Semantic parsing

Computational Linguistics

Alexander Koller

09 January 2024

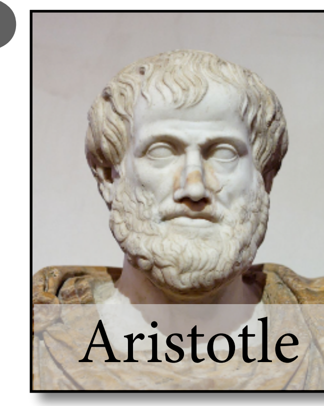
Overview

1. Semantics in computational linguistics
   (up to 2000).

2. Classical semantic parsing
   (Geoquery, early 2000s).

3. Semantic graph parsing:
   - graph-based (JAMR, 2014)
   - compositional (AM dependency parsing, 2019)
   - finetuning of pretrained models (SPRING, 2021)

Computing with meanings



- Ancient problem: *inference.*
  - How can we tell whether a sentence follows from others?
  - Can we compute this automatically?

All men are mortal.

Socrates is a man.

Therefore, Socrates is mortal.

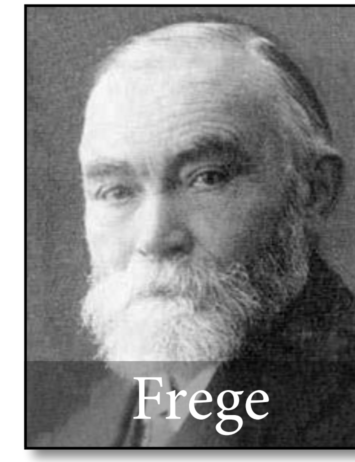
(Syllogism, Aristotle, 350 BC)

At the other end of Pennsylvania Avenue, people began to line up for a White House tour.

People formed a line at the end of Pennsylvania Avenue.

(MNLI dataset, 2018)

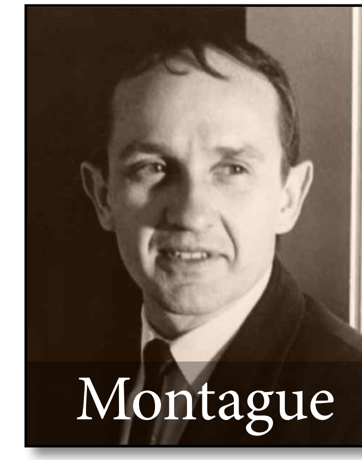
Formal meaning representations



- Aristotle with more modern tools (ca. 2000):
  - Compute *meaning representation* in some formal language (e.g. predicate logic)
  - so that it captures something relevant about the sentence's meaning (e.g. its *truth conditions*)
  - and then use reasoning tools for the formal language (e.g. a *theorem prover* for predicate logic)

All men are mortal.	$\forall x.\ \mathrm{man}(x) \rightarrow \mathrm{mortal}(x)$
Socrates is a man.	man(s)
Therefore, Socrates is mortal.	mortal(s)

Compositional semantics



S → NP VP	$\langle S \rangle = \langle NP \rangle(\langle VP \rangle)$
VP → V NP	$\langle VP \rangle = \lambda y\ \langle NP \rangle(\langle V \rangle(y))$
NP → Det N	$\langle NP \rangle = \langle Det \rangle(\langle N \rangle)$
NP → John	$\langle NP \rangle = \lambda P\ P(j^*)$
V → eats	$\langle V \rangle = eat'$
Det → a	$\langle Det \rangle = \lambda P \lambda Q \exists x\ P(x) \wedge Q(x)$
N → sandwich	$\langle N \rangle = sw'$

when you apply this syntax rule …

… construct λ-term for parent from λ-terms for children like this

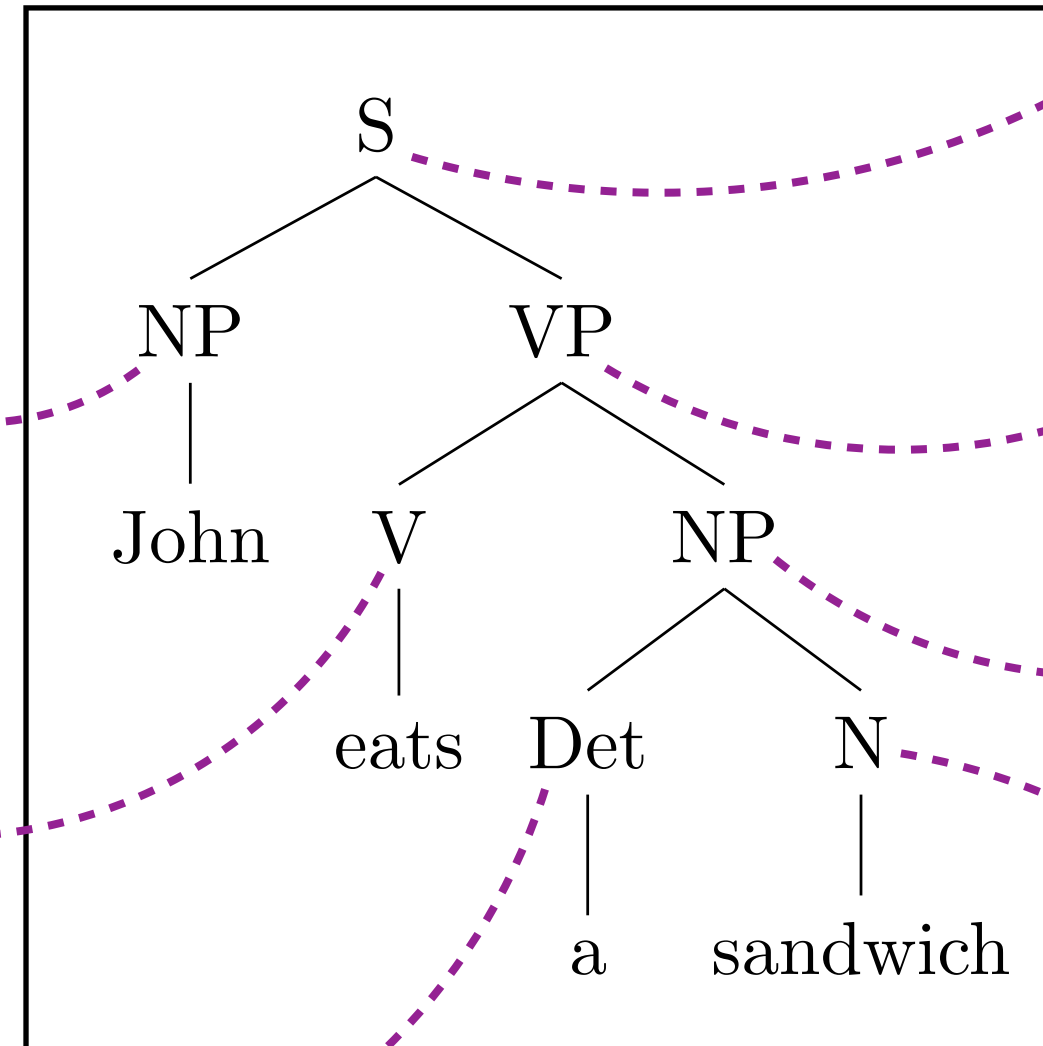
Example

$(\lambda P\ P(j^*))\ (\lambda y\ \exists x\ \text{sandwich'}(x) \wedge \text{eat'}(y)(x))$

$\rightarrow_\beta (\lambda y\ \exists x\ \text{sandwich'}(x) \wedge \text{eat'}(y)(x))(j^*)$

$\rightarrow_\beta \exists x\ \text{sandwich'}(x) \wedge \text{eat'}(j^*)(x)$

$\lambda P\ P(j^*)$



$\lambda y\ (\lambda Q \exists x\ \text{sandwich'}(x) \wedge Q(x))(\text{eat'}(y))$

