Ling 5801: Lecture Notes 20
Lambda calculus semantics

Contents

We've seen how vague 'gist'-like semantic vectors might help with sentence processing.

We've also seen meanings of computer language statements defined as procedures / things to do:

```
for x in range( 1, 5 ):          ## iterate four times
  print( x )                     ## print iteration number
```

But real semantics (meanings) of natural language sentences aren't vague or procedural:

Half the time someone hits a nut with a rock, the nut opens.

Sentences like this seem to specify (probabilistic) inferences to use in planning:

$$\mathsf{P}(\text{nut } n \text{ opens} \mid \text{agent } a \text{ hits nut } n \text{ with rock } r) = 0.5$$

Knowing this, the next time we want to open a nut we might use a rock instead of a stick.

This **useful knowledge** is often what we want to get out of natural language sentences.

20.1 Decision theory [von Neumann & Morgenstern, 1944]

Here's a model of conscious thought for artificial agents, to serve as context...

Conscious thought is a **decision process**: we choose actions to maximize **average expected utility**.

A decision process assumes a set of **plans** p, a **world model** m and a **reward** $\mathsf{R}(m)$.

For example:

- p may be a plan to walk to a hill,
- m may include our location and the knowledge that a step may take us closer to a goal,

- $R(m)$ may be the prestige we get from reaching the hill.

Average (over time t) expected reward for a plan p is a sum over **outcomes** o of **actions** a in p:

$$\mathsf{AEU}(t, m, p) = \overbrace{\mathsf{R}(m)}^{\text{reward}} \cdot \begin{cases} \text{if } \exists_a \overbrace{p \wedge m \to a}^{\text{next action } a \text{ of } p}: & \overbrace{\sum_o \mathsf{P}(o \mid m \wedge a)}^{\text{sum all outcome events } o \text{ of } a} \cdot \overbrace{\mathsf{AEU}(t{+}1, \underbrace{m \wedge a \wedge o}_{\text{new } m \text{ at next } t}, p)}^{\text{repeat with } m, a \text{ and } o \text{ as new } m} \\ \text{otherwise:} & \frac{1}{t} \end{cases}$$

(This assumes the plan p is perfectly specific: at most one action a for each possible world m.)

For example, if the goal hill is one step away, we get a reward in one step, so $\mathsf{AEU}(t, m, p) = 1$.

But if it's muddy and we slip half the time and don't go anywhere, then:

$$\mathsf{AEU}(t, m, p) = \begin{cases} .5 \text{ (no slip)} & \times \frac{1}{1} \text{ (arrive in 1 step)} \\ +.5 \text{ (slip)} & \times \begin{cases} .5 \text{ (no slip)} & \times \frac{1}{2} \text{ (arrive in 2 steps)} \\ +.5 \text{ (slip)} & \times \begin{cases} .5 \text{ (no slip)} & \times \frac{1}{3} \text{ (arrive in 3 steps)} \\ +.5 \text{ (slip)} & \times \ldots \end{cases} \end{cases} \end{cases}$$
$$\approx .7$$

So if we have two plans (clear path and muddy path) and we know mud slows us, we can avoid it.

We speculate language provides us an advantage by letting us **share** plans and world knowledge.

So what are these plans and world knowledge?

20.2 Basic parts of meanings (ontology)

We assume sentences express **propositions** – things that have **truth values** (type `t`): True or False.

- Declaratives (e.g. *Some nuts are toxic.*) are simple propositions. We use these in planning.
- Imperatives (e.g. *Get me a bucket!*) are also propositions: *Speaker wants a bucket.*
- Interrogatives (e.g. *Where is Spain?*) are also propositions: *Speaker wants to know ...*

These propositions may involve **entities** (type `e`):

- 'Count' entities (e.g. birds, nuts, etc.)
- Minimal parts (e.g. water molecules, infinitesimals of continuous measure of weight, etc.)
- Eventualities (e.g. that time that bird ate that nut; regions of time where a proposition holds)
- Reified propositions (e.g. that a bird ate a nut, which we can be claimed, desired, asked, etc.)
- Numbers (e.g. 3, $\sqrt{3}$, etc.)

20.3 Lambda calculus notation [Church, 1940]

Propositions about entities are described using **lambda calculus** expressions of various types α, β:

- $\langle \alpha\text{-expr} \rangle \rightarrow \langle \beta{\rightarrow}\alpha\text{-expr} \rangle \ \langle \beta\text{-expr} \rangle$ – expressions can **apply** functions $\beta{\rightarrow}\alpha$ to arguments β.

  E.g.: BeingOdd 3 means *three is odd*.

  (Complex β are parenthesized for disambiguation: $\langle \alpha\text{-expr} \rangle \rightarrow \langle (\beta){\rightarrow}\alpha\text{-expr} \rangle \ \langle \beta\text{-expr} \rangle$.)
- $\langle \beta{\rightarrow}\alpha\text{-expr} \rangle \rightarrow \lambda_{\langle \beta\text{-var} \rangle} \ \langle \alpha\text{-expr} \rangle$ – expressions can **abstract** functions into input β, output α.

