

Expressions		Types
John, Mary, 1, 2, ...	constants } } terms	individuals
x, y, z, x₁, y₁, z₁, x₂, ...	variables }	individuals
happy, even	unary predicate constants	unary relations
love, >	binary predicate constants	binary relations
love (John, Mary)		
love(Mary, x)		
happy(x)		
even(x)	} formulas	truth-values
(x > 1)		
∀x(love(Mary, x) → happy(x))		
∃x(even(x) ∧ (x > 1))		

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* $\| \cdot \|^{M}$, built up recursively on the basis of the basic interpretation function I , assigns to every expression α its *semantic value* $\| \alpha \|^{M}$ in a given model \mathbf{M} . (More precisely, $\| \alpha \|^{M,g}$, 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 $\| \text{love} \|^{M}$, and to the predicate constant **happy**, a unary relation (property) $\| \text{happy} \|^{M}$. 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 $\| \text{love} \|^{M}$.

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

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

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

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

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

[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 $\text{Term} = \text{Var} \cup \text{Const}$;

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: $a, b, c, a_1, \dots, \text{John, Mary, } \dots, 0, 1, \dots$

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 (\neg): If $\varphi \in \text{Form}$, then $\neg\varphi \in \text{Form}$.

R3 (\wedge): If $\varphi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\varphi \wedge \psi) \in \text{Form}$.

R4 (\vee): If $\varphi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\varphi \vee \psi) \in \text{Form}$.

R5 (\rightarrow): If $\varphi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\varphi \rightarrow \psi) \in \text{Form}$.

R6 (\leftrightarrow): If $\varphi \in \text{Form}$ and $\psi \in \text{Form}$, then $(\varphi \leftrightarrow \psi) \in \text{Form}$.

R7 ($\forall v$): If v is a variable and $\varphi \in \text{Form}$, then $\forall v\varphi \in \text{Form}$.

R8 ($\exists v$): If v is a variable and $\varphi \in \text{Form}$, then $\exists v\varphi \in \text{Form}$.

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

Open and closed formulas

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

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

$\text{FreeVar}(\neg\varphi) = \text{FreeVar}(\varphi)$;

$\text{FreeVar}(\varphi \wedge \psi) = \text{FreeVar}(\varphi \vee \psi) = \text{FreeVar}(\varphi \rightarrow \psi) = \text{FreeVar}(\varphi \leftrightarrow \psi)$
 $= \text{FreeVar}(\varphi) \cup \text{FreeVar}(\psi)$

$\text{FreeVar}(\forall v\varphi) = \text{FreeVar}(\exists v\varphi) = \text{FreeVar}(\varphi) - \{v\}$.

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

Examples:

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

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

Closed formulas: **love(Mary, Mary)**, $\exists y(\text{happy}(y) \wedge \forall x \text{love}(y, x))$

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. $I(\mathbf{c}) \in D$ for every individual constant \mathbf{c} and $g(\mathbf{v}) \in D$ for every variable \mathbf{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. $I(\mathbf{R}) \subseteq D^n$ 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 $\|\varphi\|^{M,g}$ for the semantic value of an expression φ relative to \mathbf{M} , g .

Basic Expressions

A. If α is a variable, then $\|\alpha\|^{M,g} = g(\alpha)$.

B. If α is a constant, then $\|\alpha\|^{M,g} = I(\alpha)$

(so $\|\alpha\|^{M,g} \in D$ if α is an individual constant or variable and $\|\alpha\|^{M,g} \subseteq D^n$ if α 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 $P \in \text{Pred-1}$ and $T \in \text{Term}$, then $\|P(T)\|^{M,g} = 1$ iff $\|T\|^{M,g} \in \|P\|^{M,g}$.

More general rule: If $R \in \text{Pred-}n$ and $T_1, \dots, T_n \in \text{Term}$, then $\|R(T_1, \dots, T_n)\|^{M,g} = 1$ iff $\langle \|T_1\|^{M,g}, \dots, \|T_n\|^{M,g} \rangle \in \|R\|^{M,g}$.

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

S2 (\neg): $\|\neg\varphi\|^{M,g} = \neg\|\varphi\|^{M,g}$.

S3 (\wedge): $\|(\varphi \wedge \psi)\|^{M,g} = \|\varphi\|^{M,g} \wedge \|\psi\|^{M,g}$.

S4 (\vee): $\|(\varphi \vee \psi)\|^{M,g} = \|\varphi\|^{M,g} \vee \|\psi\|^{M,g}$.