$\rightarrow_\beta \lambda y\ \exists x\ \text{sandwich'}(x) \wedge \text{eat'}(y)(x)$

$(\lambda P \lambda Q \exists x\ P(x) \wedge Q(x))(\text{sandwich'})$

$\rightarrow_\beta \lambda Q \exists x\ \text{sandwich'}(x) \wedge Q(x)$

eat'

sw'

$\lambda P \lambda Q \exists x\ P(x) \wedge Q(x)$

1/3

Semantic parsing

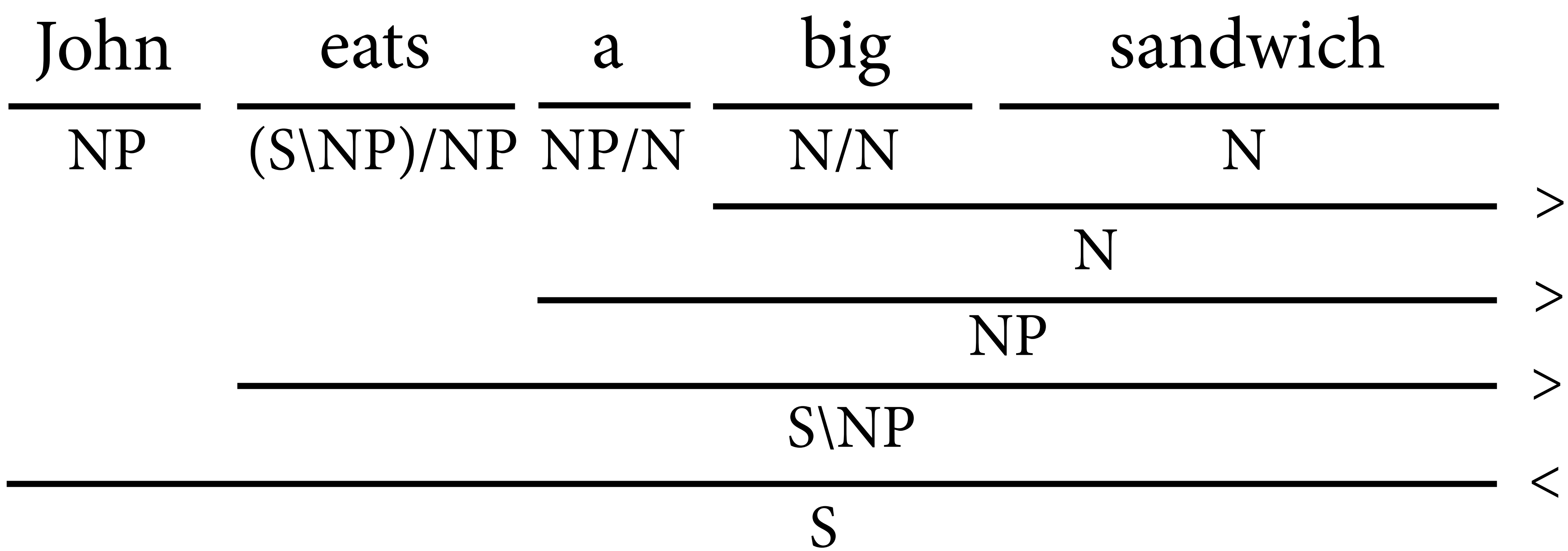
- Open issue in classical semantics construction: Where do we get large grammar that supports it?

- Current approach in CL is *semantic parsing*: learn mapping from sentence to formal meaning representation using statistical methods.

- Started with Geoquery corpus (880 sentences):

> What is the smallest state by area?
> $\texttt{answer}(x_1, \texttt{smallest}(x_2, \texttt{state}(x_1), \texttt{area}(x_1, x_2)))$

Combinatory categorial grammar



Semantics in CCG

$$\dfrac{\mathrm{X}: a}{\mathrm{Y/(Y\backslash X)}: \lambda P.P(a)}{>}\mathrm{T} \qquad \dfrac{\mathrm{X/Y}: f \qquad \mathrm{Y/Z}: g}{\mathrm{X/Z}: \lambda x.f(g(x))}{>}\mathrm{B} \qquad \dfrac{\mathrm{X/Y}: f \qquad \mathrm{Y\backslash Z}: g}{\mathrm{X\backslash Z}: \lambda x.f(g(x))}{>}\mathrm{Bx}$$

$$\dfrac{\mathrm{X}: a}{\mathrm{Y\backslash (Y/X)}: \lambda P.P(a)}{<}\mathrm{T} \qquad \dfrac{\mathrm{Y\backslash Z}: g \qquad \mathrm{X\backslash Y}: f}{\mathrm{X\backslash Z}: \lambda x.f(g(x))}{<}\mathrm{B} \qquad \dfrac{\mathrm{Y/Z}: g \qquad \mathrm{X\backslash Y}: f}{\mathrm{X/Z}: \lambda x.f(g(x))}{<}\mathrm{Bx}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{John}}{\mathrm{NP}: h^*}}{\mathrm{S/(S\backslash NP)}: \lambda P.P(h^*)}{>}\mathrm{T} \qquad \dfrac{\text{eats}}{\mathrm{(S\backslash NP)/NP}: eat'}}{\mathrm{S/NP}: \lambda x.(\lambda P.P(h^*))(eat'(x)) \Rightarrow_\beta \lambda x.eat'(x)(h^*)}{>}\mathrm{B} \qquad \dfrac{\text{a sandwich}}{\mathrm{NP}: sandwich'}}{\mathrm{S}: (\lambda x.eat'(x)(h^*))(sandwich') \Rightarrow_\beta eat'(sandwich')(h^*)}{>}$$

Zettlemoyer & Collins

GENLEX: build candidates for lexicon entries

Rules		Categories produced from logical form
Input Trigger	Output Category	$\arg\max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$
constant c	$NP : c$	$NP : texas$
arity one predicate p_1	$N : \lambda x.p_1(x)$	$N : \lambda x.state(x)$
arity one predicate p_1	$S\backslash NP : \lambda x.p_1(x)$	$S\backslash NP : \lambda x.state(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.state(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$	$N/N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$	$(N\backslash N)/NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$
an $\arg\max$ / $\min$ with second argument arity one function f	$NP/N : \lambda g.\arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g.\arg\max(g, \lambda x.size(x))$
an arity one numeric-ranged function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

(Zettlemoyer & Collins 2005, 2007)

Zettlemoyer & Collins

overall learning algorithm

Algorithm:

- For $t = 1 \dots T$

Step 1: (Lexical generation)

- For $i = 1 \dots n$:
  - Set $\lambda = \Lambda_0 \cup \text{GENLEX}(S_i, L_i)$.
  - Calculate $\pi = \text{PARSE}(S_i, L_i, \lambda, \bar{\theta}^{t-1})$.
  - Define λ_i to be the set of lexical entries in π.
- Set $\Lambda_t = \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$

Step 2: (Parameter Estimation)

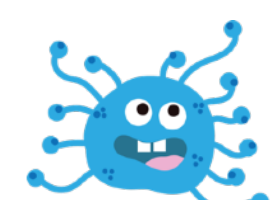
- Set $\bar{\theta}^t = \text{ESTIMATE}(\Lambda_t, E, \bar{\theta}^{t-1})$

analogous to Viterbi EM

Λ are CCG lexicons; λ contains new lexicon entries.
π is a CCG parse tree.
θ are parameters of a log-linear probability model.

