low (Szubert et al., 2018). Beschke and Menzel (2018) give interesting examples of AMR fragments that would be difficult to derive compositionally, e.g., “settled on Indianapolis for its board meeting”, where the AMR attaches Indianapolis as the location of the meeting and the meeting as the thing that was settled on (reflecting the inference settle on LOCATION for ACTIVITY ⇒ settle on [ACTIVITY at LOCATION]). We leave analysis of these types of structures to future work.

6.7 CONCLUSIONS

We have given the linguistic motivation for a particular method of deriving AMR semantic graphs using CCG. Our specification of AMR subgraphs and CCG combinators ensures a tight correspondence between syntax and semantics, which we have illustrated for a variety of linguistic constructions (including light verb construction semantics, which to the best of our knowledge has not previously been explored for CCG). Future empirical work can make use of this framework to induce CCG lexicons for AMR parsing.
CHAPTER 7

AN ANALYSIS OF CONCORDANCE AND DISCORDANCE BETWEEN CCG AND AMR

7.1 INTRODUCTION

Having developed in previous chapters a theory of CCG composition with AMR semantics and a large dataset of comprehensive AMR alignments, it is now possible to study, at a practical level, the relationship between our new alignments and CCG syntax. There are two questions we would like to answer about our AMR alignments with respect to syntax. First, to what extent is aligned AMR for English a linguistically compositional semantic schema—in other words can the subgraphs associated with our AMR alignments be composed into the correct graph using syntactically valid operations? Second, to what degree is CCG syntax in particular a useful inductive bias for inferring the forward composition? To answer these questions, we perform analyses comparing the structure of our AMR alignments with the structure of CCG syntax for the same sentences.

One advantage of using CCG for this analysis is that CCG maintains a correspondence between syntactic and semantic elements—a characteristic called syntax-semantics isomorphism or a transparent syntax-semantics interface.

We propose that an external edge (an edge connecting two aligned subgraphs) in an AMR should correspond to a CCG dependency (see (Clark et al., 2002)), with aligned tokens corresponding to the head and dependent tokens of the dependency. In cases where this occurs, we say that the AMR and CCG structures are concordant and otherwise we say they are discordant. We interpret concordance to mean that the semantic structure in AMR could be derived by relying on CCG. We interpret discordance as either the result of errors (CCG parse or AMR alignment errors) or as a case of AMR not being entirely linguistically compositional or linguistically interpretable. We analyze examples of discordance to find systematic differences between AMR and CCG syntactic
structures, as well as to understand the impact parsing error would have if CCG is to be used as an inductive bias in an AMR parser or generator. Finally, we discuss whether efforts can be made to make AMR and AMR alignments more linguistically interpretable and linguistically compositional.

7.2 RELEVANT WORK

There has not been a previous study of the linguistic compositionality of AMR or its relationship with syntax, on a large scale. However, there has been research on aligning AMR and syntax, usually on a small scale (Szubert et al., 2018; Chen and Palmer, 2017; Chu and Kurohashi, 2016), as well as parsing AMR using a syntactic formalism, (Misra and Artzi, 2016; Artzi et al., 2015; Beschke, 2019; Beschke and Menzel, 2018), and on AMR graph composition without directly relying on syntax (Groschwitz et al., 2018; Koller, 2015; Jones et al., 2012; Chiang et al., 2013; Peng et al., 2015; Peng and Gildea, 2016; Björklund et al., 2016).

7.2.1 ALIGNING AMR WITH SYNTAX

One syntactic analysis of AMR is conducted by Szubert et al. (2018). Szubert et al. (2018) use a rule-based algorithm to align AMR subgraphs with Universal Dependency subtrees allowing hierarchical (nested) alignments between AMR and a syntactic parse. Their study did not include an analysis of composition and was done for several hundred sentences. Several other alignment systems attempt to incorporate syntax into AMR alignments. Chen and Palmer (2017) use an EM algorithm to align AMR nodes with tokens in a Universal Dependencies parse in an unsupervised way while using syntactic, named-entity, and semantic-role features. Chu and Kurohashi (2016) use a supervised algorithm to align AMR subgraphs with constituency parse subtrees.

7.2.2 AMR PARSGING WITH SYNTAX

Some prior research has focused on combining AMR with CCG. Artzi et al. (2015) and Misra and Artzi (2016) develop an AMR parser using CCG by reformulating AMR graphs as logical forms in lambda calculus. Beschke and Menzel (2018) treats AMR subgraphs as functions with free variables.
and uses them as the semantics in the CCG lexicon. This requires definitions of the combinators that operate directly on AMR subgraphs rather than lambda calculus expressions. Ch. 6 builds on the strategy of Beschke and Menzel (2018) to develop a theoretical formalism of AMR semantics for CCG with a complete set of combinators for deriving graph-structures as AMR semantics.

7.2.3 AMR Graph Composition without Syntax

Several other works have studied compositional grammars and parsers for AMR without building AMRs in lockstep with syntax.\(^1\) Notably Groschwitz et al. (2018) and Groschwitz (2019) take a similar strategy to our analysis of associating lexical entries to AMR subgraphs and using composition by means of an AM-algebra to derive the full AMR graph. Other researchers use hyperedge replacement grammars—another graph grammar—to derive AMR graphs (Koller, 2015; Jones et al., 2012; Chiang et al., 2013; Peng et al., 2015; Peng and Gildea, 2016; Björklund et al., 2016)

7.3 Tools for Analysis

CCG Dependencies were first described by Clark et al. (2002) as a way to capture deep syntax from a CCG parse. CCG dependencies were later included in the CCGBank datasets (Hockenmaier and Steedman, 2007; Honnibal et al., 2010) and have been used for CCG parser evaluation as a way to abstract away from arbitrary choices in the order of operations when deriving a CCG parse.

7.3.1 CCG Dependency Graphs

An example of a syntactic dependency graph is depicted in Figure 7.1. CCG dependencies are a way to represent the syntactic structure of a sentence in a way that abstracts away from the order of derivation. This is important because CCG often allows many different orders for phrases to be combined that still result in the equally correct constituency structures. CCG dependency abstract

\(^1\)These approaches may use syntactic features, but the process of composition is done with semantic composition only, without relying on the syntactic side of grammatical composition.
Figure 7.1: CCG dependency graph and corresponding parse for the sentence “Most of the students want to visit New York when they graduate.” The edge labels describe which argument the dependency points to, where the label $i : A$ is the $i$th argument (starting from the rightmost) requiring a phrase of type $A$. To make them more visually distinct, we use blue edges for CCG dependencies and red edges for AMR composition graph dependencies.

away from the distinction between application and composition and are more frequently used in evaluation of CCG parsers because they present an unambiguous way of representing correct CCG derivations.

In Figure 7.1, each dependency points from a word to the head of a phrase which fills one of that word’s syntactic arguments. Deriving these dependencies requires a notion of co-indexing supertag substructures, for example between the NPs in $\text{want} ((S_{nto}\backslash NP_1)/(S_b\backslash NP_1))$ and $\text{visit} ((S_b\backslash NP_1)/NP_2)$ where deriving the correct dependencies requires that we know that NP$_1$—which appears twice in the supertag of $\text{want}$ and once in the supertag of $\text{visit}$—is all the same NP. Thus, CCG dependencies
capture deep (rather than shallow) syntactic structures such as relationships between subjects and embedded clauses in a control structure. This property makes it an ideal tool for comparing syntax to AMR.

**CCG Dependencies vs. CCG Derivations.** While CCG derivations and CCG dependencies contain much of the same information, there are several advantages to relying on CCG dependencies for the analyses in this chapter:

1. Each dependency points from the head of a phrase acting as a function to the head of a phrase acting as its argument. This allows us to easily identify which part of a phrase corresponds to the semantic argument and which part corresponds to the predicate.
2. Often several CCG derivations are equally good derivations of the same correct semantics. CCG dependencies are agnostic to these differences. CCG dependencies only represent whether a word was taken as an argument, not the order in which words were composed.
3. CCG derivation often resolves semantic dependencies (such as long-distance dependencies) in multiple steps of derivation. We take the position that resolving a semantic dependency in multiple steps is just as useful as resolving it in one step. CCG’s ability to resolve semantic dependencies in multiple steps (without relying on transformations) is an important feature, both for interpretability and parser performance. CCG dependencies are agnostic to the number of steps needed for resolving a dependency, which make them ideal for analysis.
4. CCG dependencies leverage co-indexing within and between supertags, and so they include dependencies for control structures (e.g., in “John wants to leave” there is a dependency from *leave* to *John*). Since these dependencies are difficult to infer without co-indexing, CCG dependencies can contain more information than the syntactic half of a CCG parse.
5. CCG dependencies capture the same information as CCG constituents in that any connected subgraph over a token span is a constituent in some derivation, so CCG dependencies and derivations are generally equivalent when it comes to analyzing spans (Note that the converse
is not necessarily true: a constituent in some derivation may not correspond to a connected subgraph. For example “of the students” in Figure 7.1).

There are some caveats to CCG dependencies as an analytical tool. One caveat is that there are certain design decisions of CCG dependencies that are separate from design decisions of CCG or CCGBank, such as the decision to skip over prepositions when drawing the dependency for a predicate that takes a PP as a core argument. See for example Figure 7.1 where in “Most of the students”, there is a dependency from most to students instead of a dependency from most to of, which syntactically should be the correct argument. This means that the analyses in this chapter will have some of their results determined by properties of CCG dependencies rather than CCG itself.

7.3.2 AMR COMPOSITION GRAPH

Given a set of comprehensive AMR alignments as described in 4, we can re-imagine the AMR graph as a composition graph, where each aligned subgraph acts as a node and each external edge corresponds to a dependency. Figure 7.2 illustrates the composition graph of our example sentence. If our goal is to parse an AMR in a compositional way, knowing the aligned subgraphs for each span and the knowing the composition graph is enough to uniquely determine the correct AMR graph. In the opposite direction, we can infer the composition graph given the complete AMR and the alignments.

