

Lecture 6. Logic II. Predicate Logic.

1. Predicate Logic.....	1
1.0. Informal introduction.....	1
1.1. Syntax.....	4
1.2. Semantics.....	6
2. Axioms and theories.....	8
2.1. Tautologies, contradictions and contingencies.....	8
2.2. Logical equivalence and laws of Predicate Logic.....	8
2.3. Axioms and theories.....	8
Homework 6.....	10

Reading: Predicate Logic: Chapter 7: 7.1 – 7.2, Chapter 13: 13.1.2 of PMW, pp. 135 –152, 321-331. **Axioms and theories:** Chapter 8: 8.1 (179-183), 8.5.1-8.5.4 (198 – 205).

1. Predicate Logic

1.0. Informal introduction

Predicate Logic (or Predicate Calculus) is the most well known and in a sense the prototypical example of a formal language. On the other hand, Predicate Logic (PL) was not just invented by logicians. It was in a way extracted from the natural language as some special and important part of it. But for a long time it was used mostly for purposes of mathematics (and metamathematics) and was elaborated as a formal language.

In studying Predicate Logic we would like to demonstrate features of formal languages which are most important for us: the notions of model and model-theoretic semantics, and the Principle of Compositionality (which we used already in Statement Logic).

We begin with some examples and remarks. More exact definitions are given below.

The sentences *John loves Mary* and *Everyone whom Mary loves is happy* can be represented as formulas of PL:

<i>John loves Mary</i>	love (John, Mary)
<i>Everyone whom Mary loves is happy</i>	$\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$

The formula $\forall x(\text{even}(x) \rightarrow (x > 1))$ says that there are even numbers greater than 1.

Formulas and other *expressions* of PL are built from individual constants (or simply “constants”), (individual) variables, predicate constants (or predicate symbols), logical connectives (the same as in statement logic) and quantifiers. Each expression belongs to a certain *type*. The type structure of PL is very simple: individuals, relations of different arities, and truth-values.

In our examples we use the following expressions:

Expressions		Types
John, Mary, 1, 2, ...	constants \square	individuals
	\square terms	
x, y, z, x₁, y₁, z₁, x₂, ...	variables \square	individuals
happy, even	unary predicate constants	unary relations
love, >	binary predicate constants	binary relations
love (John, Mary)	\square	
love(Mary, x)	\square	
happy(x)	\square	
even(x)	\square formulas	truth-values
(x > 1)	\square	
$\square x(\text{love}(\text{Mary}, x) \square \text{happy}(x))$	\square	
$\square x(\text{even}(x) \& (x > 1))$	\square	

The first five formulas are examples of *atomic formulas*. Four of them are written in *prefix* notation and the formula $(x > 1)$ is in *infix* notation. The prefix notation will be our legal one but we will use sometimes the infix notation for examples like our arithmetic one where it is traditional.

We can consider Predicate logic as an extension of Statement Logic (in fact, it is better to consider Statement Logic as a very simple but important part of Predicate Logic). Atomic formulas of Predicate Logic such as **love (John, Mary)** or **even(x)** can be considered as representations of atomic statements of Statement Logic (although to evaluate atomic formulas we need to evaluate variables). Complex formulas of PL are constructed from more simple ones with the help the same connectives as in SL and also with the help of quantifiers.

Expressions of Predicate Logic are *interpreted* in *models*. The structure common to all of the models in which a given language is interpreted (the *model structure* for the model-theoretic interpretation of the given language) reflects certain basic presuppositions about the “structure of the world” that are implicit in the language. For PL, any given model consists of the set of truth values $\{1,0\}$, a *domain* D which is some set of objects (or entities), and some n-ary relations on this set.

A *model*, or *interpreted model*, consists of a model structure plus a (“lexical”, or “basic”) interpretation function I which assigns semantic values to all constants.

$$\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$$

So models of PL presuppose a very simple structure of the world. Objects of such a model have no inner structure, they are just “points” of the domain, and relations are collections of tuples of these objects. The interpretation function I links individual constants with elements of domain and predicate symbols with relations.

