Invitation to Semantic Graphs

By Yu Cao
04/30/2021

Proper semantic representations?

- Depending on who you ask
    - Logic (symbolic)
    - Vectors (numeric)

Higher-Order Logic

- Expressive power
- Compositional transparency
- Model-theoretical interpretability

A boy walked a dog.

$\exists x y v \; \mathrm{boy}(x) \wedge \mathrm{ag}(v, x) \wedge \mathrm{walk}(v) \wedge \mathrm{th}(v, y) \wedge \mathrm{dog}(y)$

a. *a boy* $\lambda P \, \exists x \; \mathrm{boy}(x) \wedge P(x)$

b. *walk* $\lambda x y \, \exists v \; \mathrm{ag}(v, x) \wedge \mathrm{walk}(v) \wedge \mathrm{th}(v, y)$

Higher-Order Logic

- Lexicon induction
  sentence formula $\rightarrow$ *token* formula
- Unification
  equating formulas by substituting their *free variables*

$$f(g) \equiv \exists x y v \; \mathrm{boy}(x) \wedge \mathrm{ag}(v, x) \wedge \mathrm{walk}(v) \wedge \mathrm{th}(v, y) \wedge \mathrm{dog}(y)$$

$$f = \lambda P \, \exists x \; \mathrm{boy}(x) \wedge P(x),$$

$$g = \lambda x \, \exists y v \; \mathrm{ag}(v, x) \wedge \mathrm{walk}(v) \wedge \mathrm{th}(v, y) \wedge \mathrm{dog}(y).$$

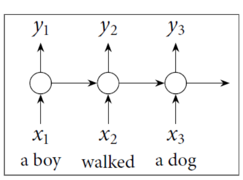
- HO unification is undecidable
  Huet (1973); Goldfarb (1981) a.o.
  Monadic 2nd-order unif is decidable (Levy et al., 2004)

Word Embedding

- Distributional semantics (Firth, 1968; Harris, 1954)
- Token → $v \in \mathbb{R}^n$
  s.t. similar distribution ~ proximate points
  (Devlin et al., 2019; Kiros et al., 2015; Peters et al., 2018)
- ~~Semantic parsing, symbolic lexicon induction~~

Word Embedding

- Sentential semantics
  "Combines" word embeddings according to a certain NN.
  e.g. Recurrent neural network (Elman, 1991)



- Compositionality
  Yes. But ...
- Model-checking
  No. Thanks

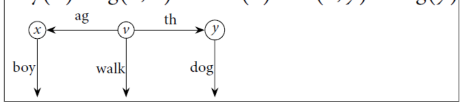
Graphs

- Computational tractability & formal interpretability
  - Directed graph-based formalisms
    (Banarescu et al., 2013; Berglund et al., 2017; Bos, 2016; Bos et al., 2017; Kalouli and Crouch, 2018; Kuhlmann and Oepen, 2016; Liang et al., 2013; Stabler, 2018)
  - Hybrid (modal) Logic
    (Baldridge and Kruijff, 2002; Kruijff, 2001;White, 2006)
  - Discourse Repr Theory ⚠
    (Kamp et al., 2011; Kamp and Reyle, 1993)
- Flexibly interpretable
  Only encode the **relations** between objects
  Interpretation subject to each use case
- Flexibly constructable
  Inependent of the interpretation of graphs
  ✓ Monadic-2nd-order variables only

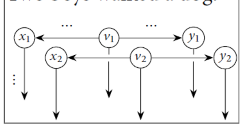
Graphs

- 1st-order déjà vu?

$\text{boy}(x) \wedge \text{ag}(v, x) \wedge \text{walk}(v) \wedge \text{th}(v, y) \wedge \text{dog}(y)$



Two boys walked a dog.



Every boy walked a dog.
Every even number is divisable by 2.

- **Plurality & quantification**

Graphs

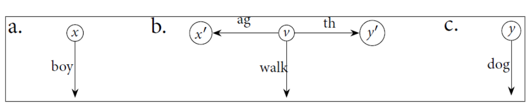
... TOWARDS A FORMALISM

- Representation (1/4)
  - thematic relations
  - modification
  - co-reference
  - plurality & quantification
  - ~~conjunction & disjunction~~
  - ~~intensionality (& modality)~~
- Interpretation (2/4)
  - direct | ~~indirect~~
  - model-theoretical intepreter