So a central question of interest is to what degree does the AMR composition graph structurally resemble (or is inferrable from) CCG dependency structure. This will tell us how often a semantic dependency—as interpreted by AMR with alignments—corresponds to a syntactic dependency—as interpreted by CCG, by means of CCGBank data and CCG dependencies. Answering this question will allow us to draw conclusions about the degree to which our AMR alignments are linguistically compositional and to what degree CCG syntax is likely to be a viable inductive bias.

Note that because the direction of dependencies is from functions to arguments, adjuncts will generally be the head of a dependency rather than the dependent, because in CCG an adjunct takes its object as an argument. This also matches the semantic dependency structure of AMR. For example,
Most of the students want to visit New York when they graduate."

The edge labels describe which argument the dependency points to, where the label $i : rel$ is the $i$th argument (from the closest argument on the right to the furthest argument on the right, followed by the closest argument on the left to the furthest argument on the left) which fills edge $rel$. To make them more visually distinct, we use blue edges for CCG dependencies and red edges for AMR composition graph dependencies.

a phrase like “big house” has a dependency from “big” to “house” in the AMR composition graph, associated with an edge $x : mod \ \text{big-01}$ aligned to “big” in the AMR.

For this reason, we are actually interested in the aligned argument structures in our AMR alignments, rather than the structure of the AMR graph itself. That’s so that we can draw a correspondence between incoming edges, which are a part of an adjunct’s argument structure, and CCG dependencies, to show that the composition works as expected.

7.3.3 Differences in Design Decisions of AMR and CCG

Some sources of disagreement are expected and are a result of design differences between AMR and CCG. One frequent source of disagreement is what to consider the head of a phrase. In AMR, the semantic head is used as the root of a subgraph, but what AMR considers a head will not always be the same as what CCG considers a head. Figure 7.3 shows examples of various phrase types with AMR and CCG dependencies for comparison.

Below are cases of discordance which are the result of design differences between AMR and CCG:
Coordination. Since CCG parsers often include special combinators for dealing with coordination, there are generally fewer if any dependencies involving coordinating conjunctions and and or. Instead of treating and and or as functions, CCG parsers use a special supertag "conj" which allows them to combine two constituents of the same type regardless of what that type is. Since and and or do still have an AMR semantics but no CCG dependencies, we see a systematic difference between the two schemas. Further, in AMR coordinating conjunctions are the semantic head of a coordinated phrase, which differs from how most syntactic formalisms describe coordinated phrases.

Prepositions. In the case of AMR, there are three possible structures a preposition could correspond to.

1. Core role: A preposition could evoke a core role of a nearby word, such as to evoking :ARG2 in “I gave a snack to the duck” (give-01 :ARG0 i :ARG1 snack :ARG2 duck).

2. Non-core role: A preposition could evoke a non-core relation such as :location or :time in which case it would be aligned to a single relation.

3. Subgraph: A preposition could be aligned to a subgraph such as behind to behind :op1 in “the cat went behind the house” (go-02 :ARG0 cat :ARG4 (behind :op1 house)). In this case, behind is taken as a core argument but is treated as the head of the prepositional phrase.

This further complicates our analysis of prepositions since in all other cases AMR treats the object of preposition as the head.

This can make it complicated to draw correspondences between AMR and CCG prepositional phrases. Firstly, AMR and CCGBank may disagree about whether a preposition corresponds to a core or non-core role, in which case the structures will be discordant.² Since there is quite a bit of ambiguity in distinguishing between core and non-core roles, even for expert annotators, we expect this distinction to be difficult to handle. Secondly, AMR sometimes treats prepositions as

²Because this is a major difference between AMR and CCGBank data, it may be worth modifying a CCG supertagger to better handle these discrepancies in future work. AMR core roles are based on PropBank, and it may be possible, either in a rule-based way or by tuning a model on new data, to develop a CCG supertagger that identifies core roles in a PropBank style rather than a CCGBank style. This would go a long way to making CCG more amenable to composition of AMR semantics.
<table>
<thead>
<tr>
<th>1. Eventive Noun</th>
<th>2. (Noun) Coordination</th>
<th>3. (Verb) Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMR</td>
<td>AMR</td>
<td>AMR</td>
</tr>
<tr>
<td>math teacher</td>
<td>John cooked chicken and rice</td>
<td>John ate and left</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Preposition Core Role</th>
<th>5. Preposition Non-Core Role</th>
<th>6. Preposition Subgraph</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMR</td>
<td>AMR</td>
<td>AMR</td>
</tr>
<tr>
<td>John went to the store</td>
<td>John shopped in the store</td>
<td>The cat went behind the house</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AMR</td>
<td>AMR</td>
<td>AMR</td>
</tr>
<tr>
<td>Should we eat here?</td>
<td>I think John is busy</td>
<td>John said that he likes cooking</td>
</tr>
</tbody>
</table>

**Figure 7.3: Examples of concordance and discordance.** Each graph illustrates the relevant dependencies for syntactic phenomena to show how the dependencies differ between our AMR composition graphs and CCG dependency graphs. AMR dependencies are displayed above each phrase in red, while CCG dependencies are displayed below in blue. Dependencies with no corresponding dependency in the opposite formalism are dashed. (CCG dependencies are depicted without preprocessing. See §7.5.1)

the semantic head and sometimes doesn’t whereas CCG dependencies always treat the object of preposition as the head, resulting in discordant structures.

**Predicative Adjectives.** AMR treats predicative adjectives as the head of the clause they occur in. So, for the sentence “John is busy.”, busy-01 will be the head of the entire phrase. This differs from CCG which treats the copula as the head.
**Embedded Clauses.** Some complementizers and relativizers such as *that*, *which*, etc. are treated as the head of an embedded clause by CCG dependencies. This differs from AMR which always treats the embedded verb as the head.

**Eventive Nouns.** One feature of AMR is that it attempts to abstract away from parts of speech. A consequence is that many nouns correspond to event frames in AMR and can have the same semantic roles as a verb evoking the same event frame. So whereas in CCG, nouns tend to be arguments only, in AMR eventive nouns are much more likely to act as a predicate and take other words as arguments. For example “math teacher” which in CCG features a dependency from *math* (N/N) to *teacher* (N), but which in AMR has a dependency in the opposite direction with *math* (math) as an argument of *teacher* (person :ARG0-of (teach-01 :ARG1 ...)).

**Alignments to Punctuation.** Our AMR alignments do include some alignments to punctuation, particularly alignments of the form “?” → amr-unknown, “.” → multi-sentence, and “;” → and as well as others. These alignments are useful in that they allow a parser to use punctuation to infer parts of an AMR that are otherwise unpronounced in a sentence. Since CCG dependencies do not include punctuation in the elements that can participate, there is a predictable difference in structure between AMR and CCG for these alignments.

**Empty Syntactic Dependencies.** CCG dependencies contain many edges that do not correspond to any AMR semantic role or relation. These especially arise for null-semantic alignments in AMR, such as *the*, *is*, *will*, etc. which are unaligned in our AMR alignments because they do not evoke a semantics that AMR includes in its representation. Thus, these tokens have CCG dependencies that do not correspond to any semantics in the AMR. Empty semantic dependencies also arise in the case of raising where in an example like “John seems really busy” there is a syntactic relationship between the raising verb *seems* and the raised subject *John* but there is no corresponding semantic relationship.
**Multi-Token Spans.** All CCG dependencies are from one token to another token. However, our AMR alignments are based on spans rather than individual tokens. Keeping this in mind, we design our analysis to consider AMR and CCG structure to be concordant as long as the head and dependent of a CCG dependency are contained in the correct AMR-aligned spans. There is also a possibility of errors due to an AMR-aligned span not having the expected structure in the CCG dependencies. To test for this, we perform a separate analysis of to what extent AMR-aligned spans correspond to connected subgraphs within the CCG dependencies graph.

### 7.4 Data

We rely on AMR alignment data and CCG parse data and we use both automatic and gold data for each. Our base dataset for our analysis is the LDC2020T02 dataset (Knight et al., 2020) of $\approx$60,000 sentences along with the Little Prince dataset of $\approx$1,500 AMRs. This dataset includes automatic comprehensive alignments, as discussed in Ch. 4 and 5, as well as 350 gold aligned sentences.

We use the CCG Parser EasySRL (Lewis et al., 2015) to add automatic CCG dependencies to our sentence\(^3\) EasySRL is a syntactic/semantic parser based on an A* parsing algorithm which jointly parses CCG as well as semantic roles.

Since parser and aligner error is one of the concerns for our analysis, we use the CCG Rebank corpus (Honnibal et al., 2010) of gold CCG parses and dependencies as a source of gold syntax. CCG Rebank and LDC2020T02 are both built (partially) on top of the Wall Street Journal corpus and have a small number of sentences in common. Table 7.1 shows the number of sentences included in four datasets: (1) our dataset of automatic alignments and CCG dependency graphs, (2) a dataset of gold CCG dependencies from the CCG Rebank corpus and automatic alignments, (3) a dataset of gold alignments with automatic CCG dependencies, and (4) 6 sentences with both gold alignments and CCG dependencies.

\(^3\)For some sentences, EasySRL fails to generate a parse, so we limited our analysis to 49,867 where the parser produced a CCG dependency graph.
7.5 Methodology

This section will illustrate what it means for AMR and CCG to be concordant or discordant in terms of structure from several points of view. This section will describe our preprocessing steps, our lexical analysis, our structural analysis, and our detailed analyses of spans and reentrancies.