  E.g.: λ_x BeingANut $x \wedge$ BeingToxic x means *toxic nuts*.

  (Complex β are parenthesized for disambiguation: $\langle (\beta){\rightarrow}\alpha\text{-expr} \rangle \rightarrow \lambda_{\langle \beta\text{-var} \rangle} \ \langle \alpha\text{-expr} \rangle$.)
- $\langle \alpha\text{-expr} \rangle \rightarrow (\ \langle \alpha\text{-expr} \rangle\ )$ – expressions can be **parenthesized**.

Note: functions with one or more entities as input and a truth value as output are called **predicates**.

Note: functions with one entity as input and a truth value as output are also **sets**.

In general, expressions can ground out in a variety of constants and variables:

- $\langle \alpha\text{-expr} \rangle \rightarrow \langle$[A-Za-z0-9]+$\rangle$ – expressions can be defined as **constants**.

  E.g.: $\langle$e-expr$\rangle \rightarrow$ 3, $\langle$t-expr$\rangle \rightarrow$ True, $\langle$e$\rightarrow$t-expr$\rangle \rightarrow$ BeingANut, $\langle$e$\rightarrow$e$\rightarrow$t-expr$\rangle \rightarrow$ Eating.
- $\langle \alpha\text{-expr} \rangle \rightarrow \langle \alpha\text{-var} \rangle$ – expressions can be **variables**.
- $\langle \alpha\text{-var} \rangle \rightarrow \langle [a\text{-}z] \rangle$ – variables can be (italicized) letters.

  E.g.: $\langle$e-var$\rangle \rightarrow x$.

We'll also allow some familar function notations that don't fit this (prefix) format:

- $\langle$t-expr$\rangle \rightarrow \langle$t-expr$\rangle \ \wedge \ \langle$t-expr$\rangle$ – propositions can be (infix) **conjunctions**.

20.4 Generalized Quantifiers (derived functions) [Barwise & Cooper, 1981]

Most propositions involve **quantifiers**, to specify number/proportion of entities that hold properties.

Cardinal quantifiers compare counts of intersected restrictor R and nuclear scope S arguments:

- $\langle$e$\rightarrow$(e$\rightarrow$t)$\rightarrow$(e$\rightarrow$t)$\rightarrow$t-expr$\rangle \rightarrow$ Count$_<$ where $[\![\text{Count}_< \, n \, R \, S \,]\!]_m \Leftrightarrow |\, [\![R]\!]_m \cap [\![S]\!]_m \,| < n$
- $\langle$e$\rightarrow$(e$\rightarrow$t)$\rightarrow$(e$\rightarrow$t)$\rightarrow$t-expr$\rangle \rightarrow$ Count$_\leq$ where $[\![\text{Count}_\leq \, n \, R \, S \,]\!]_m \Leftrightarrow |\, [\![R]\!]_m \cap [\![S]\!]_m \,| \leq n$
- $\langle$e$\rightarrow$(e$\rightarrow$t)$\rightarrow$(e$\rightarrow$t)$\rightarrow$t-expr$\rangle \rightarrow$ Count$_=$ where $[\![\text{Count}_= \, n \, R \, S \,]\!]_m \Leftrightarrow |\, [\![R]\!]_m \cap [\![S]\!]_m \,| = n$
- $\langle$e$\rightarrow$(e$\rightarrow$t)$\rightarrow$(e$\rightarrow$t)$\rightarrow$t-expr$\rangle \rightarrow$ Count$_\geq$ where $[\![\text{Count}_\geq \, n \, R \, S \,]\!]_m \Leftrightarrow |\, [\![R]\!]_m \cap [\![S]\!]_m \,| \geq n$
- $\langle$e$\rightarrow$(e$\rightarrow$t)$\rightarrow$(e$\rightarrow$t)$\rightarrow$t-expr$\rangle \rightarrow$ Count$_>$ where $[\![\text{Count}_> \, n \, R \, S \,]\!]_m \Leftrightarrow |\, [\![R]\!]_m \cap [\![S]\!]_m \,| > n$

where $[\![\varphi]\!]_m$ means φ is true in some world model m.