S5 (\rightarrow): $\|(\varphi \rightarrow \psi)\|^{M,g} = \|\varphi\|^{M,g} \rightarrow \|\psi\|^{M,g}$.

S6 (\leftrightarrow): $\|(\varphi \leftrightarrow \psi)\|^{M,g} = \|\varphi\|^{M,g} \leftrightarrow \|\psi\|^{M,g}$.

S7 ($\forall v$): $\|\forall v\varphi\|^{M,g} = 1$ iff for all $d \in D$, $\|\varphi\|^{M,g[d/v]} = 1$.

S8 ($\exists v$): $\|\exists v\varphi\|^{M,g} = 1$ iff there is a $d \in D$ such that $\|\varphi\|^{M,g[d/v]} = 1$.

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:

$$\|\forall v\varphi\|^{M,g} = \bigwedge_{d \in D} \|\varphi\|^{M,g[d/v]}.$$

$$\|\exists v\varphi\|^{M,g} = \bigvee_{d \in D} \|\varphi\|^{M,g[d/v]}.$$

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

If $\varphi \in \text{Form}$, then: $\|\varphi\|^M = 1$ iff for all assignments g , $\|\varphi\|^{M,g} = 1$.

$\|\varphi\|^M = 0$ iff for all assignments g , $\|\varphi\|^{M,g} = 0$.

Otherwise $\|\varphi\|^M$ is undefined.

Note that if formula φ is closed then it is true or false in any model **M** independent of assignment (i.e. $\|\varphi\|^M = 1$ or $\|\varphi\|^M = 0$).

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, **M1** = $\langle D, I_1 \rangle$ and **M2** = $\langle D, I_2 \rangle$, $D = \{j, m\}$, $I_1(\text{John}) = I_2(\text{John}) = j$, $I_1(\text{Mary}) = I_2(\text{Mary}) = m$, $I_1(\text{love}) = \{\langle j, j \rangle, \langle j, m \rangle, \langle m, m \rangle, \langle m, j \rangle\}$, $I_1(\text{happy}) = \{j, m\}$, $I_2(\text{love}) = \{\langle j, j \rangle, \langle m, j \rangle\}$, $I_2(\text{happy}) = \{m\}$.

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

$$\|\text{love}(\text{Mary}, x)\|^{M_2,g} = 1 \text{ and } \|\text{happy}(x)\|^{M_2,g} = 0.$$

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 \Leftrightarrow 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 $\varphi(x)$, $\psi(x)$, etc., for any formula which contains at least one free instance of the variable x .

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

$$\neg \forall x \varphi(x) \Leftrightarrow \exists x \neg \varphi(x)$$

$$\forall x \varphi(x) \Leftrightarrow \neg \exists x \neg \varphi(x)$$

$$\neg \forall x \neg \varphi(x) \Leftrightarrow \exists x \varphi(x)$$

$$\forall x \neg \varphi(x) \Leftrightarrow \neg \exists x \varphi(x)$$

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

2.3. Axioms and theories: axiomatizing trees

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

For an example domain, let's look at trees again.

Reference: Zwicky, Arnold M., Jr., and Isard, Stephen. 1963. Some aspects of tree theory. Working Paper. Bedford, MA: The MITRE Corporation.

We are going to reproduce the axioms and definitions from sections IA and IB of their paper, where they use first-order predicate logic to define unordered trees and ordered trees, omitting part IC, which defines labeled trees. (**Note:** First-order predicate logic quantifies only over individuals. Second-order predicate logic allows quantification over predicates as well as over individuals.) We will change some of their notation for convenience and familiarity.

Description	Zwicky and Isard	This handout
universal quantifier	(X)	$\forall x$
and	•	\wedge
negation	\sim	\neg
dominates	D	\Downarrow
immediately dominates	\overline{D}	\downarrow
precedes	B ("before")	\ll
immediately precedes	\overline{B}	$<$
equals by definition	<u>for</u>	$=_{def}$

Below we follow Zwicky and Isard with just a few differences in notation, occasionally paraphrasing and annotating.