7.5.1 Preprocessing

In some cases CCG dependencies differ systematically from AMR in ways that are consistent and can be addressed in a rule-based way. We apply the following preprocessing steps to our CCG dependencies before starting our analysis:

1. Coordination: In coordinated phrases, AMR uses a conjunction as the semantic head. For example, and :op1 cat :op2 dog has a root and which functions as the head of the subgraph. We alter each dependency graph such that any token with supertag “conj” is treated as the head of its arguments. Thus dependencies are altered to be from/to and instead of its arguments. (We only do this for dependencies where the associated combinator is applied after the coordinated phrase is complete.)

2. Predicative Adjectives, Nouns, and Prepositions: AMR treats predicative adjectives (and other non-verbs) as the root or head of a phrase. We modify each dependency graph such that dependencies point to/from the predicate instead of a copula.

3. that-Clauses: CCG dependencies treat the word that as the head of embedded clauses that begin with that. We move dependencies to point to/from the main verb of the embedded clause.

Table 7.1: Number of sentences in each dataset.

<table>
<thead>
<tr>
<th>Alignments / Syntax</th>
<th>automatic</th>
<th>gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>automatic</td>
<td>49,867</td>
<td>351</td>
</tr>
<tr>
<td>gold</td>
<td>324</td>
<td>6</td>
</tr>
</tbody>
</table>
4. Adverbs and other Adjuncts: CCG dependencies include some (arguably superfluous) dependencies from adverbs and adverbial prepositional phrases to the subjects of verbs they modify. Because most adverbs have a supertag like (S\NP)/(S\NP), the first NP is technically considered one of its syntactic arguments. We remove these dependencies from an adverb to a noun.

7.5.2 Lexical Analysis: Comparing Syntactic and Semantic Argument Structures

One question of interest is whether the semantic argument structure represented in AMR is structurally similar to or can be inferred from the syntactic argument structures represented in CCG. We could for example create a supertag for each span inferred from our AMR alignments in terms of the number and types of arguments appearing on the left and the right, and then we would be able to compare these AMR-flavored supertags to the corresponding CCG supertags to see if they match. That would give us insight into how similar AMR and CCG argument structures are at the word level, and how useful CCG supertags would be for inferring the right AMR argument structures.

Since we are not actually interested in the atomic types, we will only consider the number of arguments and whether they appear on the left or the right for the purposes of our analysis. We can think of this information as equivalent to a supertag template of the form:

\[ A[B_1B_2 \ldots]/C_1C_2 \ldots \] (7.1)

where \( A \) stands in for the alignment’s resulting syntactic type, \( B_1, B_2, \ldots \) stand for arguments appearing on the left, and \( C_1, C_2, \ldots \) stand for arguments appearing on the right. If we ignore the syntactic types of \( A, B_1, B_2, \ldots, C_1, C_2, \ldots \) then this supertag template captures the basic structure of AMR argument structures, namely the number of required arguments appearing on the left and the number appearing on the right. We can then compare to the same structure derived from CCG dependencies to evaluate how similar the argument structures are while ignoring atomic types.
**Empty Syntactic Dependencies.** While we are generally more interested in whether AMR dependencies correspond to a CCG dependency, it is also of interest to see how often the converse is true. As a consequence of its design, CCG does have many dependencies which do not correspond to AMR semantics. These dependencies do not tell us whether an AMR is linguistically compositional, but they may affect the prospect of CCG dependencies as an inductive bias for AMR parsing. If it is difficult for a model to tell the difference between a CCG dependency with a corresponding semantics and one without semantics, including empty syntactic dependencies as an input to a parser may result in more parser errors. With this in mind, we include empty syntactic dependencies in our lexical analysis (as a part of our supertag templates) but we do not include them in our structural analysis in the next section.

### 7.5.3 Structural Analysis: Comparing CCG Dependencies with AMR Alignments

We measure agreement by comparing AMR alignments to CCG dependencies. From a set of comprehensive AMR alignments, we build an AMR composition graph, whose dependencies should ideally correspond to CCG dependencies. The goal is to see if a relationship, such as a semantic role between a head and dependent, corresponds to a CCG dependency between the same head and dependent. CCG dependencies represent compositionality: they are drawn from a word which serves as a function to each of its arguments.

Next we develop an operational definition of structural concordance and discordance between AMR alignments and CCG dependencies. For a given AMR alignment with aligned semantic roles (these may be incoming or outgoing edges), we consider the AMR alignment and CCG dependencies to agree if:

1. there is an AMR semantic role with a function aligned to span A and an argument aligned to span B,
2. there is a corresponding CCG dependency from a token W to a token V, and
3. W is in A and V is in B
For example in the phrase “John took a bath”, an AMR dependency from “took a bath” to “John” would be concordant with a CCG dependency from “took” to “John”. This is a type of recall evaluation metric because we are only interested in whether each AMR relation corresponds to a CCG relation, while it is perfectly fine for a CCG relation to have no corresponding semantics.

One issue for our analysis is that our AMR alignments are associated with spans rather than tokens, while CCG dependencies are always between tokens. For this reason, we consider AMR composition graph dependencies to be concordant with a CCG dependency if the head/dependent token in CCG is contained in the span of the AMR dependency head/dependent. This is a weaker constraint than strict equality of heads and dependents.

7.5.4 Further Analyses

Besides the structural and lexical analyses, we conducted several analyses of more specific phenomena.

Reentrancies. We chose to analyze concordance of AMR reentrancies separately from our structural analysis. CCG is designed to capture certain phenomena which correspond to reentrancies in AMR, including coordination and various types of control. Yet, many reentrancy types are not handled by CCG because they are not syntactic in nature, but correspond to phenomena like coreference and pragmatic inference. For the cases where CCG is expected to provide a solution, we perform the same evaluation as in our structural analysis but for reentrancies only.

Multi-token Spans. Given that our AMR alignments are based on spans while CCG dependencies are based on tokens, we also evaluate whether each multi-token span is linguistically compositional in the CCG dependency structure. We consider a multi-token span to be linguistically compositional if its dependencies form a connected graph (ignoring edge direction because of issues arising from adjuncts) in the CCG dependency graph.
Table 7.2: Definitions and examples for various sources of discordance.

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition &amp; Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>preposition: core role</strong></td>
<td><strong>Definition:</strong> A preposition is core in AMR but non-core in CCG.</td>
</tr>
<tr>
<td></td>
<td>“… help himself from apologizing for anything and everything.”</td>
</tr>
<tr>
<td></td>
<td>AMR: <em>apologizing</em> is a core argument of <em>help</em> signalled by <em>from</em>.</td>
</tr>
<tr>
<td></td>
<td>CCG: <em>from apologizing</em> is an adjunct. (error)</td>
</tr>
<tr>
<td><strong>preposition: non-core role</strong></td>
<td><strong>Definition:</strong> A preposition is non-core in AMR but core in CCG.</td>
</tr>
<tr>
<td></td>
<td>“…drive over to the next town…”</td>
</tr>
<tr>
<td></td>
<td>AMR: <em>to the next town</em> is an adjunct.</td>
</tr>
<tr>
<td></td>
<td>CCG: <em>town</em> is a core argument of <em>drive</em>.</td>
</tr>
<tr>
<td><strong>preposition: attachment</strong></td>
<td><strong>Definition:</strong> AMR and CCG attach a prepositional phrase to different parts of the sentence.</td>
</tr>
<tr>
<td></td>
<td>“Ending the <em>war in</em> Iraq”</td>
</tr>
<tr>
<td></td>
<td>AMR: <em>in</em> modifies <em>war</em>.</td>
</tr>
<tr>
<td></td>
<td>CCG: <em>in</em> modifies <em>Ending</em>. (error)</td>
</tr>
<tr>
<td><strong>preposition: object</strong></td>
<td><strong>Definition:</strong> AMR and CCG assign different tokens as the object of preposition.</td>
</tr>
<tr>
<td></td>
<td>“…physics of the <em>earth</em>, sun, moon system”</td>
</tr>
<tr>
<td></td>
<td>AMR: <em>system</em> is the prepositional object.</td>
</tr>
<tr>
<td></td>
<td>CCG: <em>earth</em> is erroneously identified as the prepositional object.</td>
</tr>
</tbody>
</table>

Example sentences are taken from our base dataset. Each example demonstrates a discordant AMR dependency. The dependency head is in bold, the dependent is underlined. Some relevant phrases for CCG are in italics.
<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
</table>
| eventive nouns       | **Definition:** A noun takes a semantic argument in AMR, but does not take the same argument in CCG. | “... no maximum age limit ...”  
AMR: *limit* takes *age* as an argument (*age* is the thing being limited).  
CCG: *age* takes *limit* as an argument. |
| coordination scope   | **Definition:** AMR and CCG assign different scopes to a coordinated phrase, resulting in a difference in dependencies. | “So we ignore those in an institution or long term care .”  
AMR: the coordinated phrases are *an institution, long term care*.  
CCG: the coordinated phrases are *those in an institution, long term care*. (error) |
| phrase head mismatch | **Definition:** AMR and CCG assign the same constituent as an argument but the constituent has a different phrase head in CCG versus AMR. | “Keep in mind insurance companies are not in the business of ...”  
AMR: *Keep in mind* takes *business* as a dependent, which is treated as the semantic head of the entire clause.  
CCG: *Keep* takes *in* as a dependent, which is treated as the syntactic head of the entire clause (See Preprocessing §7.5.1). |
Table 7.4: Definitions and examples for various sources of discordance (continued part 2).

