Ling 5801: Lecture Notes 14
Decision Process Semantics

We've 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 what about statements in a natural language like English?

A company in Boston will pay 45 cents for each brick.

These statements are not things to do; they're more like ***rules*** to use in planning.

we have n bricks in Boston $\rightarrow$ we have $45 \cdot n$ cents

For example, if we only know the following, we might initially plan to sell our bricks in Columbus:

We have 1000 bricks in Columbus.
Most companies will pay 40 cents for each brick.
Each brick weighs 4 pounds.
It costs .002 cents to move a pound of cargo one mile.
Boston is 700 miles from Columbus.

But if we are told about the Boston company above, we might prefer to ship our bricks there!

14.1 Lambda expressions

Linguists typically define the meanings or *semantics* of sentences using *lambda expressions*.

Lambda expressions consist of the following:

- **variables**: x – terms that can be bound to various input values

  (Similar to variables that serve as parameters in Python functions.)

- **abstractions**: $\lambda_x \ \ldots x \ldots$ – un-named functions with parameter x that return value $\ldots x \ldots$.

  For example: $(\lambda_n \ n/2)$ takes a number and returns half of that number.

- **applications**: $f \ g$ – passing term g as an input to function f.

  For example: $(\lambda_n \ n/2) \ 6$ applies the above function to 6, returning 3.

Typically linguistic meanings also assume **term types**, similar to class types in Python:

- a type for **entity** objects, like bricks and cents.
- a type for **elementary predications** or event objects, which are entities that have participants.
- a type for **truth value** objects: specifically true and false.

- types for **predicate** functions, like Giving, from eventualities and entities to truth values.

  (These predicates can define **sets** of entities that return true.)

- a type for **quantifier** functions, like All, from sets to functions from sets to truth values.

  (For example, All Cat Pet returns true if the set of cats is a subset of the set of pets.)

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

For example, if we assume functions for predicates BeingACar, BeingAWheel, and Having, and quantifiers Most and Four, we can represent:

Most cars have four wheels.

as:

$$\begin{array}{l}\text{Most } (\lambda_c \text{ BeingACar } c)\\ \quad (\lambda_{c'} \text{ Four } (\lambda_w \text{ BeingAWheel } w)\\ \quad\quad (\lambda_{w'} \text{ Some } (\lambda_h \text{ Having } h\ c'\ w')\\ \quad\quad\quad (\lambda_{h'} \text{ true})))\end{array}$$

Or, if we assume functions for predicates BeingACompany, BeingABrick, Giving, Causing, etc., and quantifiers Some, TenThousand, FortyFive, and All, we can represent:

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

as:

$$\begin{array}{l}\text{Some } (\lambda_c \text{ BeingACompany } c)\\ \quad (\lambda_{c'} \text{ TenThousand } (\lambda_b \text{ BeingABrick } b)\\ \quad\quad (\lambda_{b'} \text{ Some } (\lambda_o \text{ BeingAOne } o)\\ \quad\quad\quad (\lambda_{o'} \text{ FortyFive } (\lambda_p \text{ BeingAPenny } p)\\ \quad\quad\quad\quad (\lambda_{p'} \text{ All } (\lambda_g \text{ Giving } g\ o'\ b'\ c')\\ \quad\quad\quad\quad\quad (\lambda_{g'} \text{ Some } (\lambda_h \text{ Giving } h\ c'\ p'\ o')\\ \quad\quad\quad\quad\quad\quad (\lambda_{h'} \text{ Causing } g'\ h'))))))\end{array}$$

Note that quantifiers like Some and Four can be generalized as $\text{Cardinality}_{\geq}\ 1$ and $\text{Cardinality}_{\geq}\ 4$, and quantifiers like Most and All can be generalized as $\text{Ratio}_{>}\ 0.5$ and $\text{Ratio}_{\geq}\ 1.0$.

Also note that expressions with ratio quantifiers can be used on novel entities as rules in planning.

14.2 Semantic structures

Sentences can be decoded into lambda expressions using a graphical representation.

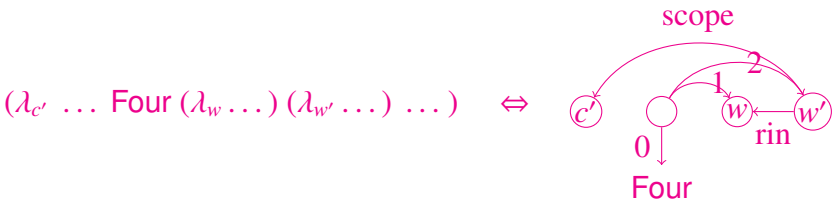
Vertices represent variables over entities and elementary predications.

Vertices for variables over elem. predications have numbered edges to predicates and participants:

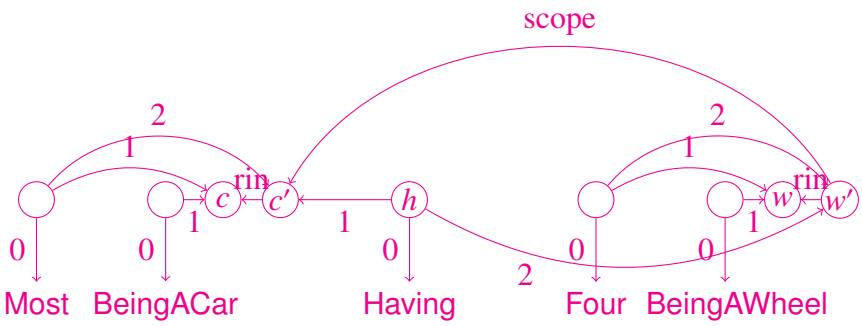
Quantifiers can be similarly decomposed (although there are no variables over quantifications):

A ‘restriction inheritance’ edge inherits constraints from the first set of a quantifier to second set.

An additional ‘scope’ edge indicates the nesting structure (parent) of each quantified variable:



Complete tree:



14.3 Are semantic structures real?

Maybe... (Rasmussen & Schuler, 2018)

Human associative memory can be modeled graphically:

- vertices: **activation patterns in cortex** define referential states in attentional focus
  (modeled as **vectors** of activation for each neuron or neural cluster)
- edges: **potentiation in hippocampal neurons** define cued associations between these patterns
  (modeled as **matrices** of synaptic sensitivities between neurons or neural clusters)
  (weighted by sum of outer products of associated vectors)

Recall moves attentional focus vector from referential state to referential state.

Consistent with observations of patients with hippocampal lesions having no storage of semantics.

- Shallow processing still possible if semantic context features bleed into activation patterns.

References

Rasmussen, N. E., & Schuler, W. (2018). Left-corner parsing with distributed associative memory produces surprisal and locality effects. *Cognitive Science*, *42*(S4), 1009–1042. Retrieved from `http://dx.doi.org/10.1111/cogs.12511` doi: 10.1111/cogs.12511