Proportional quantifiers compare ratios of $R \cap S$ over R arguments:

- $\langle e{\to}(e{\to}t){\to}(e{\to}t){\to}t\text{-expr}\rangle \to \mathsf{Ratio}_{<}$ where $[\![\mathsf{Ratio}_{<}\ n\ R\ S]\!]_m \Leftrightarrow |[\![R]\!]_m \cap [\![S]\!]_m| \div |[\![R]\!]_m| < n$
- $\langle e{\to}(e{\to}t){\to}(e{\to}t){\to}t\text{-expr}\rangle \to \mathsf{Ratio}_{\leq}$ where $[\![\mathsf{Ratio}_{\leq}\ n\ R\ S]\!]_m \Leftrightarrow |[\![R]\!]_m \cap [\![S]\!]_m| \div |[\![R]\!]_m| \leq n$
- $\langle e{\to}(e{\to}t){\to}(e{\to}t){\to}t\text{-expr}\rangle \to \mathsf{Ratio}_{=}$ where $[\![\mathsf{Ratio}_{=}\ n\ R\ S]\!]_m \Leftrightarrow |[\![R]\!]_m \cap [\![S]\!]_m| \div |[\![R]\!]_m| = n$
- $\langle e{\to}(e{\to}t){\to}(e{\to}t){\to}t\text{-expr}\rangle \to \mathsf{Ratio}_{\geq}$ where $[\![\mathsf{Ratio}_{\geq}\ n\ R\ S]\!]_m \Leftrightarrow |[\![R]\!]_m \cap [\![S]\!]_m| \div |[\![R]\!]_m| \geq n$
- $\langle e{\to}(e{\to}t){\to}(e{\to}t){\to}t\text{-expr}\rangle \to \mathsf{Ratio}_{>}$ where $[\![\mathsf{Ratio}_{>}\ n\ R\ S]\!]_m \Leftrightarrow |[\![R]\!]_m \cap [\![S]\!]_m| \div |[\![R]\!]_m| > n$

Note: these last ones, in order to be defined for infinite sets, are defined on expected cardinalities of sampled subsets of entities D_K as sample size K tends to infinity:

$$|A| \div |B| \stackrel{\text{def}}{=} \lim_{K\to\infty} \mathsf{E}_{D_K \sim \pi}\, |D_K \cap A| \div |D_K \cap B|$$

The first argument of a quantifier is called the **restrictor**; the second is the **nuclear scope**.

If we assume some predicates have eventualities as arguments, these can be quantified as well:

$\mathsf{Most}\ (\lambda_x\ \mathsf{BeingANut}\ x)\ (\lambda_x\ \mathsf{No}\ (\lambda_e\ \mathsf{BeingAnEvent}\ e)\ (\lambda_e\ \mathsf{Growing}\ e\ x))$ (*Most nuts never grow.*)

Proportional quantifiers define **conditional probabilities** – a basis for probabilistic reasoning:

$\mathsf{Most}\ (\lambda_x\ \mathsf{BeingANut}\ x)\ (\lambda_x\ \mathsf{BeingEdible}\ x) \quad \Leftrightarrow \quad \mathsf{P}(\mathsf{BeingEdible}\ x\,|\,\mathsf{BeingANut}\ x) > .5$

20.5 Scope

Note that quantifier expressions can occur ('scope') as arguments of other quantifier expressions.

For example, assuming $\mathsf{Most}\ R\ S \Leftrightarrow \mathsf{Ratio}_{>}\ 0.5\ R\ S$, and $\mathsf{Some}\ R\ S \Leftrightarrow \mathsf{Count}_{>}\ 0\ R\ S$:

Most cars have four wheels.

can be represented as:

$$\begin{array}{l}\mathsf{Most}\ (\lambda_c\ \mathsf{BeingACar}\ c)\\ \quad (\lambda_c\ \mathsf{Count}_{=}\ 4\ (\lambda_w\ \mathsf{BeingAWheel}\ w)\\ \quad\quad (\lambda_w\ \mathsf{Some}\ (\lambda_h\ \mathsf{Having}\ h\ c\ w)\\ \quad\quad\quad (\lambda_h\ \mathsf{true})))\end{array}$$

More complex: assuming $\mathsf{Some}\ R\ S \Leftrightarrow \mathsf{Count}_{>}\ 0\ R\ S$ and $\mathsf{All}\ R\ S \Leftrightarrow \mathsf{Ratio}_{=}\ 1.0\ R\ S$:

A company will (always) pay (anyone) 45 cents each for 10,000 bricks.

can be represented as:

$$\begin{array}{l}\text{Some } (\lambda_c \text{ BeingACompany } c)\\ \quad (\lambda_c \text{ Count}_{=} \text{ 10000 } (\lambda_b \text{ BeingABrick } b)\\ \qquad (\lambda_b \text{ All } (\lambda_o \text{ BeingAOne } o)\\ \qquad\quad (\lambda_o \text{ All } (\lambda_g \text{ Giving } g\ o\ b\ c)\\ \qquad\qquad (\lambda_g \text{ Count}_{=} \text{ 45 } (\lambda_p \text{ BeingAPenny } p)\\ \qquad\qquad\quad (\lambda_p \text{ Some } (\lambda_h \text{ Giving } h\ c\ p\ o)\\ \qquad\qquad\qquad (\lambda_h \text{ Causing } g\ h))))))\end{array}$$

