

Logic - Notes

Dom Hutchinson

February 10, 2020

Contents

1	Introduction	2
1.1	Alphabets & Strings	2
1.2	Countable Sets	2
2	First-Order Languages	3
2.1	Induction of Terms & Formulae	6
2.2	Free Variables	7
3	Semantics of First-Order Languages	8

1 Introduction

1.1 Alphabets & Strings

Definition 1.1 - Alphabet

An *Alphabet* is a set of symbols from which *Strings* can be created.

Definition 1.2 - String

A *String* over a set \mathcal{A} is any sequence $\alpha := \langle a_1, \dots, a_n \rangle$ where $a_1, \dots, a_n \in \mathcal{A}$.
N.B. Here we say α has *length* n and $\alpha \in \mathcal{A}^n$.

Definition 1.3 - Power Set

Let \mathcal{A} be an alphabet. We define

$$\mathcal{A}^* := \bigcup_{n \in \mathbb{N}} \mathcal{A}^n = \{ \langle a_1, \dots, a_n \rangle : n \in \mathbb{N}; a_1, \dots, a_n \in \mathcal{A} \}$$

This means \mathcal{A}^* is the set of all possible strings over alphabet \mathcal{A} .

Remark 1.1 - Concatenating Strings

Define *Strings* $\alpha := \langle a_1, \dots, a_n \rangle \in \mathcal{A}^n$ and $\beta := \langle b_1, \dots, b_m \rangle \in \mathcal{A}^m$.

We define *Concatenation* of α & β as $\alpha\beta := \langle a_1, \dots, a_n, b_1, \dots, b_m \rangle$ Note that

$$\alpha\beta \neq \langle \alpha, \beta \rangle = \langle \langle a_1, \dots, a_n \rangle, \langle b_1, \dots, b_m \rangle \rangle$$

N.B. Sometimes the following notation is used $\alpha * \beta$.

Example 1.1 - English Alphabet

If we define an alphabet $\mathcal{A} := \{ 'a', \dots, 'z' \}$ then $\langle 't', 'h', 'i', 's' \rangle$ is a *String* of \mathcal{A} .

Remark 1.2 - Ambiguity when using multiple Alphabets

Consider the *Alphabets* $\mathcal{A}_1 := \{0, 1, \dots, 9\}$ & $\mathcal{A}_2 := \mathbb{N}$.

Then we are unsure which of the following definitions of 123 is valid

$$\langle 123 \rangle, \langle 12, 3 \rangle, \langle 1, 23 \rangle, \langle 1, 2, 3 \rangle$$

Remark 1.3 - $\mathcal{A} := \{0, 1\}$ is sufficient to describe any language - binary

Remark 1.4 - Describing Formal Languages

When describing a *Formal Language* we need to provide two things

- i) An *Alphabet* which defines what symbols are allowed.
- ii) A *Grammar* which defines what combinations of symbols are allowed.

1.2 Countable Sets

Definition 1.1 - Countable Set

A set X is said to be *Countable* if

$$\begin{aligned} &\exists \text{ a surjection } f : \mathbb{N} \rightarrow X \\ &\exists \text{ an injection } f : X \rightarrow \mathbb{N} \end{aligned}$$

Definition 1.2 - Countably Infinite Set

A set X is said to be *Countably Infinite* if \exists a bijection $f : X \rightarrow \mathbb{N}$.

Theorem 1.1 - Power set is Countable

If set \mathcal{A} is *countable* then \mathcal{A}^* is *countable*.

Proof 1.1 - Theorem 1.1

Let $f : \mathcal{A} \rightarrow \mathbb{N}$ (This function exists trivially since we define \mathcal{A} to be countable).

Define the following function $g(\cdot) : \mathcal{A}^* \rightarrow \mathbb{N}$

$$g(\langle a_1, \dots, a_n \rangle) := p_1^{f(a_1)+1} \dots p_n^{f(a_n)+1}$$

where p_i is the i^{th} prime.

Since each natural number can be described by a unique composition of primes and since $f(\cdot)$ is injective, then $g(\cdot)$ is injective.

