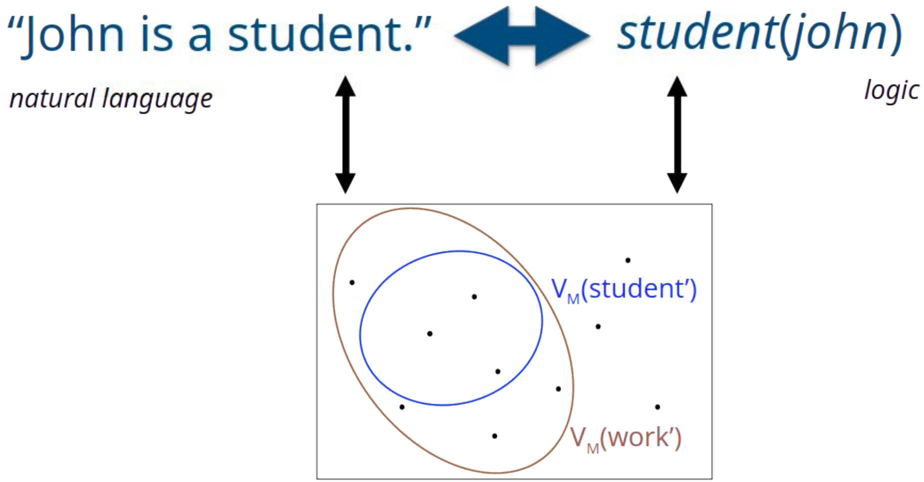
Type Theory

Week 3

Slides and materials based on the courses by
Noortje Venhuizen and Mareike Hartmann

Truth-conditional semantics

- Assumption: a logical formula captures the truth-conditions of an NL sentence—they are true in the same possible models.



Recap: truth, validity, and entailment

- A formula φ is **true** in a model M iff:
  - $[[\varphi]]^{M,g} = 1$ for every variable assignment g

- A formula φ is **valid** ($\vDash \varphi$) iff:
  - φ is true in all models

- A formula φ is **satisfiable** iff:
  - there is *at least one* model M such that φ is true in M

- A set of formulas Γ *entails* formula φ ($\Gamma \vDash \varphi$) iff:
  - φ is true in every model in which all formulas in Γ are true
  - the elements of Γ are called the **premises** or **hypotheses**
    - φ is called the **conclusion**

First-order logic

- Predication and quantification over individual entities

- First-order logic talks about:
  - Individual objects: $V_M(john) \in U_M$, $g(x) \in U_M$
  - Properties of and relations between individual objects: *happy*(*john*), *love*(*john*, *mary*)
  - Quantification over individual objects: $\forall x(happy(x))$

Limitations of first-order logic

- FOL is not expressive enough to capture all meanings that can be expressed by basic natural language expressions:
    - Predicate modifiers: *"Jumbo is a <u>small</u> elephant"*
    - Second-order predicates: *"being rich is a <u>state of mind</u>"*
    - Non-logical sentence operators: *"<u>yesterday</u>, it rained"*
    - Higher-order quantification: *"Bill and John have <u>the same</u> hair color"*

- What system can capture these phenomena?
    - Simple idea: introduce higher order predication & quantification

Russell's Paradox

- What if we extend the FOL interpretation of predicates, and simply interpret higher-order predicates as sets *of sets* of properties?
  - For every predicate P, define a set $\{x \mid P(x)\}$
    - higher order predicates are defined as sets of sets, e.g., $\{P \mid H(P)\}$

- … But: this means that we can formally define a set $S = \{X \mid X \notin X\}$ representing the set of all sets that are not members of themselves
  - Does *S* belong to itself?

Russell's Paradox

- What if we extend the FOL interpretation of predicates, and simply interpret higher-order predicates as sets *of sets* of properties?
  - For every predicate P, define a set $\{x \mid P(x)\}$
    - higher order predicates are defined as sets of sets, e.g., $\{P \mid H(P)\}$

- … But: this means that we can formally define a set $S = \{X \mid X \notin X\}$ representing the set of all sets that are not members of themselves
  - Does *S* belong to itself? **Paradox!**
  - We need a more restricted way of talking about properties and relations between properties

Type Theory: basic and complex types

- In Type Theory, all non-logical expressions are assigned a type (that may be **basic** or **complex**), which restricts how they can be combined.

