LING5702: Lecture Notes 2
Models of Thought

Contents

Language seems similar to thought, but we can distinguish them.

Let's start with thought.

2.1 Decision theory [von Neumann & Morgenstern, 1944]

We model thought as a **decision process**: choosing actions to maximize **average expected utility**.

(There's lots of other thought: enjoying, reminiscing, etc., but this more directly helps us survive.)

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 position or status 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:

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

(It assumes the plan p is perfectly specific: at most one next action a for each possible model 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:

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

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?

2.2 Typed lambda calculus [Church, 1940]

Plans and world knowledge must be able to describe vast, hypothetical sets of entities or relations.

We use **typed lambda calculus** expressions, which concisely and precisely **denote** these things.

We'll write $[\![\dots]\!]^m$ for the **interpretation** or **denotation** of expression '$\dots$' in world model m.

(World models may be thought of as **sets of sentences or propositions** that describe a world.)

(These may be underspecified, so they are also **sets of possible worlds**: $[\![\dots]\!]^m = \forall_{w \in m} [\![\dots]\!]^w$.)

Typed lambda calculus expressions have the following **types**:

1. **propositions**: things that can be true or false, like the proposition that it is sunny — they each denote a **truth value** as evaluated in m, e.g. $[\![\mathsf{ItsSunny}]\!]^m = \mathbf{True}$);
2. **entity terms**: references to things that can be predicated over, like people and places — they each denote an **entity** in m if one exists, e.g. $[\![\mathsf{MyHill}]\!]^m = \mathbf{Hill1}$, or are ill-formed if not;
3. **functions** from any type as input to any type as output (including other functions) — they denote **sets** of input-output pairs, e.g. $[\![\mathsf{Muddy}]\!]^m = \{\langle \mathbf{Hill1}, \mathbf{True} \rangle, \langle \mathbf{Hill2}, \mathbf{False} \rangle, \dots\}$.

Typed lambda calculus expressions are constructed using the following **rules**:

1. **applications** of functions f to arguments x to get outputs $f\ x$, like $\overbrace{\underbrace{\mathsf{Muddy}}_{\text{function}}\ \underbrace{\mathsf{MyHill}}_{\text{entity term}}}^{\text{proposition (truth value)}}$ — this retrieves the (unique) output: $[\![f\ x]\!]^m = h$ such that $\langle [\![x]\!]^m, h \rangle \in [\![f]\!]^m$, e.g. $\mathbf{True}$;

2. **abstractions** over argument variables x to get functions $\lambda_x \ldots x \ldots$, like $\overbrace{\lambda_x \underbrace{\mathsf{Muddy}\ x}_{\text{proposition (truth value)}}}^{\text{function from entity term } x \text{ to proposition}}$ —

   this creates a set of pairs $[\![\lambda_x \ldots x \ldots]\!]^m = \{\langle x, [\![\ldots x \ldots]\!]^m\rangle \mid x \in m\}$ (constrained to x's type).

   Truth output defines set of input: $[\![\lambda_x \ldots x \ldots]\!]^m = \{x \mid x \in m, [\![\ldots x \ldots]\!]^m\}$, so $[\![\mathsf{Muddy}]\!]^m = \{\textbf{Hill1}\}$.

The most common functions we need are:

1. **predicates**: e.g. $\overbrace{\overbrace{\mathsf{Person}}^{\text{predicate}}\ \overbrace{x}^{\text{entity}}}^{\text{proposition}}$ or $\overbrace{\overbrace{\mathsf{At}}^{\text{pred.}}\ \overbrace{x}^{\text{ent.}}\ \overbrace{y}^{\text{ent.}}}^{\text{proposition}}$, map entity terms to truth values (propositions) —

   for extra entities, use functions as output: $[\![\mathsf{At}]\!]^m = \{\langle \textbf{Me}, \underbrace{\{\langle \textbf{Hill1}, \textbf{True}\rangle, \langle \textbf{Hill2}, \textbf{False}\rangle, \ldots\}}_{\text{output is another function with second entity as input!}}\rangle, \ldots\}$;