Evaluation results

System	Variable Free			Lambda Calculus		
	Rec.	Pre.	F1	Rec.	Pre.	F1
Cross Validation Results						
KRISP	71.7	**93.3**	81.1	–	–	–
WASP	74.8	87.2	80.5	–	–	–
Lu08	81.5	89.3	**85.2**	–	–	–
λ-WASP	–	–	–	86.6	92.0	89.2
Independent Test Set						
ZC05	–	–	–	79.3	**96.3**	87.0
ZC07	–	–	–	86.1	91.6	88.8
UBL	81.4	89.4	**85.2**	85.0	94.1	**89.3**
UBL-s	**84.3**	85.2	84.7	**87.9**	88.5	88.2



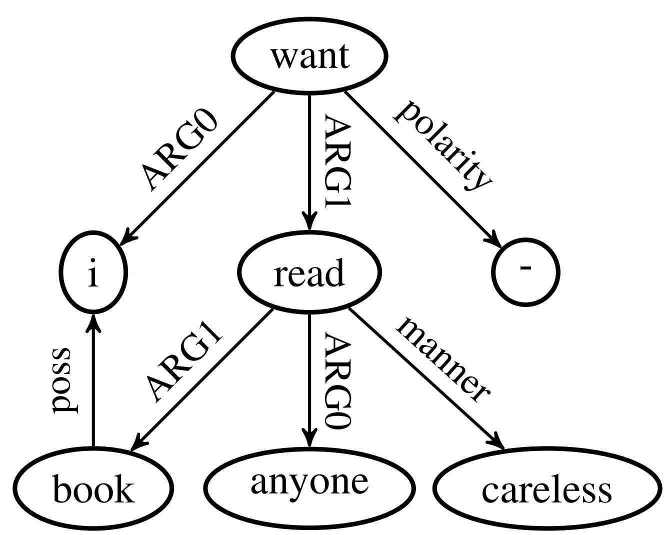
2/3

(on Geoquery 880 corpus)

Abstract Meaning Representations

- Limitations of Geoquery:
  - semantic representations are trees — (too) easy
  - very small

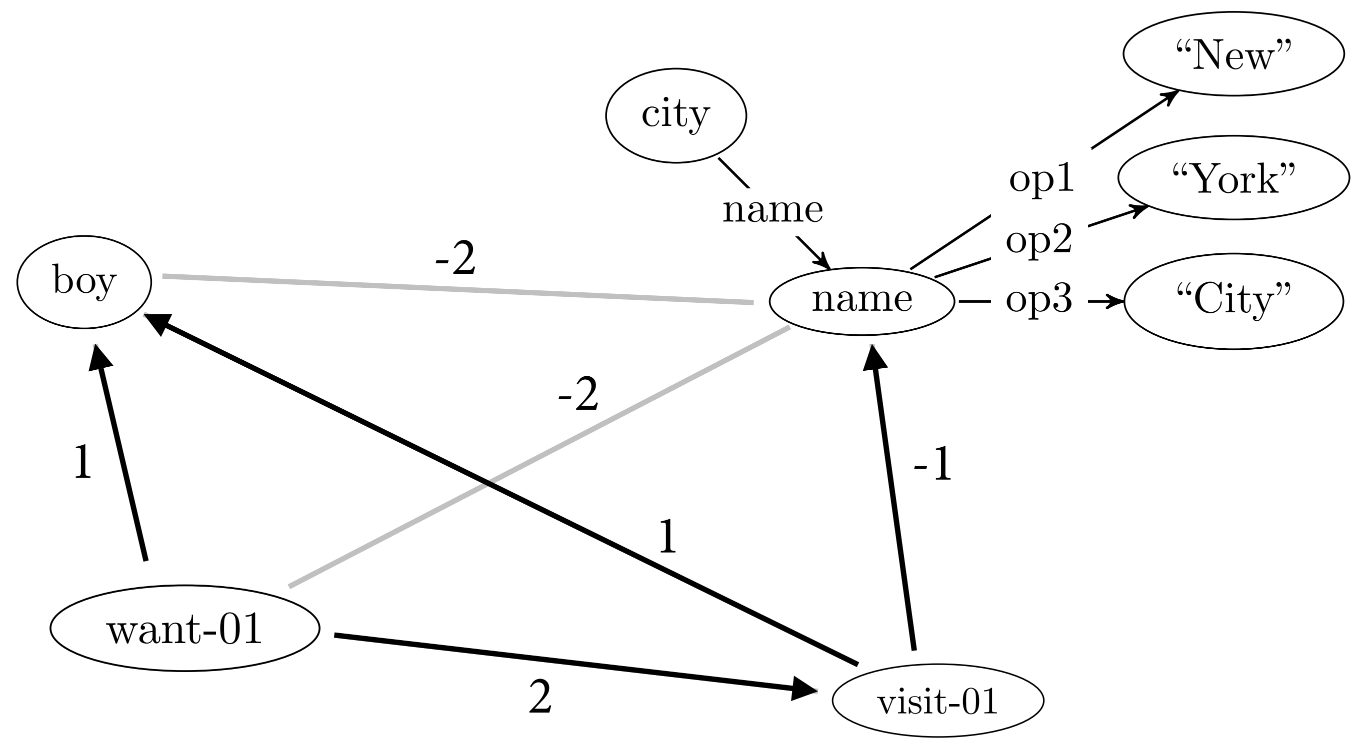
- Since 2014, much larger corpora available, e.g. AMRBank: ~40k AMRs, graphs as semantic representations.



"I don't want anyone to read my book carelessly."

Dependency-style AMR parsing

"The boy wants to visit New York City."



Concept Identification: determine atomic graph for each word.

Relation Identification: add all edges with positive weight; then repeatedly add least negative edge that connects subgraphs.

JAMR; Flanigan et al. 2014

Issues with JAMR

- JAMR can draw edge between any two nodes; syntactic structure of sentence used only indirectly.

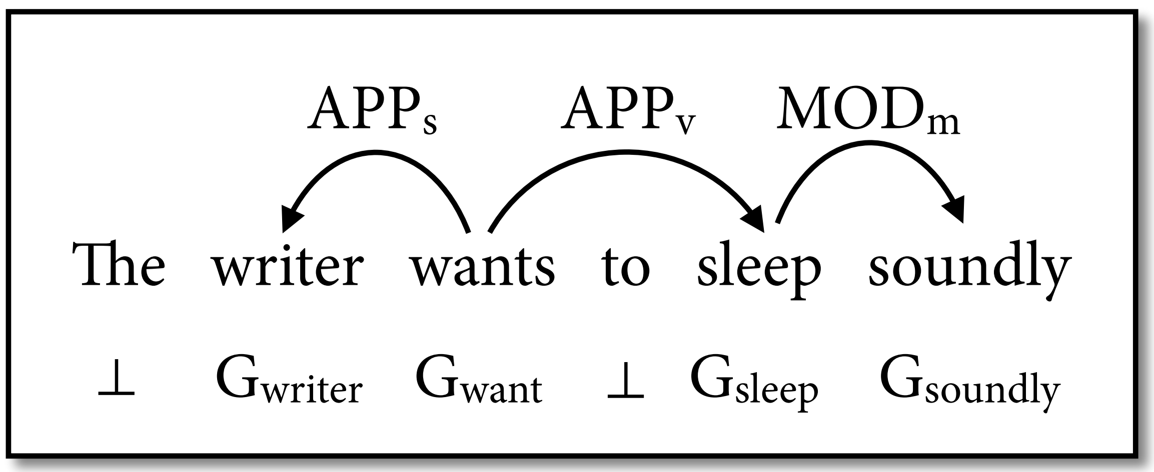
- Semantic representations for words don't know anything about their semantic arguments.

- Edges for control verbs added arbitrarily, not because linguistic principle of control discovered.