Thus there exists an injection from \mathcal{A}^* to \mathbb{N} , making \mathcal{A}^* countable.

Theorem 1.2 - If \mathcal{A} is countable, then so are $\mathcal{A}^*, (\mathcal{A}^*)^*, \dots$

2 First-Order Languages

Definition 2.1 - First-Order Language, \mathcal{L}

The *Alphabet* of a *First-Order Language*, comprises of the following, pairwise disjoint, categories (and nothing else)

- i) Negation, \neg , and implication, \rightarrow .
- ii) For all, \forall .
- iii) Infinitely many variables, $\{v_0, v_1, \dots\}$.
- iv) Parentheses, (\cdot) , and comma $,$.
- v) Equality, \equiv , which is the only logical predicate symbol with 2-arity.
- vi) A set of constant symbols, $\{c_1, c_2, \dots\}$. (Possibly empty)
- vii) For each $n \geq 1$, a set of n -arity function symbols $\{f_1^n, f_2^n, \dots\}$. (Possibly empty)
- viii) For each $n \geq 1$, a set of n -arity non-logical predicate symbols $\{P_1^n, P_2^n, \dots\}$. (Possibly empty)

N.B. We denote the set of variables by $Var := \{v_0, v_1, \dots\}$; denote a language as \mathcal{L} and the alphabet of \mathcal{L} as $\mathcal{A}_{\mathcal{L}}$.

N.B. In this course *Alphabets* are restricted to being *Countable*.

Definition 2.2 - Negation, \neg

Negation returns in the inverse of a predicate (DO I MEAN PREDICATE)

P	$\neg P$
T	F
F	T

Definition 2.3 - Implication, \rightarrow

Implication returns whether one predicate being true necessarily implies a second predicate being true

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

Remark 2.1 - *First-Order Languages don't have \wedge , \vee , \exists*

Alphabets for *First-Order Languages* do not contain propositional connectives for AND, \wedge , OR, \vee , or EXISTS, \exists since they can be expressed as a combination of negation & implication.

$$\begin{aligned} P \vee Q &\iff \neg P \rightarrow Q \\ P \wedge Q &\iff \neg(P \rightarrow \neg Q) \\ \exists x \text{ st } P(x) \text{ is true} &\iff \neg(\forall x, \neg P(x)) \end{aligned}$$

P	Q	$\neg P$	$\neg P \rightarrow Q$	P	Q	$\neg Q$	$P \rightarrow \neg Q$	$\neg(P \rightarrow \neg Q)$
T	T	F	T	T	T	F	F	T
T	F	F	T	T	F	T	T	F
F	T	F	T	F	T	F	T	F
F	F	F	F	F	F	T	T	F

Example 2.1 - *Recursive Definition*

Consider the following, normal, definition

$$x \text{ is a multiple of } 5 \iff \exists y \in \mathbb{Z} \text{ st } y \cdot 5 = x$$

We can instead use the recursive definition

- i) 0 is a multiple of 5.
- ii) If n is a multiple of 5 then $n + 5$ is a multiple of 5.

Definition 2.4 - *\mathcal{L} -Term & Complexity*

Let \mathcal{L} be a *First-Order Language*.

We define *\mathcal{L} -Terms & Complexity*, $cp(\cdot)$, together using the following *recursive definition*

- i) If s is a variable or a constant symbol, then s is an *\mathcal{L} -Term* with $cp(s) = 0$.
N.B. Terms with $cp(\cdot) = 0$ are called *Atomic Terms*.
- ii) If f is a function symbol with k -arity & if a_1, \dots, a_k are *\mathcal{L} -Terms* then $f(a_1, \dots, a_k)$ is an *\mathcal{L} -Term* with complexity

$$cp(f(a_1, \dots, a_k)) := \max\{cp(a_1), \dots, cp(a_k)\} + 1$$

N.B. Terms with $cp(\cdot) \geq 1$ are called *Compound Terms*.

- iii) Nothing else is an *\mathcal{L} -Term*

N.B. We denote the set of \mathcal{L} - Terms by $T_{\mathcal{M}_{\mathcal{L}}}$.

