Notation

Formally we write, e.g.

\[ \forall v_1 (v_1 \rightarrow \bot) \rightarrow (\forall v_2 (v_1 \land v_2 \rightarrow \bot) \rightarrow \bot) \]

but informally we write

\[ \forall x (x \neq 0 \rightarrow \exists y \ x = Sy) \]

Recall

(\neg \alpha) \quad \text{abbreviates} \quad (\alpha \rightarrow \bot)

(\alpha \lor \beta) \quad \text{abbreviates} \quad ((\neg \alpha) \rightarrow \beta)

(\alpha \land \beta) \quad \text{abbreviates} \quad (\neg (\alpha \rightarrow (\neg \beta))

(\alpha \rightarrow \beta) \quad \text{abbreviates} \quad ((\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha))

\exists x \alpha \quad \text{abbreviates} \quad (\neg \forall x (\neg \alpha))

u = t \quad \text{abbreviates} \quad ut

u < t \quad \text{abbreviates} \quad u < t

u \neq t \quad \text{abbreviates} \quad (\neg = ut)

u \neq t \quad \text{abbreviates} \quad (\neg < ut)

We can drop parentheses as follows

1. Drop outermost parentheses
2. \neg, \forall, \exists apply to as little as possible
   e.g. \forall x \beta \; \text{is} \; (\forall x \alpha \rightarrow \beta) \; \text{not} \; \forall x (\alpha \rightarrow \beta)
3. \land, \lor apply to as little as possible
   after rule 2
4. repeated connectives are grouped on the right
   e.g. \alpha \rightarrow \beta \rightarrow \gamma \; \text{is} \; \alpha \rightarrow (\beta \rightarrow \gamma)
Truth and Models (2.2)

So far our language of first-order logic is just a bunch of symbols. We have yet to (formally) give these symbols meaning. We do this with structures.

Informally, a structure for a first-order language tells us

1. What collection of objects the quantifiers $\forall$ and $\exists$ refer to.
2. What the function, constant, and predicate symbols refer to.

Formally, a structure $\mathcal{L}$ for a given first-order language is a function from the set of parameters such that

1. $\mathcal{L}$ assigns to $\forall$ a nonempty set $\mathcal{L}|$ called the universe of $\mathcal{L}$.

2. $\mathcal{L}$ assigns to each $n$-place predicate symbol $P^n$ an $n$-ary relation $P^n \subseteq \mathcal{L}|^n$.

3. $\mathcal{L}$ assigns to each constant symbol $c$ a member $c_\mathcal{L} \in \mathcal{L}|$.

4. $\mathcal{L}$ assigns to each $n$-place function symbol $f^n$ an $n$-ary function $f^\mathcal{L} : \mathcal{L}|^n \to \mathcal{L}|$. 

\[\text{Fraktur A} \quad \text{Written in book as } \mathcal{A}\]
Remarks

- \( \mathcal{L}_1 \) is always nonempty
- For a constant symbol \( c \),
  \( c \) is a 0-place function symbol and
  \( c^\alpha \) can be thought of as the 0-ary function
  \[
  c^\alpha : \mathcal{L}_1 ^0 \to \mathcal{L}_1
  \]
  \( \left( \text{this is the set } \{< >\} \right) \)
  so \( c^\alpha () = c^\alpha \).
- \( = \) and \( 1 \) are parameters (but there
  meaning is always fixed);
  \( = \) is the relation \( \{< a, b > \in \mathcal{L}_1 ^2 \mid a = b \} \)
  \( 1 \) is the relation \( \emptyset \subseteq \mathcal{L}_1 ^0 \)

Normal Example

Language of elementary number theory

\( \mathcal{L}_1 = \mathbb{N} \)

- \( <^\alpha = \{ < n, m > \in \mathbb{N} ^2 \mid n < m \} \)
- \( S^\alpha : \mathbb{N} \to \mathbb{N} \) s.t. \( S^\alpha (n) = n+1 \)
- \( \ast^\alpha : \mathbb{N} ^2 \to \mathbb{N} \) s.t. \( \ast^\alpha (n, m) = n \cdot m \)
- \( +^\alpha : \mathbb{N} ^2 \to \mathbb{N} \) s.t. \( +^\alpha (n, m) = n + m \)
- \( E^\alpha : \mathbb{N} ^2 \to \mathbb{N} \) s.t. \( E^\alpha (n, m) = n ^ m \)
- \( 0^\alpha = 0 \)

Weird Example

Language of set theory

\( \mathcal{L}_1 = \mathbb{N} \)

- \( \epsilon^\alpha = \{ < n, m > \in \mathbb{N} ^2 \mid n < m \} \)
Example.
Language with a 2-place predicate symbol $E$.
(Language of graph theory)

$\mathcal{L} = \{a, b, c, d\}$

$E^\mathcal{L} = \{\langle a, b \rangle, \langle b, a \rangle, \langle b, c \rangle, \langle c, c \rangle\}$

$a \xrightarrow{b} c \xrightarrow{d}$
Our goal is to define for a sentence $\sigma$, and structure $\mathcal{O}$, "$\sigma$ is true in $\mathcal{O}$" written in symbols as

$$\text{For } \sigma$$

To do this, we will consider this setup:

- $\mathcal{O}$ is a structure
- $s : V \to |\mathcal{O}|$ is a function from a set $V$ of variables which assigns each variable a value in $|\mathcal{O}|$.

Then we will define

- For terms $t$: $s(t)$ by recursion
- For atomic formulas $\phi$: $\text{For } \phi[s]$
- For wffs $\phi$: $\text{For } \phi[s]$ by recursion
- For sentences $\sigma$: $\text{For } \sigma$