An *interpretation* $\underline{\quad}^M$, built up recursively on the basis of the basic interpretation function I , assigns to every expression φ its *semantic value* $\underline{\varphi}^M$ in a given model \mathbf{M} . (More precisely, $\underline{\varphi}^{M,g}$, where g is an assignment function, evaluating variables). These semantic values must correspond to the types of the expressions. Thus, in our examples, to the individual constants **John** and **Mary** are assigned certain objects, individual variables take their values in the set of objects (entities), to the predicate constant **love** is assigned a binary relation $\underline{\text{love}}^M$, and to the predicate constant **happy**, a unary relation (property) $\underline{\text{happy}}^M$. Formulas receive truth values. The formula **love (John, Mary)** is true in the model \mathbf{M} if the pair of objects corresponding to the constants **John** and **Mary** belongs to the relation $\underline{\text{love}}^M$.

The formula $\forall x(\text{love}(\text{Mary}, x) \supset \text{happy}(x))$ is true in \mathbf{M} iff:
 for every object d in the domain,
 $d \supset \underline{\text{happy}}^M$ if $\langle \underline{\text{Mary}}^M, d \rangle \in \underline{\text{love}}^M$.

Restating the last statement more carefully and more generally requires talking about semantic values relative to a model *and an assignment g of values to variables*.

The notation $g[d/x]$ means: The variable assignment which is identical to g except for the (possible) difference that $g[d/x]$ assigns the individual d to the variable x .

The complication of needing to talk about $g[d/x]$ comes from formulas with more than one variable, like

$\forall x \exists y(\text{love}(y, x) \supset \text{happy}(x))$ and
 $\forall y \exists x(\text{love}(y, x) \supset \text{happy}(x))$.

So let us restate more carefully, according to the semantics given in **1.1** below, the truth conditions for the formula: $\forall x(\text{love}(\text{Mary}, x) \supset \text{happy}(x))$:

$\underline{\forall x(\text{love}(\text{Mary}, x) \supset \text{happy}(x))}^{M,g} = 1$ iff for each d in D , if $\langle \underline{\text{Mary}}^{M,g[d/x]}, d \rangle \in \underline{\text{love}}^{M,g[d/x]}$, then $d \in \underline{\text{happy}}^{M,g[d/x]}$.

[Exercise 4 offers practice in deriving such truth conditions compositionally.]

So, the meaning of a formula (a sentence) is its truth-conditions: to know the meaning of a formula is to know what the world (the model) must be like if the formula is true. Knowing the meaning of a sentence does not require knowing whether the sentence is *in fact* true; it only requires being able to discriminate between situations in which the

sentence is true and situations in which the sentence is false. Semantics which is based on truth-conditions is called *model-theoretic*.

According the *Principle of Compositionality* the semantics of these formulas – their interpretation in every given model – is defined by semantic rules (rules S1 – S8 below), which correspond in a direct way to the syntactic rules (rules R1—R8 below). The semantics of the whole is based on the semantics of parts by means of this pairing of semantic interpretation rules with syntactic formation rules.

1.1. Syntax

Syntactic Categories:

terms: let Var be a set of (individual) variables and Const be a set of (individual) constants; put $\text{Term} = \text{Var} \sqcup \text{Const}$;

predicate (predicate symbols or predicate constants): let Pred be a set of predicates partitioned into subsets of 1-place (unary) predicates Pred-1, 2-place (binary) predicates Pred-2, ..., n-place predicates Pred-n, ...

formulas (Form).

Notation and examples:

Term(s): (i) (individual) variables: $x, y, z, x_1, y_1, z_1, x_2, \dots$

(ii) (individual) constants: $a, b, c, a_1, \dots, \text{John}, \text{Mary}, \dots, 0, 1, \dots$

Pred-1: **run, walk, happy, calm, ..., even, odd, ...**

Pred-2: **love, kiss, like, see, ..., >, ...**

...

As in Statement Logic, we define the set Form of formulas of PL as a set of words (strings) in the basic vocabulary (alphabet) with the help of recursive rules (operations on the set of formulas). The basic vocabulary (alphabet) consist of the set Term of terms, the set Pred of predicate symbols, the symbols of logical connectives (the same as in SL), quantifier symbols \forall and \exists , and punctuation symbols – left and right brackets and comma (“(”, “)”, “,”).