Note that the meaning changes a lot when the quantifiers are a bit different:

$$\begin{array}{l}\text{Some } (\lambda_c \text{ BeingACompany } c)\\ \quad (\lambda_c \text{ Count}_{=} \text{ 45 } (\lambda_p \text{ BeingAPenny } p)\\ \qquad (\lambda_p \text{ Count}_{=} \text{ 10000 } (\lambda_b \text{ BeingABrick } b)\\ \qquad\quad (\lambda_b \text{ All } (\lambda_o \text{ BeingAOne } o)\\ \qquad\qquad (\lambda_o \text{ All } (\lambda_g \text{ Giving } g\ o\ b\ c)\\ \qquad\qquad\quad (\lambda_g \text{ Some } (\lambda_h \text{ Giving } h\ c\ p\ o)\\ \qquad\qquad\qquad (\lambda_h \text{ Causing } g\ h))))))\end{array}$$

This is very different from vectoral semantics where small differences are smoothed away!

20.6 Reified propositions and natural logic subsumption

We can associate variables with propositions by adding them as arguments to quantifiers:

$$\langle \text{e}{\to}\text{e}{\to}(\text{e}{\to}\text{t}){\to}(\text{e}{\to}\text{t}){\to}\text{t-expr}\rangle \to \text{ReifRatio}_o \qquad \text{ReifRatio}_o\ p\ n\ R\ S \Leftrightarrow p = \{m \mid [\![\text{Ratio}_o\ n\ R\ S\ ]\!]_m\}$$
$$\langle \text{e}{\to}\text{e}{\to}(\text{e}{\to}\text{t}){\to}(\text{e}{\to}\text{t}){\to}\text{t-expr}\rangle \to \text{ReifCount}_o \qquad \text{ReifCount}_o\ p\ n\ R\ S \Leftrightarrow p = \{m \mid [\![\text{Count}_o\ n\ R\ S\ ]\!]_m\}$$

where a subsumption predicate ($p \sqsupseteq q$) is defined to be a weak form of entailment:

$$\begin{array}{rcl}\{\langle m, ..\rangle \mid [\![\varphi]\!]_m\} \sqsupseteq \{\langle m, ..\rangle \mid [\![\varphi \wedge \psi]\!]_m\} & & \\ \{\langle m, ..\rangle \mid [\![Q_{\geq}\ n\ R\ S\ ]\!]_m\} \sqsupseteq \{\langle m, ..\rangle \mid [\![Q_{\geq}\ n'\ R\ S\ ]\!]_m\} & \Leftrightarrow & n' > n \\ \{\langle m, ..\rangle \mid [\![Q_{\leq}\ n'\ R\ S\ ]\!]_m\} \sqsupseteq \{\langle m, ..\rangle \mid [\![Q_{\leq}\ n\ R\ S\ ]\!]_m\} & \Leftrightarrow & n' > n \\ \{\langle m, ..\rangle \mid [\![Q_{\geq}\ n\ R\ (\lambda_x\ \varphi)]\!]_m\} \sqsupseteq \{\langle m, ..\rangle \mid [\![Q_{\geq}\ n\ R\ (\lambda_x\ \psi)]\!]_m\} & \Leftrightarrow & \{\langle m, .., x\rangle \mid [\![\varphi]\!]_m\} \sqsupseteq \{\langle m, .., x\rangle \mid [\![\psi]\!]_m\} \\ \{\langle m, ..\rangle \mid [\![Q_{\leq}\ n\ R\ (\lambda_x\ \psi)]\!]_m\} \sqsupseteq \{\langle m, ..\rangle \mid [\![Q_{\leq}\ n\ R\ (\lambda_x\ \varphi)]\!]_m\} & \Leftrightarrow & \{\langle m, .., x\rangle \mid [\![\varphi]\!]_m\} \sqsupseteq \{\langle m, .., x\rangle \mid [\![\psi]\!]_m\}\end{array}$$

where $Q \in \{\text{Count}, \text{Ratio}\}$ and $\langle m, x, x', ..\rangle$ is a tuple of a world model m and entities $x, x', .. \in m$.

(Note that these, unlike full entailment, can be evaluated just using the form of the quantifier!)

This kind of weak entailment is sometimes called **natural logic** [van Benthem, 1986].

These subsumption predicates can be used to evaluate whether something was said, for example.

20.7 Other operators

Generalized quantifiers are powerful enough that we don't need other operators.

We can use a 'None' quantifier (and a uniquely satisfied predicate 'Unit') to implement **negation**:

$$(\neg\ \mathsf{ItsRainy}) \ \Leftrightarrow\ (\mathsf{Count}_{=}\ 0\ (\lambda_x\ \mathsf{Unit}\ x)\ (\lambda_x\ \mathsf{ItsRainy}))$$

We can use negation to implement **disjunction** (via DeMorgan's law):

$$(\mathsf{ItsCloudy} \vee \mathsf{ItsSunny}) \ \Leftrightarrow\ (\neg\ ((\neg\ \mathsf{ItsCloudy}) \wedge (\neg\ \mathsf{ItsSunny})))$$

And we can use disjunction to implement **implication** (via double negation law):

$$(\mathsf{ItsRainy} \rightarrow \mathsf{ItsCloudy}) \ \Leftrightarrow\ ((\neg\ \mathsf{ItsRainy}) \vee \mathsf{ItsCloudy})$$

or more directly using an 'All' quantifier:

$$(\mathsf{ItsRainy} \rightarrow \mathsf{ItsCloudy}) \ \Leftrightarrow\ (\mathsf{All}\ (\lambda_x\ \mathsf{Unit}\ x \wedge \mathsf{ItsRainy})\ (\lambda_x\ \mathsf{ItsCloudy}))$$

This simplifies our natural logic subsumption!