2. **conjunctions**: e.g. $\overbrace{\overbrace{\mathsf{Person}\ x}^{\text{proposition}} \overbrace{\wedge}^{\text{conj.}} \overbrace{\mathsf{At}\ x\ \mathsf{MyHill}}^{\text{proposition}}}^{\text{proposition}}$, map truth values to truth values —

   this is an 'infix' operator, equivalent to a regular function: $[\![\varphi \wedge \psi]\!]^m \Leftrightarrow [\![\mathsf{And}\ \varphi\ \psi]\!]^m$;

3. **generalized quantifiers**: e.g. $\overbrace{\overbrace{\mathsf{All}}^{\text{g.q.}}\ \overbrace{(\overbrace{\lambda_x}^{\text{lambda}}\ \overbrace{\mathsf{Person}\ x}^{\text{proposition}})}^{\text{'restriction' set}}\ \overbrace{(\overbrace{\lambda_x}^{\text{lambda}}\ \overbrace{\mathsf{At}\ x\ \mathsf{MyHill}}^{\text{proposition}})}^{\text{'nuclear scope' set}}}^{\text{proposition}}$, map sets to truth values.

   The lambdas in the 'restriction' and 'nuclear scope' sets bind entity variables for use within.

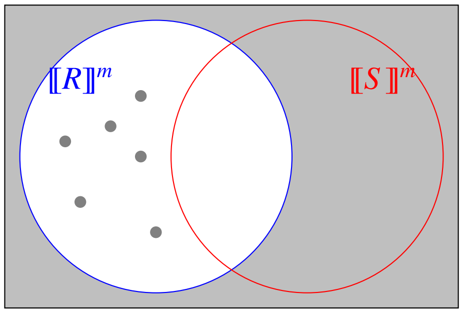
Practice 2.1:

Using the predicates $\mathsf{Dog}\ x$, which means x is a dog, and $\mathsf{Mammal}\ x$, which means x is a mammal, write a typed lambda calculus expression stating that all dogs are mammals.

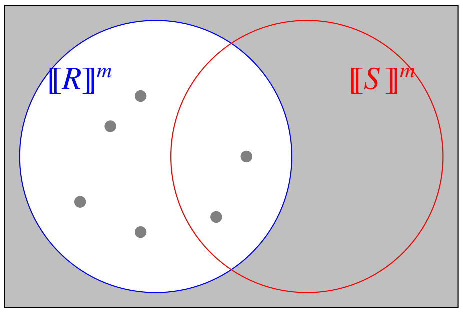
2.3 Generalized quantifiers [Barwise & Cooper, 1981]

Generalized quantifiers compare denotations of intersections of restriction R and nuclear scope S:

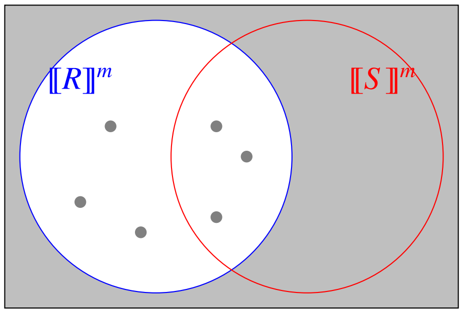
- $[\![\mathsf{None}\ R\ S]\!]^m \Leftrightarrow |[\![R]\!]^m \cap [\![S]\!]^m| = 0$ — true if none of the R's are S's:



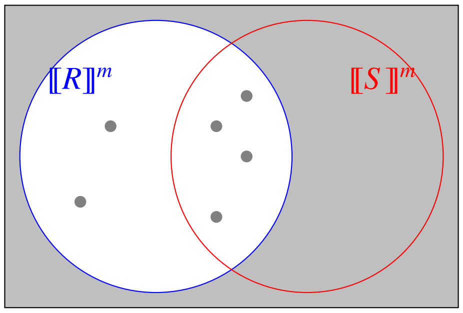
- $[\![\text{Some } R\ S]\!]^m \Leftrightarrow |[\![R]\!]^m \cap [\![S]\!]^m| > 0$ — true if some of the R's are S's:



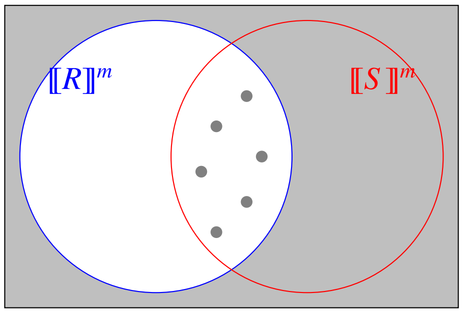
- $[\![\text{Half } R\ S]\!]^m \Leftrightarrow \frac{|[\![R]\!]^m \cap [\![S]\!]^m|}{|[\![R]\!]^m|} = 0.5$ — true if half of the R's are S's:



- $[\![\text{Most } R\ S]\!]^m \Leftrightarrow \frac{|[\![R]\!]^m \cap [\![S]\!]^m|}{|[\![R]\!]^m|} > 0.5$ — true if most of the R's are S's:



- $[\![\text{All } R\ S]\!]^m \Leftrightarrow \frac{|[\![R]\!]^m \cap [\![S]\!]^m|}{|[\![R]\!]^m|} = 1.0$ — true if all of the R's are S's:



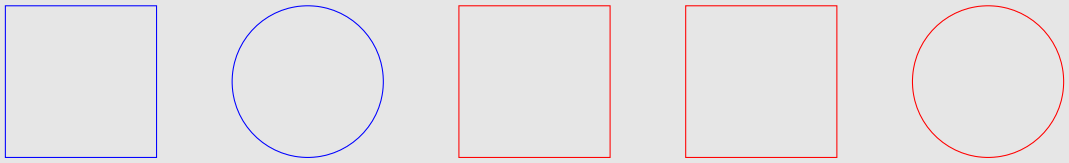
Note the similarity to diagrams of conditional probabilities from the previous lecture notes.

Generalized quantifiers represent conditional probabilities $\mathsf{P}_m(S \mid R)$, so we use them for reasoning!

(Specifically, we use them to represent probabilistic outcome events o in our decision processes.)

Practice 2.2:

Given a world m of Shape entities (where Red and Square have their usual meanings):



what is the value of the following lambda calculus expression?

$$[\![\ \text{Most}\ (\lambda_x\ \text{Shape}\ x\ \wedge \text{Red}\ x)\ (\lambda_x\ \text{Square}\ x)\ ]\!]^m$$

Practice 2.3:

Given the same set of shapes above, what is the value of the following lambda calculus expression?

$$[\![\ \text{Most}\ (\lambda_x\ \text{Shape}\ x)\ (\lambda_x\ \text{Square}\ x\ \wedge \text{Red}\ x)\ ]\!]^m$$

2.4 Complex (nested) propositions

Recall that quantifiers are propositions that can contain propositions:

$$\overbrace{\text{All}\ (\lambda_x\ \overbrace{\text{Person}\ x}^{\text{proposition}})\ (\lambda_x\ \overbrace{\text{At}\ x\ \text{MyHill}}^{\text{proposition}})}^{\text{proposition}}$$

This means we can stuff them inside each other like Russian dolls or turduckens:

$$\overbrace{\text{All}\ (\lambda_x\ \overbrace{\text{Person}\ x}^{\text{proposition}})\ (\lambda_x\ \overbrace{\text{Some}\ (\lambda_y\ \overbrace{\text{Place}\ y}^{\text{proposition}})\ (\lambda_y\ \overbrace{\text{At}\ x\ y}^{\text{proposition}}))}^{\text{proposition}}}^{\text{proposition}}$$

This is called **nesting** or **scoping**.

