LING3804: Lecture Notes 8
Logical Reasoning

Humans seem to intend precise meanings for sentences they utter, with well-defined entailments.

Linguists usually represent these meanings using unambiguous formal logical expressions.

This lecture defines a very common logic and maps it to a neural mechanism for logical entailment.

Contents

8.1 Basic parts of meaning (ontology) (Church, 1940; Carnap, 1947)

Linguistic meaning is usually defined in terms of sets of **values** for each of a set of **types**:

1. **truth values** (type `t`): judgments about statements in particular real or hypothetical worlds, simple examples: True (tautology), False (contradiction);
2. **entities** (type `e`): referents of proper nouns in particular real or hypothetical worlds, examples: Person43, Course101, GrainOfSand5324, InfinitesimalOfVolume3387, . . . ;
3. **functions** (type $(\alpha)\rightarrow\beta$ for any other types α and β, including other functions):
   examples: Prof of type `e→t`, Teach of type `(e)→(e)→t`, . . .
   (functions of type `(e)→t` or `(e)→(e)→t` or `(e)→(e)→(e)→t`, . . . are called **predicates**)
   (predicates of type `(e)→t` are called **characteristic functions of sets**, or just **sets**);
4. **propositions** (type `p`): partial descriptions of desired, believed or claimed world states, examples: ItsSnowing, . . . (these can also be entire logical expressions, as we'll see later);
5. **individuals** (type `i`): referents that may or may not be distinct in various hypothetical worlds, examples: BruceWayne, Batman, . . . (they're just names, in case you don't like predicates).

These values are not necessarily represented in the brain nor mentioned in logical expressions. They're just objects we can describe in whatever real or hypothetical world we may reason about.

8.2 Logical expressions (Church, 1940)

Now we can define rules to build tree-like logical expressions, as with syntax in natural language. These rules will use the above types, just like syntax rules used syntactic categories.

We start with two kinds of simple expression of any type γ, shown as schematized grammar rules:

1. **constants** $\overset{\gamma}{\underset{\varphi}{\wedge}}$ — examples of φ of type γ: Person43 of type e, Prof of type e→t, ...

2. **variables** $\overset{\gamma}{\underset{\chi}{\wedge}}$ — examples of χ of type γ: x, y of type e, ...

We'll also define two basic rules to build complex expressions for any types α and β:

3. **applications** $\overset{\beta}{\underset{\varphi}{(\alpha)\to\beta}}\ \overset{}{\underset{\psi}{\alpha}}$ of functions φ of type $(\alpha)\to\beta$ to arguments ψ of type α:

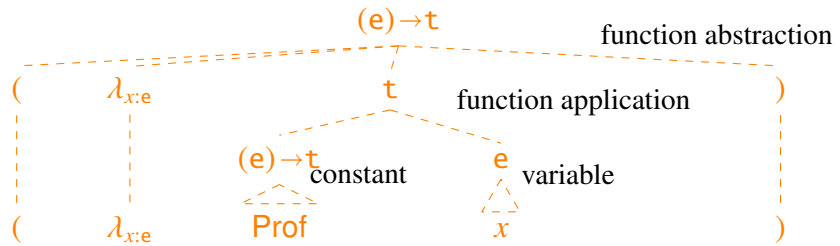
   example $\varphi\,\psi$ of type β: Prof Person43 of type t, where Prof is type (e)→t and Person43 is e,

4. **abstractions** $\overset{(\alpha)\to\beta}{(\ \lambda_{\chi:\alpha}\ \underset{\varphi}{\beta}\ )}$ defining functions on input variables χ of type α with output φ:

   example of $(\lambda_{x:\alpha}\ \varphi)$ of type $(\alpha)\to\beta$: $(\lambda_{x:\mathrm{e}}$ True) of type (e)→t, where True is type t.

This is called **lambda calculus** because of the abstraction operator.