Example 2.2 - *\mathcal{L} -Term & Complexity*

Let $\{c, d, f, g, h, p\} \subseteq \mathbb{L}$ with c, d being constants, g, p being unary functions & f, h being binary functions.

Show that the following is an *\mathcal{L} -Term* & find its *Complexity*

$$h(g(f(x, c)), p(d))$$

- i) x is an *\mathcal{L} -Term* with $cp(x) = 0$ by (i).

- ii) c & d are \mathcal{L} -Terms with $cp(c) = 0 = cp(d)$ by (i).
- iii) $f(x, c)$ is an \mathcal{L} -Term with $cp(f) = \max 0, 0 + 1 = 1$ by (ii).
- iv) $p(d)$ is an \mathcal{L} -Term with $cp(f) = \max 0 + 1 = 1$ by (ii).
- v) $g(f(x, c), p(d))$ is an \mathcal{L} -Term with $cp(g) = \max 1, 1 + 1 = 2$ by (ii).
- vi) $h(g(f(x, c), p(d)))$ is an \mathcal{L} -Term with $cp(h) = \max 2 + 1 = 3$ by (ii).

Thus $h(g(f(x, c), p(d)))$ is an \mathcal{L} -Term with Complexity 3.

Notation 2.1 - More readable Functions

WE often write $x \circ y$ instead of $\circ(x, y)$ as it is more readable (even though the later is technically the only correct notation). Similarly, $x + y$ instead of $+(x, y)$.

Definition 2.5 - Atomic Formulae

Let \mathcal{L} be a First-Order Language.

The atomic \mathcal{L} -Formulae are those strings over $\mathcal{A}_{\mathcal{L}}$ of the form

$$R(t_1, \dots, t_n) \text{ for } n \in \mathbb{N}$$

where R is a predicate symbol of \mathcal{L} with n -arity and t_1, \dots, t_n are \mathcal{L} -terms.

N.B. $\equiv (t_1, t_2)$ is an Atomic \mathcal{L} -Formula for each \mathcal{L} terms t_1, t_2 .

Definition 2.6 - \mathcal{L} -Formulae & Complexity

We define \mathcal{L} -Formulae & Complexity, $cp(\cdot)$, together using the following recursive definition

- i) If $\phi \in \mathcal{A}_{\mathcal{L}}^*$ is an Atomic \mathcal{L} -Formula then ϕ is an \mathcal{L} -Formula with $cp(\phi) = 0$.
- ii) If ϕ is an \mathcal{L} -Formula with $cp(\phi) = n$ then $\neg\phi$ is an \mathcal{L} -Formula with $cp(\neg\phi) = n + 1$.
- iii) If ϕ & ψ are \mathcal{L} -Formulae then $\phi \rightarrow \psi$ is an \mathcal{L} -Formula with $cp(\phi \rightarrow \psi) = \max\{cp(\phi), cp(\psi)\} + 1$.
- iv) if ϕ is an \mathcal{L} -Formula then $\forall x\phi$ is an \mathcal{L} -Formula with $cp(\forall x\phi) = cp(\phi) + 1$, where x is a variable.

N.B. Complexity is just a measure of the syntactic complexity, not semantic. Notice how $cp(\neg\neg\phi) = cp(\phi) + 2$.

Remark 2.2 - Formulae are uniquely readable & parsable

Example 2.3 - \mathcal{L} -Formulae Complexity

Let $\{R, f\} \subset \mathcal{L}$ be binary operations.

Show that the following is an \mathcal{L} -Formula

$$\forall v_0 (\neg R(f(v_0, v_2), v_2) \longrightarrow \underbrace{\equiv (v_0, v_2)}_{v_0 \equiv v_2})$$

- i) v_0, v_2 are \mathcal{L} -Terms.
- ii) $f(v_0, v_2)$ is an \mathcal{L} -Term.
- iii) $R(f(v_0, v_2), v_2)$ is an \mathcal{L} -Formula with $cp(\cdot) = 0$.
- iv) $\neg R(f(v_0, v_2), v_2)$ is an \mathcal{L} -Formula with $cp(\cdot) = 0 + 1 = 1$.

v) $\equiv (v_0, v_2)$ is an \mathcal{L} -Formula with $cp(\cdot) = 0$.

vi) $\neq R(f(v_0, v_2), v_2) \longrightarrow \equiv (v_0, v_2)$ is an \mathcal{L} -Formula with $cp(\cdot) = \max\{0, 1\} + 1 = 2$.

vii) $\forall v_0 (\neg R(f(v_0, v_2), v_2) \longrightarrow \equiv (v_0, v_2))$ is an \mathcal{L} -Formula with $cp(\cdot) = 2 + 1 = 3$.