Graphs

... TOWARDS A FORMALISM

- Construction (3/4)
  - Graph unification – glue graphs by fusing vertices



  - Syntax-semantics interface to **parition** referents
- Application (4/4)
  - Distributivity, non-scoping distributivity
  - Cross-categorial coordination, branching quantification
  - Scope permutation, inverse linking, scope islands

- Exceptionally scoping indefinites, existential vs. distributional scope
- Strong & weak cross-over, donkey anaphors, accessibility, binder roof constraint

Reading semgraphs

- Enter through the **root** to traverse a graph G
- G be uniquely rooted & connected

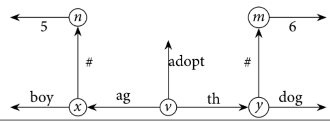
Definition 1.2. We say e *encycles* e' if both edges lie in a common cycle and e' would not be in any cycle without e and, if $\gamma(e)$ is equality, any equality edge sharing its head with e.

Definition 1.3. A *root* is a vertex v such that either v has no in-edge; or every in-edge $\overrightarrow{uv}$ encycles some out-edge $\lambda\,\overrightarrow{vw}$.

Reading semgraphs

- Thematic relation (w/ cardinality)

(2.35) Five boys adopted six dogs.



(2.36) There is some g such that

a. $v^g \subseteq adopt_w$, $x^g \subseteq boy_w$, $y^g \subseteq dog_w$;

b. $|x^g| = 5, |y^g| = 6$;

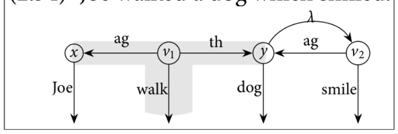
c. $x^g = \bigcup_{a \in v^g} ag_w(a)$, $y^g = \bigcup_{a \in v^g} th_w(a)$.

- *The* insight of DRT:
  existence & logical conjunction **implicitly** indicated

Reading semgraphs

- λ-edge "such that"

(2.34) Joe walked a dog which smiled.

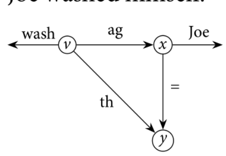


- To guide graph traversal
- To shift vertex valuation order

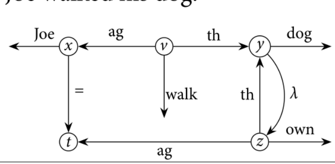
Reading semgraphs

- Co-reference

(2.10) Joe washed himself.



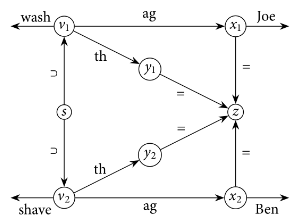
(2.11) Joe walked his dog.



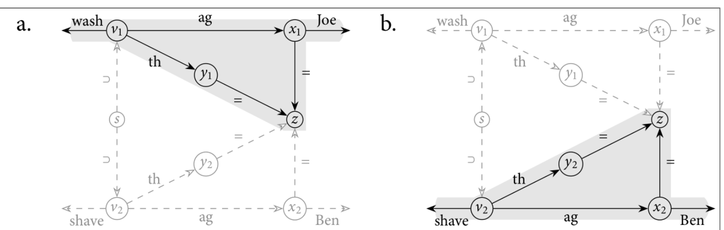
Reading semgraphs

- Co-reference (w/ conjunction)

(2.12) Joe washed and Ben shaved himself.



(2.13) a. b.



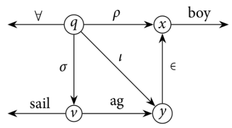
- What if the equality direction is reversed?

Reading semgraphs

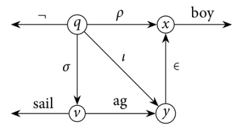
- Quantification: **counting via iteration**

```
for x in X:
    statements about x
```

(2.20) a. Every boy sailed.

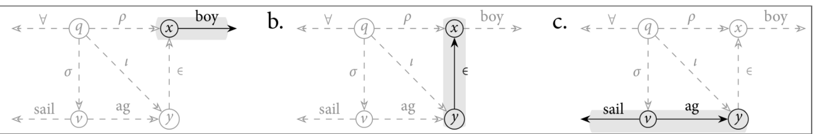


b. No boy sailed.



- Counting (classes of) valuations

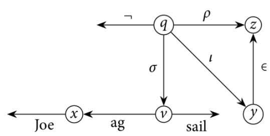
(2.21) a. b. c.