We can draw trees for expressions in lambda calculus, just like in natural languages like English:



Practice 8.1:

Assuming:

- Prof is of type $(\mathtt{e}) \to \mathtt{t}$, and
- Cardinality is of type $((\mathtt{e}) \to \mathtt{t}) \to \mathtt{e}$,

draw a tree for the following lambda calculus expression, showing type labels on all branches:

$$\text{Cardinality } (\lambda_{x:\mathtt{e}} \text{ Prof } x).$$

8.3 Simplification by variable substitution using beta reduction

We can simplify application of function abstractions via **beta reduction**.

This is just variable substitution: $(\lambda_{\chi:\alpha}\ \varphi)\ \omega = [\varphi]_{\chi \mapsto \omega}$, replacing χ with ω via the following rules:

$$[\varphi]_{\chi\mapsto\omega} = \varphi \quad \text{if } \varphi \text{ is a constant,} \qquad [\varphi\ \psi]_{\chi\mapsto\omega} = [\varphi]_{\chi\mapsto\omega}\ [\psi]_{\chi\mapsto\omega},$$
$$[\varphi]_{\chi\mapsto\omega} = \varphi \quad \text{if } \varphi \text{ is a variable and } \varphi \neq \chi, \qquad [(\lambda_{\psi:\beta}\ \varphi)]_{\chi\mapsto\omega} = (\lambda_{\psi:\beta}\ [\varphi]_{\chi\mapsto\omega}) \quad \text{if } \psi \neq \chi,$$
$$[\chi]_{\chi\mapsto\omega} = \omega, \qquad [(\lambda_{\chi:\alpha}\ \varphi)]_{\chi\mapsto\omega} = (\lambda_{\chi:\alpha}\ \varphi).$$

For example:

$$\begin{aligned}
(\lambda_{x:\mathtt{e}}\ \text{Like}\ x\ x)\ \text{Person43} &= [\text{Like}\ x\ x]_{x\mapsto\text{Person43}} && \text{by definition}\\
&= [\text{Like}\ x]_{x\mapsto\text{Person43}}\ [x]_{x\mapsto\text{Person43}} && \text{function application}\\
&= [\text{Like}\ x]_{x\mapsto\text{Person43}}\ \text{Person43} && \text{variable matching substituendum}\\
&= [\text{Like}]_{x\mapsto\text{Person43}}\ [x]_{x\mapsto\text{Person43}}\ \text{Person43} && \text{function application}\\
&= [\text{Like}]_{x\mapsto\text{Person43}}\ \text{Person43}\ \text{Person43} && \text{variable matching substituendum}\\
&= \text{Like Person43 Person43} && \text{constant}
\end{aligned}$$

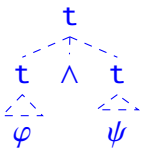
Practice 8.2:

Beta reduce the following expression: $(\lambda_{x:\mathtt{e}}\ x)$ Person43.

8.4 Logical functions: conjunction and negation

Some functions (on truth values) are so useful, they're just considered part of the logic.

We have an 'infix' rule for conjunction (we could use a regular 'prefix' function, but that's messier):

5. **conjunctions**  of conjuncts φ and ψ of type $\mathtt{t}$:

   example of $\varphi \wedge \psi$: Prof Person43 $\wedge$ True of type $\mathtt{t}$, where both conjuncts are type $\mathtt{t}$.

   Conjunctions also accommodate beta reduction: $[\varphi \wedge \psi]_{\chi \mapsto \omega} = [\varphi]_{\chi \mapsto \omega} \wedge [\psi]_{\chi \mapsto \omega}$.

We also have a special symbol for negation:

6. **negations**  of statement φ of type $\mathtt{t}$:

   example of $\neg\, \varphi$: $\neg$ Prof Person43 of type $\mathtt{t}$, where argument statement is of type $\mathtt{t}$.

   Negations also accommodate beta reduction: $[\neg\, \varphi]_{\chi \mapsto \omega} = \neg\, [\varphi]_{\chi \mapsto \omega}$.