Notation 2.2 - *Convention for common operators*

To make formulae more readable we generally make the following allowances in notation

$$\begin{aligned} t_1 &\equiv t_2 & \text{for } &\equiv (t_1, t_2) \\ t_1 &\not\equiv t_2 & \text{for } &\neg \equiv (t_1, t_2) \\ t_1 &< t_2 & \text{for } &< (t_1, t_2) \\ t_1 &\not\equiv t_2 & \text{for } &\neg \equiv (t_1, t_2) \end{aligned}$$

Further, when a formula is encapsulated by parentheses then we will often suppress the outermost parentheses (only), as they do not affect anything.

$$\phi \longrightarrow (\psi \longrightarrow \theta) \text{ for } (\phi \longrightarrow (\psi \longrightarrow \theta))$$

Definition 2.7 - *More complex operators*

- AND, $(\phi \wedge \psi) := \neg(\phi \longrightarrow \neg\psi)$.
- OR, $(\phi \vee \psi) := (\neg\phi \longrightarrow \psi)$.
- IFF, $(\phi \longleftrightarrow \psi) := (\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi)$.
- EXISTS, $(\exists x\phi) := \neg\forall x \neg\phi$.

Notation 2.3 - *Sets of \mathcal{L} Features*

- $T_{\mathcal{M}_{\mathcal{L}}} :=$ Set of \mathcal{L} -Terms.
- $F_{\mathcal{M}_{\mathcal{L}}} :=$ Set of \mathcal{L} -Formulae.
- $\text{Var} :=$ Set of Variables.

Proposition 2.1 - $T_{\mathcal{M}_{\mathcal{L}}}$ & $F_{\mathcal{M}_{\mathcal{L}}}$ are always countable in this course since we assume \mathcal{L} to be finite.

2.1 Induction of Terms & Formulae

Theorem 2.1 - *Inheritance of a Property - \mathcal{L} -Terms*

Let P be a property of \mathcal{L} -Terms.

Suppose the following to be true

- i) All Atomic \mathcal{L} -Terms have property P .
- ii) $\forall k \in \mathbb{N}, \forall$ function symbols f with k -arity: If \mathcal{L} -Terms t_1, \dots, t_k have property P then $f(t_1, \dots, t_k)$ has P .

Then every \mathcal{L} -Term has property P .

Proof 2.1 - *Theorem 2.1*

This is a proof by contradiction. Suppose that i) & ii) are true but there exists some \mathcal{L} -Term which does not have P .

Let t be an \mathcal{L} -Term with minimum complexity st t does not have P .

Then $cp(t) \neq 0$ otherwise i) would be untrue.

Thus $t \equiv f(t_1, \dots, t_k)$ by the minimality of $cp(t)$.

We know that t_1, \dots, t_k have P .

Thus $f(t_1, \dots, t_k)$ has P . This is a contradiction.

Theorem 2.2 - Inheritance of a Property - \mathcal{L} -Formulae

Let P be a property of \mathcal{L} -Formula.

Suppose the following to be true

- i) All Atomic \mathcal{L} -Formulae have property P .
- ii) If $\phi, \psi \in F_{\mathcal{M}_{\mathcal{L}}}$ have P then $\neg\phi, (\phi \rightarrow \psi)$ & $\forall x\phi$ have P to.

Then every \mathcal{L} -Formulae has property P .

Theorem 2.3 - Number of Parentheses

Every \mathcal{L} -Formula has as many left parentheses as right parentheses.

Every \mathcal{L} -Term has as many left parentheses as right parentheses.

Proof 2.2 - Theorem 2.3

This is a proof by induction.