2.5 Example: grid-world navigation

Now we can make a plan and world model that moves a person toward a hill in a grid world:

1. a **plan** p to move Me to MyHill (assuming the following predicates:
   - CurrentTime t is true for the most recent time point t in an AEU evaluation;
   - PrecedeOrEqual s t is true if time s precedes or is equal to t;
   - TryMoveToward t a x is true if a tries to move toward x at time t;

   where 'All's iterate over entities, 'None's give conditions):

$$\begin{array}{l} \text{All } (\lambda_t \text{ CurrentTime } t \wedge \\ \quad\quad \text{None } (\lambda_s \text{ PrecedeOrEqual } 0\ s\ \wedge \text{PrecedeOrEqual } s\ t) \\ \quad\quad\quad\quad (\lambda_s \text{ At } s \text{ Me MyHill})) \\ \quad (\lambda_t \text{ TryMoveToward } t \text{ Me MyHill}) \end{array}$$

   (Here time '0' is when the plan is created – I have to reach the goal *after* that to succeed.)

   When used in an AEU, this gives us our actions a – in this case: TryMoveToward predicates.

2. **world knowledge** m that moving through mud may fail (assuming the following predicates:
   - Adjacent x y is true if grid squares x and y are adjacent;
   - Aligned x y z is true if a grid square y lies on a line from x to z;
   - Muddy y and Clear y are true if grid square y is muddy or clear, respectively;
   - ConsecutiveTime t u is true if time t immediately precedes time u;

   where 'Half's give probability cost):

$$\begin{array}{l} \text{All } (\lambda_{t,a,z} \text{ TryMoveToward } t\ a\ z) \\ \quad (\lambda_{t,a,z} \text{ All } (\lambda_x \text{ At } t\ a\ x) \\ \quad\quad\quad (\lambda_x \text{ All } (\lambda_y \text{ Adjacent } x\ y\ \wedge \text{Aligned } x\ y\ z\ \wedge \text{Muddy } y) \\ \quad\quad\quad\quad (\lambda_y \text{ Half } (\lambda_u \text{ ConsecutiveTime } t\ u)\ (\lambda_u \text{ At } u\ a\ y) \wedge \\ \quad\quad\quad\quad\quad \text{Half } (\lambda_u \text{ ConsecutiveTime } t\ u)\ (\lambda_u \text{ At } u\ a\ x)))) \end{array}$$

   and that moving through clear terrain always succeeds:

$$\begin{array}{l} \text{All } (\lambda_{t,a,z} \text{ TryMoveToward } t\ a\ z) \\ \quad (\lambda_{t,a,z} \text{ All } (\lambda_x \text{ At } t\ a\ x) \\ \quad\quad\quad (\lambda_x \text{ All } (\lambda_y \text{ Adjacent } x\ y\ \wedge \text{Aligned } x\ y\ z\ \wedge \text{Clear } y) \\ \quad\quad\quad\quad (\lambda_y \text{ All } (\lambda_u \text{ ConsecutiveTime } t\ u)\ (\lambda_u \text{ At } u\ a\ y)))) \end{array}$$

   When used in an AEU, this gives us our outcome events o – in this case: At predicates.

Dropping these plans and world models into the AEU function defines a rational decision process.

Since they are *simple* and *work*, they are in some sense a 'natural' representation of complex ideas.

We'll therefore use these expressions as the complex ideas that get communicated using language.

But note these look very different from how we might represent these ideas in natural language.

2.6 Extra: quantifiers over substances

Generalized quantifiers model **substances** as sets of infinitesimal (arbitrarily small) 'minimal parts':

$$\mathsf{Most}\ (\lambda_v\ \mathsf{Contain\ MilkyWayGalaxy}\ v)\ (\lambda_v\ \mathsf{EmptySpace}\ v)$$

Proportions over infinite sets of infinitesimals can be well defined using random sampling:

$$[\![\mathsf{Most}\ R\ S\,]\!]^m \Leftrightarrow \lim_{K\to\infty} \mathsf{E}_{D_K\sim\pi} \frac{|D_K \cap [\![R]\!]^m \cap [\![S\,]\!]^m|}{|D_K \cap [\![R]\!]^m|} > 0.5$$

(Here $\mathsf{E}_{D_K\sim\pi}$ ... is expected value of ... for K-element set D_K randomly drawn from distribution π.)