- Basic types:
    - ***e***: the type of individual terms (“entities”)
    - ***t***: the type of formulas (“truth-values”)

- Complex types: if π, σ are types, then $\langle \pi, \sigma \rangle$ is a type
    - A functor expression that takes an expression of type π as its argument and returns an expression of type σ
    - Sometimes written as $(\pi \rightarrow \sigma)$

Type Theory: types & function application

- Types for first-order expressions:
    - Individual constants (*Luke*, *Death Star*): *e*
    - One-place predicates (*to walk*, *to be a jedi*):
    - Two-place predicates (*to admire*, *to fight with*):
    - Three-place predicates (*to give*, *to introduce*):

Type Theory: types & function application

- Types for first-order expressions:
    - Individual constants (*Luke*, *Death Star*): e
    - One-place predicates (*to walk*, *to be a jedi*): $\langle e, t\rangle$
    - Two-place predicates (*to admire*, *to fight with*):
    - Three-place predicates (*to give*, *to introduce*):

Type Theory: types & function application

- Types for first-order expressions:
  - Individual constants (*Luke*, *Death Star*): e
  - One-place predicates (*to walk*, *to be a jedi*): $\langle e, t\rangle$
  - Two-place predicates (*to admire*, *to fight with*): $\langle e, \langle e, t\rangle\rangle$
  - Three-place predicates (*to give*, *to introduce*):

Type Theory: types & function application

- Types for first-order expressions:
  - Individual constants (*Luke*, *Death Star*): e
  - One-place predicates (*to walk*, *to be a jedi*): $\langle e, t\rangle$
  - Two-place predicates (*to admire*, *to fight with*): $\langle e, \langle e, t\rangle\rangle$
  - Three-place predicates (*to give*, *to introduce*): $\langle e, \langle e, \langle e, t\rangle\rangle\rangle$

- **Function application**: Combining a functor of complex type $\langle \pi, \sigma\rangle$ with an appropriate argument of type π—results in an expression of type σ
  - *jedi*: $\langle e, t\rangle$, *luke*: e $\mapsto$ *jedi*(*luke*): t
  - *admire*: $\langle e,\langle e, t\rangle\rangle$, *luke*: e $\mapsto$ *admire*(*luke*): $\langle e, t\rangle$

More examples of types

- **Higher-order** expressions:
  - Predicate modifiers (*expensive*, *small*): $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$
  - Second-order predicates (*state of mind*): $\langle\langle e, t\rangle, t\rangle$
  - Degree particles (*very*, *too*): $\langle\langle\langle e, t\rangle, \langle e, t\rangle\rangle, \langle\langle e, t\rangle, \langle e, t\rangle\rangle\rangle$
  - Sentence operators*** (*yesterday*, *unfortunately*): $\langle t, t\rangle$

- If π, σ are basic types, $\langle \pi, \sigma\rangle$ can be abbreviated as $\pi\sigma$. The types of predicate modifiers and second-order predicates can then be more conveniently written as: $\langle et, et\rangle$ and $\langle et, t\rangle$

Type Theory: Vocabulary

- Non-logical constants: a (possibly empty) set of non-logical constants for every type *σ*: CON_σ
  - such that the sets for all distinct types are pairwise disjoint
- Variables: an infinite set of variables for every type σ (VAR_σ)
- Logical symbols: ∀, ∃, ¬, ∧, ∨, →, ↔, =
- Brackets: (,)

Type Theory: Syntax

- For every type σ, the set of well-formed expressions (WFFs) WE_σ is defined as follows:
  - $CON_\sigma \subseteq WE_\sigma$ and $VAR_\sigma \subseteq WE_\sigma$;
  - If $\alpha \in WE_{\langle\pi, \sigma\rangle}$, and $\beta \in WE_\pi$, then $\alpha(\beta) \in WE_\sigma$ (function application)
- If A, B are in WE_t, then $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$ are in WE_t
  - If A is in WE_t and x is a variable of *arbitrary* type, then $\forall xA$ and $\exists xA$ are in WE_t
  - If α, β are well-formed expressions *of the same type*, then $\alpha = \beta \in WE_t$;
- Nothing else is a well-formed expression

Type inferencing

- Based on the syntactic structure of a sentence, we can derive its logical form, which defines how functions and arguments are combined

- Each expression that constitutes the logical form obtains a type, which can be inferred from the function-argument structure