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition:</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverted dependency</td>
<td>A dependency from A to B in AMR corresponds to a dependency from B to A in CCG (excluding eventive nouns). This often happens in cases of type raising and similar phenomena.</td>
<td>“Who Drops Out After Super Tuesday?” AMR: <em>Drops Out</em> takes <em>Who</em> as a semantic argument. CCG: <em>Who</em> (<em>S[wq]</em>/(S[dcl]\NP)) takes the rest of the sentence as a syntactic argument.</td>
</tr>
<tr>
<td>non-projective</td>
<td>A semantic argument in AMR results in a non-projective dependency, and can thus not be applied without permutation, type-raising or some other mechanism.</td>
<td>“He ’s . . . only in out of spite .” AMR: <em>only</em> modifies <em>spite</em>, but only after it is taken as an argument of <em>out of</em> (aligned to cause-01) CCG: <em>only</em> and <em>spite</em> do not have a dependency.</td>
</tr>
</tbody>
</table>

7.6 Discussion

7.6.1 Parsing and Alignment Errors

One obvious source of discordance between AMR and CCG is model error, either on the side of CCG parsing or on the side of our AMR aligner. We reported evaluations for all combinations of automatic/gold CCG and AMR alignments in order to investigate the contributions of model error. Our results show that while model error is a contributor of discordance, it is not the main contributor. For example, our structural concordance recall shows a difference of 7.17% (68.63% vs. 61.46%) as a contribution of gold alignments and 3.98% (65.44% vs. 61.46%) as a contribution of gold syntax, while our supertag template accuracy shows a difference of 4.13% (67.98% vs. 63.85%) as
a contribution of gold alignments and yields a lower result with gold syntax. Even after including
 gold data, these scores show room for improvement, with a large part of the remaining discordance
 resulting from design differences between AMR and CCG.

7.6.2 Sources of Discordance by Type

Table 7.4 provides definitions and detailed examples of each source of structural discordance
 identified by our heuristics. Eventive nouns, prepositions, and coordination stand out as the three
 largest sources of structural discordance in our data, with the first two being dominant sources of
discordance even in our gold data. This tells us that, while parser and alignment error do contribute
to the structural discordance we see, differences in design decisions are the biggest contributors of
structural differences between AMR and CCG for these cases. I would note that eventive nouns,
prepositions, and coordinated structures each create challenges that (1) may be unique to AMR
and (2) may necessitate unique solutions rather than an entirely end-to-end approach. Eventive
nouns have argument structures which are unique for each lexeme, some of which may be very rare.
Prepositions require a distinction between core roles (which generally depend on the head lexeme)
and non-core roles (which do not), as well as considerations of what is a likely object of preposition
and where a PP should be attached. Coordinating conjunctions have argument structures which
depend on the number of phrases being coordinated and need to have scopes which are syntactically
and semantically similar. Each of these issues creates problems for semantic parsing and motivates
specialized solutions. One lesson suggested by our analysis is that specialized modules for dealing
with each of eventive nouns, prepositions, and coordination may be a fruitful area of future research
for semantic parsing.

4We also anticipate that some instances of discordance are the result of annotator artifacts which differ
between AMR and CCG rather than differences in design, though we don’t have a way to separate out these
cases.
5or possibly annotator artifacts
7.6.3 Phrase Head Mismatch

One difference in design that may make it difficult to incorporate CCG syntax into AMR in future work is the tendency of AMR and CCG to prefer different heads of a phrase. In some cases the differences are systematic based on part-of-speech. For example, most modals are the head of a clause instead of the main verb: In “Some cats can swim”, can (possible-01) is the sentence head in AMR, while swim is the sentence head in CCG. In other cases, particular lexemes will be assigned as the phrase head in AMR regardless of their part-of-speech: In “John will likely pass his exam”, AMR may treat the adverb likely (likely-01) as the semantic head since it takes the remainder of the clause as a semantic argument. One possible workaround would be treating modals in AMR with an inverse role (i.e. :ARG1-of likely-01) and then rearranging the graph to make them the root in postprocessing.

7.6.4 Comparison to Other Analyses of Compositionality of AMR

A number of previous works discuss the degree to which AMRs exhibit noncompositional structure. Bender et al. (2015) points out that AMR annotations sometimes go beyond the compositional ‘sentence meaning’ and incorporate elements of ‘speaker meaning’. During their analysis, Szubert et al. (2018) found the rate of noncompositional structures to be relatively low but existing. Beschke and Menzel (2018) give interesting examples of AMR fragments that would be difficult to derive compositionally, e.g., “settled on Indianapolis for its board meeting”, where the AMR attaches Indianapolis as the location of the meeting and the meeting as the thing that was settled on (reflecting the inference settle on LOCATION for ACTIVITY ⇒ settle on [ACTIVITY at LOCATION]). One insight that can be gained from the analyses in this chapter is that an AMR can be noncompositional in two senses. First, it can be noncompositional in the sense of not adhering to a particular analysis of the ordered composition of a sentence—such as one given by CCG or another trusted grammar formalism. This first sense of noncompositionality might be resolved by relying on a syntactic analysis which is conscientious and flexible towards design decisions of AMR. Second, an AMR can be noncompositional in the sense that the derivation of the complete AMR requires some additional
knowledge. This includes cases where the AMR has additional pragmatic information, or where the AMR has a graph structure that cannot reasonably be inferred from ordered composition. We believe it is still worth while to consider how AMRs with this 2nd sense of noncompositionality might still be derived in a linguistically interpretable way. For example, a study of what sorts of graph transformations might be necessary to instill particular kinds of pragmatic knowledge into an AMR would be a valuable next step. We leave this type of investigation to future work.

7.7 Results

The following sections introduce our results for our structural and lexical analyses, as well as our study of reentrancies, multi-token spans, and empty syntactic dependencies.

Table 7.5: Concordance recall by dependency.

<table>
<thead>
<tr>
<th></th>
<th>Concordance Recall by Dependency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>automatic</td>
</tr>
<tr>
<td></td>
<td>$S = 49,867$</td>
</tr>
<tr>
<td>all</td>
<td>61.46</td>
</tr>
<tr>
<td>- verb (31%)</td>
<td>75.67</td>
</tr>
<tr>
<td>- noun (16%)</td>
<td>36.36</td>
</tr>
<tr>
<td>- preposition (15%)</td>
<td>51.47</td>
</tr>
<tr>
<td>- adjective (11%)</td>
<td>75.62</td>
</tr>
<tr>
<td>- adverb (10%)</td>
<td>65.08</td>
</tr>
<tr>
<td>- conjunction (9%)</td>
<td>53.86</td>
</tr>
<tr>
<td>- other (8%)</td>
<td>60.96</td>
</tr>
</tbody>
</table>

This table illustrates the percentages of external edges in AMR which correspond to a CCG dependency with the correct parent and child. This is considered a recall score since it only considers whether an AMR relation has a matching CCG relation and not the other way around. The score is based on dependencies after pre-processing and does not include reentrancy edges or edges aligned to punctuation in AMR. $S$ is the number of sentences and $N$ is the number of (AMR) dependencies.
7.7.1 Structural Analysis

Table 7.5 shows our main results of dependency concordance recall by dependency—which measures the percentage of AMR composition graph dependencies which have a corresponding CCG dependency. We show results for combinations of gold and automatic data, and we further break down results by the type of dependency—whether it is a part of a predicate, an adjunct, or a single relation. One insight that can be seen here is that the highest results come from gold alignments and gold alignments/syntax datasets, suggesting that aligner error is a prominent source of discordant structure. Table 7.6 further breaks down the discordant dependencies for each dataset into subcategories based on heuristics designed to explain the cause of discordance for that dependency.

Table 7.6: Discordance details.

<table>
<thead>
<tr>
<th>Breakdown of Discordant Dependencies (%)</th>
<th>automatic</th>
<th>gold alignments</th>
<th>gold syntax</th>
<th>gold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S = 49,867$</td>
<td>$S = 324$</td>
<td>$S = 351$</td>
<td>$S = 6$</td>
</tr>
<tr>
<td></td>
<td>$N = 176,435$</td>
<td>$N = 830$</td>
<td>$N = 1,304$</td>
<td>$N = 16$</td>
</tr>
<tr>
<td>all</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00 (16/16)</td>
</tr>
<tr>
<td>- prepositions</td>
<td>20.02</td>
<td>17.47</td>
<td>22.09</td>
<td>18.75 (3/16)</td>
</tr>
<tr>
<td>- PP attachment</td>
<td>7.64</td>
<td>6.02</td>
<td>7.67</td>
<td>6.25 (1/16)</td>
</tr>
<tr>
<td>- AMR non-core vs. CCG core</td>
<td>6.10</td>
<td>7.11</td>
<td>7.21</td>
<td>6.25 (1/16)</td>
</tr>
<tr>
<td>- object of preposition</td>
<td>3.68</td>
<td>2.17</td>
<td>4.98</td>
<td>0.00 (0/16)</td>
</tr>
<tr>
<td>- AMR core vs. CCG non-core</td>
<td>2.46</td>
<td>2.17</td>
<td>1.92</td>
<td>6.25 (1/16)</td>
</tr>
<tr>
<td>- eventive nouns</td>
<td>15.03</td>
<td>17.47</td>
<td>19.02</td>
<td>18.75 (3/16)</td>
</tr>
<tr>
<td>- coordination scope</td>
<td>13.15</td>
<td>16.14</td>
<td>7.82</td>
<td>0.00 (0/16)</td>
</tr>
<tr>
<td>- non-projective</td>
<td>6.82</td>
<td>3.61</td>
<td>6.44</td>
<td>6.25 (1/16)</td>
</tr>
<tr>
<td>- inverted dependency</td>
<td>6.21</td>
<td>5.42</td>
<td>7.67</td>
<td>12.50 (2/16)</td>
</tr>
<tr>
<td>- phrase head mismatch</td>
<td>5.27</td>
<td>6.63</td>
<td>5.44</td>
<td>12.50 (2/16)</td>
</tr>
<tr>
<td>- negation attachment</td>
<td>2.13</td>
<td>2.29</td>
<td>0.92</td>
<td>0.00 (0/16)</td>
</tr>
<tr>
<td>- other</td>
<td>29.18</td>
<td>28.44</td>
<td>29.07</td>
<td>31.25 (5/16)</td>
</tr>
</tbody>
</table>

This table breaks discordant edges into categories based on heuristic types. (1) prepositional refers to mismatches associated with a preposition—with various sub-types denoting discordant (1a) PP attachment, (1b) & (1d) core vs. non-core, (1c) object of preposition—(2) Eventive nouns are AMR edges aligned to a noun with no matching CCG dependency, (3) coordination scope are edge to/from a coordinating conjunction where the scope differs between AMR and CCG resulting in a mismatching dependency, (4) non-projective refers to edges whose dependent is outside of the contiguous scope of which the predicate is the head (creating a non-projective dependency), (5) inverted dependency refers to edges where a corresponding dependency exists, but its direction is reversed, (6) phrase head mismatch refers to edges where the dependent constituent is the same in AMR and CCG but where a different phrase head is assigned in AMR than CCG resulting in a discordant dependency, and (7) negation attachment refers to edges assigning negative polarity which attach to dependents in AMR than CCG (such as attaching to a modal instead of a main verb). $S$ is the number of sentences and $N$ is the number of (AMR) dependencies.
Finally, Table 7.7 shows concordance recall for various token distances to demonstrate the effect of long-distance dependencies on concordance between AMR and CCG. It is generally true that long-distance dependencies increase the chances of model error, but it can be seen from this table that even the gold datasets show a trend that long-distance dependence show more discordant structure.