Let P be the property “Has as many left parentheses as right”.

Base Case - When ϕ is an Atomic \mathcal{L} Formula it trivially has equal number of parentheses.

Inductive Case

Let ϕ & ψ be arbitrary \mathcal{L} -Formulae.

Assume that $P(\phi)$ & $P(\psi)$ hold.

We need to show that $P(\neg\phi)$, $P(\phi \rightarrow \psi)$ & $P(\forall x\phi)$ all hold.

We do not need to show $P(\neg\psi)$, $P(\psi \rightarrow \phi)$ & $P(\forall x\psi)$ hold as ϕ & ψ are arbitrary.

We have that $\neg\phi$ and $\forall x\phi$ don't add any brackets, so P holds.

We have that $(\phi \rightarrow \psi)$ add one left & one right parentheses (although they are often suppressed), thus P holds.

Thus by the process of mathematical induction P holds for all \mathcal{L} -Formulae.

N.B. The proof for \mathcal{L} -Terms is very similar.

2.2 Free Variables

Definition 2.1 - Variable Function, $var(\cdot)$

Define $var : \mathcal{A}_{\mathcal{L}}^* \rightarrow 2^{\text{Var}}$ st $var(s)$ is the set of all variables in string s .

Example 2.4 - $\text{Var}(\cdot)$

$$\begin{aligned} var(f(x, f(y, c))) &= \{x, y\} \\ var(f(c, f(c, c))) &= \emptyset \\ var(\equiv, \equiv, \equiv) &= \emptyset \text{ nonsense strings are acceptable} \end{aligned}$$

Definition 2.2 - Free Variables

Free Variables are variables whose value are ambiguous in an \mathcal{L} -Formula.

Definition 2.3 - Free Variable Function, $FV(\cdot)$

We recursively define $FV(\phi)$ for \mathcal{L} -Formulae as ϕ as follows

- i) $FV(\phi) = var(\phi)$ if ϕ is an Atomic \mathcal{L} -Formula.
- ii) $FV(\neg\phi) = FV(\phi)$.
- iii) $FV((\phi \rightarrow \psi)) = FV(\phi) \cup FV(\psi)$.

iv) $FV(\forall x\phi) = FV(\phi) \setminus \{x\}$.

Example 2.5 - Free Variable Function

$$\begin{aligned} FV(\forall x(P(y) \rightarrow Q(x))) &= FV(P(y) \rightarrow Q(x)) \setminus \{x\} \\ &= [FV(P(y)) \cup FV(Q(x))] \setminus \{x\} \\ &= [\{y\} \cup \{x\}] \setminus \{x\} \\ &= \{y\} \end{aligned}$$

Proposition 2.2 - Free Variable Function for more complex operators

$$\begin{aligned} FV(\phi \wedge \psi) &= FV(\neg(\phi \rightarrow \neg\psi)) \text{ by definition of } \wedge \\ &= FV(\phi) \cup FV(\psi) \\ FV(\phi \vee \psi) &= FV(\neg\phi \rightarrow \psi) \text{ by definition of } \vee \\ &= FV(\phi) \cup FV(\psi) \\ FV(\exists x\phi) &= FV(\neg\forall x\neg\phi) \text{ by definition of } \exists \\ &= FV(\phi) \setminus \{x\} \end{aligned}$$

Definition 2.4 - Closed \mathcal{L} -Term

Let t be an \mathcal{L} -Term.

If $var(t) = \emptyset$ then t is called a *Closed \mathcal{L} -Term*.

Definition 2.5 - \mathcal{L} -Sentence

Let ϕ be an \mathcal{L} -Formula.

If $FV(\phi) = \emptyset$ then ϕ is called an *\mathcal{L} -Sentence*.

Example 2.6 - \mathcal{L} -Sentence

$$\begin{aligned} FV(\forall x(P(x) \rightarrow \exists y R(y, x))) &= FV((P(x) \rightarrow \exists y R(y, x)) \setminus \{x\}) \\ &= FV(P(x)) \cup FV(\exists y R(y, x)) \setminus \{x\} \\ &= \{x\} \cup (FV(R(y, x) \setminus \{y\}) \setminus \{x\}) \\ &= \{x\} \cup (\{y, x\} \setminus \{y\}) \setminus \{x\} \\ &= \{x\} \cup \{x\} \setminus \{x\} \\ &= \emptyset \end{aligned}$$

Remark 2.3 - \mathcal{L} -Sentences have no Free Variables and thus no ambiguity in meaning.