Basis of recursion:

R1: If $P \in \text{Pred-1}$ and $T \in \text{Term}$, then $P(T) \in \text{Form}$.

If $R \in \text{Pred-2}$ and $T_1, T_2 \in \text{Term}$, then $R(T_1, T_2) \in \text{Form}$.

More general rule: If $R \in \text{Pred-n}$ and $T_1, \dots, T_n \in \text{Term}$, then $R(T_1, \dots, T_n) \in \text{Form}$

Formulas defined by this rule are called *atomic formulas*.

Recursive Syntactic Rules (operations on the set of formulas):

R2 (\Box): If \Box Form, then $\Box\Box$ Form.

R3 ($\&$): If \Box Form and \Box Form, then $(\Box \& \Box)$ Form.

R4 ($\)$: If \Box Form and \Box Form, then $(\Box \)$ Form.

R5 (\Box): If \Box Form and \Box Form, then $(\Box \Box \Box)$ Form.

R6 (\Box): If \Box Form and \Box Form, then $(\Box \Box \Box)$ Form.

R7 ($\Box v$): If v is a variable and \Box Form, then $\Box v\Box$ Form.

R8 ($\Box v$): If v is a variable and \Box Form, then $\Box v\Box$ Form.

Note, that rules **R7** and **R8** represent sets of operations, one operation for every variable from Var .

Open and closed formulas

Let \Box Form. Denote by $\text{Var}(\Box)$ the set of all variables of the formula \Box . Let us define by induction the set $\text{FreeVar}(\Box)$ of *free variables* of the formula \Box :

$\text{FreeVar}(\Box) = \text{Var}(\Box)$ for every atomic formula \Box ;

$\text{FreeVar}(\Box\Box) = \text{FreeVar}(\Box)$;

$\text{FreeVar}(\Box \& \Box) = \text{FreeVar}(\Box \) = \text{FreeVar}(\Box \Box \Box) = \text{FreeVar}(\Box \Box \Box)$
 $= \text{FreeVar}(\Box) \cup \text{FreeVar}(\Box)$

$\text{FreeVar}(\Box v\Box) = \text{FreeVar}(\Box v\Box) = \text{FreeVar}(\Box) - \{v\}$.

A formula \Box is called *closed* (or a *sentence*) if the set of its free variables is empty ($\text{FreeVar}(\Box) = \emptyset$). Formulas which are not closed are called *open*.

Examples:

Atomic formulas: **love(Mary, x), calm(Peter)**

Open formulas: **love(Mary, x), $\Box x(\text{calm}(x) \Box \text{happy}(y))$**

Closed formulas: **love(Mary, Mary), $\Box y(\text{happy}(y) \& \Box x \text{love}(y, x))$**

1.2. Semantics

Model structure: Domain D of entities (individuals)

The set of truth values $\{1,0\}$

I : Interpretation function which assigns semantic values to all constants (individual constants in Term and predicates in Pred-1, Pred-2, ... Pred- n)

$$\mathbf{M} = \langle D, I \rangle$$

Set G of assignment functions $g: \text{Var} \rightarrow D$. So every function g assigns variables to elements of domain D .

Semantic Types assigned to Syntactic Categories:

Term: entities, individuals. The semantic values of this type are the members of D (i.e. $I(c) \in D$ for every individual constant c and $g(v) \in D$ for every variable v and every assignment function g).

Pred-1: sets (of entities). Semantic values of this type are members of $\wp(D)$.

($\wp(D)$ is the power set (the set of all subsets) of D).

Pred-2: relations between entities (sets of pairs). Values: members of $\wp(D \times D)$.

Pred- n : n -place relations; sets of n -tuples of entities. Values: members of $\wp(D \times \dots \times D)$. (I.e. $I(R) \in D^n$ for every n -ary predicate symbol R).

Form: Truth values. Values: members of $\{0,1\}$.

Given a model \mathbf{M} and some assignment function g we will consider semantic interpretation relative to \mathbf{M}, g and use the notation $\llbracket _ \rrbracket_{\mathbf{M},g}$ for the semantic value of an expression $_$ relative to \mathbf{M}, g .