**Table 7.7: Concordant structure by token distance.**

<table>
<thead>
<tr>
<th>Distance</th>
<th>automatic</th>
<th>gold alignments</th>
<th>gold syntax</th>
<th>gold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S = 49,867$</td>
<td>$S = 324$</td>
<td>$S = 351$</td>
<td>$S = 6$</td>
</tr>
<tr>
<td></td>
<td>$N = 457,816$</td>
<td>$N = 2,646$</td>
<td>$N = 3,773$</td>
<td>$N = 58$</td>
</tr>
<tr>
<td>1</td>
<td>76.21</td>
<td>81.10</td>
<td>71.53</td>
<td>84.62 (22/26)</td>
</tr>
<tr>
<td>2</td>
<td>71.65</td>
<td>77.48</td>
<td>72.17</td>
<td>64.29 (9/14)</td>
</tr>
<tr>
<td>3</td>
<td>62.70</td>
<td>63.93</td>
<td>67.95</td>
<td>50.00 (3/6)</td>
</tr>
<tr>
<td>4</td>
<td>50.88</td>
<td>59.42</td>
<td>52.11</td>
<td>66.67 (2/3)</td>
</tr>
<tr>
<td>5-9</td>
<td>33.25</td>
<td>43.72</td>
<td>37.98</td>
<td>42.86 (3/7)</td>
</tr>
<tr>
<td>10+</td>
<td>17.88</td>
<td>32.67</td>
<td>26.29</td>
<td>50.00 (1/2)</td>
</tr>
</tbody>
</table>

This table illustrates the effect of distance between tokens on the likelihood of AMR and CCG sharing the same structure. Note that long-distance dependencies are more difficult to predict and can correlate with parser error, however even the gold data demonstrates the effect of a correlation between token distance and discordant structure to some degree. $S$ is the number of sentences and $N$ is the number of (AMR) dependencies.

### 7.7.2 Lexical Analysis

Table 7.8 shows the percentage of alignments such that the AMR argument structure matches the CCG argument structure.

We are also interested in the prevalence of our supertag templates in AMR versus CCG. Table 7.9 shows the three most common supertag templates in our AMR data for several parts of speech and their frequencies in AMR and CCG.
For a given alignment, the AMR and CCG argument structures are considered to match if they have the same number of left-appearing arguments and the same number of right-appearing arguments. Spans are considered a single unit with internal dependencies ignored. We do not include aligned punctuation or null-aligned spans in our analysis. \( S \) is the number of sentences and \( N \) is the number of alignments.

Arguments whose dependencies are removed in pre-processing are colored gray.
Table 7.10 shows the concordance recall for reentrancies. Given the lower performance for automatic reentrancy alignments, the gold alignments dataset is likely the best indicator of concordance across reentrancy categories. Certain reentrancy types—such as control, adjunct control, and coordination—correspond to syntactic phenomena meant to be captured by CCG, while others—such as coref, repetition, and pragmatic—are non-syntactic and are not expected to be captured by CCG. Nevertheless, we can see that concordance recall for control (44%), adjunct control (28%), and coordination (54%), while higher than other reentrancy types are still quite low compared to the concordance recall for normal dependencies.

### Table 7.10: Concordance for reentrancies.

<table>
<thead>
<tr>
<th>Recall by AMR Reentrancy (%)</th>
<th>automatic</th>
<th>gold alignments</th>
<th>gold syntax</th>
<th>gold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S = 49,867$</td>
<td>$S = 324$</td>
<td>$S = 351$</td>
<td>$S = 6$</td>
</tr>
<tr>
<td></td>
<td>$N = 97,476$</td>
<td>$N = 521$</td>
<td>$N = 779$</td>
<td>$N = 7$</td>
</tr>
<tr>
<td>all</td>
<td>40.55</td>
<td>44.53</td>
<td>40.31</td>
<td>71.43 (5/7)</td>
</tr>
<tr>
<td>- primary (40%)</td>
<td>66.26</td>
<td>70.72</td>
<td>67.21</td>
<td>66.67 (2/3)</td>
</tr>
<tr>
<td>- coref (23%)</td>
<td>10.95</td>
<td>6.25</td>
<td>11.61</td>
<td>50.00 (1/2)</td>
</tr>
<tr>
<td>- pragmatic (12%)</td>
<td>26.55</td>
<td>9.09</td>
<td>21.05</td>
<td>- (0/0)</td>
</tr>
<tr>
<td>- control (10%)</td>
<td>36.99</td>
<td>44.44</td>
<td>24.21</td>
<td>100.00 (2/2)</td>
</tr>
<tr>
<td>- coordination (8%)</td>
<td>41.00</td>
<td>54.00</td>
<td>46.43</td>
<td>- (0/0)</td>
</tr>
<tr>
<td>- repetition (3%)</td>
<td>5.17</td>
<td>0.00</td>
<td>30.00</td>
<td>- (0/0)</td>
</tr>
<tr>
<td>- unmarked adjunct control (3%)</td>
<td>29.49</td>
<td>25.00</td>
<td>39.13</td>
<td>- (0/0)</td>
</tr>
<tr>
<td>- comparative (1%)</td>
<td>36.22</td>
<td>41.67</td>
<td>12.50</td>
<td>- (0/0)</td>
</tr>
<tr>
<td>- adjunct control (0.2%)</td>
<td>54.17</td>
<td>27.78</td>
<td>-</td>
<td>- (0/0)</td>
</tr>
</tbody>
</table>

This table shows the percentage of concordant edges (edges which correspond to a CCG dependency) for AMR reentrancies. Not surprisingly, CCG dependencies capture some types of reentrancy relations—such as control and coordination—but not others. The reentrancy types coreference, repetition, and pragmatic are not expected to correspond to syntactic relationships at all. $S$ is the number of sentences and $N$ is the number of AMR reentrancies (each of which corresponds to a dependency).

### MWE Errors.  
We generally find that the longer a multi-token span is, the less likely it is to correspond to a connected component of the CCG dependency structure, making it difficult to associate large spans aligned in AMR with a particular set of syntactic dependencies in CCG.
Table 7.11: Compositionality of multi-token spans.

<table>
<thead>
<tr>
<th></th>
<th>automatic</th>
<th>gold alignments</th>
<th>gold syntax</th>
<th>gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S = 49,867 )</td>
<td>59.28</td>
<td>60.44</td>
<td>72.90</td>
<td>71.43 (5/7)</td>
</tr>
<tr>
<td>( N = 97,476 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 2 tokens (74%)</td>
<td>77.26</td>
<td>83.65</td>
<td>85.28</td>
<td>83.33 (5/6)</td>
</tr>
<tr>
<td>- 3 tokens (17%)</td>
<td>10.02</td>
<td>5.71</td>
<td>0.00</td>
<td>0.00 (0/1)</td>
</tr>
<tr>
<td>- 4 tokens (5%)</td>
<td>2.20</td>
<td>7.14</td>
<td>0.00</td>
<td>- (0/0)</td>
</tr>
</tbody>
</table>

\( S \) is the number of sentences and \( N \) is the number of spans with more than 2 tokens.

Table 7.11 shows the percentages of multi-token spans that form a connected graph in terms of CCG dependencies. Our expectation is that each span should correspond to a constituent in CCG and should thus form a connected subgraph of the CCG dependencies graph. When this is true, it allows us to interpret multi-token spans as still being syntactically (though not semantically) compositional. What we find is that spans with 3 or more tokens generally do not have this property. Many of the disconnected spans result from the way we identified multi-token spans for AMR. We identified a sequence of tokens as an alignable span if it was contained in a list of multiword expressions known to have non-compositional meaning. Many of these MWEs include function words whose syntactic relationship with a head word is not guaranteed. For example the phrase “according to” is annotated as a multiword expression in our data but its CCG dependencies do not form a connected subgraph.

7.8 Future Work

One major issue highlighted by the analyses in this chapter is that while CCG is a flexible grammar formalism, CCGBank, the main dataset of CCG parses, was created with particular design decisions, and those design decisions differ in many dimensions form the design decisions of AMR. CCGBank offers a dialect of CCG with its own interpretations of what counts as a core role, how noun-noun compounds are handled, etc. We believe that developing an AMR-friendly dialect of CCG, along with a supertagger for predicting that dialect, will be an important area of future work for our goal of adding compositionality to AMR. An AMR-friendly dialect of CCG would need to have core roles...
based on PropBank frames, would need to properly identify and handle eventive nouns, and may need to have better analysis of coordination than current CCG parsers. The CCG-AMR formalism presented in Ch. 6 and the challenges identified in this chapter provide a good theoretical basis for what such a dialect should look like. The next steps would be developing a CCG supertagger designed for AMR-friendly output. One avenue of research could be augmenting an existing supertagger with conversion rules and additional classification decisions to produce the desired output. Another avenue could be developing a dataset of AMR-friendly CCG data and tuning an existing supertagger on that new data.