Practice 8.3:

Assuming:

- Prof is of type $(\mathtt{e}) \to \mathtt{t}$,
- Person43 is of type $\mathtt{e}$, and
- True is of type $\mathtt{t}$,

draw a tree for the following lambda calculus expression, showing type labels on all branches:

$$\text{Prof Person43} \ \wedge \ \text{True}.$$

Practice 8.4:

Beta reduce the following expression: $(\lambda_{x:\mathtt{e}}\ \text{Prof}\ x\ \wedge\ \text{True})\ \text{Person43}$.

8.5 Entailment (Hilbert & Bernays, 1934; Gentzen, 1935)

Generally we have two ways of acquiring new knowledge as logical propositions:

- from other people, via language (we'll discuss that in a subsequent lecture),
- from existing knowledge (other logical propositions, often called a 'theory'), via entailment.

Entailment $\pi_1, \pi_2, \ldots \Rightarrow \kappa$ is a process of inferring conclusion statements κ from premises π_1, π_2, ... These entailments can be chained up in a proof, just like equations in a mathematical derivation.

The 'theory' which provides premises for logical entailment is analogous to 'context' in an LLM. A proof is then analogous to a 'chain of thought' output of an LLM, as we'll see.

We can define two simple entailments involving conjunction and negation:

- **conjunction elimination (CE)** — for all φ, ψ of type `t`:

$$\varphi \wedge \psi \;\Rightarrow\; \varphi \qquad \varphi \wedge \psi \;\Rightarrow\; \psi$$

- **negated conjunction introduction (NCI)** — for all φ, ψ of type `t`:

$$\neg\,\varphi \;\Rightarrow\; \neg\,(\varphi \wedge \psi) \qquad \neg\,\psi \;\Rightarrow\; \neg\,(\varphi \wedge \psi)$$

Here they are used in a simple proof:

$$\text{Prof Kim} \wedge \text{Adult Pat} \;\Rightarrow\; \text{Prof Kim} \qquad \text{(CE)}$$

These entailment rules follow from axioms of **intuitionistic logic** (Brouwer, 1907; Heyting, 1930). Defining meaning by constructed proof is **proof theory** (Hilbert & Bernays, 1934; Gentzen, 1935).

8.6 Finite cardinality and counting (Heyting, 1930)

One of the most fundamental things we can reason about is **cardinality**: the sizes of sets of entities.

First, if our theory introduces K entity constants, we add new entity constants for numbers 0 to K. (We also add equality and arithmetic operators to add, subtract, and multiply, with usual meanings.) Then we define a total order on this expanded set of entities and define a partial count predicate:

- Infimum/Supremum: add new entity constants for the first/last entity in sorted order,
- Successor χ: the next entity in sorted order after χ (and the supremum succeeds itself),
- PartialCount $\kappa\ \sigma\ \chi$: asserts that set σ includes at least κ elements counting up to element χ.

where the following entailments define the partial count predicate (the first is an axiom):

$$\Rightarrow \text{PartialCount } 0\ \sigma\ \text{Infimum} \qquad \text{(CIntro)}$$

$$\text{PartialCount } \kappa\ \sigma\ \chi \Rightarrow \text{PartialCount } \kappa\ \sigma\ (\text{Successor } \chi) \qquad \text{(CHold)}$$

$$\sigma\ (\text{Successor}\ \chi) \wedge \text{PartialCount}\ \kappa\ \sigma\ \chi \Rightarrow \text{PartialCount}\ (\kappa + 1)\ \sigma\ (\text{Successor}\ \chi) \qquad \text{(CAdd)}$$

So, for the expressions of type e in the theory Prof Kim ∧ Adult Pat, we conjoin to our theory:

$$\begin{aligned}\text{Successor Infimum} &= \text{Kim} \wedge\\ \text{Successor Kim} &= \text{Pat} \wedge\\ \text{Successor Pat} &= 0 \wedge\\ \text{Successor } 0 &= 1 \wedge\\ \text{Successor } 1 &= 2 \wedge\\ \text{Successor } 2 &= \text{Supremum},\end{aligned}$$

and for the expressions of type (e)→t in this theory, we conjoin:

$$\begin{aligned}&\text{PartialCount } 0 \text{ Prof Infimum} \wedge\\ &\text{PartialCount } 0 \text{ Adult Infimum}.\end{aligned}$$