Basic Expressions

A. If $_$ is a variable, then $\llbracket _ \rrbracket_{\mathbf{M},g} = g(_)$.

B. If $_$ is a constant, then $\llbracket _ \rrbracket_{\mathbf{M},g} = I(_)$

(so $\llbracket _ \rrbracket_{\mathbf{M},g} \in D$ if $_$ is an individual constant or variable and $\llbracket _ \rrbracket_{\mathbf{M},g} \in D^n$ if $_$ is an n -ary predicate symbol).

Again, as in the case of Statement Logic, the semantics of PL is the assigning of truth values to our formulas. We will do it recursively with the help of operations and operators on the set of truth values. Operations corresponding to connectives will be the same as in SL. Operators corresponding to quantifiers will be defined in rules **S7** and **S8** below.

Basis of recursion:

S1: If $P \sqsubseteq \text{Pred-1}$ and $T \sqsubseteq \text{Term}$, then $_P(T) _^{M,g} = 1$ iff $_T _^{M,g} \sqsubseteq _P _^{M,g}$.

More general rule: If $R \sqsubseteq \text{Pred-n}$ and $T_1, \dots, T_n \sqsubseteq \text{Term}$, then $_R(T_1, \dots, T_n) _^{M,g} = 1$ iff

$$\langle _T_1 _^{M,g}, \dots, _T_n _^{M,g} \rangle \sqsubseteq _R _^{M,g}.$$

Recursive Semantic Rules (operations and operators) If $\square, \square \sqsubseteq \text{Form}$ and v is a variable, then:

S2 (\square): $_ \square \square _^{M,g} = \square _ \square _^{M,g}$.

S3 ($\&$): $_ (\square \& \square) _^{M,g} = _ \square _^{M,g} \& _ \square _^{M,g}$.

S4 ($()$): $_ (\square \square) _^{M,g} = _ \square _^{M,g} _ \square _^{M,g}$.

S5 (\square): $_ (\square \square \square) _^{M,g} = _ \square _^{M,g} \square _ \square _^{M,g}$.

S6 (\square): $_ (\square \square \square) _^{M,g} = _ \square _^{M,g} \square _ \square _^{M,g}$.

S7 ($\square v$): $_ \square v \square _^{M,g} = 1$ iff for all $d \sqsubseteq D$, $_ \square _^{M,g[d/v]} = 1$.

S8 ($\square v$): $_ \square v \square _^{M,g} = 1$ iff there is a $d \sqsubseteq D$ such that $_ \square _^{M,g[d/v]} = 1$.

The notation $g[d/x]$ means: The variable assignment which is identical to g except for the (possible) difference that $g[d/x]$ assigns the individual d to the variable x .

Rules (operators) **S7** and **S8** can be reformulated as (possibly infinite) conjunctions and disjunctions on the set of truth values:

$$_ \square v \square _^{M,g} = \&_{d \sqsubseteq D} _ \square _^{M,g[d/v]}.$$

$$_ \square v \square _^{M,g} = \vee_{d \sqsubseteq D} _ \square _^{M,g[d/v]}.$$

Truth: Some formulas are true independent of the choice of assignment; those can be called true relative to just \mathbf{M} , i.e. simply true on the given interpretation.

If $\square \sqsubseteq \text{Form}$, then: $_ \square _^M = 1$ iff for all assignments g , $_ \square _^{M,g} = 1$.

$_ \square _^M = 0$ iff for all assignments g , $_ \square _^{M,g} = 0$.

Otherwise $_ \square _^M$ is undefined.

Note that if formula \square is closed then it is true or false in any model \mathbf{M} independent of assignment (i.e. $_ \square _^M = 1$ or $_ \square _^M = 0$).

Example

Consider the example of a very simple PL language containing two individual constants **John** and **Mary**, just two binary predicate symbols **love** (binary) and **happy** (unary) (see examples in 1.0. above).

Let us consider two models, $\mathbf{M1} = \langle D, I_1 \rangle$ and $\mathbf{M2} = \langle D, I_2 \rangle$, $D = \{j, m\}$, $I_1(\mathbf{John}) = I_2(\mathbf{John}) = j$, $I_1(\mathbf{Mary}) = I_2(\mathbf{Mary}) = m$, $I_1(\mathbf{love}) = \{ \langle j, j \rangle, \langle j, m \rangle, \langle m, m \rangle, \langle m, j \rangle \}$, $I_1(\mathbf{happy}) = \{j, m\}$, $I_2(\mathbf{love}) = \{ \langle j, j \rangle, \langle m, j \rangle \}$, $I_2(\mathbf{happy}) = \{m\}$.

