1. Types of inferences

The types of inferences that can hold between two sentences, A and B:

- Entailment
- Conversational implicature
- Conventional implicature (communicative effects that are just part of particular words meanings, but non-truth-conditionally, such as *but* and *therefore* as compared to *and*)
- No relation

We can determine the inference type of two of the inferences by applying different tests.

'A and not B'

- contradictory
  - Entailment
- not contradictory
  - Conversational Implicature
  - Conventional implicature or No relation
    - cancelable/defeasible
    - reinforceable
    - non-detachable
    - calculable by Grice’s maxims
Grice’s maxims

Quantity:  – Make your contribution as informative as needed.
          – Don’t make your contribution more informative than is needed.

Quality:  – Do not say anything you believe to be false.
          – Do not say anything for which you lack adequate evidence.

Relation: Be relevant.

Manner:  – Avoid obscurity.
          – Avoid ambiguity.
          – Be brief.
          – Be orderly.

Scalar implicatures

• Some families of words are organized into scales:
  – \(<all, many, some>\)
  – \(<and, or>\)
  – \(<certainly, probably, possibly>\)
  – \(<love, like>\)
  – \(<freezing, cold>\)

• Entailment: higher items entail lower items
  A: There were many casualties.
  B: There were some casualties.

          – Entailment: A entails B, ‘there were many casualties and it’s not the case that there were some casualties’ is contradictory.

• Implicature: asserting lower item on scale implicates the negation of higher items on scale
  A: There were some casualties.
  B: There were not many casualties.

          – No entailment: A does not entail B, ‘there were some casualties and it’s not the case that there were not many casualties’ or, canceling out the double negation ‘there were some casualties and there were many casualties’ is not contradictory.

          – Cancelable/Defeasible: There were some casualties, in fact there were many.
          – Reinforceable: There were some casualties, but not many.
          – Non-detachable: Implicature is retained when ‘some’ is replaced with a synonym like ‘a few.’ ‘There were a few casualties’ still implicates ‘There were not many casualties.’
          – Calculable: The implicature is a consequence of the Maxim of Quantity, which states that one should make the most informative statement. When a speaker uses a weaker expression from a scale of informativeness, they implicate that a stronger expression does not hold. Since some is less informative than many, many some here implicates not many.
1.1. **Exercise.** For each of the following pair of sentences, determine whether A entails, implicates, or is in no relation to B. Justify your answers.

1. **A:** Lee kissed Kim passionately.
   **B:** Lee kissed Kim.

2. **A:** Lee kissed Kim passionately.
   **B:** Kim kissed Lee.

3. **A:** Mary used to swim a mile daily.
   **B:** Mary no longer swims a mile daily.

4. **A:** Mary likes her presents.
   **B:** Mary doesn’t love her presents.

References for types of inferences:
- Class lectures 1/15-1/17, 4/23-4/25
- Section notes 1/28, 2/25
- Reading 1/17 : CMG 1
- Reading 4/23-25 : Grice

1.2. **Entailments in models.**

A sentence \( \phi \) entails a sentence \( \psi \) iff, for all models \( M \),

- if \( \models M = 1 \) then \( \models M = 1 \)
- if \( M \) satisfies \( \phi \), then \( M \) satisfies \( \psi \)
- if \( M \models \phi \) then \( M \models \psi \)

We have intuitions about whether one sentence entails another in English. We want our model to be able to compute entailments in a way that is consistent with our intuitions. Recall that in mini-assignment 5, our translation of *small* in our model gave us entailment relations that did not correspond to our intuitions. This mismatch lead us to change our translation.

Mini-assignment 5:
\[
\text{tr}(\text{small}) = \text{small'}
\]

Afterwards:
\[
\text{tr}(\text{small}) = \lambda P_{(e,t)}[\lambda z[P(z) \land \text{smallfora'}(P)(z)]]
\]