If we aren't given all properties of all entities, we can't count set members exactly, so we define:

- CardinalityMinorant $\kappa\ \sigma$: asserts κ is a minorant (lower bound) on the cardinality of σ,
- CardinalityMajorant $\kappa\ \sigma$: asserts κ is a majorant (upper bound) on the cardinality of σ.

defined in terms of count, where K is the number of entity constants introduced in our theory:

$$\text{CardinalityMinorant}\ \kappa\ \sigma \Leftrightarrow \text{PartialCount}\ \kappa\ \sigma\ \text{Supremum} \qquad \text{(CMin)}$$
$$\text{CardinalityMajorant}\ \kappa\ \sigma \Leftrightarrow \text{PartialCount}\ (K - \kappa)\ (\lambda_{x:\mathsf{e}}\ \neg\ \sigma\ x)\ \text{Supremum} \qquad \text{(CMaj)}$$

E.g., for a theory Prof Kim ∧ Adult Pat, we can infer a tight minimum cardinality for Prof:

$$\begin{aligned}\text{PartialCount } 0 \text{ Prof Infimum} &\Rightarrow \text{PartialCount } 1 \text{ Prof Kim} && \text{(CIntro and CAdd)}\\ &\Rightarrow \text{PartialCount } 1 \text{ Prof Pat} && \text{(CHold)}\\ &\Rightarrow \text{PartialCount } 1 \text{ Prof } 0 && \text{(CHold)}\\ &\Rightarrow \text{PartialCount } 1 \text{ Prof } 1 && \text{(CHold)}\\ &\Rightarrow \text{PartialCount } 1 \text{ Prof } 2 && \text{(CHold)}\\ &\Rightarrow \text{PartialCount } 1 \text{ Prof Supremum} && \text{(CHold)}\\ &\Rightarrow \text{CardinalityMinorant } 1 \text{ Prof} && \text{(CMin)}\end{aligned}$$

and a weaker lower bound on cardinality for Prof:

$$\begin{aligned}\text{PartialCount } 0 \text{ Prof Infimum} &\Rightarrow \text{PartialCount } 0 \text{ Prof Kim} && \text{(CIntro and CHold)}\\ &\Rightarrow \text{PartialCount } 0 \text{ Prof Pat} && \text{(CHold)}\\ &\Rightarrow \text{PartialCount } 0 \text{ Prof } 0 && \text{(CHold)}\end{aligned}$$

$$\Rightarrow \text{PartialCount } 0 \text{ Prof } 1 \quad \text{(CHold)}$$
$$\Rightarrow \text{PartialCount } 0 \text{ Prof } 2 \quad \text{(CHold)}$$
$$\Rightarrow \text{PartialCount } 0 \text{ Prof Supremum} \quad \text{(CHold)}$$
$$\Rightarrow \text{CardinalityMinorant } 0 \text{ Prof} \quad \text{(CMin)}$$

and a tight maximum cardinality for Prof, observing that $K = 2$ for this theory:

$$\text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x) \text{ Infimum} \Rightarrow \text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x) \text{ Kim} \quad \text{(CIntro, CHold)}$$
$$\Rightarrow \text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x) \text{ Pat} \quad \text{(CHold)}$$
$$\Rightarrow \text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x)\ 0 \quad \text{(CHold)}$$
$$\Rightarrow \text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x)\ 1 \quad \text{(CHold)}$$
$$\Rightarrow \text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x)\ 2 \quad \text{(CHold)}$$
$$\Rightarrow \text{PartialCount } 0\ (\lambda_{x:\mathsf{e}} \neg \text{Prof } x) \text{ Supremum} \quad \text{(CHold)}$$
$$\Rightarrow \text{CardinalityMajorant } 2 \text{ Prof} \quad \text{(CMaj)}$$

since $\neg$ Prof does not appear in our theory (and contradicts Prof Kim via intuitionistic logic).