I. Axioms (identified with A) and definitions (identified with D) for trees, in first order calculus (= first order predicate logic)

A. Unordered Tree (think “mobile”: ordering among sister nodes is unspecified.)

We begin with the notion of an (unordered) *tree*; in a model for this theory, a universe U of objects called *nodes* and one binary relation \Downarrow (“double down-arrow”) on U must be specified. $\Downarrow(x, y)$ is read “ x dominates y ”.

$$A1. \forall x \forall y \forall z (\Downarrow(x, y) \wedge \Downarrow(y, z) \rightarrow \Downarrow(x, z))$$

\Downarrow is transitive.

$$A2. \forall x \forall y (\Downarrow(x, y) \rightarrow \neg \Downarrow(y, x))$$

\Downarrow is asymmetric.

$$D1. \text{Root}(x) =_{\text{def}} \forall y (y \neq x \rightarrow \Downarrow(x, y))$$

x is a *root* iff it dominates every node other than itself.

$$A3. \exists! x \text{Root}(x)$$

There is exactly one root. (See homework problem 7 about defining $\exists!$)

$$D2. \Downarrow(x, y) =_{\text{def}} \Downarrow(x, y) \wedge \neg \exists z (\Downarrow(x, z) \wedge \Downarrow(z, y))$$

x *immediately dominates* (Z&I: *covers*) y iff x dominates y and no z intervenes.

$$A4. \forall x (\neg \text{Root}(x) \rightarrow \exists! y \Downarrow(y, x))$$

Every node except the root has a unique immediate dominator (is uniquely covered.)

Discussion (not in Z&I): There is no axiom that says directly whether the relation \Downarrow is reflexive or not. Do the other axioms already settle it? What are the reflex/sym/trans properties of immediate dominance \Downarrow ? Is either \Downarrow or \downarrow a partial order? A linear order?

How would we have to change axiom A2 if we wanted to make \Downarrow reflexive?

B. Ordered Tree

A tree is ordered by the specification of an additional binary relation \ll on U . $\ll(x, y)$ is read “ x precedes y ”.

$$A5. \forall x \forall y \forall z (\ll(x, y) \wedge \ll(y, z) \rightarrow \ll(x, z))$$

\ll is transitive.

$$A6. \forall x \forall y (\ll(x, y) \rightarrow \neg \ll(y, x))$$

\ll is asymmetric.

$$A7. \forall x \forall y (\ll(x, y) \rightarrow \exists z (\downarrow(z, x) \wedge \downarrow(z, y)))$$

If x precedes y , both are immediately dominated by the same node.

$$D3. \text{Init}(x, y) =_{\text{def}} \downarrow(y, x) \wedge \neg \exists z \ll(z, x)$$

x is an *initial* (*first daughter*) of y iff y immediately dominates x and nothing precedes x .

$$D4. \prec(x,y) =_{def} \prec\prec(x,y) \wedge \neg\exists z (\prec\prec(x,z) \wedge \prec\prec(z,y))$$

x is *the predecessor* of y iff x precedes y and no z intervenes.

$$D5. \text{Fin}(x,y) =_{def} \downarrow(y,x) \wedge \neg\exists z \prec\prec(x,z)$$

x is a *final (last daughter)* of y if y immediately dominates x and nothing succeeds¹ x.

$$A8. \forall x (\neg\exists y \text{Fin}(x,y) \rightarrow \exists!z \prec(x,z))$$

If x is not the final of some y, then x has a unique successor. [BHP: unless x is root?]

$$D6. \text{Term}(x) =_{def} \neg\exists y \downarrow\downarrow(x,y)$$

x is *terminal* if it dominates nothing.

$$A9. \forall x (\neg\text{Term}(x) \rightarrow \exists!y \text{Init}(y,x))$$

Every node that is not terminal has a unique initial (first daughter).

See homework problem 6 for a chance to work with these and other axioms.

Homework 8. for Tues October 17

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. Feel free to consult answers from 2001 and 2004 on the website and then annotate your work if you learn something from what's posted, or if you have any questions. The last problem (for the most part new this year), about trees, is the most linguistically interesting one, and there should be something in it for everyone, so try to include it.

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

[2] **NOT in 2006** (But you'll see it in answer sheets from 2001, 2004, so we're leaving it here. The handout was different then, and had 'parent' examples instead of 'tree' examples.) 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

¹ We follow Z&I and use *succeed* as the converse of *precede* in the informal discussion. Similarly, *successor* is used with "the obvious meaning".

4) The predicate logic formula $\forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x))$ 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. $\| \forall x(\text{love}(\text{Mary}, x) \rightarrow \text{happy}(x)) \|^{M1,g} = 1$ iff for each d in D ,
 $\| \text{love}(\text{Mary}, x) \rightarrow \text{happy}(x) \|^{M1,g[d/x]} = 1$.