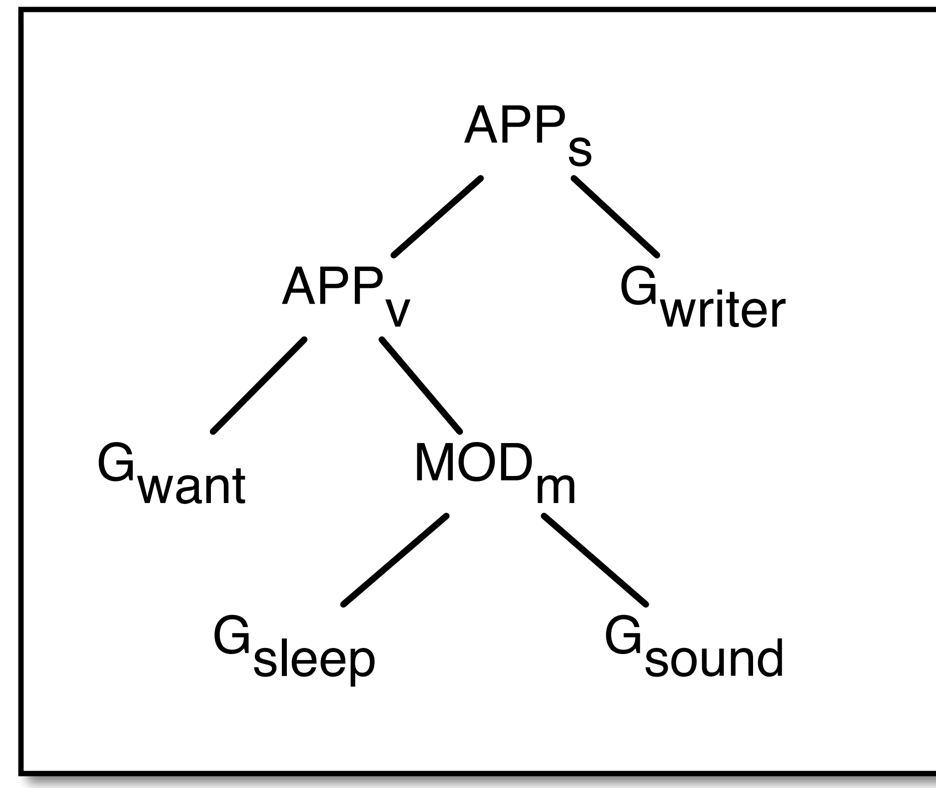
- No notion of compositionality!

Compositional semantic parsing

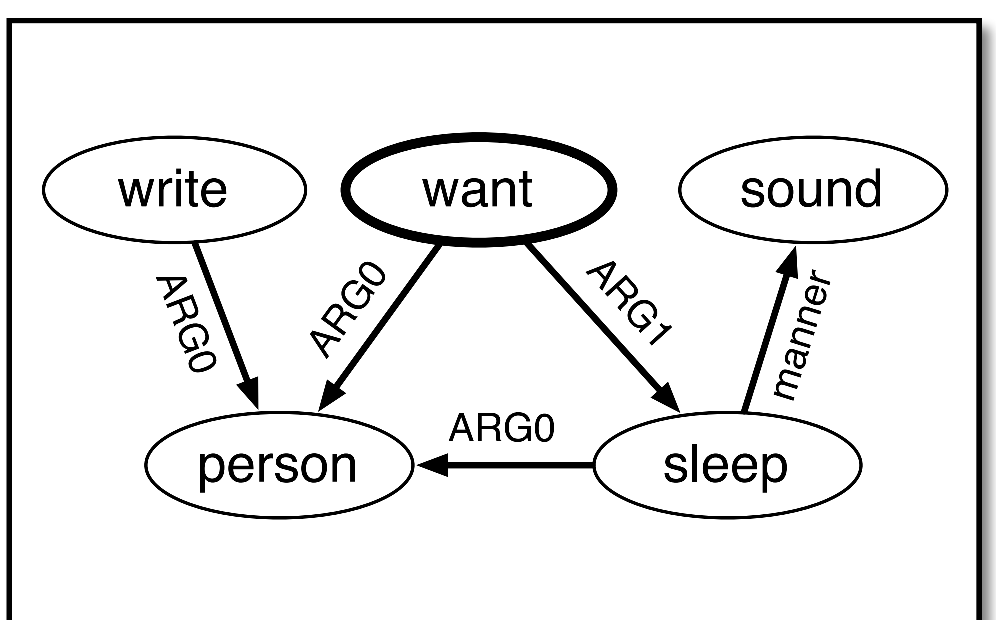
Learn to predict a (semantic) dependency tree,
which then decodes into an AM term, which evaluates to a graph.



AM dependency tree



Term over AM graph algebra

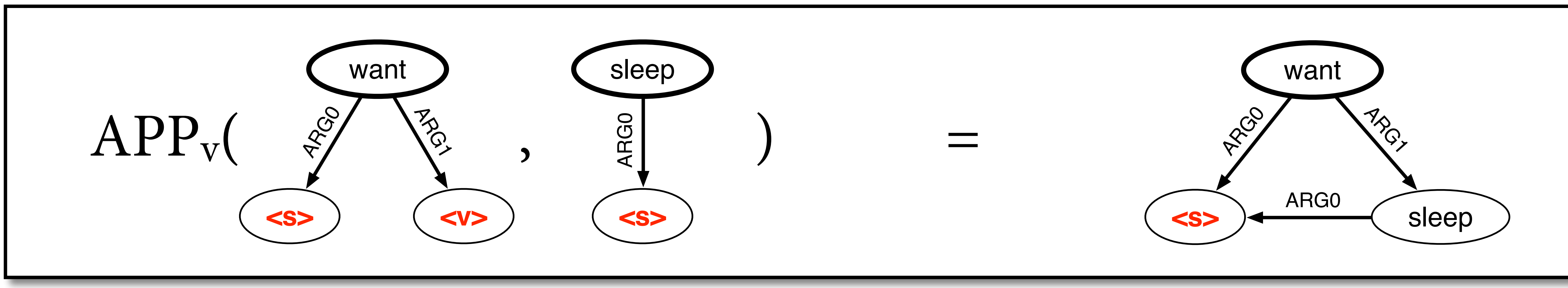


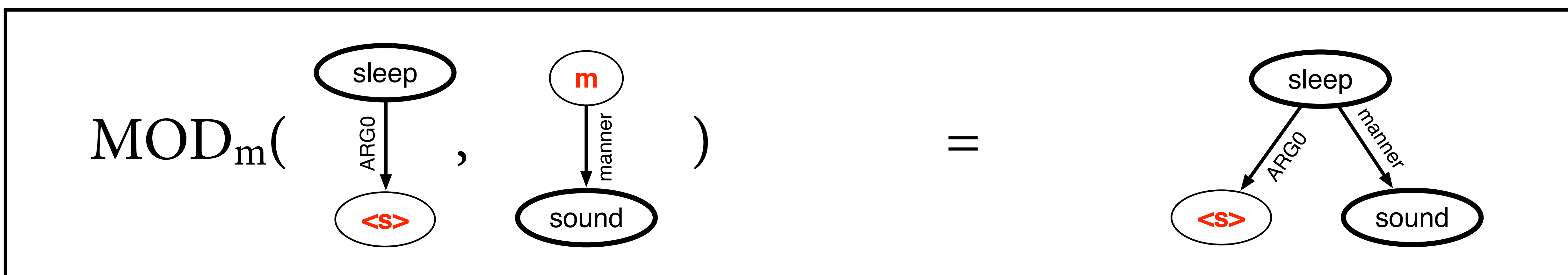
graph (e.g. AMR)

(Groschwitz et al., ACL 2018)