References for entailments in models:
- Class lecture 2/5
- Problem set 2
- Mini-assignment 5
- Section notes 3/4
- Reading DWP
2. Set theory

\[ x \in A \quad \text{x is a member of } A \]
\[ x \notin A \quad \text{x is not a member of } A \]
\[ B \subseteq A \quad \text{B is a subset of } A \text{ iff every member } B \text{ is a member } A \]
\[ B \subset A \quad \text{B is a proper subset of } A \text{ iff every member of } B \text{ is a member } A, \]
and there is at least one member of \( A \) that is not a member of \( B \)

\[ \emptyset = \text{def} \{ \} \]

there is one set with no members at all, the null set

\[ \emptyset \subseteq A \quad \emptyset \text{ is a subset of every set} \]

\[ \emptyset \notin A \quad \emptyset \text{ is not a member of every set, but it can be a member of a specific set} \]

\[ \wp(A) = \text{def} \{ B \mid B \subseteq A \} \]

the powerset of \( A \) is the set whose members are all the subsets of \( A \)

\[ A \cup B = \text{def} \{ x \mid x \in A \text{ or } x \in B \} \]

the union of sets \( A \) and \( B \) is the set where members are the objects which are members of \( A \) or of \( B \) or of both

\[ A \cap B = \text{def} \{ x \mid x \in A \text{ and } x \in B \} \]

the intersection of sets \( A \) and \( B \) is the set whose members are just the members of both \( A \) and \( B \)

\[ A - B = \text{def} \{ x \mid x \in A \text{ and } x \notin B \} \]

the set of members of \( A \) which are not members of \( B \)

\[ A' = \text{def} \{ x \mid x \notin A \} = U - A \]

\[ \langle x, y \rangle \quad \text{is an ordered pair, } \langle x, y \rangle \neq \langle y, x \rangle \text{ unless } x = y \]

\[ A \times B = \text{def} \{ \langle x, y \rangle \mid x \in A \text{ and } y \in B \} \]

the set of ordered pairs where the first member of the pair is in \( A \) and the second member of the pair is in \( B \)

2.1. The characteristic function of a set. If \( A \) is the set of individuals and \( S \) is any subset of \( A \), we define a function \( f_S \) on the set \( A \) by letting for each \( a \) in \( A \):

\[
\begin{cases} 
1 & \text{if } a \in S \\
0 & \text{if } a \notin S 
\end{cases}
\]

2.2. Relations and Functions.

Relation: \( R \) is a relation on \( A \) if \( R \subseteq A \times A \)

Function: A relation \( R \) from \( S \) to \( T \) is a function iff it is:

single-valued: each element in the domain (\( S \)) is paired with just one element in the codomain (\( T \))

into/total: the domain of \( R \) is equal to \( S \)
A function from $S$ to $T$ can be:

**onto/surjective**: the image of $R$ is equal to $T$

**one-to-one/injective**: each element in the codomain ($T$) is assigned to just one element in the domain ($S$)

**one-to-one correspondence/bijective**: both surjective and injective

References for set theory:

- Class lectures 1/17-1/24
- Section notes 1/28
- Reading: PTW, DWP

3. Older languages

- Simple languages $L_0, L_{0E}$:
  - basic expressions, logical constants, formation rules (recursive/phrase-structure grammar), semantic rules, semantic values
  - truth tables
  - scope of logical operators
  - differences between $L_0$ and $L_{0E}$
  - function-valued functions

- Predicate Logic:
  - models (domain, interpretation functions), truth relative to a model $M$
  - variables, quantifiers, scope (free/bound variables), assignment functions
  - multiple quantifiers and scope ambiguity
  - translating to and from predicate logic

- Adjectives: different types of adjectives intersective, subsective, non-subsective

4. Current Language

4.1. $L_\lambda$: Syntax.

- Types: $e$ (entities like names) and $t$ (truth values)

- $\langle \sigma, \tau \rangle$ is the type of functions that take arguments of type $\sigma$ and map them to things of type $\tau$.

- For every type $\tau$, $ME_\tau$ is the set of meaning expressions of type $\tau$

- If $\eta \in ME_{\langle \sigma, \tau \rangle}$ and $\alpha \in ME_\sigma$, then $\eta(\alpha) \in ME_\tau$

- If $\alpha \in ME_a$ and $u \in Var_b$, then $\lambda u[\alpha] \in ME_{\langle b, a \rangle}$

- $\lambda$-conversion:
When we have an expression of the form \( \lambda u[\eta](\alpha) \), we can get rid of the \( \lambda u \) in front, and the \( (\alpha) \) at the end, and replace all \( u \)s in \( \eta \) with \( \alpha \)s, and we’re guaranteed to get an equivalent expression as long as no variables in \( \alpha \) get “captured” when we copy and paste it into \( \eta \).