20.8 World models

World models are like 'game engines' that represents the world for the interlocutor agent.

It can be expected to:

- represent count entities.
- represent (blobs of) infinitesimal or other overly numerous entities.
- represent relations (or cover relations, if partial-order like containinment).
- represent onset and offset times for temporary relations, to calculate relations at other times.
- calculate subsumptions over reified propositions (using lambda calculus form).

20.9 Mapping from syntax

We compose lambda calculus expressions the same way we build trees, using semiring substitution.

First intialize unit spans with triples of probability, lambda term and syntactic category:

$$v_{\top}(w) = \begin{cases} \langle \mathsf{P}(\mathsf{most}\,|\,\mathbf{N\text{-}b(N\text{-}aD)}, \ldots),\ \lambda_R\ \lambda_S\ \mathsf{Most}\ R\ S,\ \mathbf{N\text{-}b(N\text{-}aD)} \rangle & \text{if } w = \mathsf{most} \\ \langle \mathsf{P}(\mathsf{shapes}\,|\,\mathbf{N\text{-}aD}, \ldots),\ \lambda_x\ \mathsf{BeingAShape}\ x,\ \mathbf{N\text{-}aD} \rangle & \text{if } w = \mathsf{shapes} \\ \langle \mathsf{P}(\mathsf{are}\,|\,\mathbf{V\text{-}aN\text{-}b(A\text{-}aN)}, \ldots),\ \lambda_P\ \lambda_Q\ P\ Q,\ \mathbf{V\text{-}aN\text{-}b(A\text{-}aN)} \rangle & \text{if } w = \mathsf{are} \\ \langle \mathsf{P}(\mathsf{red}\,|\,\mathbf{A\text{-}aN}, \ldots),\ \lambda_Q\ Q\ (\lambda_x\ \mathsf{BeingRed}\ x),\ \mathbf{A\text{-}aN} \rangle & \text{if } w = \mathsf{red} \\ \langle \mathsf{P}(\mathsf{square}\,|\,\mathbf{A\text{-}aN}, \ldots),\ \lambda_Q\ Q\ (\lambda_x\ \mathsf{BeingSquare}\ x),\ \mathbf{A\text{-}aN} \rangle & \text{if } w = \mathsf{square} \\ \vdots \end{cases}$$

(Recall $v_\top$ is just the semiring identity for $\otimes$, now extended to depend on word w.)

We then extend 'prod_pair' to compose terms $a = \langle p_a,\ \varphi_a,\ c_a \rangle$ and $b = \langle p_b,\ \varphi_b,\ c_b \rangle$:

$$a \otimes b = \begin{cases} \langle \mathsf{P(Aa,}\, \upsilon \,|\, \tau, \ldots) \cdot p_a \cdot p_b,\ \varphi_b\, \varphi_a,\ \tau \rangle & \text{if } c_a = \upsilon,\ c_b = \tau\text{-}\mathbf{a}\upsilon \\ \langle \mathsf{P(Ab,}\, \upsilon \,|\, \tau, \ldots) \cdot p_a \cdot p_b,\ \varphi_a\, \varphi_b,\ \tau \rangle & \text{if } c_a = \tau\text{-}\mathbf{b}\upsilon,\ c_b = \upsilon \\ \langle \mathsf{P(Ma,}\, \upsilon \,|\, \tau, \ldots) \cdot p_a \cdot p_b,\ \lambda_x\, \varphi_a\, (\lambda_P\, P\, x) \wedge \varphi_b\, x,\ \tau \rangle & \text{if } c_a = \upsilon\text{-}\mathbf{a}\mathbf{N},\ c_b = \tau \\ \langle \mathsf{P(Mb,}\, \upsilon \,|\, \tau, \ldots) \cdot p_a \cdot p_b,\ \lambda_x\, \varphi_b\, (\lambda_P\, P\, x) \wedge \varphi_a\, x,\ \tau \rangle & \text{if } c_a = \tau,\ c_b = \upsilon\text{-}\mathbf{a}\mathbf{N} \\ \vdots \end{cases}$$

where Aa, Ab, Ma, Mb, ... are just names for the corresponding operations on lambda terms.