AM algebra

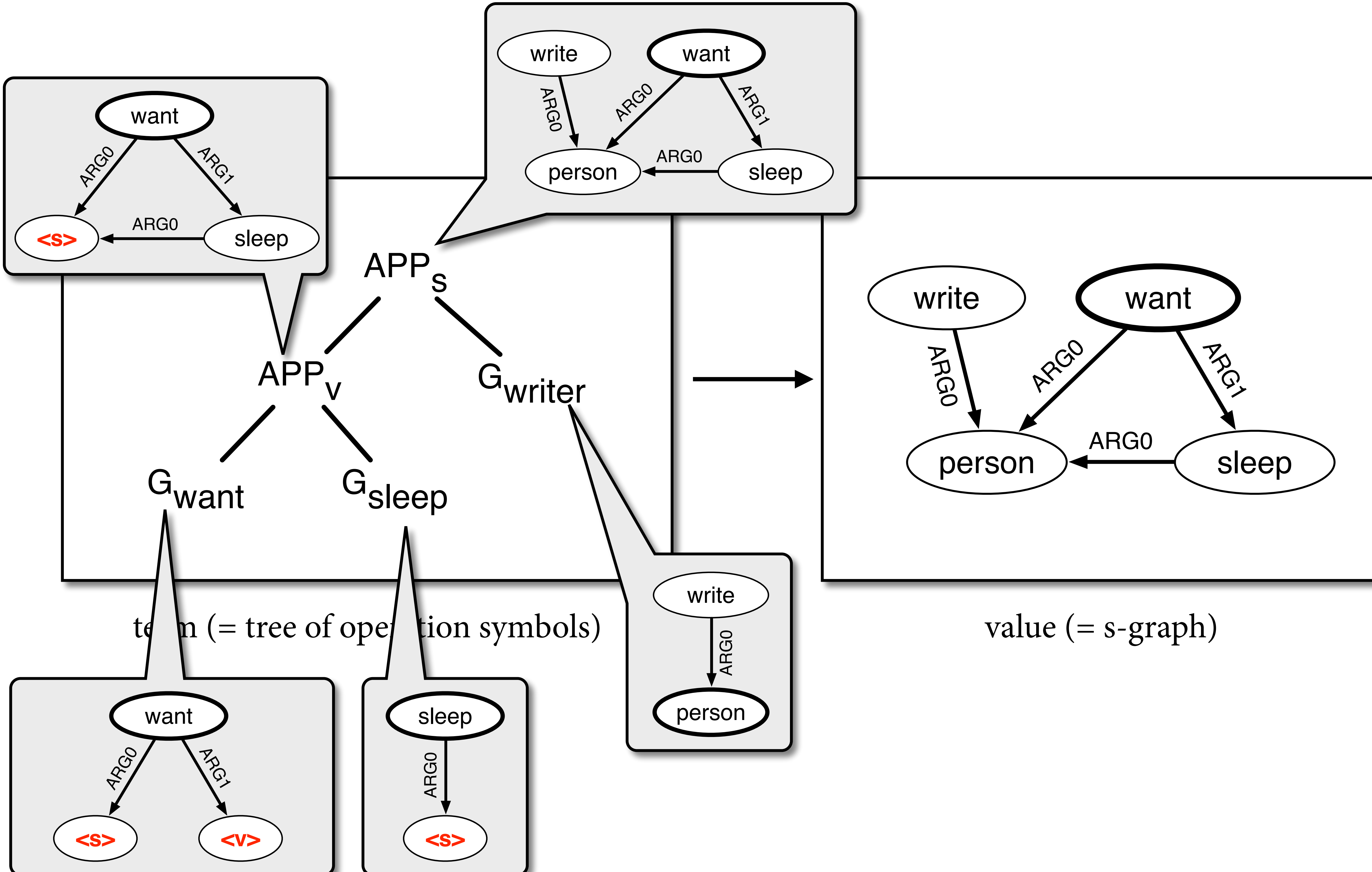
Two operations for combining s-graphs:
Apply (= head + complement), Modify (= head + modifier).





(Groschwitz et al., IWCS 2017; inspired by Copestake et al. 2001)

AM terms

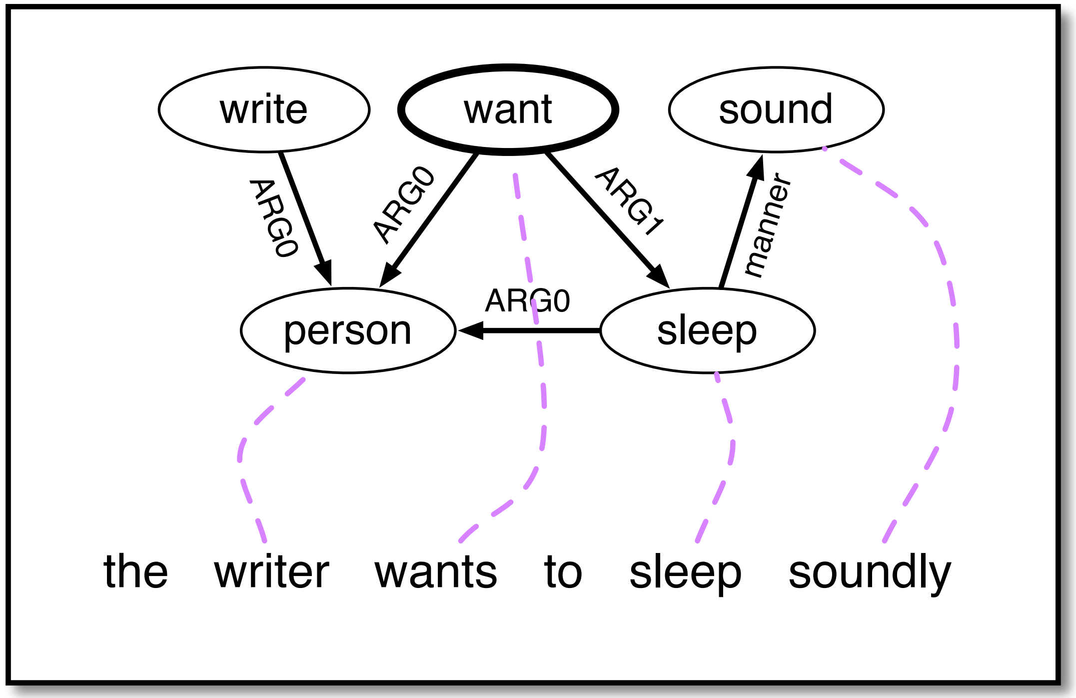


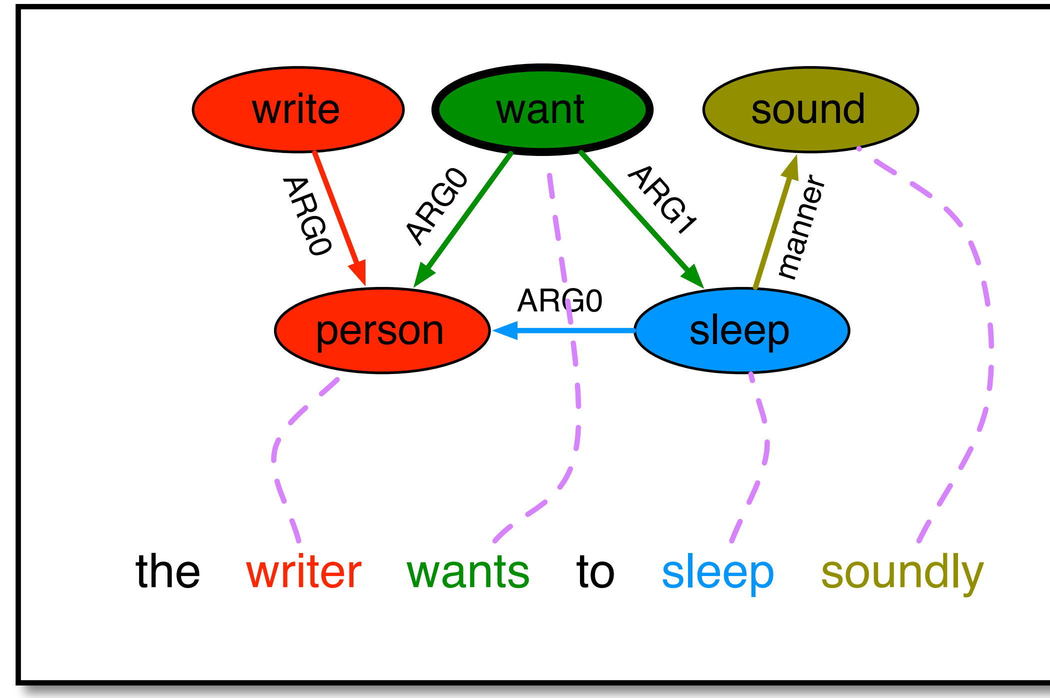
te m (= tree of ope tion symbols)