7.9 CONCLUSIONS

To create a linguistically interpretable approach to semantic parsing, we need to understand how AMR can be broken apart into form meaning pairs and how those form meaning pairs can be composed. Further, the process and order of composition cannot be completely arbitrary. To be linguistically interpretable, the composition must follow the same syntactic constraints that we see in natural language. In this chapter, I have offered a detailed analysis of the extent to which AMR and CCGBank data share the same structure, and explanations of when and why they do not. While some of the dissimilarity of structure may be due to particularities of CCG, we can see other cases where AMR differs structurally from what linguists generally interpret as the structure of natural language, at least at a syntactic level. I have identified contexts where AMR parsing may be difficult and present unique challenges due to its dissimilarity with syntax, and I have illustrated cases where the complexity of AMR semantics might be resolved with task-specific approaches. The analysis in this chapter also motivates a need for a more AMR-friendly dialect of syntax which is capable of handling more complex argument structures in a flexible way. I have detailed in this chapter subtasks of AMR derivation which are particularly complex in order to motivate avenues for improvement for future semantic parsing research.
In this dissertation, I have taken up a goal of augmenting a scalable semantic graph representation to make it linguistically interpretable. I have presented a novel formulation of AMR alignment, which is comprehensive with respect to AMR substructures and whose alignments correspond to wide variety of linguistic phenomena including argument structures, non-core roles, coreference, control, and ellipsis. I have designed and released a corpus of automatically generated alignments for English AMR data as well as several hundred manually annotated sentences for tuning and evaluation. I implemented a probabilistic, structure-aware alignment algorithm to automatically align English sentences to AMRs without supervision, with higher coverage, accuracy, and variety than alignments from existing AMR aligners. I designed a formulation of AMR as graph semantics in CCG and accompanying combinatorial rules of CCG for deriving a full AMR graph in an interpretable way. Lastly, I conducted an empirical analysis of the compatibility and structural similarity/dissimilarity of AMR with automatically generated CCG parse data, ultimately demonstrating the complexity of integrating AMR with a syntactic formalism and the need for AMR-friendly dialects of syntax. I present this work as a foundation for linguistically interpretable and explicit graph semantic representations for improving interpretability of models in future NLP research.

8.1 Lessons for Future AMR Research

The overarching goal of this work has been investigating and enhancing the interpretability that AMR adds to natural language processing tasks. One insight demonstrated by this dissertation is that there are dimensions of AMR which deserve to be explored further in future research and need to be analyzed and derived in a more explicit way if AMR is going to serve as a tool for
interpretability. In particular, relations, reentrancies, and ellipsis fit this description. This perspective is quite different from the perspective of most state-of-the-art AMR parsers which tend to rely on end-to-end models leveraging transfer learning and work extremely well, but without the benefit of interpretability. Another insight demonstrated in this dissertation is that there are challenges for AMR parsing, especially syntax-based and linguistically interpretable AMR parsing, that may warrant more attention. In particular, eventive nouns, prepositional phrases, and coordination all pose difficult challenges and evaluating AMR parsers in these categories may be illuminating. Lastly, a valuable next step for the goal of incorporating syntactic knowledge into AMR and AMR parsing will be developing varieties of syntax which are flexible towards and conscientious of design decisions of AMR.

8.2 Future Work

There are a number of avenues of research which are well-motivated by this dissertation and would be worthwhile to pursue in future work. I group these into several categories based on the goals involved. First, for my long-term goal of incorporate analyzability and compositionality into AMR parsing, I plan to:

1. Incorporate LEAMR alignments as defined in Ch. 4 into an existing AMR parser.
2. Tune an existing CCG supertagger to predict AMR-friendly supertags to be integrated with AMR semantics.
3. Incorporate AMR-friendly CCG syntax into an existing AMR parser, either as a process of CCG derivation or as syntactic features.

This research would be enabled by the work presented in this dissertation: the LEAMR alignments, the AMR-CCG formalism given in Ch. 6, and the analyses and identification of challenges given in Ch. 7. A linguistically interpretable AMR parser could then be relied on by NLP applications to encode semantic knowledge in a such way that linguistic decisions are transparent at each step. Second, I plan to investigate some of the challenging problems for interpretable AMR parsing that I identified in Ch. 7 and how well transfer learning can address them. I will:
5. Conduct a “BERTology” investigation of identified AMR parsing subtasks for coordination, event nouns, and prepositions. This would involve training a simple classifier over a pre-trained language model to see how well it can handle these tasks.

This research would be enabled by the LEAMR alignments presented in Ch. 4 and the analyses presented in Ch. 7 where I identified challenging problems for compositionally deriving AMRs. It would also reveal the strengths and limitations of transfer learning techniques that currently dominate state-of-the-art AMR parsing. Lastly, I plan to experiment with AMR as a tool for adding interpretability and inductive biases to an NLP system. I will:

6. Incorporate AMR parsing sub-tasks such as lexical prediction as auxiliary tasks for an NLP system in a multitask setting.

This avenue of research is enabled by the LEAMR alignments presented in Ch. 4 which make it possible to break up AMR prediction into linguistically meaningful subtasks, each of which might be useful to any of a number of NLP applications which require semantic knowledge or reasoning.
Alignments in this scheme are designed to be mutually exclusive and comprehensive alignments between connected AMR subgraphs and English token spans. This alignment scheme also includes several layers of alignments with primary alignments for nodes and edges and secondary alignments for duplicate subgraphs and reentrancies. Table A.1 shows the alignments for the AMR in Figure A.1. Readers can see how to annotate these alignments by hand in Table A.2. Annotation files use a tsv format with columns (1) node or edge id, (2) readable label, (3) aligned span, (4) additional labels. The first and second columns are automatically generated. Annotators fill in the remaining two columns with the correct aligned span and special labels in the case of duplicate subgraph and reentrancy alignments.

(j / join-01
 :ARG0 (p / person
  :name (p2 / name :op1 (x0 / "Pierre") :op2 (x1 / "Vinken")))
  :age (t / temporal-quantity :quant x3 / 61
   :unit (y / year)))
 :ARG1 (b / board
  :ARG1-of (h / have-org-role-91
   :ARG0 p
   :ARG2 (d2 / director
    :mod (e / executive :polarity (x4 / -))))))
 :time (d / date-entity :month (x5 / 11) :day (x6 / 29)))

Figure A.1: AMR for the sentence “Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.”
Table A.1: Alignments for the sentence “Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.”

<table>
<thead>
<tr>
<th>Primary Alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pierre Vinken</td>
</tr>
<tr>
<td>61 years old</td>
</tr>
<tr>
<td>will</td>
</tr>
<tr>
<td>join</td>
</tr>
<tr>
<td>the</td>
</tr>
<tr>
<td>board</td>
</tr>
<tr>
<td>as</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>nonexecutive</td>
</tr>
<tr>
<td>director</td>
</tr>
<tr>
<td>Nov. 29</td>
</tr>
<tr>
<td>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Secondary Alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>director</td>
</tr>
</tbody>
</table>

A.1 General Principles

- Comprehensive/Mutually Exclusive: Every node & edge must be aligned to exactly one token span
- When it is completely ambiguous which of two spans to align to, you can align to whichever appears first in the sentence. For example, when a name is repeated several times in the sentence, you may align to the first mention.
- Connected: nodes aligned to a span must form a connected subgraph with two exceptions:
  - Duplicate Subgraphs: A span may be aligned to multiple subgraphs if one subgraph is a duplicate of the others. This is often necessary when dealing with ellipsis where there is more semantic content in the AMR than is pronounced in the sentence and thus several identical parts of the AMR must be aligned to the same span. See the section below on how to align duplicate subgraphs.
Table A.2: Alignment annotation for the sentence “Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.”

<table>
<thead>
<tr>
<th>#Node alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
</tr>
<tr>
<td>p</td>
</tr>
<tr>
<td>p2</td>
</tr>
<tr>
<td>x0</td>
</tr>
<tr>
<td>x1</td>
</tr>
<tr>
<td>t</td>
</tr>
<tr>
<td>x3</td>
</tr>
<tr>
<td>y</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>d2</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>x4</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>x5</td>
</tr>
<tr>
<td>x6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#Edge alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>j :ARG0 p</td>
</tr>
<tr>
<td>p :name p2</td>
</tr>
<tr>
<td>p2 :op1 x0</td>
</tr>
<tr>
<td>p2 :op2 x1</td>
</tr>
<tr>
<td>p :age t</td>
</tr>
<tr>
<td>t :quant x3</td>
</tr>
<tr>
<td>t :unit y</td>
</tr>
<tr>
<td>j :ARG1 b</td>
</tr>
<tr>
<td>b :ARG1-of h</td>
</tr>
<tr>
<td>h :ARG0 p</td>
</tr>
<tr>
<td>h :ARG2 d2</td>
</tr>
<tr>
<td>d2 :mod e</td>
</tr>
<tr>
<td>e :polarity x4</td>
</tr>
<tr>
<td>j :time d</td>
</tr>
<tr>
<td>d :month x5</td>
</tr>
<tr>
<td>d :day x6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#Reentrancy alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>h :ARG0 p</td>
</tr>
</tbody>
</table>

Annotation files are stored in tsv format with 4 columns: (1) node or edge id, (2) readable label, (3) aligned span, (4) additional labels.
– Cases like never: A span may be aligned to two terminal nodes that have the same parent. For example, never will generally align to :polarity - :time ever, two nodes and two edges which share the same parent.