To simplify logical forms, we'll define substitution in function application using **beta reduction**:

$$(\lambda_A\ \ldots A \ldots A \ldots A \ldots)\ B \ \ = \ \ \ldots B \ldots B \ldots B \ldots$$

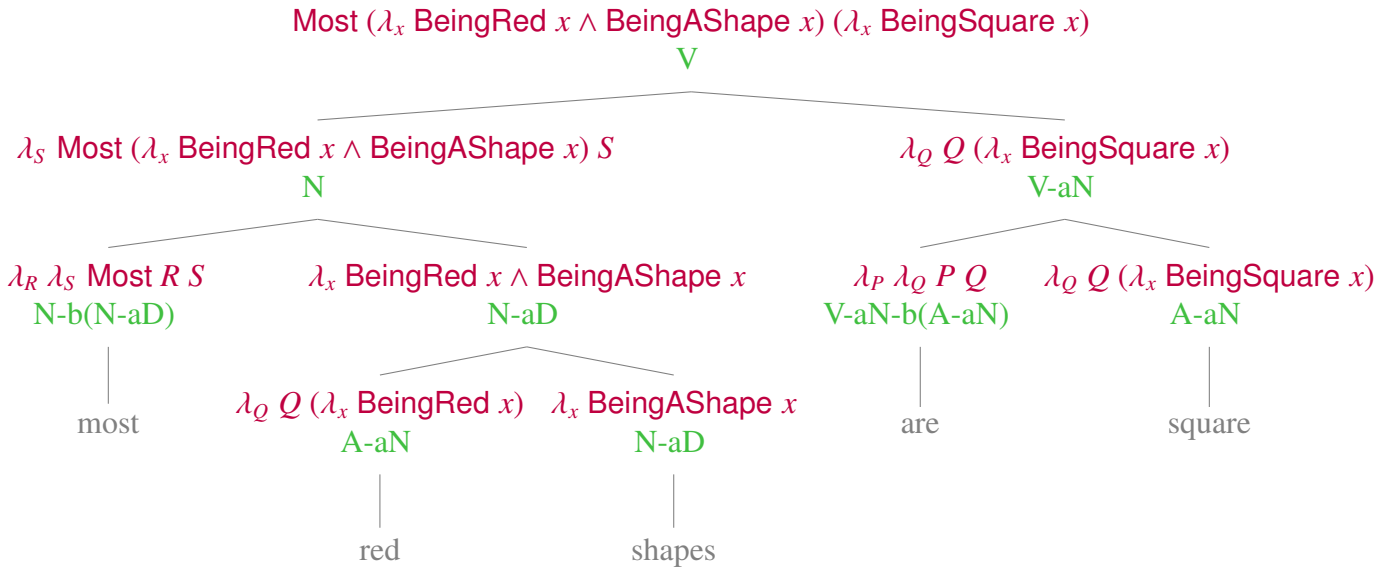
For example (as happens in '*red shapes*' below):

$$\begin{aligned} (\lambda_Q\ Q\ (\lambda_x\ \mathsf{BeingRed}\ x))\ (\lambda_P\ P\ x) &= (\lambda_P\ P\ x)\ (\lambda_x\ \mathsf{BeingRed}\ x) \\ &= (\lambda_x\ \mathsf{BeingRed}\ x)\ x \\ &= \mathsf{BeingRed}\ x \end{aligned}$$

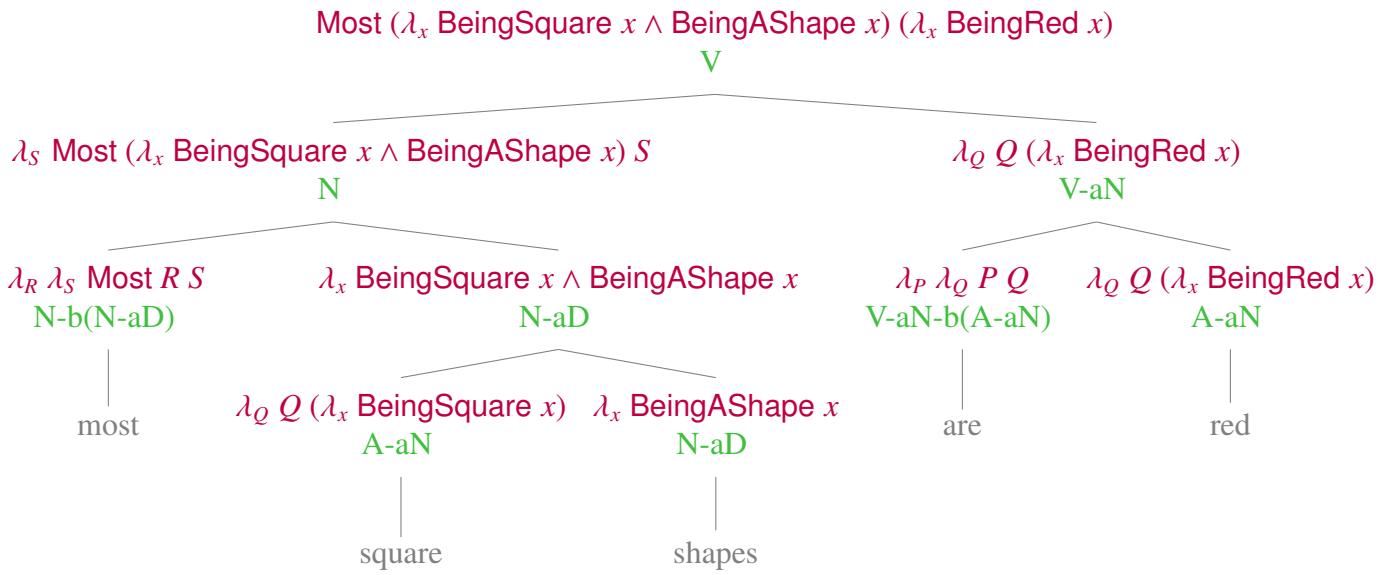
Another example (as happens in '*are square*' below):