3 Semantics of First-Order Languages

Definition 3.1 - \mathcal{L} -Structure

Let \mathcal{L} be a first-order language.

An \mathcal{L} -Structure is an ordered pair $\mathfrak{M} = \langle D, \mathfrak{I} \rangle$

i) D is a non-empty set.

ii) \mathfrak{I} is a function on the non-logical symbols of \mathcal{L} st

- For each predicate symbol $P \in \mathcal{L}$ with n -arity.

$$\mathfrak{I}(P) \subset D^n$$

- For each function symbol f of \mathcal{L} with n -arity

$$\mathfrak{I}(f) : D^n \rightarrow D$$

- For each constant symbol c of \mathcal{L}

$$\mathfrak{I}(c) \in D$$

N.B. D is the domain, \mathfrak{I} is the interpretation.

Notation 3.1 - \mathcal{L} -Structure

For ease we use the following notation wrt \mathcal{L} -Structure

$$|\mathfrak{M}| := D \quad f^{\mathfrak{M}} := \mathfrak{I}(f) \quad c^{\mathfrak{M}} := \mathfrak{I}(c) \quad p^{\mathfrak{M}} = \mathfrak{I}(p)$$

Example 3.1 - \mathcal{L} -Structure

Let $\mathcal{L}_{\text{Ring}} := \{\bar{0}, \bar{1}, \bar{+}, \bar{\cdot}\}$ where $\bar{+}$ & $\bar{\cdot}$ are binary functions and $\bar{0}$ & $\bar{1}$ are constants.

(This is the language for ring theory)

We use the overline to distinguish language symbols from standard symbols.

Define

$$\begin{aligned} D &:= \mathbb{R} \\ \mathfrak{I}(\bar{0}) &= 0 \in \mathbb{R} \\ \mathfrak{I}(\bar{1}) &= 1 \in \mathbb{R} \\ \mathfrak{I}(\bar{+}) &: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \text{ with } (a, b) \mapsto (a + b) \\ \mathfrak{I}(\bar{\cdot}) &: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \text{ with } (a, b) \mapsto (a \cdot b) \end{aligned}$$

We recall $\langle D, \mathfrak{I} \rangle$ is the standard model of the real field.

N.B. We can alternatively write $\langle D, \mathfrak{I} \rangle = \langle \mathbb{R}, 0, 1, +, \cdot \rangle$ for neatness.

Definition 3.2 - Variable Assignment

A *Variable Assignment* over an \mathcal{L} -Structure is a function which maps from the set of variables to the domain of the \mathcal{L} -Structure.

$$s : \text{Var} \rightarrow |\mathfrak{M}|$$

Definition 3.3 - Extension of Variable Assignment

Let \mathfrak{M} be an \mathcal{L} -Structure & s be a variable assignment over \mathfrak{M} .

The function $\bar{s} : T_{\mathfrak{M}_{\mathcal{L}}} \rightarrow |\mathfrak{M}|$ is defined using the following recursion

- i) $\bar{s}(t) = s(t)$ if $t \in \text{Var}$.
- ii) $\bar{s}(t) = t^{\mathfrak{M}}$ if t is a constant symbol.
- iii) $\bar{s}(f(t_1, \dots, t_k)) = f^{\mathfrak{M}}(\bar{s}(t_1), \dots, \bar{s}(t_k))$.

Example 3.2 - Variable Assignment

Let \mathfrak{M} be the standard model of the real field.