(Think of this as setting K high enough to be reliable and ensure a non-zero denominator.)

This lets us use the same quantifier functions (e.g. Most) for objects and substances.

2.7 Extra: propositions about arbitrary probabilities

We can further generalize quantifiers as **cardinal** (Count) and **proportional** (Ratio) quantifiers.

We can yet further generalize these as **upward-entailing** ($Q_\geq$) and **downward-entailing** ($Q_\leq$):

$$[\![\mathsf{Count}_\geq\ n\ R\ S\,]\!]^m \Leftrightarrow |[\![R]\!]^m \cap [\![S\,]\!]^m| \geq n \qquad \text{(e.g. } \mathsf{Some}\ R\ S \Leftrightarrow \mathsf{Count}_\geq\ 1\ R\ S\text{)}$$

$$[\![\mathsf{Count}_\leq\ n\ R\ S\,]\!]^m \Leftrightarrow |[\![R]\!]^m \cap [\![S\,]\!]^m| \leq n \qquad \text{(e.g. } \mathsf{None}\ R\ S \Leftrightarrow \mathsf{Count}_\leq\ 0\ R\ S\text{)}$$

$$[\![\mathsf{Ratio}_\geq\ n\ R\ S\,]\!]^m \Leftrightarrow \frac{|[\![R]\!]^m \cap [\![S\,]\!]^m|}{|[\![R]\!]^m|} \geq n \qquad \text{(e.g. } \mathsf{All}\ R\ S \Leftrightarrow \mathsf{Ratio}_\geq\ 1\ R\ S\text{)}$$

$$[\![\mathsf{Ratio}_\leq\ n\ R\ S\,]\!]^m \Leftrightarrow \frac{|[\![R]\!]^m \cap [\![S\,]\!]^m|}{|[\![R]\!]^m|} \leq n \qquad \text{(e.g. } \mathsf{Few}\ R\ S \Leftrightarrow \mathsf{Ratio}_\leq\ .5\ R\ S\text{)}$$

with variants $Q_=$, $Q_<$, $Q_>$ for other comparison operators defined in terms of these:

$$\begin{aligned} Q_=\ n\ R\ S &\Leftrightarrow Q_\leq\ n\ R\ S \wedge Q_\geq\ n\ R\ S \\ Q_<\ n\ R\ S &\Leftrightarrow Q_\leq\ n\ R\ S \wedge \neg Q_\geq\ n\ R\ S \\ Q_>\ n\ R\ S &\Leftrightarrow Q_\geq\ n\ R\ S \wedge \neg Q_\leq\ n\ R\ S \end{aligned}$$

Now we can express arbitrary claims about probability:

$$\mathsf{P(Edible\ |\ Nut)} > .20 \Leftrightarrow \mathsf{Ratio}_>\ .20\ (\lambda_x\ \mathsf{Nut}\ x)\ (\lambda_x\ \mathsf{Edible}\ x)$$

2.8 Extra: intensions (propositions about propositions)

We now have a formal system to reason about complex ideas based on sets of entities or tuples.

But what if we have to do something when someone *wants to eat*, where *to eat* is a proposition?

Propositions denote truth values, but the speaker doesn't want '**False**' (whatever that would mean).

So, define argument propositions as **intensions** – sets of satisfying possible worlds [Carnap, 1947]:

$$\begin{aligned} [\![\mathsf{IntensionOfRatio}_o\ i\ n\ R\ S\ ]\!]^m &\iff i = \{w \mid [\![\mathsf{Ratio}_o\ n\ R\ S\ ]\!]^w\} \cap m \\ [\![\mathsf{IntensionOfCount}_o\ i\ n\ R\ S\ ]\!]^m &\iff i = \{w \mid [\![\mathsf{Count}_o\ n\ R\ S\ ]\!]^w\} \cap m \end{aligned}$$

(Worlds are completely specified. World models may be incomplete, subsuming many worlds.)