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

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

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

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

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

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

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) TREES. Axiomatizing the notion of a syntactic tree structure. We began to look at tree notions at the end of Lecture 3 (section 4, Trees). Now it will be good to begin thinking more seriously about how to axiomatize a useful notion of “tree”, and even better to think about how variations in the choice of axioms correspond to variations in what counts as a possible tree.

Below there are a variety of possible subproblems to work on – pick some that seem the right degree of challenging and/or interesting for you. [Most of this problem was new in 2004, and the addition of the 1963 Zwicky and Isard working paper about trees is new in 2006. This problem has been somewhat revised since the 2004 homework.]

- a) Review the constraints (axioms) suggested in Lecture 3 and write them down using predicate logic.
- b) Compare the axiomatization of trees in Zwicky and Isard to that proposed in Chapter 16, pp 431-448, of Partee, ter Meulen and Wall and to what was suggested in Lecture 3. Can you find examples of (a) disagreements, (b) the same thing said in different ways, (c) places where one axiomatization leaves something open that the other settles?
- c) Look at other definitions of trees in the linguistic literature, and consider how they relate to one or more of the three approaches mentioned just above. The same things can often be defined in many different ways, but the substantive notions may also differ.

For instance, look at the handed-out paper, Blackburn, Gardent, and Meyer-Viol 1993. ‘Talking about trees’, in *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL- 93)*, 21 - 29. Utrecht, the Netherlands. Just try to read section 2 on page 2 (and not necessarily all of it), and see if you can figure out the correspondence between our Chapter 3 notions of dominance and precedence and what you find there. (Note about that paper: ignore all occurrences of “3D” and all oddly placed occurrences of “=”; those are evidently bits of coding that should have stayed invisible.) They define a basic notion of tree without mentioning ordering, and then define an ordering on daughter nodes separately to end up with a definition of “Finite Ordered Trees”. When we defined left-right order, we didn’t mention anything about dominance. Try to figure out whether the results are equivalent; we don’t expect a formal proof, just some argumentation, and if possible an ‘informal proof’ (i.e. a solidly convincing argument.)

There is a separate handout “Trees” (thanks to Chris Potts) with several alternative characterizations of trees spelled out formally, plus a handout “A sampling of papers about trees” with references to still more papers that include definitions of trees.

Terminology note for reading the first definition (from Blackburn et al 2001): If S is a relation, S^* is the “ancestral” of that relation, its reflexive transitive closure. S^+ is also a kind of ancestral of S , but it omits the reflexive part. It’s just the “positive” transitive closure.

- d) Study the definitions and axioms in Zwicky and Isard (1963). Answer some of the questions included in the text above, and raise and answer similar questions about other parts of their system (e.g. the part about labeling nodes.)

One interesting point noted there is that to give all the axioms in first order predicate calculus, you need to take ‘dominates’ and ‘precedes’ as basic; if you use second-order predicate calculus, then you can take “immediately dominates” and “immediately precedes” as basic and define “dominates” and “precedes” by an inductive (recursive) definition. (See the footnote on page 4: it is the “and nothing else” clause that can’t be written in first-order predicate calculus.)

One nice thing about this paper is that it starts with basic definitions, and then shows how various further classes of trees (and “forests”) can be defined.

e) Go through the definitions of trees on Chris’s “Trees” handouts, and see what, if anything, corresponds in each of them to (i) ‘dominates’ and ‘immediately dominates’, (ii) ‘precedes’ (left-right) and ‘immediately precedes’. (You will always find something that corresponds somehow to domination; you won’t always find any left-right order relation.)

- If you find ‘immediately dominates’, it should be irreflexive, asymmetric, and intransitive. ‘Dominates’, if defined explicitly, should be transitive; it may or may not be defined to be reflexive. What do you find in each of those tree definitions?
- Does each of those tree definitions require that no node can have two mothers? If so, how? If not, give an example of a ‘tree’ where some node has two mothers but the whole ‘tree’ satisfies the definition. (You can do this and the rest for any or all of the given definitions, plus the one in PtMW.)
- Does each of those tree definitions find a way to prevent the possibility where node a dominates b, b dominates c, and c dominates a?
- Can you find any interesting idiosyncracies in any of the definitions? Do you have any questions about any of them? Does anything look like a possible mistake?

f) Go back to the related homework exercise from Homework 3, involving playing with alternative axioms and examining consequences, and try some aspect of it now.

7) The notation $\exists!x P(x)$ means there is one and only one x such that $P(x)$. Find a way to write that using only the basic quantifiers and connectives of predicate logic.