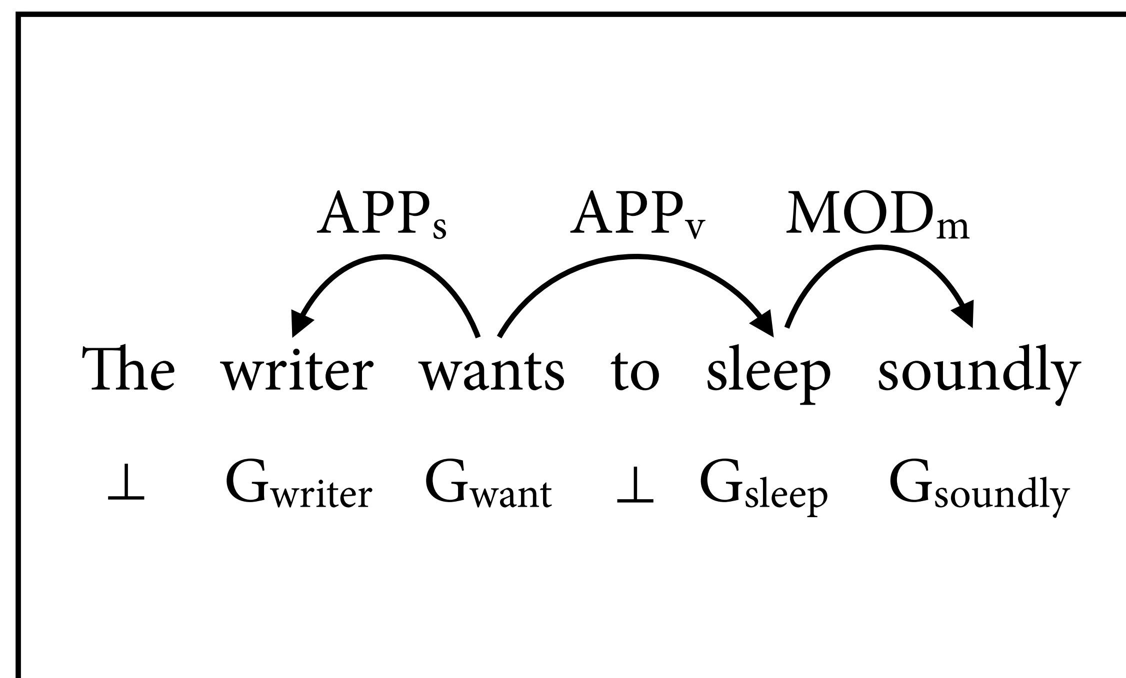
value (= s-graph)

(Groschwitz et al., IWCS 2017)

Converting training data







Idea: Represent AM term
as *AM dependency tree.*

Then semantic parser needs to learn:

- dependency parsing
- supertagging.

(Groschwitz et al., ACL 2018; *SEM 2021)

Parsing

- Convert (string, graph) training data into (string, supertags + dependencies) training data.

- Train neural supertagger + dependency parser to assign scores to supertags + dependencies.
    - easier than predicting the whole graph; compositional!

- At evaluation time, compute highest-scoring well-typed dependency tree.
    - well-typedness requirement makes this NP-complete
    - solve approximately with CKY-style parsing algorithm