A.2 SPANS

• Spans are chunks of one or more tokens that are used to identify named entities or multiword expressions as a single unit. All alignments need to specify a span which must be a contiguous list of token indices (e.g., “5,6,7” but not “4,8”). In this schema, spans must be mutually exclusive and cannot be nested (Though that is allowed by other researchers, for example, in Named Entity Recognition).

• Named Entities: Names such as John Smith or Democratic Republic of the Congo will be aligned as a span. Any nodes or edges, etc. aligned to these spans must specify the token index of every token in the span.

• Multiword Expressions: When several tokens form a multiword expression where the words together form a specific idiomatic meaning, possibly unrelated to the tokens’ individual meanings, we annotate these cases as a span. For example, in order to and look after are multiword expressions and will be treated as spans. In cases where a multiword expression is discontiguous (e.g. “think the plan over”), align only to the first contiguous part of the expression (i.e. think).

• Dates: When aligning date and time expressions, the span should include relevant date and time entities (e.g., “September 9th, 2021”) and cardinal or ordinal numbers (e.g., “3 years”), but should not include modifiers, prepositions, or postpositions at the beginning or end of the expression (e.g. “earlier this year”, “one week ago”). If an expression is an interval, both date expressions should be included (e.g. “5 to 7”, “September to November”). If the date expression includes a multiword expression, make sure this is included (e.g. “10 years old”).

131
• Quantities & Money: For currency or other quantities any units or cardinal numbers should be included (e.g. “$ 30 billion”) but no modifiers, prepositions, or postpositions at the beginning or end of the expression (e.g., “more than 30 kilos”).

A.3 NODES

It is usually intuitive which node and which span should be aligned. Here are a few notes for resolving ambiguous cases.

• Named entities: named entities will generally be aligned to several nodes and edges including a node specifying the type of entity (e.g. New York => city, name, “New”, “York”). A rare exception is when the entity type is pronounced by a separate token in the sentence such as “the company IBM” where the alignment must be company => company and IBM => name, “International”, “Business”, “Machines”. The :name edge is then aligned with IBM.

• Punctuation: Alignments to punctuation may be rare but are necessary in a few notable cases:
  – and nodes can be aligned to comma (,) or semicolon (;) if there is no other span to align to.
  – multi-sentence nodes will usually be aligned to a period (.), semi-colon (;), or other punctuation introducing a new sentence.
  – :mode interrogative will usually be aligned to a question mark (?).
  – Rarely, mean-01 must be aligned to a colon (:) or other punctuation.

• AMR specific concepts: some nodes in AMR are notational and don’t correspond to a specific linguistic frame or entity (e.g., multi-sentence, include-91, have-org-role-91). Often these may be aligned as a part of a larger subgraph, or in some cases to an unaligned token or punctuation. Use your best judgement to determine which span is more indicative of the concept. I.e., which span, if you saw it, would most accurately be predictive of the concepts presence in the gold AMR.

• Comparative constructions: comparative expressions like bigger, more than, less, most, too, as...as may be aligned to the concept have-degree-91. This is a fairly complex frame
which often takes multiple arguments and triggers a reentrancy. Given the large number of constructions it can occur with, it is important to align it correctly. If the sentence has a discontiguous multiword expression such as \textit{as...as} or \textit{more...than}, align to the first span to appear in the sentence.

A.4 Edges

- \texttt{:ARGX} edges should always be aligned to the same span as the parent (\texttt{:ARGX-of} edges should always be aligned to the child)
- \texttt{:opX} edges should always be aligned with the parent.
- \texttt{:sntX} edges should always be aligned with the parent.
- \texttt{:domain} edges should always be aligned with the parent. (Don’t align these edges to copula tokens as it tends to make parsing difficult.)
- \texttt{:poss} and \texttt{:part} edges may be aligned to a token (i.e. 's or of) or with the parent.
- \texttt{:name} edges should always be aligned with the child. (Usually, the child and parent will be aligned to the same span.)
- Other edges may be aligned to the same span as the child or to an unaligned span such as a preposition or subordinating conjunction. (e.g., \texttt{:location} => \textit{in} or \texttt{:time} => \textit{at})

A.5 Duplicate Subgraphs

\begin{verbatim}
(w/work-09
 :ARG1 (i/it)
 :ARG1-of (r/resemble-01
 :ARG2 (w2/work-09
 :ARG1 (p/program
 :name (n/name
 :op1 "Medicare")))))
\end{verbatim}

\textbf{Figure A.2: AMR for “It would work like medicare ...”}.
In Figure A.2, there are two work-09 concepts in the AMR but only one pronounced work token in the sentence. We can blame this on ellipsis. Under that hypothesis, the sentence could alternatively be pronounced “It would work similarly to how medicare works.” Since we have to annotate the sentences we are given, we handle these situations in a secondary layer of alignments for duplicate subgraphs. We will say that w2/work-09 is a duplicate of w/work-09, that the primary alignments will include the alignment work => w/work-09, but there will also be a secondary alignment in the duplicate subgraphs layer that looks like work => w2/work-09. This allows us to align every node between the two layers and it allows us to specify that the second alignment is the result of ellipses or some other linguistic phenomena. As a matter of convention, the first node or subgraph (in the linearized AMR) gets the primary alignment and any duplicates are given secondary alignments. When annotating, you can annotate duplicate subgraph alignments using an asterisk (*) before the token indices. (In the case of more than one duplicate, you can use two asterisks (**), etc.)

Table A.3 shows an example annotation:

**Table A.3: Annotation of duplicate subgraphs in “It would work like medicare . . .”**

<table>
<thead>
<tr>
<th>#Node alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#Edge alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>w :ARG1 i</td>
</tr>
<tr>
<td>w2 :ARG1 p</td>
</tr>
</tbody>
</table>

### A.6 Reentrancies

In addition to all edges being aligned, any edge which is a reentrancy (an edge which re-uses a previously named concept as an argument) must be aligned to a span which signals the reentrancies. We can identify reentrancies by looking at any edges where the target has multiple parents. There are several linguistic reasons an AMR might have a reentrancy, so reentrancies are also annotated with a reentrancy type. Below is an example:
Figure A.3: AMR for “I actually had some other classmates there, and was going to call them”

Table A.4: Annotation of reentrancy alignments for the sentence “I actually had some other classmates there, and was going to call them.”

<table>
<thead>
<tr>
<th>#Reentrancy alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>h :ARG0 i have-03 :ARG0 i _</td>
</tr>
<tr>
<td>h :ARG1 c have-03 :ARG1 classmate _</td>
</tr>
<tr>
<td>c2 :ARG0 i call-02 :ARG0 i 8 coordination</td>
</tr>
<tr>
<td>c2 :ARG1 c call-02 :ARG1 classmate 13 coref</td>
</tr>
</tbody>
</table>

Note that above in Figure A.3 and Table A.4, any edges where the target has multiple parents gets listed row by row. For each target one edge is annotated with an underscore (_) marking that as the primary parent. Any other reentrancies are aligned to a token span which is responsible for triggering the reentrancy and annotated with the reentrancy type from the list in Table A.5:

Above, one reentrancy call-02 :ARG0 i is aligned to *and* which triggers the reentrancy by coordination. The other reentrancy call-02 :ARG1 classmate is aligned to *them* which triggers a reentrancy through coreference.
Table A.5: Explanation of reentrancy types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COREF</td>
<td>Coreference triggered by a pronoun (including possessive or reflexive)</td>
</tr>
<tr>
<td>REPETITION</td>
<td>Coreference triggered by a repeated name or non-pronominal phrase</td>
</tr>
<tr>
<td>COORDINATION</td>
<td>Reentrancies triggered by COORDINATION of two or more phrases sharing an argument</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Reentrancies triggered by control verbs, control nouns, or control adjectives.</td>
</tr>
<tr>
<td>ADJUNCT CONTROL</td>
<td>Reentrancies triggered by control within an adjunct phrase</td>
</tr>
<tr>
<td>UNMARKED ADJUNCT CONTROL</td>
<td>Reentrancies triggered by control within an adjunct phrase with only a bare verb and no subordinating conjunction</td>
</tr>
<tr>
<td>COMPARATIVE CONTROL</td>
<td>Reentrancies triggered by a comparative construction</td>
</tr>
<tr>
<td>PRAGMATIC</td>
<td>Reentrancies that must be resolved using context</td>
</tr>
</tbody>
</table>

A.6.1 Coreference

1. Pronominal: Reentrancies triggered by a pronoun, including possessive and reflexive pronouns, are aligned to the pronoun and annotated with the label COREF.

2. Repetition: Reentrancies triggered by a named entity or other coreferential phrase being repeated in the sentence are aligned to the repeated phrase and annotated with the label REPETITION. (This includes cases where the phrase is not exactly the same but is coreferential. E.g., “Chinese . . . China . . . ” or “United States . . . U.S. . . . ”)

A.6.2 Coordination

In a sentence with VP coordination such as “The people cheered and celebrated.”, the same argument (or arguments) is shared by multiple verbs. This can happen not just for verb phrases but for any phrase which takes one or more semantic arguments. In cases where a reentrancy occurs because of coordination, the reentrancy is aligned to the coordinating conjunction (usually and) and annotated with the label COORDINATION. You also need to decide which edge is the primary parent edge and
which ones are annotated as reentrancies. By convention, given coordination of the form “A and B”, and edges out of A are annotated as primary edges and the edges out of B are treated as reentrancies.