Let s be a variable assignment over \mathfrak{M} st $s(x) = s(y) = \pi$. Then

$$\begin{aligned} \bar{s}(x \bar{+} y) &= \bar{+}^{\mathfrak{M}}(\bar{s}(x), \bar{s}(y)) \\ &= \bar{+}^{\mathfrak{M}}(s(x), s(y)) \\ &= \bar{+}^{\mathfrak{M}}(\pi, \pi) \\ &= \pi + \pi \\ &= 2\pi \end{aligned}$$

Theorem 3.1 -

Let s be a variable assignment over \mathfrak{M} , $x \in \text{Var}$ & $d \in |\mathfrak{M}|$.

A new variable assignment $\frac{sd}{x}$ over \mathfrak{M} is defined as

$$\frac{sd}{x}(y) = \begin{cases} d & \text{if } y = x \\ s(y) & \text{otherwise} \end{cases}$$

Definition 3.4 - Satisfaction Relation

Let \mathfrak{M} be an \mathcal{L} -Structure & s be a variable assignment over \mathfrak{M} .

The *Satisfaction Relation*, $\mathfrak{M}, s \models \phi$ between \mathfrak{M}, s and \mathcal{L} -Formula ϕ is recursively defined as

- i) $\mathfrak{M}, s \models t_1 \equiv t_2$ iff $\bar{s}(t_1) = \bar{s}(t_2)$.
- ii) $\mathfrak{M}, s \models P(t_1, \dots, t_k)$ iff $\langle \bar{s}(t_1), \dots, \bar{s}(t_k) \rangle \in P^{\mathfrak{M}} \subset D^k$.
- iii) $\mathfrak{M}, s \models \neg\phi$ iff $\mathfrak{M}, s \not\models \phi$.
- iv) $\mathfrak{M}, s \models \phi \rightarrow \psi$ iff if $\mathfrak{M}, s \models \phi$ then $\mathfrak{M}, s \models \psi$.
- v) $\mathfrak{M}, s \models \forall x\phi$ iff for all $d \in |\mathfrak{M}|$, $\mathfrak{M}_{\frac{sd}{x}} \models \phi$.

Proposition 3.1 - Extension of Satisfaction Relation

$$\begin{array}{lll}
 \mathfrak{M}, s \models \phi \wedge \psi & \text{iff} & \mathfrak{M}, s \models \phi \text{ and } \mathfrak{M}, s \models \psi \\
 \mathfrak{M}, s \models \phi \vee \psi & \text{iff} & \mathfrak{M}, s \models \phi \text{ or } \mathfrak{M}, s \models \psi \\
 \mathfrak{M}, s \models \phi \leftrightarrow \psi & \text{iff} & \mathfrak{M}, s \models \phi \text{ iff } \mathfrak{M}, s \models \psi \\
 \mathfrak{M}, s \models \exists x\phi & \text{iff} & \mathfrak{M}, s_{\frac{d}{x}} \models \phi \text{ for some } d \in |\mathfrak{M}|
 \end{array}$$

Definition 3.5 - Model

Let $\Phi \subset \text{Fml}_{\mathcal{L}}$, a subset of formulae of a first order language \mathcal{L} .

\mathfrak{M}, s is a *model* of Φ iff $\mathfrak{M}, s \models \phi$ for all $\phi \in \Phi$.

N.B. This is denoted $\mathfrak{M}, s \models \Phi$.

Example 3.3 - Model

Let \mathfrak{M} be the standard model of ring theory & s be a variable assignment over \mathfrak{M} st $s(v_1) = 3$ & $s(v_2) = -\pi$.

$$\begin{aligned}
 & \mathfrak{M}, s \models \bar{0} < v_1 + v_2 \\
 \iff & <^{\mathfrak{M}}(\bar{s}(\bar{0}), \bar{s}(v_1 + v_2)) \\
 \iff & <^{\mathfrak{M}}(\bar{0}^{\mathfrak{M}}, \bar{+}^{\mathfrak{M}}(\bar{s}(v_1), \bar{s}(v_2))) \\
 \iff & <^{\mathfrak{M}}(0, \bar{+}^{\mathfrak{M}}(s(v_1), s(v_2))) \\
 \iff & 0 < 3 + (-\pi) \\
 \implies & \mathfrak{M}, s \not\models \bar{0} < v_1 + v_2
 \end{aligned}$$