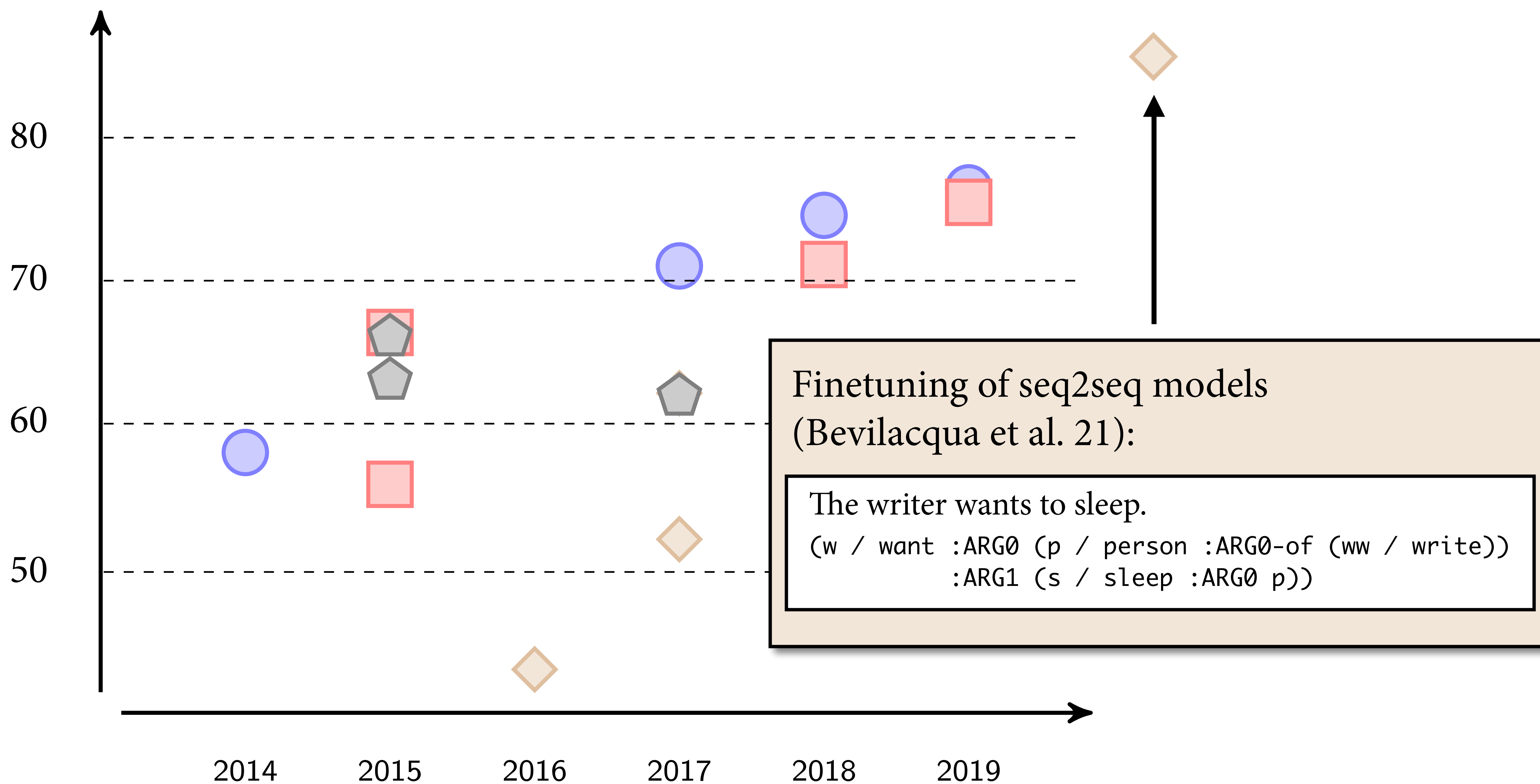
Parsing across graphbanks

	DM		PAS		PSD		EDS		AMR 2015	AMR 2017
	id F	ood F	id F	ood F	id F	ood F	Smatch F	EDM	Smatch F	Smatch F
Groschwitz et al. (2018)	-	-	-	-	-	-	-	-	70.2	71.0
Lyu and Titov (2018)	-	-	-	-	-	-	-	-	73.7	74.4 ±0.16
Zhang et al. (2019)	-	-	-	-	-	-	-	-	-	**76.3** ±0.1
Peng et al. (2017) Basic	89.4	84.5	92.2	88.3	77.6	75.3	-	-	-	-
Dozat and Manning (2018)	93.7	88.9	94.0	90.8	81.0	79.4	-	-	-	-
Buys and Blunsom (2017)	-	-	-	-	-	-	85.5	85.9	60.1	-
Chen et al. (2018)	-	-	-	-	-	-	**90.9**[1,2]	**90.4**[1]	-	-
This paper (GloVe)	90.4 ±0.2	84.3 ±0.2	91.4 ±0.1	86.6 ±0.1	78.1 ±0.2	74.5 ±0.2	87.6 ±0.1	82.5 ±0.1	69.2 ±0.4	70.7 ±0.2
This paper (BERT)	**93.9** ±0.1	**90.3** ±0.1	**94.5** ±0.1	**92.5** ±0.1	**82.0** ±0.1	**81.5** ±0.3	90.1 ±0.1	84.9 ±0.1	**74.3** ±0.2	75.3 ±0.2
Peng et al. (2017) Freda1	90.0	84.9	92.3	88.3	78.1	75.8	-	-	-	-
Peng et al. (2017) Freda3	90.4	85.3	92.7	89.0	78.5	76.4	-	-	-	-
This paper, MTL (GloVe)	91.2 ±0.1	85.7 ±0.0	92.2 ±0.2	88.0 ±0.3	78.9 ±0.3	76.2 ±0.4	88.2 ±0.1	83.3 ±0.1	(70.4)[3] ±0.2	71.2 ±0.2
This paper, MTL (BERT)	**94.1** ±0.1	**90.5** ±0.1	**94.7** ±0.1	**92.8** ±0.1	**82.1** ±0.2	**81.6** ±0.1	90.4 ±0.1	85.2 ±0.1	(74.5)[3] ±0.1	75.3 ±0.1

- First semantic parser that does well across all six major graphbanks.
- Established new states of the art through use of pretrained BERT embeddings.
- Small improvements through multi-task learning on multiple graphbanks.
- Transition-based parser can be extremely fast (~10k tokens/sec) and accurate.

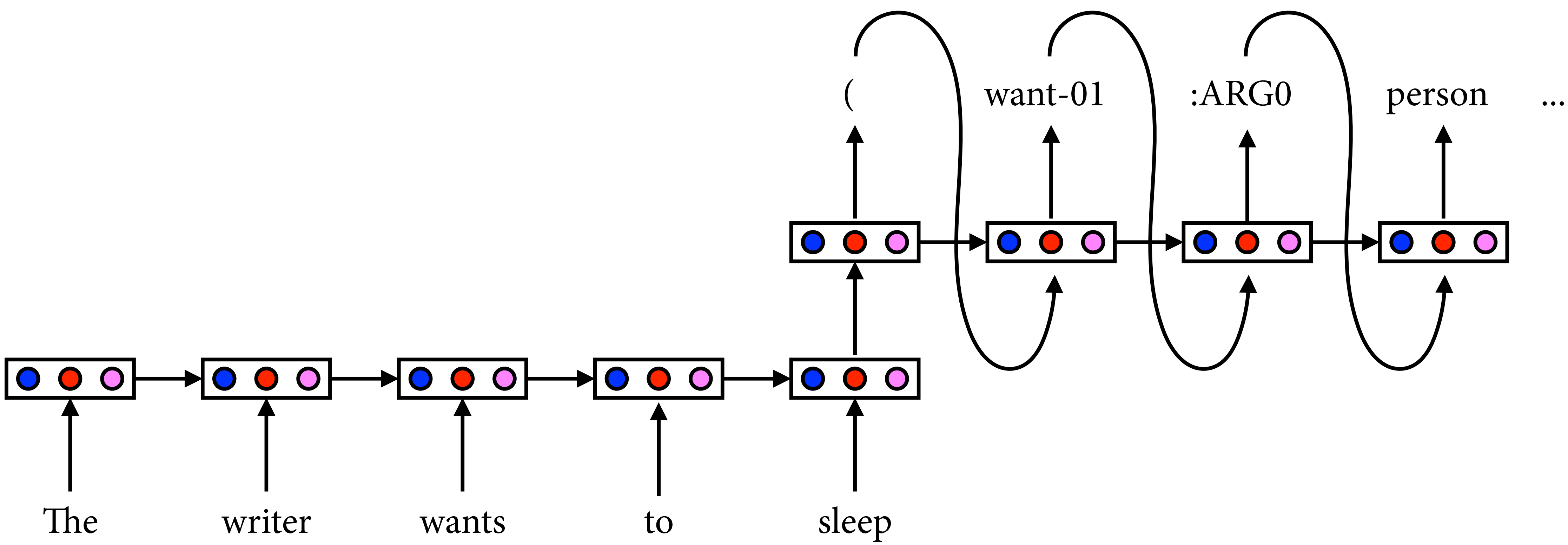
(Lindemann et al., ACL 2019; EMNLP 2020)

Progress in semantic parsing



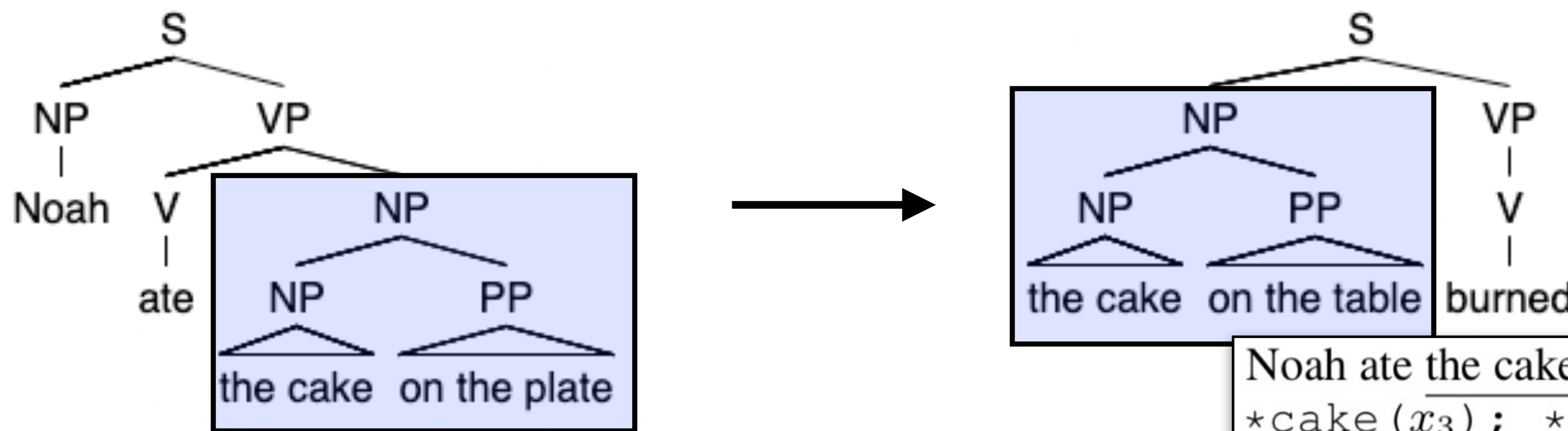
Parsing accuracies on the AMR graphbank up to 2019; slide from ACL 2019 tutorial by Koller, Oepen, Sun.

