Review of the Art of Logic

Hans Halvorson

February 14, 2013

1 Syntax

1 Definition. A first-order *language* L consists of a set of symbols, divided as follows:

- constant symbols; for example, $c_1, c_2, c_3, \ldots$;
- function symbols; for example, f, g;
- relation symbols; for example, P, R.

Each function and relation symbol is assumed to come with a fixed "arity," i.e. a number of trailing blank spaces.

2 Definition. The *alphabet* of a first-order language L consist of:

- elements of L, the so-called "non-logical vocabulary";
- elements of the list: $(,), =, x, y, z, x_1, x_2, x_3, \ldots$

(Here the comma separates members of the alphabet of L; it is not itself in that alphabet.) The symbols $x, y, z, x_1, x_2, x_3, \ldots$ are called *variables*.

Certain strings of symbols are called *terms* of the language L. But for the sake of precision, we define terms along with their free variables.

3 Definition. The *terms* of L are defined inductively by:

- each variable v is a term with $\mathrm{FV}(v) = v$;
- each constant symbol c is a term with $\mathrm{FV}(c) = \emptyset$;

- if $t_1, \ldots, t_n$ are terms and f is a n-place function symbol, then $f(t_1, \ldots, t_n)$ is a term with $\mathrm{FV}(f(t_1, \ldots, t_n)) = \mathrm{FV}(t_1)\mathrm{FV}(t_2), \ldots, \mathrm{FV}(t_n)$.

4 Example. Consider the language L with two binary function symbols $+$ and $\cdot$ and with two constant symbols 0 and 1. Then the following are terms of this language:

$$+(x, y) \qquad +(x, x) \qquad +(0, 1) \qquad \cdot(+(x, y), +(x, y))$$

Can you say which variables are free in each term?

Other strings of symbols are called L formulas.

5 Definition. The *formulas* of L are defined inductively by:

- If t_1 and t_2 are terms, then $t_1 = t_2$ is a formula with $\mathrm{FV}(t_1 = t_2) = \mathrm{FV}(t_1), \mathrm{FV}(t_2)$;
- If R is a n-place relation symbol, and $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is a formula with $\mathrm{FV}(R(t_1, \ldots, t_n)) = \mathrm{FV}(t_1), \ldots, \mathrm{FV}(t_n)$;
- If ϕ and ψ are formulas, then so are $\phi \wedge \psi, \phi \vee \psi, \phi \to \psi$ and $\neg\phi$. The free variables are just concatenations of the free variables in the subformulas.
- If ϕ is a formula such that x is included in $\mathrm{FV}(\phi)$, then $\exists x\phi$ is a formula with $\mathrm{FV}(\exists x\phi) = \mathrm{FV}(\phi) - \{x\}$.
- If ϕ is a formula such that x is included in $\mathrm{FV}(\phi)$, then $\forall x\phi$ is a formula with $\mathrm{FV}(\forall x\phi) = \mathrm{FV}(\phi) - \{x\}$.

6 Definition. An L-formula ϕ is called a *sentence* just in case $\mathrm{FV}(\phi) = \emptyset$.

7 Example. Use the same language as above, but for convenience, we write the function symbols in infix notation rather than prefix notation. So, instead of writing $+(x, y)$ we write $x + y$. Then the following are formulas of this language:

$$x + y = y + x \qquad \forall x \forall y(x + y = y + x) \qquad 0 = 1$$

Can you say which of these formulas are sentences?

8 Note. Sometimes it is convenient to introduce an additional symbol $\perp$ whose intended meaning is "the absurd," or more technically, "a sentence that is always false." In most cases, one can think of $\perp$ as an abbreviation for the sentence $\forall x(x \neq x)$, or for the sentence $\phi \wedge \neg\phi$, where ϕ is any sentence of the relevant language.

2 Derivation rules

We are going to define a relation "$\Rightarrow$" that holds between finite sets of sentences (on the left) and a single sentence (on the right). Thus, $\Gamma \Rightarrow \phi$ will mean that the set of sentences Γ stands in the relation $\Rightarrow$ to the sentence ϕ. This string of symbols "$\Gamma \Rightarrow \phi$" is called a *sequent*, and its intended meaning is that "there is a *derivation* or *proof* of ϕ from Γ."