4.2. L\( _\lambda \): Semantics.

- Expressions are assigned semantic values relative to a model \( M = \langle A, F \rangle \) and an assignment \( g \).
- Types are associated with sets of possible denotations:
  - \( D_e = A \)
  - \( D_t = \{1, 0\} \)
  - For all types \( \sigma \) and \( \tau \), \( D_{\langle \sigma, \tau \rangle} \) is the set of all functions from \( D_\sigma \) to \( D_\tau \)
- For all types \( \tau \) and expressions \( \eta \), if \( \eta \in ME_\tau \), \([\eta]^{M,g} \in D_\tau \)
- Expressions get their semantic values from the model or assignment:
  - Constant symbol \( \alpha \): \([\alpha]^{M,g} = F(\alpha) \)
  - Variable \( u \): \([u]^{M,g} = g(u) \)
- Quantification:
  - Where \( u \) is of type \( \tau \), \([\forall u \phi]^{M,g} = 1 \text{ iff for all } a \in D_\tau, [\phi]^{M,g_a} = 1 \)
  - Where \( u \) is of type \( \tau \), \([\exists u \phi]^{M,g} = 1 \text{ iff there is } a \in D_\tau \text{ such that } [\phi]^{M,g_a} = 1 \)
- \( \lambda \) abstraction defines a new function that takes the argument and uses it as the value for the bound variable.
  - For every variable \( u \) of type \( \sigma \), and any expression \( \eta \) of type \( \tau \), \([\lambda u[\eta]]^{M,g} = f \text{, where } f \text{ is the unique function in } D_{\langle \sigma, \tau \rangle} \text{ such that, for all } a \in D_\sigma, f(a) = [\eta]^{M,g_a} \).

4.3. English Syntax.

(1) \( S \rightarrow DP \ VP \)
  - \( DP \rightarrow Nm \)
  - \( VP \rightarrow V \)
  - \( VP \rightarrow V \ DP \)
  - \( VP \rightarrow VP \ ConjP \)
  - \( ConjP \rightarrow Conj \ VP \)
  - \( S \rightarrow S \ ConjP \)
  - \( ConjP \rightarrow Conj \ S \)
  - \( NP \rightarrow Adj \ NP \)
  - \( NP \rightarrow N \)
  - \( VP \rightarrow is \ a \ NP \)
  - \( VP \rightarrow are \ NP \)
  - \( DP \rightarrow D \ NP \)
  - \( NP \rightarrow NP \ PP \)
  - \( PP \rightarrow P \ DP \)
VP → V CP
CP → that S

(2) Constraint: in any tree of the following form, X and Y must be the same category and must have translations of the same type:

X
  X ConjP
    Conj Y
(That is, you can’t coordinate a VP with an S).

(3) We can attach a quantificational element (type ⟨⟨e,t⟩,t⟩) DP subscripted with an n to a S tree containing e_n.

S
  DP_n S
    ...e_n...

(4) We can attach a quantificational element (type ⟨⟨e,t⟩,t⟩) DP subscripted with an n to any node of type ⟨e,t⟩ containing e_n, keeping the category the same.

NP ⟨⟨e,t⟩⟩
  DP_n NP ⟨⟨e,t⟩⟩
    ...e_n...

4.4. Translating English to L_\Lambda.

(1) a. For every word α, tr(α) will be specified for our “dictionary”.
   b. For every complex node γ made up of α and β, tr(γ) = tr(α)(tr(β)) or tr(γ) = tr(β)(tr(α)), whichever is a valid expression of L_\Lambda.
   c. If γ made up of α and β and neither tr(β)(tr(α)) nor tr(α)(tr(β)) is a valid expression of L_\Lambda, γ is not a grammatically valid node.
   d. If β is a node with only one child α, then tr(β) = tr(α).

(2) a. Constraint: If any constituent does not receive a translation under the above rules, it is not grammatical.
   b. Constraint: If an S receives a translation of any type besides t, it is ungrammatical.

(3) tr(is a α) = tr(are α) = tr(α)

(4) Translating sentence embedding element that:

tr(CP) = \lambda u_s[tr(S)]

that S
4.5. QR Recipes.

(1) When attaching to S nodes:

\[ \text{tr}(S) = \text{tr}(\text{DP}) (\lambda v_{n,e}[\text{tr}(S_1)]) \]

\[
\begin{array}{ccc}
\text{DP}_n & \text{S}_1
\end{array}
\]

\[
\langle (e, t), t \rangle
\]

\[
\ldots \text{e}_n \ldots
\]

- Pronounced with DP\textsubscript{n} in the position of e\textsubscript{n}.