$$\begin{aligned} (\lambda_P\ \lambda_Q\ P\ Q)\ (\lambda_S\ S\ (\lambda_x\ \mathsf{BeingSquare}\ x)) &= \lambda_Q\ (\lambda_S\ S\ (\lambda_x\ \mathsf{BeingSquare}\ x))\ Q \\ &= \lambda_Q\ Q\ (\lambda_x\ \mathsf{BeingSquare}\ x) \end{aligned}$$

Example translation of '*Most red shapes are square.*':



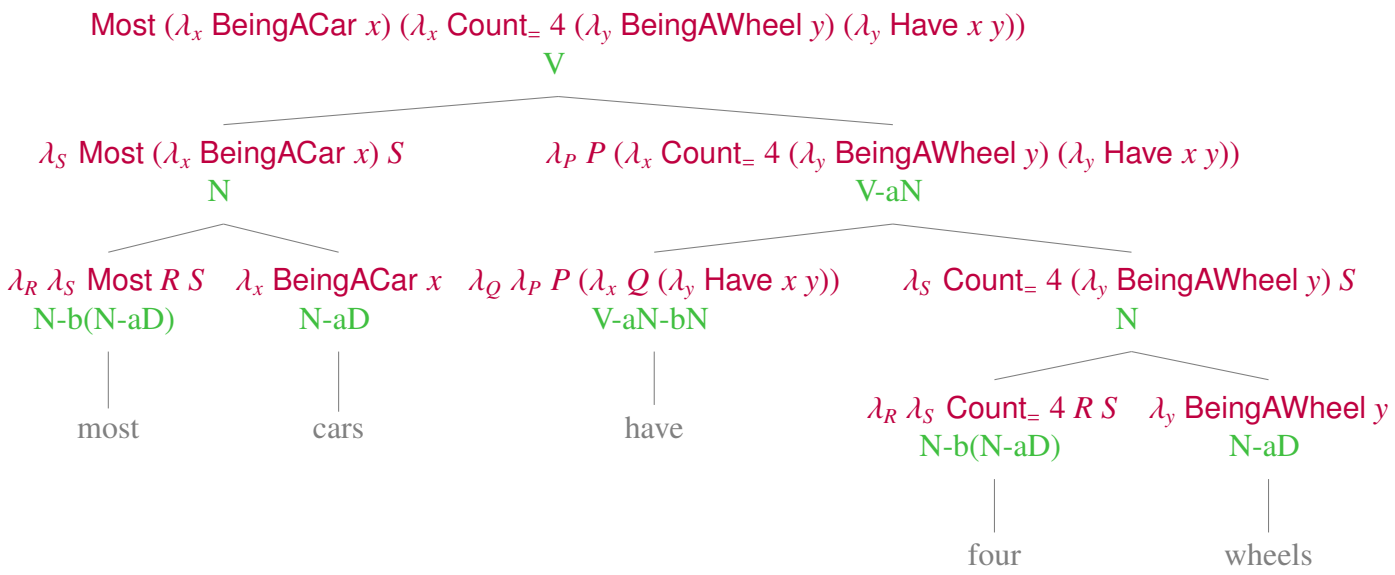
as distinct from '*Most square shapes are red.*':