It is easy to see that formulas **love (John, Mary)** and **love (Mary, John)** are both true in $\mathbf{M1}$ but only the second one is true in $\mathbf{M2}$. Formula $\exists x(\mathbf{love}(\mathbf{Mary}, x) \wedge \mathbf{happy}(x))$ is true in $\mathbf{M1}$ and false in $\mathbf{M2}$ because for assignment $g(x) = j$ we have

$$\underline{\mathbf{love}(\mathbf{Mary}, x)}_{\mathbf{M2}, g} = 1 \text{ and } \underline{\mathbf{happy}(x)}_{\mathbf{M2}, g} = 0.$$

2. Axioms and theories.

2.1. Tautologies, contradictions and contingencies

As in Statement Logic, some closed formulas of Predicate Logic are always true, i.e. they are true in every model. These are called tautologies. Formulas which are false in every model are called contradictions. All the other formulas are called *contingencies*: their truth values depend on models; they are true in some models and false in others.

2.2. Logical equivalence and laws of Predicate Logic

Two closed formulas are *logically equivalent* if they have the same truth value in every model. As in statement logic, we will use the symbol \equiv to denote logical equivalence between two arbitrary formulas. There are a number of important equivalences, which can be considered as laws of predicate logic. In the following we write $\exists(x)$, $\forall(x)$, etc., for any formula which contains at least one free instance of the variable x .

We can show that next laws hold. They are called *Laws of Quantifier Negations*:

$$\exists x \neg \phi(x) \equiv \neg \forall x \phi(x)$$

$$\forall x \neg \phi(x) \equiv \neg \exists x \phi(x)$$

$$\exists x \phi(x) \equiv \neg \forall x \neg \phi(x)$$

$$\forall x \phi(x) \equiv \neg \exists x \neg \phi(x)$$

In PMW on pp. 147-152 you can find more useful laws of PL.

2.3. Axioms and theories

Contingent formulas can be used to differentiate models, in axioms and theories, describing axiomatizable classes of models.

Let us begin with an example.

Example: how to describe properties of relations ‘parent’ and ‘grandparent’

Consider the example of a tiny PL language containing just two binary predicate symbols **parent** and **grandparent**.

It is easy to see that the formula (i) is tautology: it is true in every model.

$$(i) \quad \forall x \forall y (\text{parent}(x,y) \supset \text{parent}(x,y))$$

And formula (ii) is a contradiction: it is false in every model.

$$(ii) \quad \forall x \forall y (\text{parent}(x,y) \ \& \ \neg \text{parent}(x,y))$$

Formula (iii)

$$(iii) \quad \forall x \forall z (\text{grandparent}(x,z) \supset \exists y (\text{parent}(x,y) \ \& \ \text{parent}(y,z)))$$

is true in model **M** where a given pair of individuals **a** and **c** stand in the **grandparent**-relation, i.e.

$$\langle a,c \rangle \in \text{grandparent}_M$$

iff there exists an individual **b** such that **a** is a parent of **b** and **b** is a parent of **c**, i.e.

$$\langle a,b \rangle \in \text{parent}_M \ \text{and} \ \langle b,c \rangle \in \text{parent}_M.$$

We can see that formula (iii) selects the class of models in which the relation **grandparent** has some properties which the relation expressed by English *grandparent* has in the real world. We can consider formula (iii) as *axiom* describing the relation **grandparent** in real world.