Practice 8.5:

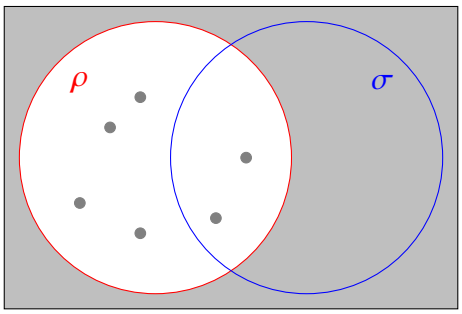
Use the above entailments to obtain a lower bound on cardinality for Adult given the theory Adult Kim $\wedge \neg$ Adult Pat. Identify the entailment rules you used.

8.7 Generalized quantifiers (Barwise & Cooper, 1981)

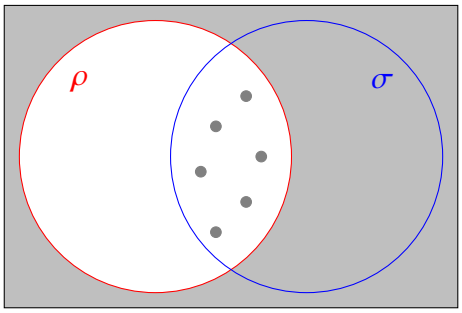
With these basic rules we can define quantifiers to make general statements over variables.

Generalized quantifiers compare sizes of intersections of restriction ρ and nuclear scope σ sets:

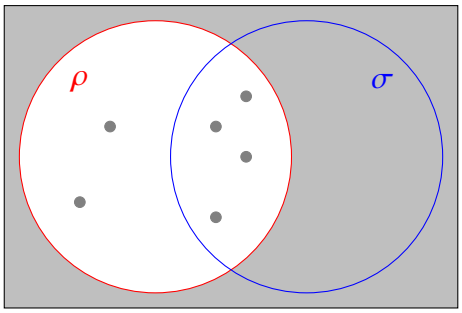
- Some $\rho\ \sigma \Leftrightarrow$ CardinalityMinorant $1\ (\lambda_{x:\mathsf{e}}\ \rho\ x \wedge \sigma\ x)$ — true if provably some ρ are σ:



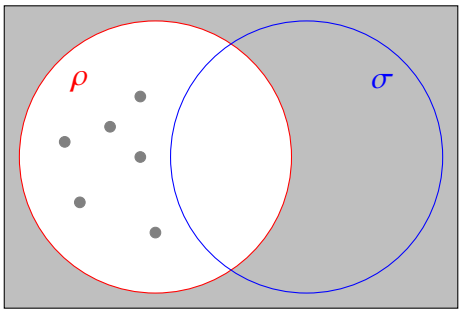
- $\mathsf{All}\ \rho\ \sigma \Leftrightarrow \mathsf{Some}\ (\lambda_{\kappa:\mathsf{e}}\ \mathsf{CardinalityMajorant}\ \kappa\ \rho)$
  $(\lambda_{\kappa:\mathsf{e}}\ \mathsf{CardinalityMinorant}\ \kappa\ (\lambda_{x:\mathsf{e}}\ \rho\ x \wedge \sigma\ x))$ — true if provably all ρ are σ:



- $\mathsf{Most}\ \rho\ \sigma \Leftrightarrow \mathsf{Some}\ (\lambda_{\kappa:\mathsf{e}}\ \mathsf{CardinalityMajorant}\ (2 \times \kappa)\ \rho)$
  $(\lambda_{\kappa:\mathsf{e}}\ \mathsf{CardinalityMinorant}\ \kappa\ (\lambda_{x:\mathsf{e}}\ \rho\ x \wedge \sigma\ x))$ — true if provably most ρ are σ:



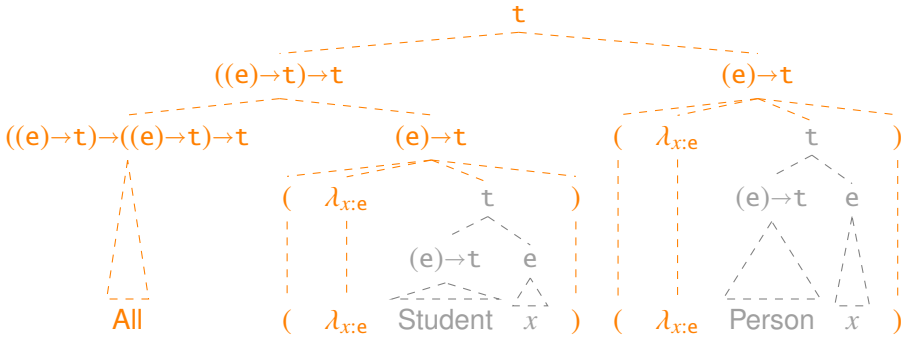
- $\mathsf{None}\ \rho\ \sigma \Leftrightarrow \mathsf{Some}\ (\lambda_{\kappa:\mathsf{e}}\ \mathsf{CardinalityMajorant}\ \kappa\ \rho)$
  $(\lambda_{\kappa:\mathsf{e}}\ \mathsf{CardinalityMinorant}\ \kappa\ (\lambda_{x:\mathsf{e}}\ \rho\ x \wedge \neg\ \sigma\ x))$ — true if provably no ρ are σ:



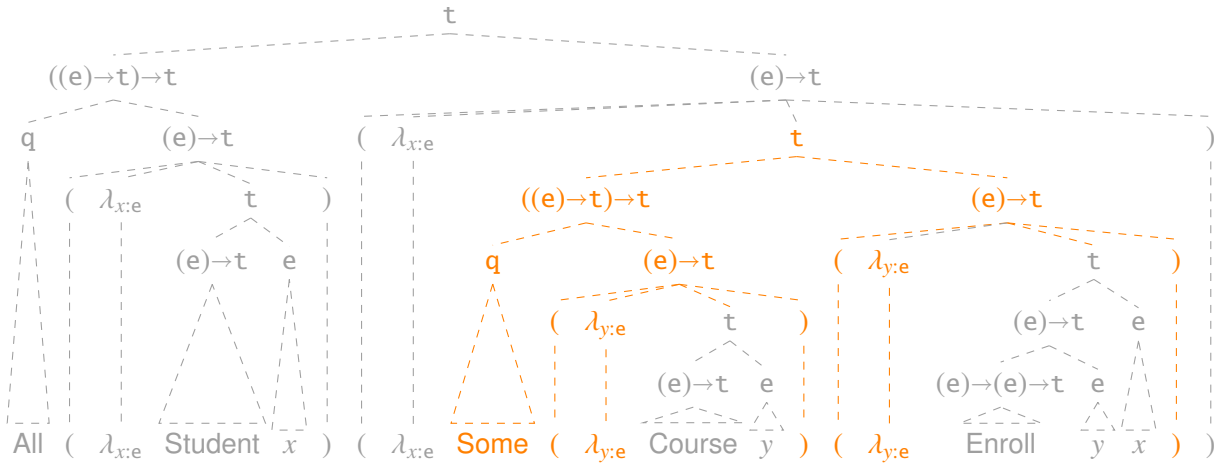
Note the similarity to diagrams of conditional probabilities from the earlier lecture notes.

Generalized quantifiers represent conditional probabilities $\mathsf{P}(\sigma|\rho)$, so we use them for reasoning!