(2) When attaching to nodes of type \langle e, t \rangle:

\[ \text{tr}(\alpha_2) = \lambda u_e [\text{tr}(\text{DP}) (\lambda v_{n,e}[\text{tr}(\alpha_1)(u_e)])] \]

\[
\begin{array}{ccc}
\text{DP}_n & \alpha_1
\end{array}
\]

\[
\langle (e, t), t \rangle
\]

\[
\langle (e, t) \rangle
\]

\[
\ldots \text{e}_n \ldots
\]

- u\textsubscript{e} is any “unused” variable of type e.

- Pronounced with DP\textsubscript{n} in the position of e\textsubscript{n}.

- Useable on any node of type \langle e, t \rangle (subject to any syntactic restrictions we discover later on), keeps category the same (so when \alpha\textsubscript{1} is an NP, so is \alpha\textsubscript{2}).

Nontrivial Translations of English Words.

- \text{tr}(\text{pronoun}) = v_{n,e}

- \text{tr}(\text{every}) = \lambda P_{(e,t)} [\lambda Q_{(e,t)} [\forall x_{e}[P_{(e,t)}(x_e) \rightarrow Q_{(e,t)}(x_e)]]]

- \text{tr}(\text{a}) = \text{tr}(\text{some}) = \lambda P_{(e,t)} [\lambda Q_{(e,t)} [\exists x_{e}[P_{(e,t)}(x_e) \land Q_{(e,t)}(x_e)]]]

- \text{tr}(\text{no}) = \lambda P_{(e,t)} [\lambda Q_{(e,t)} [\neg \exists x_{e}[P_{(e,t)}(x_e) \land Q_{(e,t)}(x_e)]]]

- \text{tr}(\text{most}) = \lambda P_{(e,t)} [\lambda Q_{(e,t)} [\lambda x_{e}[P_{(e,t)}(x_e) \land Q_{(e,t)}(x_e)] > \# \lambda y_{e}[P_{(e,t)}(y_e) \land \neg Q_{(e,t)}(y_e)]]]

- \text{tr}(\text{brown}) = \lambda P_{(e,t)} [\lambda y_{e}[P_{(e,t)}(y_e) \land \text{brown}'(y_e)]]

- \text{tr}(\text{large}) = \lambda P_{(e,t)} [\lambda y_{e}[P_{(e,t)}(y_e) \land \text{largefora}'(P_{(e,t)})(y_e)]]

- \text{tr}(\text{quick}) = \lambda P_{(e,t)} [\lambda y_{e}[P_{(e,t)}(y_e) \land \text{quickfora}'(P_{(e,t)})(y_e)]]

- \text{tr}(\text{young}) = \lambda P_{(e,t)} [\lambda y_{e}[P_{(e,t)}(y_e) \land \text{youngfora}'(P_{(e,t)})(y_e)]]

- \text{tr}(\text{in}) = \lambda x_e [\lambda P_{(e,t)} [\lambda y_{e}[P_{(e,t)}(y_e) \land \text{in}'(x_e)(y_e)]]]

- When and is used to coordinate Ss,
  \text{tr}(\text{and}) = \lambda p_{t} [\lambda q_{t} [q_{t} \land p_{t}]]

- When and is used to coordinate VPs,
  \text{tr}(\text{and}) = \lambda P_{(e,t)} [\lambda Q_{(e,t)} [\lambda x_{e}[Q_{(e,t)}(x_e) \lor P_{(e,t)}(x_e)]]]
• When or is used to coordinate Ss,
  \[ \text{tr}(or) = \lambda p_t[\lambda q_t[q_t \land p_t]] \]

• When or is used to coordinate VPs,
  \[ \text{tr}(or) = \lambda P_{(e,t)}[\lambda Q_{(e,t)}[\lambda x_e[Q_{(e,t)}(x_e) \lor P_{(e,t)}(x_e)]]] \]

Some Constants and their Types.