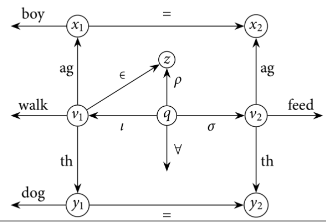
Reading semgraphs

- Quantification: verbal negation, implication

(2.22) a. Joe didn't sail.



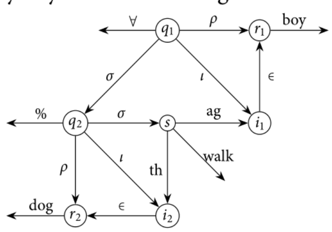
b. If a boy walks a dog, he feeds it.



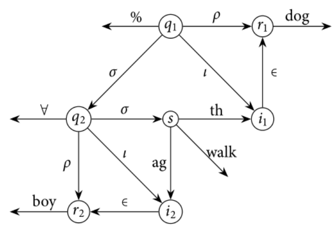
- Quantification: nesting

(2.60) Every boy walked most dogs.

a.



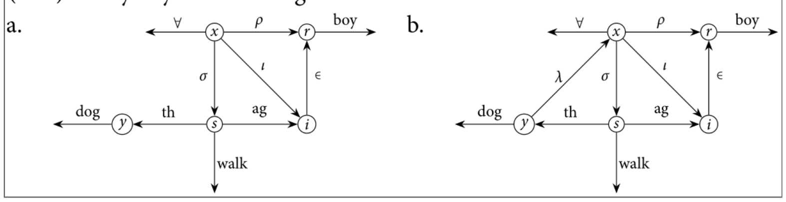
b.



Reading semgraphs

- Quantification (w/ λ-edge)

(2.72) Every boy walked a dog.

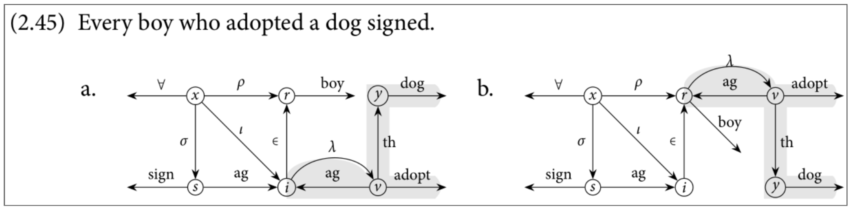


- How do you paraphrase wide-scope indefinites?

Reading semgraphs

- Quantification (w/ iterator filter)

(2.45) Every boy who adopted a dog signed.



- What's wrong w/ (b)?

Constructing semgraphs

- Graph unification – to partition referents

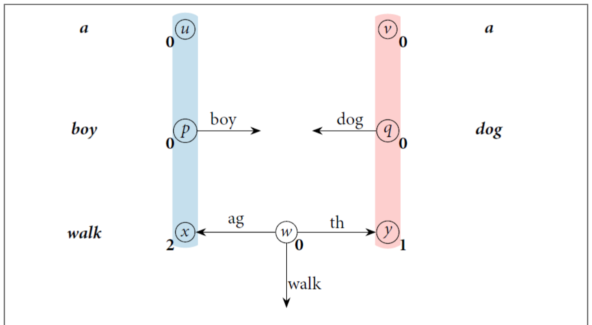


Figure 3.1: Fusing equivalence for *a boy walked a dog*.

Constructing semgraphs

- Syntax as a parition function:
  a sequence of tokens $\mapsto$ valid equivalence relations
- Categorial grammars (CGs; Ajdukiewicz, 1935; Bar-Hillel, 1953)
- The plan:
  - type each linguistic token
  - establish **atom-vertex correspondence**
  - via **type reduction** find **atom matches** ($\rightarrow$ vertex equivalence)
- Automation: **https://git.io/JtUTm**

Constructing semgraphs

... TYPING

- Expr_1 is of type $A\backslash B$ (*resp.* B/A) iff it yields an expression of type B when **concatenated** with Expr_2 of type A on its left (*resp.* right)
  e.g. $\quad see \rightarrow np\backslash s/np$
- Expr_1 is of type $A \downarrow B$ (*resp.* $B \uparrow A$) iff it yields an expression of type B
  when filling a gap in (*resp.* having a gap filled by) Expr_2 of type A.
  e.g. *a dog that* $\mathbf{a\ boy\ saw\ \_\_\ in\ a\ park} \rightarrow s \uparrow np$
  a boy saw $\mathbf{every\ dog}$ *in a park* $\rightarrow s \uparrow np \downarrow s$