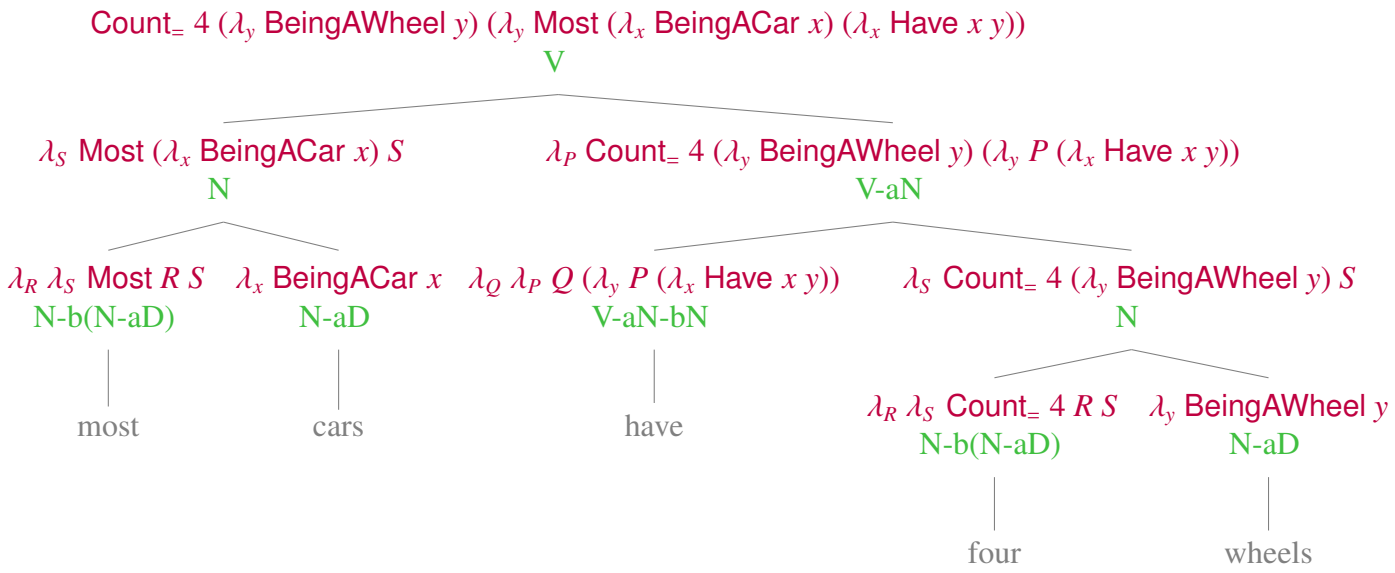
We can also model (local) scope inversion:

$$v_{\mathsf{T}}(w) = \begin{cases} \vdots & \\ \langle \mathsf{P}(\text{have} \,|\, \textbf{V-aN-bN}, \ldots),\ \lambda_Q\ \lambda_P\ P\ (\lambda_x\ Q\ (\lambda_y\ \mathsf{Having}\ x\ y)),\ \textbf{V-aN-bN} \rangle & \text{if } w = \text{have} \\ \langle \mathsf{P}(\text{have} \,|\, \textbf{V-aN-bN}, \ldots),\ \lambda_Q\ \lambda_P\ Q\ (\lambda_y\ P\ (\lambda_x\ \mathsf{Having}\ x\ y)),\ \textbf{V-aN-bN} \rangle & \text{if } w = \text{have} \\ \langle \mathsf{P}(\text{four} \,|\, \textbf{N-b(N-aD)}, \ldots),\ \lambda_R\ \lambda_S\ \mathsf{Count}_{=}\ 4\ R\ S,\ \textbf{N-b(N-aD)} \rangle & \text{if } w = \text{four} \\ \langle \mathsf{P}(\text{cars} \,|\, \textbf{N-aD}, \ldots),\ \lambda_x\ \mathsf{BeingACar}\ x,\ \textbf{N-aD} \rangle & \text{if } w = \text{cars} \\ \langle \mathsf{P}(\text{wheels} \,|\, \textbf{N-aD}, \ldots),\ \lambda_y\ \mathsf{BeingAWheel}\ y,\ \textbf{N-aD} \rangle & \text{if } w = \text{wheels} \\ \vdots & \end{cases}$$

Example translation of '*Most cars have four wheels*' with usual scope:



Example translation of '*Most cars have four wheels*' with inverted scope:



We can also model conjunction:

$$a \otimes b = \begin{cases} \vdots & \\ \langle \mathrm{P(Ca} \,|\, \tau, \ldots) \cdot p_a \cdot p_b,\ \lambda_Q\, \varphi_a\, Q \wedge \varphi_b\, Q,\ \tau \rangle & \text{if } c_a = \tau,\ c_b = \tau\text{-}\mathbf{c}\tau \\ \langle \mathrm{P(Cb} \,|\, \tau\text{-}\mathbf{c}\tau, \ldots) \cdot p_a \cdot p_b,\ \lambda_Q\, \varphi_a\, Q \wedge \varphi_b\, Q,\ \tau\text{-}\mathbf{c}\tau \rangle & \text{if } c_a = \tau\text{-}\mathbf{c}\tau\text{-}\mathbf{d}\tau,\ c_b = \tau \\ \vdots & \end{cases}$$

Not covered yet:

- Coreference (may have to be separately inferred).
- Scope (may have to be separately inferred).

References

Barwise, J., & Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, *4*.

Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, *5*(2), 56–68.

van Benthem, J. (1986). Natural logic. In *Essays in logical semantics.* Dordrecht, the Netherlands: Kluwer.

von Neumann, J., & Morgenstern, O. (1944). Theory of games and economic behavior. *Science and Society*, *9*(4), 366–369.