(Intensions are intersected with world models to ensure domains of e.g. entity constants match.)

An intension may then **entail** another if its satisfying possible worlds are a subset of the other's:

$$[\![\mathsf{Entail}\ i\ j]\!]^m \iff i \subseteq j$$

These sets would be hard to calculate! Fortunately we can define entailment in terms of structure!

We reason about these, e.g. test if claim i is in some class j, by *simplifying* rather than enumerating

(where $Q \in \{\mathsf{Count}, \mathsf{Ratio}\}$ and $\langle w, x, x', ..\rangle$ is a tuple of a possible world w and entities $x, x', .. \in w$):

$$\begin{aligned}
\overbrace{\{\langle w, ..\rangle \mid [\![\varphi \wedge \chi]\!]^w\}}^{\text{more specific intension } i} \subseteq \overbrace{\{\langle w, ..\rangle \mid [\![\psi]\!]^w\}}^{\text{more general intension } j} &\quad \text{if} \quad \{\langle w, ..\rangle \mid [\![\varphi]\!]^w\} \subseteq \{\langle w, ..\rangle \mid [\![\psi]\!]^w\} \\
\{\langle w, ..\rangle \mid [\![Q_\geq\ n\ R\ S\ ]\!]^w\} \subseteq \{\langle w, ..\rangle \mid [\![Q_\geq\ n'\ R\ S\ ]\!]^w\} &\quad \text{if} \quad n \geq n' \\
\{\langle w, ..\rangle \mid [\![Q_\leq\ n\ R\ S\ ]\!]^w\} \subseteq \{\langle w, ..\rangle \mid [\![Q_\leq\ n'\ R\ S\ ]\!]^w\} &\quad \text{if} \quad n \leq n' \\
\{\langle w, ..\rangle \mid [\![Q_\geq\ n\ R\ (\lambda_x\ \varphi)]\!]^w\} \subseteq \{\langle w, ..\rangle \mid [\![Q_\geq\ n\ R\ (\lambda_x\ \psi)]\!]^w\} &\quad \text{if} \quad \{\langle w, .., x\rangle \mid [\![\varphi]\!]^w\} \subseteq \{\langle w, .., x\rangle \mid [\![\psi]\!]^w\} \\
\{\langle w, ..\rangle \mid [\![Q_\leq\ n\ R\ (\lambda_x\ \varphi)]\!]^w\} \subseteq \{\langle w, ..\rangle \mid [\![Q_\leq\ n\ R\ (\lambda_x\ \psi)]\!]^w\} &\quad \text{if} \quad \{\langle w, .., x\rangle \mid [\![\varphi]\!]^w\} \supseteq \{\langle w, .., x\rangle \mid [\![\psi]\!]^w\}
\end{aligned}$$

For example:

$$\begin{aligned}
\overbrace{\{w \mid [\![\mathsf{ItsCloudy} \wedge \mathsf{ItsRainy}]\!]^w\}}^{\text{more specific intension}} &\subseteq \overbrace{\{w \mid [\![\mathsf{ItsCloudy}]\!]^w\}}^{\text{more general intension}} \\
\{w \mid [\![\mathsf{Count}_\geq\ 2\ (\lambda_x\ \mathsf{Hut}\ x)\ (\lambda_x\ \mathsf{Straw}\ x)]\!]^w\} &\subseteq \{w \mid [\![\mathsf{Count}_\geq\ 1\ (\lambda_x\ \mathsf{Hut}\ x)\ (\lambda_x\ \mathsf{Straw}\ x)]\!]^w\} \\
\{w \mid [\![\mathsf{Count}_\geq\ 1\ (\lambda_x\ \mathsf{Hut}\ x)\ (\lambda_x\ \mathsf{Straw}\ x \wedge \mathsf{Round}\ x)]\!]^w\} &\subseteq \{w \mid [\![\mathsf{Count}_\geq\ 1\ (\lambda_x\ \mathsf{Hut}\ x)\ (\lambda_x\ \mathsf{Straw}\ x)]\!]^w\}
\end{aligned}$$

This kind of reasoning by simplifying is sometimes called **natural logic** [van Benthem, 1986].