Remark

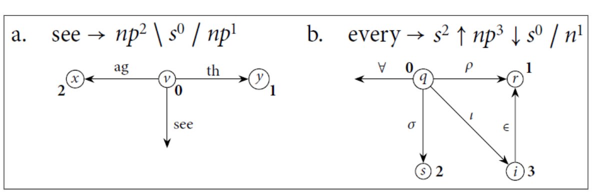
Lambek notation | Steedman notation

Continuized typing: Barker and Shan (2014); **Morrill (2011)**

Constructing semgraphs

... ATOM-VERTEX CORRESPONDENCE

Calder et al. (1988); Zeevat(1988)



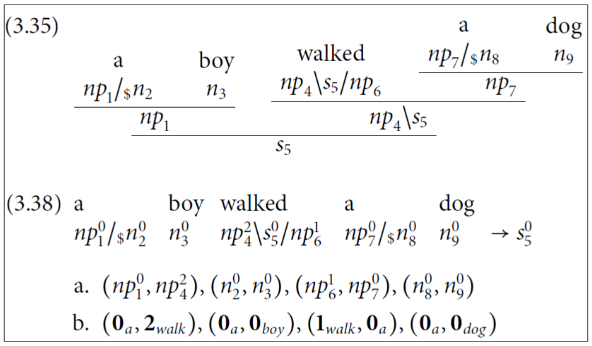
- Claim: **atom-vertex corresp is partly deterministic**
  - deterministically label atoms in types (**depth calc**)
  - lexically label vertices in graphs

Constructing semgraphs

... TYPE REDUCTION

- Combinatory

  (CCGs; Steedman, 1996; Jacobson, 1999; Steedman, 2011)



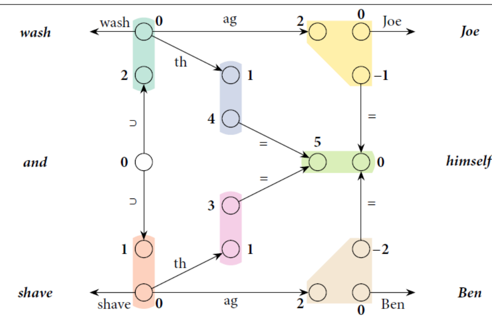
- Type-logical

  (TLGs; Barker and Shan, 2014;Moortgat, 2011;Morrill, 2011; Moot and

Retoré, 2012)

Constructing semgraphs

... TYPE REDUCTION + CO-REF RESOLUTION



(3.39) Joe washed and Ben shaved himself

np^0_1 $np^2_2\backslash s^0_3/np^1_4$ $(s^2_5/np^4_6)\backslash_{\&}(s^0_7/np^5_8)/_{\&}(s^1_9/np^3_{10})$ np^0_{11} $np^2_{12}\backslash s^0_{13}/np^1_{14}$ np^0_{15} $\to s^0_7$

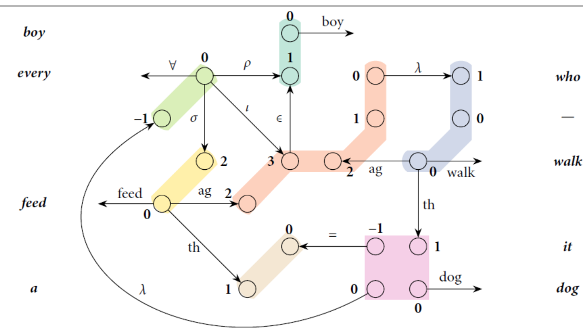
a. $(np^0_1, np^2_2), (s^0_3, s^2_5), (np^1_4, np^4_6), (np^5_8, np^0_{15}),$
$(s^1_9, s^0_{13}), (np^3_{10}, np^1_{14}), (np^0_{11}, np^2_{12})$

b. $(\mathbf{0}_{Joe}, \mathbf{2}_{wash}), (\mathbf{0}_{wash}, \mathbf{2}_{and}), (\mathbf{1}_{wash}, \mathbf{4}_{and}), (\mathbf{5}_{and}, \mathbf{0}_{himself}),$
$(\mathbf{1}_{and}, \mathbf{0}_{shave}), (\mathbf{3}_{and}, \mathbf{1}_{shave}), (\mathbf{0}_{Ben}, \mathbf{2}_{shave})$

c. $(-\mathbf{1}_{himself}, \mathbf{0}_{Joe}), (-\mathbf{2}_{himself}, \mathbf{0}_{Ben})$