Seq2seq Semantic Parsing



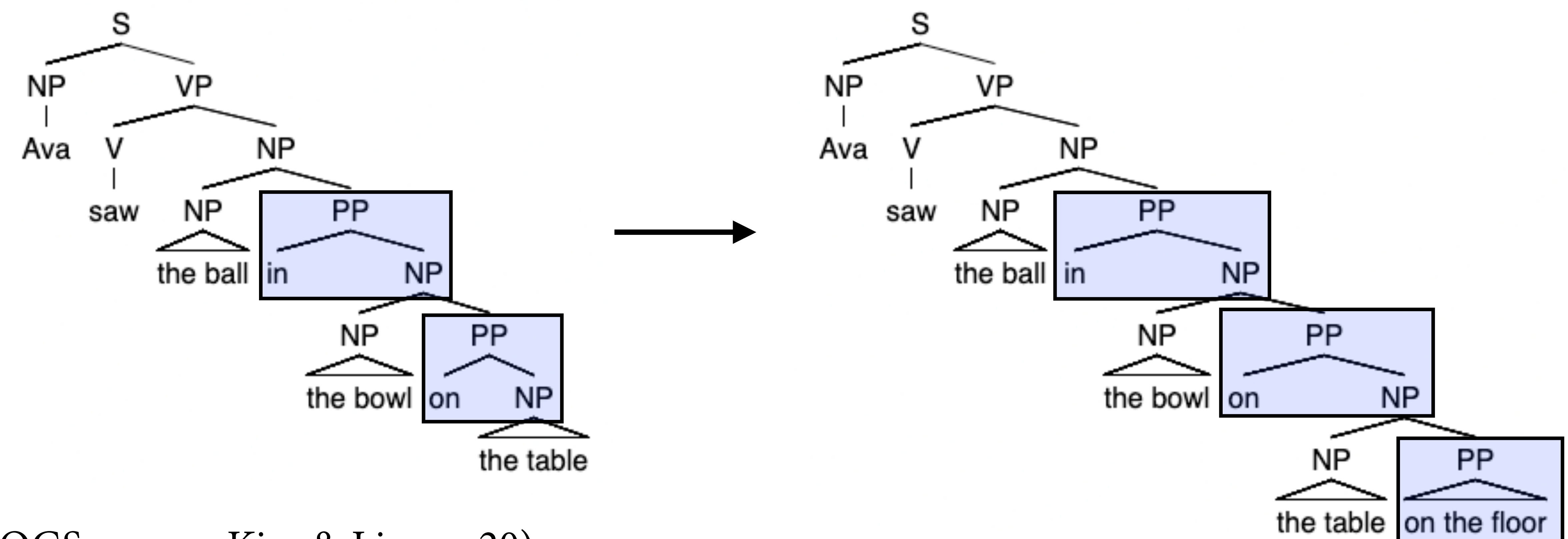
Compositional generalization

Structural: observe PPs in objects, generalize to PPs in subjects ("objPP to subjPP")



Noah ate the cake on the plate.
$\star$cake(x_3); $\star$plate(x_6);
eat.agent(x_1, Noah) $\wedge$ eat.theme(x_1, x_3)
$\wedge$ cake.nmod.on(x_3, x_6)

Structural: observe PPs up to recursion depth 2, generalize to recursion depth 3-12



(COGS corpus, Kim & Linzen 20)

Compositional helps on COGS

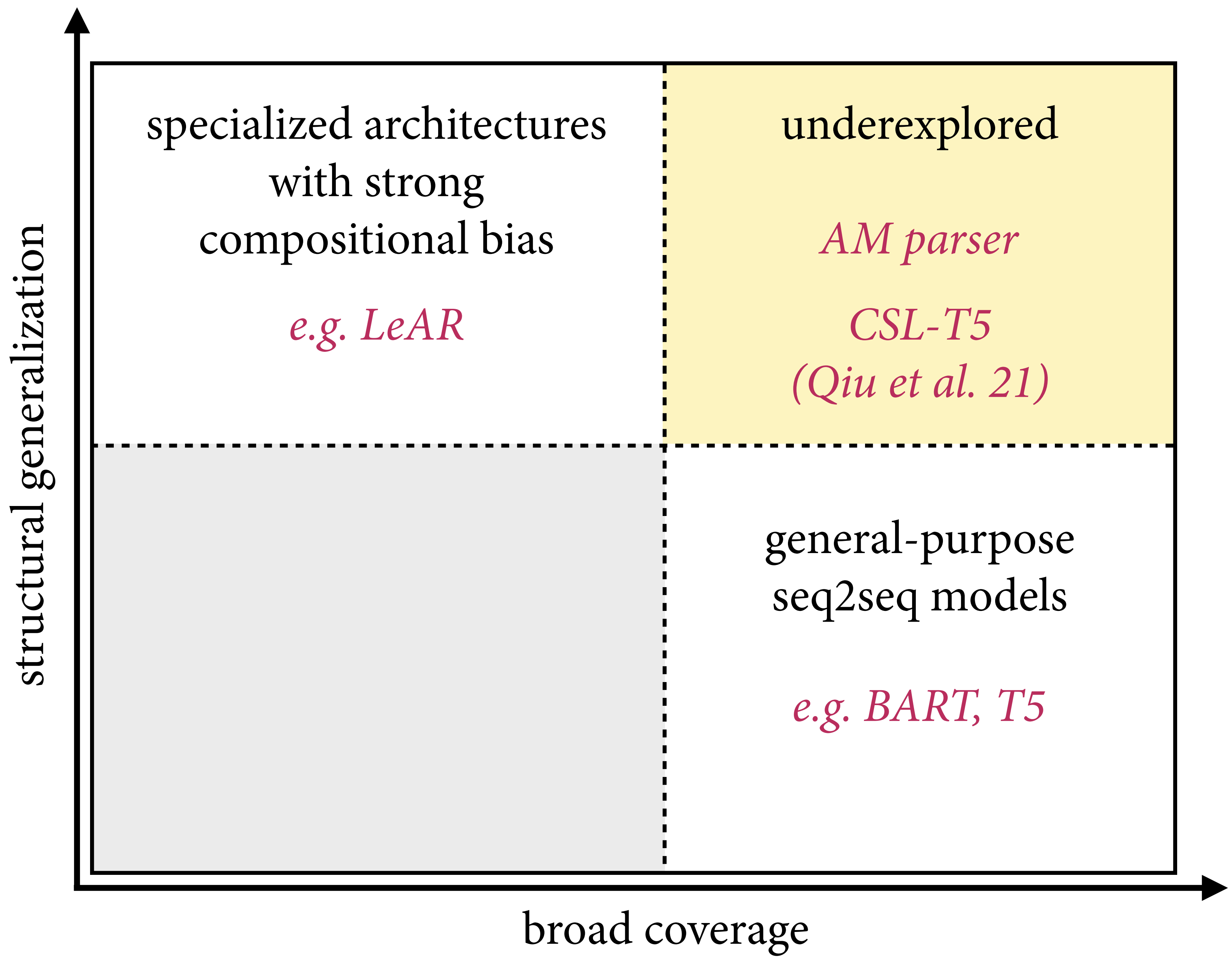
Model Class	Model	Struct: Obj to Subj PP	Struct: CP recursion	Struct: PP recursion	Lex: all 18 other types	Overall
seq2seq	BART	0	0	12	91	79
	BART+syn	0	5	8	93	80
	T5	0	0	9	97	83
	Kim and Linzen 2020	0	0	0	73	63
	Akyürek and Andreas 2021	0	0	1	96	82
	Zheng and Lapata 2022	0	12	39	99	89
	Conklin et al. 2021	0	0	0	88	75
	Csordás et al. 2021	0	0	0	95	81
	Qiu et al. 2021 *	100	100	100	100	100
structure-aware	Liu et al. 2021	93	100	99	99	99
	Weißenhorn et al. 2022	78	100	99	100	98

AM parser

*) Can still see this as a neurosymbolic model (uses compositional data augmentation); achieves perfect accuracy, up to depth seen in the augmented data.

(Weißenhorn, Donatelli, K., *SEM 2022; Yao & K., EMNLP 2022)

Semantic parsing tasks



(picture adapted from Shaw et al., ACL 2021)

Conclusion

- Challenge in compositional semantic construction: Where do we get large-scale grammars?

- Semantic parsing: Learn such grammars from corpora with semantic annotations.
  - GeoQuery: small corpus of trees
  - graphbanks, e.g. AMRBank
  - also parsing into programs and other formal languages