But our axiom (iii), which captures some “correct” properties of the given kinship relations, is evidently insufficient for a complete characterization. It admits, for example, the model (i.e. is true in the model) **M_{BAD}** which consists just of objects **a** and **b** such that:

$$\langle a,b \rangle \in \text{parent}_{M_{BAD}}$$

$$\langle b,b \rangle \in \text{parent}_{M_{BAD}}$$

$$\langle a,b \rangle \in \text{grandparent}_{M_{BAD}}.$$

Consider the formula (iv).

$$(iv) \quad \forall x \forall y (\text{parent}(x,y) \supset \neg (x = y))$$

It is true in some models admitted by the formula (iii), and false in others, for example in the “bad” model **M_{BAD}** considered above. If we add this formula (iv) as another axiom, and take axioms (iii) and (iv) together, we slightly improve the situation, excluding from the class of models corresponding to these two axioms the model **M_{BAD}** along with various other “bad” models.

But it is easy to see that even these two axioms together admit not only “correct” (“intended”) models. To describe correct models of kinship, we need some additional axioms. We will not continue that task here, but will turn to further illustrations of the notion of an axiomatic theory and its models.

Consider the formula (v).

$$(v) \quad \forall x \forall y \forall z ((\text{parent}(x,y) \ \& \ \text{parent}(y,z)) \supset \text{grandparent}(x,z))$$

It’s not difficult to show that this formula is true in all models in which the formula (iii) is true. So formula (v) is included in the theory generated by axiom (iii) and is a theorem of this theory.

And formula (vi) below is false in all models in which the formula (iii) is true, i.e. it is inconsistent with formula (iii) (and with the theory generated by that axiom).

(vi) $\forall x \exists z (\text{grandparent}(x,z) \ \& \ \exists y (\text{parent}(x,y)))$

If we were to add formula (vi) as an axiom to form the set of axioms (iii) and (vi), the resulting theory would be inconsistent, i.e. would have no models at all. And the negation of formula (vi) is in fact a theorem of the theory whose only axiom is (iii).

Axioms and axiomatizable class of models. Theories and theorems.

Let us assume that the set of predicate symbols is fixed.

To each set Σ of closed formulas there corresponds the class Σ^* of all models in which all the formulas of Σ are true.

The class Σ^* is called an *axiomatizable class of models*, and the set Σ is called the set of its *axioms*. But in Σ^* , not only the axioms of Σ may be true. The set Σ^{**} of all closed formulas which are true in Σ^* is called a *theory*, and the formulas of Σ^{**} are called the *theorems* of the theory Σ^{**} . (The axioms are a subset of the theorems; they are the *generators* of the set of theorems. The same theory may often be generated by different choices of axioms.)

Homework 6.

Choose one or more problems that are challenging for you but not impossible – this is an area where some of you have much more background than others, so we've tried to provide a range.

1) Consider models **M1** and **M2** above. Find one or two closed formulas which are false in **M1** and true in **M2**.

2) Write down axioms defining relations **grandparent**, **grandmother** (you will need the relation **mother**). Explain what motivates your axioms – what would go wrong if they were different or if you left one of them out. Mention any points at which you are having to make decisions about the constraints on those relations which could be empirically controversial. Mention any points at which you are having to make decisions about “How much of what we know about grandparents should be captured by the axioms?”.

Write the axioms in predicate logic. Also state them in English.

3) Exercises from PtMW: [Note: if you are new to predicate logic, make sure you do these; if you find them difficult, check with us right away so we can help you find help. On the other hand, if you're already comfortable with predicate logic and would be willing to volunteer to help someone who might need help, let us know.]

Chapter 7, pp. 173, 174. #1 (a,f,m), #3 a-e, #4 a-c, #5 a-d

4) The predicate logic formula $\forall x(\text{love}(\text{Mary}, x) \supset \text{happy}(x))$ is equivalent to the English sentence *Everyone who Mary loves is happy*. Draw a syntactic tree (analogous to the trees 13-11, p.326 and 13-12, p. 328), which shows how that formula (not the English sentence) is built up from its parts according to the syntactic rules of the predicate calculus (in Section 1.1 above).

- (a) Give each node a label that identifies both the syntactic category of the expression it dominates and the number of the syntactic rule by which its immediate constituents were combined (or “Basic”, if that node dominates a basic expression.) [The trees on pp 326, 328 have only rule numbers.]
- (b) Below is a derivation of the truth-conditions of the formula according to the semantic rules of the predicate calculus. Annotate each line by identifying the semantic rule that was applied anywhere within that line (show where), and the node of the tree to which it corresponds. (According to the principle of compositionality, there should be a perfect match between syntactic rule and semantic rule applied at each node.)
- (c) In addition, further annotate the syntactic tree by adding to the label of each non-terminal node the number of the *semantic* rule which was used to combine the meanings of the daughter-node expressions to get the meaning of the whole expression dominated by that node. For nodes dominating basic expressions, indicate whether the semantic rule to use is Rule A or Rule B. (If you’ve done it right, there should be a perfect correspondence between syntactic rules and semantic rules applied at a given node.)