Constructing semgraphs

… TYPE REDUCTION + PRED RESOLUTION



(3.40)

every	boy	who	—
$s_1^2{\uparrow}np_2^3{\downarrow}s_3^0/n_4^1$	n_5^0	$np_6^0\backslash_{\$}np_7^0/(s_8^1{\uparrow}np_9^0)$	$(s_{10}^0{\uparrow}np_{11}^1){\downarrow_{\$}}(s_{12}^0{\uparrow}np_{13}^1)$

walked	a	dog	fed	it	
$np_{14}^2\backslash s_{15}^0/np_{16}^1$	$np_{17}^0/_{\$}n_{18}^0$	n_{19}^0	$np_{20}^2\backslash s_{21}^0/np_{22}^1$	np_{23}^0	$\to s_3^0$

a. $(s_1^2, s_{21}^0), (np_2^3, np_6^0), (n_4^1, n_5^0), (np_7^0, np_{20}^2),$
$(s_8^1, s_{12}^0), (np_9^0, np_{13}^1), (s_{10}^0, s_{15}^0), (np_{11}^1, np_{14}^2),$
$(np_{16}^1, np_{17}^0), (n_{18}^0, n_{19}^0), (np_{22}^1, np_{23}^0)$

b. $(\mathbf{2}_{every}, \mathbf{0}_{feed}), (\mathbf{3}_{every}, \mathbf{0}_{who}), (\mathbf{1}_{every}, \mathbf{0}_{boy}), (\mathbf{0}_{who}, \mathbf{2}_{feed}),$
$(\mathbf{1}_{who}, \mathbf{0}_{\_}), (\mathbf{0}_{who}, \mathbf{1}_{\_}), (\mathbf{0}_{\_}, \mathbf{0}_{walk}), (\mathbf{1}_{\_}, \mathbf{2}_{walk}),$
$(\mathbf{1}_{walk}, \mathbf{0}_{a}), (\mathbf{0}_{a}, \mathbf{0}_{dog}), (\mathbf{1}_{feed}, \mathbf{0}_{it})$

c. $(-\mathbf{1}_{it}, \mathbf{0}_{dog}), (-\mathbf{1}_{a}, \mathbf{0}_{every})$

Remark Jäger (2005, chap. 6) gives equal semantics to pronouns and indefinites

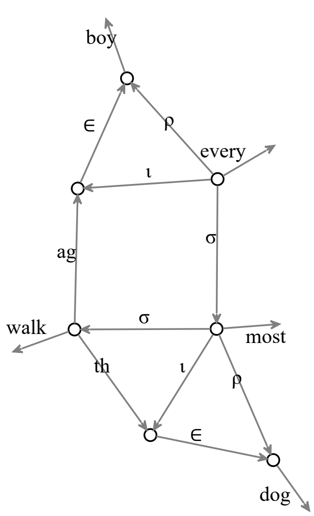
In [1]:

```
from graphviz import Source
from lambekseq.semcomp import SemComp

SemComp.load_lexicon(abbr_path='../../code/lambekseq/abbr.json',
                     vocab_path='../../code/lambekseq/schema.json')

ex = 'every boy walk most dog'
pos = 'qnt n vt qnt n'
sc = SemComp(zip(ex.split(), pos.split()))
sc.unify()
```


In [2]:

```
Source(sc.semantics[0].dot_styled)
```

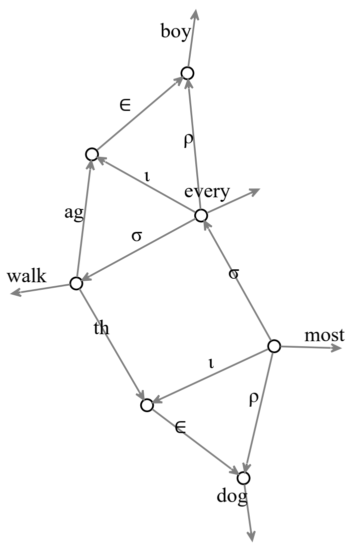
Out[2]:



In [3]:

```
Source(sc.semantics[1].dot_styled)
```

Out[3]:



In [4]:

```
sc.syntax[0].insight.printProofs()
```

```
(n_12, n_13), (n_4, n_5), (np_10, np_8), (np_
2, np_6), (s_0, s_3), (s_1, s_11), (s_7, s_9)
(n_12, n_13), (n_4, n_5), (np_10, np_8), (np_
2, np_6), (s_0, s_11), (s_1, s_7), (s_3, s_9)
```