A.6.3 Control

In sentences with control verbs, control adjectives, or control nouns, a reentrancy is triggered by a grandparent of the reentrancy’s target node. Control is a lexical property of some word or phrase where an argument of the control phrase also becomes an argument of a controlled phrase. For example, in the sentence “John was afraid to speak up”, John is a semantic argument of both afraid and speak up, and so the corresponding AMR will have a reentrancy corresponding to the relationship between John and speak up. Afraid is the control phrase and speak up is the controlled phrase which is an argument of afraid. Both the control phrase and controlled phrase take John as an argument. Since control is a lexical property of afraid and other control phrases, we align control reentrancies with the control phrase. So the edge between John and afraid (fear-01 :ARG0 person) is annotated as the primary edge and the edge between John and speak up (speak-up-02 :ARG0 person) is annotated as a reentrancy triggered by afraid. Reentrancies triggered by a control phrase are annotated with the label CONTROL. The AMR is shown in Figure A.4 and an annotation is shown in Table A.6.
### A.6.4 Adjunct Control

Several types of control do not fit the pattern described above. Most of these fall into the category of adjunct control. Reentrancies triggered by adjunct control are annotated with the label **adjunct control**. **Infinitive Purpose Clauses:** An infinitive purpose clause is a clause like “I went to the store to buy some milk” where “to buy some milk” is an adjunct clause specifying the purpose of going to the store. Infinitive purpose clauses take the infinitive form of a verb, hence the name. Reentrancies triggered by this type of clause should be aligned to the infinitival to token. **Subordinate Gerund Clauses:** Another type of adjunct control can be seen in “Mary did her homework while listening to music”. In this and similar cases, adjunct control is introduced by subordinating conjunction (such as after, while, etc.) followed by a verb in gerund form. In these cases, the reentrancy should be aligned to the subordinating conjunction. More rarely, you may see a sentence like “Mary did her homework, listening to music”. This is like a subordinate gerund clause, but no subordinating conjunction is present. In cases like this, the reentrancy should be aligned to the gerund verb and you should annotate it with the special label unmarked adjunct control.

### A.6.5 Comparative Control

Comparative expressions like bigger, more, less, most, too, as... as may be aligned to the concept have-degree-91 which often includes a reentrancy to the attribute whose degree is being described. This type of reentrancy gets the label **comparative control**. The first mention of the attribute in the linearized AMR is annotated as the primary edge, while the second attribute is annotated as a
For example, the reentrancy pointing to the node objective in Figure A.5 should be aligned to as (the first token in as...as construction above).

A.6.6 PRAGMATIC

In many cases, there is not a clear way to infer a reentrancy or how it should be resolved using either syntax or coreference. In these cases, the reentrancy is aligned to the same span as the edge itself is aligned and is annotated with the label PRAGMATIC. These often come up when a frame has an argument which is obvious to a speaker because of world knowledge but which is not otherwise easy to resolve. Reentrancies of this type are labelled pragmatic.

For example, the reentrancy pointing to the node earthquake in Figure A.6 should be aligned to precursory. The intuition is that the word precursory has an implicit semantic argument, but this semantic argument is not resolved by normal processes like syntactic composition or coreference. Instead, a natural English speaker uses their general knowledge to pick earthquake as the most likely candidate. The reentrancy should be aligned to the span with an implicit argument, which in

Figure A.5: AMR for “The key is to be as objective as possible.”

Figure A.6: AMR for “Precursory signs of the earthquake.”
this case is *precursory*. Another example:

```
(m/meet-up-04
 :ARG0 (p/person
      :name (n/name
           :op1 "John"))
 :ARG1 (p2/person
      :ARG0-of (h/have-rel-role-91
           :ARG1 p
           :ARG2 (f/friend)))
```

**Figure A.7: AMR for “John met up with a friend.”**

In Figure A.7, there is an implicit argument of *friend* answering the question “a friend of whom”. This is represented in the AMR above by a reentrancy to p/person. From context, it is obvious that “a friend” refers to a friend of John’s, but inference must be resolved by pragmatic knowledge since there is no syntactic or coreferential way to resolve it. Like in the previous example, the reentrancy should be aligned to the span with an implicit argument, which in this case is *friend*. 

140
APPENDIX B

DETAILS OF RULES AND PRE/POST-PROCESSING FOR AMR ALIGNMENT

B.1 IDENTIFYING SPANS

As a preprocessing step, sentence have their tokens grouped into spans based on three criteria, outlined in detail below:

1. Named entity spans identified by Stanza.
2. Spans matching multiword expressions from a fixed list of ≈1600
   (a) 143 prepositional MWEs from STREUSLE (Schneider and Smith, 2015)
   (b) 348 verbal MWEs from STREUSLE
   (c) 1095 MWEs taken from gold AMRs in LDC train data (any concept which is a hyphenated compound of multiple words, e.g., \textit{alma-mater} or \textit{white-collar}) and are not present in the above lists.
   (d) ≈12 hand-added MWEs
3. Any sequence of tokens which is an exact match to a name in the gold AMR (e.g., United Kingdom matches \{(n/name :op1 "United" :op2 "Kingdom")\}) is also treated as a span.
B.2 Rule-based Subgraph Alignment Preprocessing

B.2.1 Token Matching

We use three phases of rule-based alignment which attempt to align particular spans to particular AMR subgraphs:

1. Exact token matching: If there is a unique full string correspondence between a span and a name or number in the AMR, they are aligned.
2. Exact lemma matching: If there is a unique correspondence between an AMR concept and the lemma of a span (which in the case of a multiword span is the sequence of lemmas of the tokens joined by hyphens), they are aligned.
3. Prefix token matching: A span with a prefix match of length 6, 5, or 4 is aligned if it uniquely corresponds to an AMR named entity.
4. Prefix lemma matching: A span with a prefix match of length 6, 5, or 4 of its lemma is aligned if it uniquely corresponds to an concept.
5. English rules: Several hand-written rules for matching English strings to specific subgraphs are used to match constructions such as dates, currency, and some frequent AMR concepts with many different ways of being expressed, such as and and -.

- Parsing Dates, and Times
- Numbers written out (e.g., one, two, thousand, etc.)
- Currencies (e.g., $, €, etc.)
- Decades (e.g., twenties, nineties)
- and (matching and, additionally, as well, etc.)
- multi-sentence (matching punctuation)
- :polarity - (matching not, none, never, etc.)
- cause-01 (matching thus, since, because, etc.)
- amr-unknown (matching ?, who, when, etc.)
- person (matching people)
B.2.2 Graph Rules

We also perform preprocessing to expand a subgraph alignment to include some neighboring nodes. These fall into two main categories:

1. Some AMR concepts are primarily notational rather than linguistic and should be aligned together with a neighboring node. For example named entities (e.g., \((\text{country :name } (\text{n/name :op1 "United" :op2 "Kingdom"}))\)) are aligned as a unit rather than one node at a time. Likewise, date entities, and subgraphs matching \((\text{x/X-quantity :unit X :quant X})\) or \((\text{x/X-entity :value X})\) are also aligned as a unit.

2. Neighboring nodes which are associated with morphological information of the aligned span (e.g., \((\text{biggest } \rightarrow (\text{have-degree-91 :ARG1 big :ARG2 most}))\)) are added to the alignment using a series of rules for identifying comparatives, superlatives, polarity, and suffixes such as -er or -able, etc.

B.3 Rule-based Relation Alignment Preprocessing

Many of the relations are forced to be aligned in a particular way as a matter of convention. We use a similar approach to that of (Groschwitz et al., 2018).

1. \(:\text{ARGX} \) edges are automatically aligned to the same span as the parent (\(:\text{ARGX-of} \) edges are automatically aligned to the child)
2. :opX edges are automatically aligned with the parent.
3. :sntX edges are automatically aligned with the parent.
4. :domain edges are automatically aligned with the parent. (We don’t align these edges to copula. Instead, a concept with a :domain edge is thought of as a predicate which takes one argument.)
5. :name, :polarity, and :li edges are automatically aligned with the child.

B.3.1 Token Matching

Some relations take the form :prep-X or :conj-X where X is a preposition or conjunction in the sentence. We use exact match to align these relations as a preprocessing step. The relations :poss and :part may be automatically aligned to ’s or of if the correspondence is unique within a sentence.

B.4 Rule-based Reentrancy Alignment Preprocessing

Primary edges are identified as a preprocessing step before aligning reentrancies with the following rules: Any relation which is aligned to the same span as its token (any incoming edge which is a part of a span’s argument structure) is automatically made the primary edge. Otherwise, for each edge pointing to a node, we identify the spans aligned to the parent and child nodes in the subgraph layer. Whichever edge has the shortest distance between the span aligned to the parent and the span aligned to the child is identified as the primary edge. In the event of a tie, the edge whose parent is aligned to the leftmost span is identified as the primary edge. Primary reentrancy edges are always aligned to the same span the edge is aligned to in the relation layer of alignments.
APPENDIX C

ADDITION CCG-AMR DERIVATIONS

Below are full derivations illustrating raising, subject control, object control, an object control wh-question, a modal auxiliary with preposed VP adjunct, a purpose clause, coordinated purpose clauses, and right node raising with a shared main verb.

Figure C.1: Raising example: “Mary seems to practice guitar often.”
Figure C.2: Subject control example: “Mary wants to practice guitar.”

Figure C.3: Object control wh-question example: “Who did you persuade to smile?” (example suggested by a reviewer)
Figure C.4: Modal auxiliary with preposed adjunct: “Tomorrow, John may eat rice.” In the derived AMR, the temporal modifier is placed incorrectly under the modal predicate rather than the main event predicate.

Figure C.5: To-purpose example: “Mary bought a ticket to see the movie.”
Figure C.6: Coordinated purpose clauses: “John arrived to eat and to party.” Note that the PropBank predicate `arrive-01` has no :ARG0; its subject is :ARG1. The lexical semantics for infinitive purpose `to` is chosen accordingly. However, the placement in the derived AMR of the semantic conjunction and is incorrect.
BIBLIOGRAPHY


150


Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In LREC.


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. European Language Resources Association (ELRA).


178


Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020a. AMR-To-Text Generation with Graph Transformer. Transactions of the Association for Computational Linguistics, 8:19–33.