• buttercup' ∈ ME_{(e,t)}
• john' ∈ ME_{(e,t)}
• mary' ∈ ME_{(e,t)}
• ecuador' ∈ ME_{(e,t)}
• fears' ∈ ME_{(e,(e,t))}
• attacked' ∈ ME_{(e,(e,t))}
• sees' ∈ ME_{(e,(e,t))}
• loves' ∈ ME_{(e,(e,t))}
• likes' ∈ ME_{(e,(e,t))}
• trusts' ∈ ME_{(e,(e,t))}
• in' ∈ ME_{(e,(e,t))}
• tortoise' ∈ ME_{(e,t)}
• philosopher' ∈ ME_{(e,t)}
• sleeps' ∈ ME_{(e,t)}
• linguist' ∈ ME_{(e,t)}
• brown' ∈ ME_{(e,t)}
• largefora' ∈ ME_{((e,t),(e,t))}
• quickfora' ∈ ME_{((e,t),(e,t))}
• youngfora' ∈ ME_{((e,t),(e,t))}
5. Adding times

The model’s interpretation of a constant of type $\tau$ is a function from $I$ into $D_\tau$. This means that the type of one-place predicates becomes $\langle i, \langle e, t \rangle \rangle$.

Needed to add to model:

- $I = \{t_1, t_2, t_3\}$
- $\leq = \{\langle t_1, t_2 \rangle, \langle t_2, t_3 \rangle, \langle t_1, t_3 \rangle\}$

Two ways of looking at tense markers:

- Tense markers as pronouns referring to intervals (Partee)
- Tense markers as existential quantifiers over intervals (Reichenbach)

a. $\text{Hank snores}$ is true iff $\text{now}$ is a time at which Hank snores.

b. $\text{Hank snored}$ is true iff $\exists i[i < \text{now}$ and $i$ is a time at which Hank snores].

c. $\text{Hank will snore}$ is true iff $\exists i[\text{now} < i$ and $i$ is a time at which Hank snores].

5.1. Reichenbachian Tense.

<table>
<thead>
<tr>
<th>Tense</th>
<th>Aspect:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R &lt; S$</td>
<td>$E &lt; R$</td>
</tr>
<tr>
<td>$S, R$</td>
<td>$E, R$</td>
</tr>
<tr>
<td>$S &lt; R$</td>
<td>$E, R$</td>
</tr>
<tr>
<td>$S &lt; R$</td>
<td>$R \subset E$</td>
</tr>
<tr>
<td>$S &lt; R$</td>
<td>$R &lt; E$</td>
</tr>
</tbody>
</table>

Past (ex. -ed)  Perfect (have V-en)/Anterior

Present (ex. -s)  Simple

Future (ex. will)  Progressive (be V-ing)

Prospective/Posterior

6. Adding worlds

The model’s interpretation of a constant of type $\tau$ that is not a name is a function from $W$ into $D_\tau$. This means that the type of one-place predicates becomes $\langle s, \langle e, t \rangle \rangle$.

Needed to add to model:

- $W = \{w_{1,s}, w_{2,s}, w_{3,s}\}$
- An accessibility relation $R_{\langle s, (s,t) \rangle}$, interpreted by $g$

Extensions:

- $\text{tr}(\text{Buttercup is bald}) = \text{bald}'(w^*)(\text{buttercup}')$
- $\text{tr}(\text{Buttercup might be bald}) = \exists w_s[R(w^*)(w_s) \land \text{bald}'(w_s)(\text{buttercup}')]$
- $\text{tr}(\text{Buttercup must be bald}) = \forall w_s[R(w^*)(w_s) \rightarrow \text{bald}'(w_s)(\text{buttercup}')]$
Intensions:

- \( \text{tr}(\text{Buttercup is bald}) = \lambda u \, [\text{bald}'(u_s) (\text{buttercup}')] \) 
- \( \text{tr}(\text{Buttercup might be bald}) = \lambda u \, [\exists w [R(u_s)(w_s) \land \text{bald}'(w_s) (\text{buttercup}')]] \) 
- \( \text{tr}(\text{Buttercup must be bald}) = \lambda u \, [\forall w [R(u_s)(w_s) \rightarrow \text{bald}'(w_s) (\text{buttercup}')]] \) 

7. \textit{de re AND de dicto}

(1) Bond believes that a student in this class is a spy.

\textit{de re:}
- \([\text{a student in this class}]_1 \, [\text{Bond believes that } e_1 \text{ is a spy.}] \)
- belief about a particular individual
- There is a particular individual in this class, Hannah, and Bond believes her to be a spy. Bond doesn’t necessarily know that Hannah is a student.

\textit{de dicto:}
- \([\text{Bond believes that } [\text{a student in this class}]_1 e_1 \text{ is a spy.}] \)
- belief about what is said, the propositional content of a sentence
- Bond believes that there is a student or other in this class that is a spy, but he doesn’t know which student is a spy.