Generalized quantifiers are just functions with type ((e)→t)→((e)→t)→t:

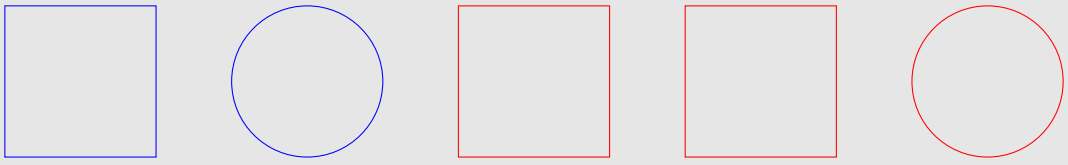


These can be **scoped** or nested (here, to save space, we substitute q for ((e)→t)→((e)→t)→t):



Practice 8.6:

Given this theory of Shape entities (where Red and Square have their usual meanings):



can we prove the following lambda calculus expression (note you are not asked for the proof)?

$$\text{Most}\ (\lambda_{x:e}\ \text{Shape}\ x\ \wedge \text{Red}\ x)\ (\lambda_{x:e}\ \text{Square}\ x)$$

Practice 8.7:

Given the same set of shapes above, can we prove the following lambda calculus expression?

$$\text{Most}\ (\lambda_{x:\mathtt{e}}\ \text{Shape}\ x)\ (\lambda_{x:\mathtt{e}}\ \text{Square}\ x\ \wedge\ \text{Red}\ x)$$

8.8 Intensions (Carnap, 1947; Pollard, 2007)

If you want something that's not true in the current world state, you don't just want 'False'.

Rather, what you want is for some whole description of some hypothetical world state to be true.

We therefore define an 'up' operator (Montague, 1970) to interpret expressions as descriptions.

7. **intension** $\overset{\mathtt{p}}{\underset{\varphi}{(\ \uparrow\ \mathtt{t}\ )}}$ of an expression φ of type $\mathtt{t}$ — essentially this is just a type marker.

   For example: $(\uparrow \text{Prof Person43})$ of type $\mathtt{p}$, where Prof Person43 is type $\mathtt{t}$.

   Intensions also accommodate beta reduction: $[\,(\uparrow \varphi)\,]_{\chi \mapsto \omega} = (\uparrow [\varphi]_{\chi \mapsto \omega})$.

We will reason about intensions in the next lecture.

8.9 Logic in associative memory (Mel'čuk, 1988; Schuler & Wheeler, 2014)

We can use associative memory to define complex ideas equivalent to lambda calculus expressions.

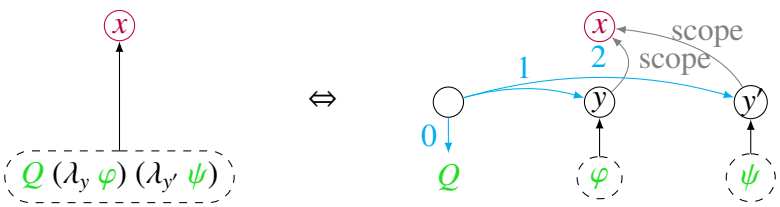
We can use neural activation patterns to represent quantified variables.

We can use cued associations to represent:

- **function arguments**: cued associations from function to argument, numbered by position,
- scope: association from a quantifier's argument variables to its immediate outscoper.

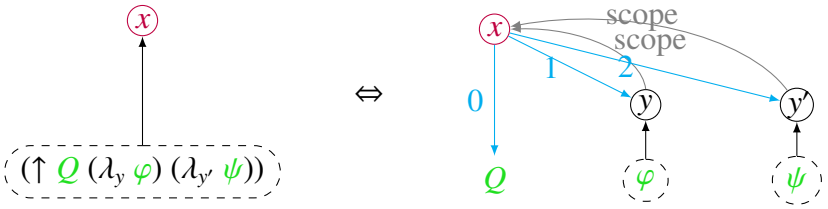
Define cued associations via recursive mappings, in context of a most recently bound variable x:

1. **generalized quantifications** (unlabeled circles) cue restrictor and nuclear scope variables:



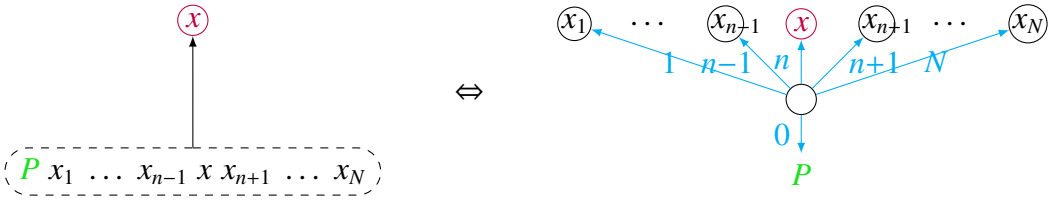
A ‘scope’ association cues the most recently bound variable as immediately outscoping.