We can also model intensions as entities, drawn from a very large domain, similar to infinitesimals.

Entailment predicates can be used to evaluate if a desired intension i is in some class j:

$$
\begin{array}{l}
\mathsf{All}\ (\lambda_t\ \mathsf{CurrentTime}\ t) \\
\quad (\lambda_t\ \mathsf{All}\ (\lambda_c\ \mathsf{Clerk}\ c) \\
\qquad (\lambda_c\ \mathsf{All}\ (\lambda_a\ \mathsf{Person}\ a) \\
\qquad\quad (\lambda_a\ \mathsf{All}\ (\lambda_x\ \mathsf{Have}\ t\ c\ x \wedge \\
\qquad\qquad\qquad \mathsf{Some}\ (\lambda_j\ \mathsf{IntensionOfCount}_{\geq}\ j\ 1\ (\lambda_u\ \mathsf{ConsecutiveTime}\ t\ u) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_u\ \mathsf{Eat}\ u\ a\ x)) \\
\qquad\qquad\qquad\quad (\lambda_j\ \mathsf{Some}\ (\lambda_i\ \mathsf{Want}\ t\ a\ i) \\
\qquad\qquad\qquad\qquad\quad (\lambda_i\ \mathsf{Entail}\ i\ j))) \\
\qquad\qquad (\lambda_x\ \mathsf{Give}\ t\ c\ a\ x))))
\end{array}
$$

(If a clerk has something, and someone wants it [perhaps among other things], give it to them.)

Here, even if the intension i that the agent wants contains other conjuncts, it still entails j:

$$
\begin{array}{l}
\mathsf{Some}\ (\lambda_i\ \mathsf{IntensionOfCount}_{\geq}\ i\ 1\ (\lambda_u\ \mathsf{ConsecutiveTime}\ 10{:}00{:}00\ u) \\
\qquad\qquad\qquad\qquad\qquad\qquad (\lambda_u\ \mathsf{Eat}\ u\ \mathsf{Me}\ \mathsf{Apple1} \wedge \mathsf{Drink}\ u\ \mathsf{Me}\ \mathsf{Juice1})) \\
\qquad (\lambda_i\ \mathsf{Want}\ 10{:}00{:}00\ \mathsf{Me}\ i)
\end{array}
$$

So if the above is true, the clerk will recognize that I want to eat an apple and give it to me.

Practice 2.4:

Using the non-intensional and intensional quantifier functions above and the predicates Kid k, Car c, Time t, Own $t\ k\ c$, and Want $k\ i$, write a lambda calculus expression stating that every kid wants to own a car at some point in time (but they don't have a particular car in mind). Note: only quantifier functions can take lambda functions $(\lambda_x \ldots)$ as arguments; all the predicates can only take entity variables as arguments.

2.9 No need for other operators

Now we're done! 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**:

$$\neg\ \mathsf{ItsRainy} \quad \Leftrightarrow \quad \mathsf{None}\ (\lambda_x\ \mathsf{Unit}\ x)\ (\lambda_x\ \mathsf{ItsRainy})$$

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

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

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

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

or more directly using an 'All' quantifier:

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

This simplifies our natural logic entailment!

Now we have some basics of meaning, let's see how we can represent them in the brain...

2.10 Review

We defined a formal system of reasoning that consists of:

1. predicates
2. conjunctions
3. generalized quantifiers, to model probabilistic inference, negation, disjunction, etc.
4. intensions, to model propositions about propositions

Coming lectures will show how to represent these things in brains and decode them from language.

References

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

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

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

[van Benthem, 1986] van Benthem, J. (1986). Natural logic. In *Essays in Logical Semantics*. Dordrecht, the Netherlands: Kluwer.

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