Semantic derivation of truth conditions:

1. $\forall x(\text{love}(\text{Mary}, x) \supset \text{happy}(x))_{M1,g} = 1$ iff for each d in D ,
 $\text{love}(\text{Mary}, x)_{M1,g[d/x]} \supset \text{happy}(x)_{M1,g[d/x]} = 1$.

2. That will hold iff for each d in D ,

$\text{love}(\text{Mary}, x)_{M1,g[d/x]} = 0$ or $\text{happy}(x)_{M1,g[d/x]} = 1$.

3. That will hold iff for each d in D ,

if $\langle \text{Mary}_{M1,g[d/x]}, x_{M1,g[d/x]} \rangle \in \text{love}_{M1,g[d/x]}$, then $x_{M1,g[d/x]} \in \text{happy}_{M1,g[d/x]}$.

4. And that will hold iff for each d in D ,

if $\langle \text{Mary}_{M1,g[d/x]}, d \rangle \in \text{love}_{M1,g[d/x]}$, then $d \in \text{happy}_{M1,g[d/x]}$.

5. I.e., if $\langle I(\text{Mary}), d \rangle \in I(\text{love})$, then $d \in I(\text{happy})$.

5) Here is a nice ‘axioms and models’ problem from Chapter 8: PtMW p. 235, Ex. 13. (BHP learned that one from Hartley Rogers, Jr., in a logic course in graduate school.) Try

it after you have read 8.5.1 – 8.5.4. This is relatively challenging. If the previous ones are easy for you, be sure to try this one.

6) Axiomatizing the notion of a syntactic tree structure. More challenging. And we will probably return to it again later. This is strictly optional for this time. But it may be fun to begin thinking about a linguistic example.

Consider syntactic structures (trees of immediate constituency) as sets of nodes together with some relations (e.g., *is a constituent*, *is an immediate constituent of*, the order from left to right (*is to the left of*, maybe something else). Try to write axioms which define all “well-formed” trees of this kind. Here is a suggestion for how to begin. Let a tree consist of a set **Node** of nodes, with two basic relations defined on **Node**: **Dom** (immediate dominance) and \sqsubset (immediate left-right precedence). Write axioms characterizing **Dom** and \sqsubset . Then you can define further relations and properties for trees, such as **Dom*** for (not-necessarily-immediate) dominance, $<$ for (not-necessarily-immediate) precedence, **Root** as a unary property of a node that is the “top” node of a tree, **Leaf** as a unary property of a node that is a “bottom” or “terminal” node of a tree (note that linguistic trees are always “upside down”, with their root on the top and their leaves on the bottom!). Then try writing axioms for when a set of nodes **Node** forms a well-formed tree. (If you want to keep going, you can try characterizing the relation **C-command** axiomatically, if you are familiar with it, and any other relations and properties of nodes that you can think of. And you could think about adding labels to the nodes – this can be done in several different ways, either by introducing a new set called **Node-labels** and a binary relation **Label** relating elements of **Node** to elements of **Node-labels**, or just by introducing a family of unary relations **S**, **NP**, **VP**, etc.)

If you want to read about one way of axiomatically characterizing trees, look at Chapter 16, pp 431-448, of Partee, ter Meulen and Wall.

If you want to play with it yourself for a while before looking at PtMW, one good way is to work with a partner and be devil’s advocates for each other. One of you proposes an axiomatization of some of the various tree-related notions and/or of what it takes to be a well-formed tree, and the other one tries to construct an unintended model – something that intuitively doesn’t fit those notions but does conform to the proposed axioms. Or maybe the axioms are too strong and you can think of something they wrongly exclude. Take turns being the proposer and the counter-example person. Keep a written record of the process and let us see it – it’s all interesting, not just the endpoint you may reach.