2. **intensions** of generalized quantifiers use the most recently bound variable as quantification:



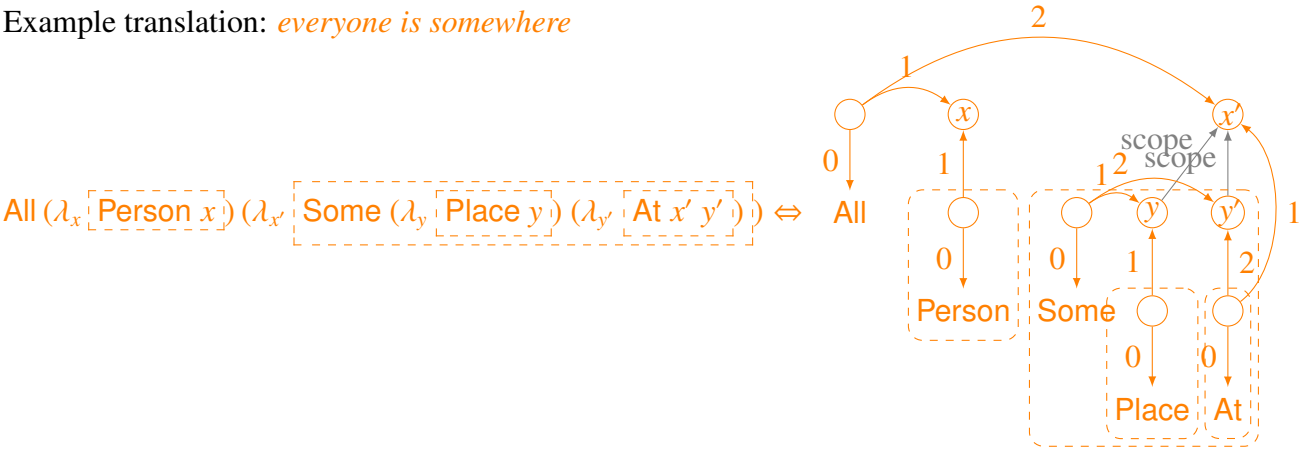
3. **conjunctions** are represented as independent graphs sharing a most recently bound variable:

4. **predications** (unlabeled circles) directly cue a predicate type and arguments:



Note that the most recently bound variable is typically one of the arguments.

Example translation: *everyone is somewhere*



In this manner, any complex idea may be stored as cued associations in associative memory.

References

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

Brouwer, L. E. J. (1907). *Over de grondslagen der wiskunde*. PhD thesis, University of Amsterdam, Amsterdam.

Carnap, R. (1947). *Meaning and Necessity: A Study in Semantics and Modal Logic*. Chicago: University of Chicago Press.

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

Gentzen, G. (1935). Untersuchungen über das logische schliesen. *Mathematische Zeitschrift*, 39, 176–210, 405–431.

Heyting, A. (1930). Die formalen regeln der intuitionistischen logik I. *Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse*, (pp. 42–56).

Hilbert, D. & Bernays, P. (1934). *Grundlagen der Mathematik*, volume 50 of *Grundlagen der mathematischen Wissenschaften*. Berlin, Heidelberg: Springer.

Mel'čuk, I. (1988). *Dependency syntax: theory and practice*. Albany: State University of NY Press.

Montague, R. (1970). Pragmatics and intensional logic. *Dialectica*, 24(4), 277–302.

Pollard, C. (2007). Hyperintensions. *Journal of Logic and Computation*, 18(2), 257–282.

Schuler, W. & Wheeler, A. (2014). Cognitive compositional semantics using continuation dependencies. In *Third Joint Conference on Lexical and Computational Semantics (*SEM'14)*.