- "Luke is a talented jedi" ↦ *talented*(*jedi*)(*luke*)

$$\dfrac{::\ e \qquad \dfrac{::\ \langle\langle e, t\rangle, \langle e, t\rangle\rangle \qquad ::\ \langle e, t\rangle}{::\ \langle e, t\rangle}}{::\ t}$$

Exercise

- Recommended strategy: Start by describing the logical form of the sentences (how are functions and arguments combined, based on the given syntactic bracketing), then derive types for all relevant sub-expressions

1. Yoda$_e$ [is [[fast[er than]] Palpatine$_e$]
2. Yoda$_e$ [is much [faster than]] Palpatine$_e$]
3. [[Han Solo]$_e$ fights] [because [[the Dark Side]$_e$ is rising]]
4. Obi-Wan$_e$ [[told [Qui-Gon Jinn]$_e$] he will take [the Jedi-exam]$_e$]

Higher-order predicates

- Higher-order quantification:

  "Leia has the same hair colour as Padmé"

  $\mapsto \exists C(\textit{hair-color}(C) \wedge C(\textit{leia}) \wedge C(\textit{padme}))$

- Higher-order equality:
  - For $p, q \in CON_t$, $p = q$ expresses *material equivalence*: $p \leftrightarrow q$
  - For $F, G \in CON_{\langle \pi, \sigma \rangle}$, $F = G$ expresses co-extensionality: $\forall x(F(x) \leftrightarrow G(x))$

Type Theory: Semantics (type domains)

- Let U be a non-empty set of entities.

- The domain of possible denotations D_σ for every type σ is given by:
  - $D_e = U$
  - $D_t = \{0, 1\}$
  - $D_{\langle\pi, \sigma\rangle}$ is the set of all functions from D_π to D_σ: ${D_\sigma}^{D\pi}$

- For any type σ, expressions of type σ denote elements of the domain D_σ

Example domains

For $M = (U, V)$, let U consist of five entities. For selected types, we have the following sets of possible denotations:

- $D_t = \{0, 1\}$
- $D_e = U = \{e_1, e_2, e_3, e_4, e_5\}$
- $D_{\langle e, t\rangle} = \{\{\}, \{e_1\}, \ldots, \{e_2, e_3, e_5\}, \ldots, \{e_1, e_2, e_5\}, \ldots\}$

Characteristic functions

- Many natural language expressions have a type $\langle \sigma, t \rangle$, expressing functions that map elements of type σ to truth values {0, 1}

- Such functions with a range of {0, 1} are called **characteristic functions** (of a set), because they uniquely specify a subset of their domain D_σ
  - The **characteristic function** of set S in a domain U is the function F_S: $U \rightarrow \{0, 1\}$ such that for all $e \in \mathrm{U}$, $F_S(e) = 1$ iff $e \in S$
  - NB: For first-order predicates, the FOL denotation (using sets) and the type-theoretic denotation (using characteristic functions) are equivalent

Model-theoretic interpretation

- A model structure for a type theoretic language is a tuple $M = (U, V)$ such that:
  - U is a non-empty domain of individuals
  - V is an interpretation function, which assigns to every $\alpha \in CON_\sigma$ an element of D_σ (where σ is an arbitrary type)

- The variable assignment function g assigns to every typed variable $v \in VAR_\sigma$ an element of the domain D_σ (where σ is an arbitrary type)—$g: VAR_\sigma \rightarrow D_\sigma$

Interpretation of expressions

- Given model structure $M = (U, V)$ and assignment g:
  - $[\![\alpha]\!]^{M,g} = V(\alpha)$ if α is a constant
  - $[\![\alpha]\!]^{M,g} = g(\alpha)$ if α is a variable

- $[\![\alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$ (function application)

Interpretation of formulas

- $[\![\alpha = \beta]\!]^{M,g} = 1$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$
- $[\![\neg\varphi]\!]^{M,g} = 1$ iff $[\![\varphi]\!]^{M,g} = 0$
- $[\![\varphi \wedge \psi]\!]^{M,g} = 1$ iff $[\![\varphi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$
- $[\![\varphi \vee \psi]\!]^{M,g} = 1$ iff $[\![\varphi]\!]^{M,g} = 1$ or $[\![\psi]\!]^{M,g} = 1$
- …

Interpretation of formulas

For any variable v of type σ:

- $[[\exists v\varphi]]^{M,g} = 1$ iff there is a $d \in D_\sigma$ such that $[[\varphi]]^{M,g[v/d]} = 1$
- $[[\forall v\varphi]]^{M,g} = 1$ iff for all $d \in D_\sigma$: $[[\varphi]]^{M,g[v/d]} = 1$

Type-theoretic interpretation: example

"Luke is a talented jedi" $\mapsto$ $talented_{\langle\langle e,\, t\rangle,\, \langle e,\, t\rangle\rangle}(jedi_{\langle e,\, t\rangle})(luke_e)$

$[\![talented(jedi)(luke)]\!]^{M,g}$

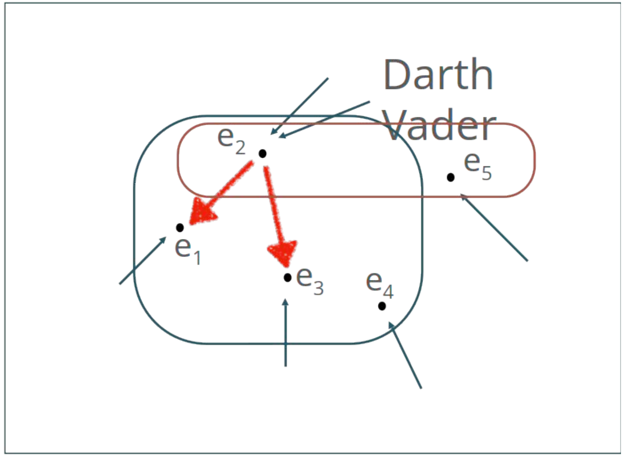
$= [\![talented(jedi)]\!]^{M,g}([\![luke]\!]^{M,g})$

$= [\![talented]\!]^{\mathrm{M,g}}([\![jedi]\!]^{M,g})\ ([\![luke]\!]^{M,g})$

$= V_M(talented)(V_M(jedi))(V_M(luke))$

Defining the right model

M:



Consider the following model *M*:

- $D_e = U_M = \{e_1, e_2, e_3, e_4, e_5\}$
- $V_M(anakin) = V_M(darth\text{-}vader) = e_2$
- $V_M(jedi) = \{e_1, e_2, e_3, e_4\}$, $V_M(dark\text{-}sider) = \{e_2, e_5\}$
- $V_M(powerful) = \{$

  $\{e_1, e_2, e_3, e_4\} \mapsto \{e_2, e_4\}$,

  $\{e_2, e_5\} \mapsto \{e_2, e_5\}$,

  …

  $\}$

Note that "*powerful*" is defined to be truth-preserving: powerful($X_{\langle e,\, t\rangle}$) $\vDash X_{\langle e,\, t\rangle}$

Meaning postulates: restricting denotations

- Some valid inferences in natural language:
  - Bill is a poor piano player $\vDash$ Bill is a piano player
  - Bill is a blond piano player $\vDash$ Bill is blond
  - Bill is a former professor $\vDash$ Bill isn't a professor

- These entailments do not hold in type theory by definition

- **Meaning postulates**: restrictions on models that constrain the possible meanings of certain words

Example: meaning postulates for adjective classes

- Restrictive or subsective adjectives (e.g., "*poor*")
  - Restriction: $[\![\textit{poor}\ \mathrm{N}]\!] \subseteq [\![\ \mathrm{N}\ ]\!]$
  - Meaning postulate: $\forall G \forall x(\textit{poor}(G)(x) \rightarrow G(x))$

- Intersective adjectives (e.g., "*blond*")
  - Restriction: $[\![\textit{blond}\ \mathrm{N}]\!] = [\![\textit{blond}]\!] \cap [\![\ \mathrm{N}\ ]\!]$
  - Meaning postulate: $\forall G \forall x(\textit{blond}(G)(x) \rightarrow (\textit{blond*}(x) \wedge G(x))$
  - NB: $\textit{blond} \in \mathrm{WE}_{\langle\langle e,\, t\rangle,\, \langle e,\, t\rangle\rangle} \neq \textit{blond*} \in \textit{WE}_{\langle e,\, t\rangle}$

- Privative adjectives (e.g.,"*former*")
  - Restriction: $[\![\textit{former}\ \mathrm{N}]\!] \cap [\![\ \mathrm{N}\ ]\!] = \varnothing$
  - Meaning postulate: $\forall G \forall x(\textit{former}(G)(x) \rightarrow \neg G(x))$