$$\begin{aligned}
 & \mathfrak{M}, s \models \forall v_2 \exists v_0, v_2 < v_0 \\
 \iff & \text{for all } d \in \mathbb{R}, \mathfrak{M}_{s_{\frac{d}{v_2}}} \models \exists v_0, v_2 < v_0 \\
 \iff & \text{for all } e \in \mathbb{R}, \mathfrak{M}(s_{\frac{d}{v_2}})_{\frac{e}{v_0}} \models v_2 < v_0 \\
 \iff & \text{for all } d \in \mathbb{R} \text{ there is } e \in \mathbb{R}, s(\frac{d}{v_2})_{\frac{e}{v_0}} v_2 < (s_{\frac{d}{v_0}})_{\frac{e}{v_0}} v_0 \\
 \iff & \text{for all } d \in \mathbb{R} \text{ there is } e \in \mathbb{R} \text{ st } d < e
 \end{aligned}$$

Theorem 3.2 -

Let $\mathcal{L}_1, \mathcal{L}_2$ be first order languages.

Define models

$$\begin{aligned}
 \mathfrak{M}_1 &= \langle D, \mathcal{I}_1 \rangle : \mathcal{L}_1 \text{ structure} \\
 \mathfrak{M}_2 &= \langle D, \mathcal{I}_2 \rangle : \mathcal{L}_2 \text{ structure}
 \end{aligned}$$

Note that D is the same for both (*i.e.* Different languages, same world).

Let $\mathcal{L} := \mathcal{L}_1 \cup \mathcal{L}_2$

- i) For all \mathcal{L} -Terms, t , for all variable assignments s_1 over \mathfrak{M}_1 & s_2 over \mathfrak{M}_2 .
 If $\begin{cases} c^{\mathfrak{M}_1} = c^{\mathfrak{M}_2} & \text{for all constants symbols that occur in } t \\ f^{\mathfrak{M}_1} = f^{\mathfrak{M}_2} & \text{for all functions symbols that occur in } t \text{ then } \bar{s}_1(t) = \bar{s}_2(t). \\ s_1(x) = s_2(x) & \text{for all variable symbols that occur in } t \end{cases}$

ii) For all $\phi \in \text{Fml}_{\mathcal{L}}$ & for all variable assignemnts s_1 over \mathfrak{M}_1 & s_2 over \mathfrak{M}_2 .

$$\text{If } \begin{cases} c^{\mathfrak{M}_1} = c^{\mathfrak{M}_2} & \text{for all constants symbols that occur in } \phi \\ f^{\mathfrak{M}_1} = f^{\mathfrak{M}_2} & \text{for all functions symbols that occur in } \phi \\ R^{\mathfrak{M}_1} = R^{\mathfrak{M}_2} & \text{for all predicate symbols that occur in } \phi \\ s_1(x) = s_2(x) & \text{for all variable symbols that occur in } \phi \end{cases} \text{ then } \mathfrak{M}_1 s_1 \models \phi \text{ iff } \mathfrak{M}_2 \models \phi.$$

N.B. If $\mathcal{L} = \emptyset$ it is not very interesting.

Proof 3.1 - Theorem 3.2 i)

This is a proof by induction on \mathcal{L} -Term.

Base Case

Let t be atomic then $\bar{s}_1(t) = \bar{s}_2(t)$ is trivial, $\bar{s}_1(x) = s_1(x) = s_2(x) = \bar{s}_2(x)$ and $\bar{s}_1(c) = c^{\mathfrak{M}_1} = c^{\mathfrak{M}_2} = \bar{s}_2(c)$.

Inductive Case

Let $t = f(t_1, \dots, t_k)$. Then

$$\begin{aligned} \bar{s}_1(t) &= f^{\mathfrak{M}_1}(\bar{s}_1(t_1), \dots, \bar{s}_1(t_k)) \\ &= f^{\mathfrak{M}_1}(\bar{s}_2(t_1), \dots, \bar{s}_2(t_k)) \text{ by inductive hypothesis} \\ &= f^{\mathfrak{M}_2}(\bar{s}_2(t_1), \dots, \bar{s}_2(t_k)) \\ &= \bar{s}_2(t) \end{aligned}$$

□