The set of valid sequents is defined in an inductive fashion — with base cases, and inductive clauses. The first base case is the Rule of Assumptions:

$$\frac{}{\phi \Rightarrow \phi}$$

The inductive clauses consist of two rules for each connective — an introduction rule and an elimination rule.

$$\frac{\Gamma \Rightarrow \phi,\ \Delta \Rightarrow \psi}{\Gamma \cup \Delta \Rightarrow \phi \wedge \psi} \qquad \frac{\Gamma \Rightarrow \phi \wedge \psi}{\Gamma \Rightarrow \phi}$$

$$\frac{\Gamma \Rightarrow \phi}{\Gamma \Rightarrow \phi \vee \psi} \qquad \frac{\Gamma \Rightarrow \phi \vee \psi,\ \Delta \cup \{\phi\} \Rightarrow \chi,\ \Theta \cup \{\psi\} \Rightarrow \chi}{\Gamma \cup \Delta \cup \Theta \Rightarrow \chi}$$

$$\frac{\Gamma \cup \{\phi\} \Rightarrow \psi}{\Gamma \Rightarrow \phi \rightarrow \psi} \qquad \frac{\Gamma \Rightarrow \phi \rightarrow \psi,\ \Delta \Rightarrow \phi}{\Gamma \cup \Delta \Rightarrow \psi}$$

$$\frac{\Gamma \cup \{\phi\} \Rightarrow \bot}{\Gamma \Rightarrow \neg\phi} \qquad \frac{\Gamma \Rightarrow \neg\neg\phi}{\Gamma \Rightarrow \phi}$$

The $\forall$ intro rule and the $\exists$ elim rule have special restrictions.

$$\frac{\Gamma \Rightarrow \forall x \phi(x)}{\Gamma \Rightarrow \phi(a)}$$

$$\frac{\Gamma \Rightarrow \phi(a)}{\Gamma \Rightarrow \forall x \phi(x)} \quad a \text{ does not occur in } \Gamma \text{ or } \phi(x)$$

$$\frac{\Gamma \Rightarrow \phi(a)}{\Gamma \Rightarrow \exists x \phi(x)}$$

$$\frac{\Gamma \Rightarrow \exists x \phi(x), \quad \Delta, \phi(a) \Rightarrow \psi}{\Gamma \cup \Delta \Rightarrow \psi} \quad a \text{ does not occur in } \Gamma,\ \Delta,\ \phi(x) \text{ or } \psi$$

Finally, the symbol = has its own introduction and elimination rules.

$$\frac{}{\Rightarrow a = a} \qquad \frac{\Gamma \Rightarrow \phi(a), \quad \Delta \Rightarrow a = b}{\Gamma \cup \Delta \Rightarrow \phi(b)}$$

9 Note. In what follows, I'll sometimes use Γ, Δ as shorthand for $\Gamma \cup \Delta$, and Γ, ϕ as shorthand for $\Gamma \cup \{\phi\}$.

10 Note. The rules for $\neg$ don't follow the pattern exactly. Indeed, there is a sense in which there is no proper elim rule for the $\neg$ symbol; and that has led some logicians to think that this rule should be dropped. If it is dropped then the resulting subsystem is called *intuitionistic* logic, and it is connected to a view in the philosophy of mathematics called *intuitionism* or *constructivism*.

11 Note. Because of the way that it is defined, the relation $\Rightarrow$ has a special feature: the collection of valid sequents is *effectively enumerable*. Basically this means that we could program a computer with the rules for producing valid sequents, and then if we allowed this program to run forever, it would produce all and only the valid sequents.

Suppose that there were also a computer program that produced all and only the *invalid* sequents. Then, I claim, there would be a program that decides which sequents are valid. Indeed, if you gave me a sequent $\Gamma \Rightarrow \phi$, I would run both programs — the one that spits out valid sequents and the one that spits out invalid sequents — and I would check $\Gamma \Rightarrow \phi$ against the outputs of both programs. The sequent $\Gamma \Rightarrow \phi$ is guaranteed to appear eventually on one of the lists, and so we will eventually get an answer to our question about whether $\Gamma \Rightarrow \phi$ is valid.

So the set of valid sequents is decidable if and only if the set of invalid sequents is effectively enumerable. Is the set of invalid sequents effectively enumerable? That's one of the questions that we will answer in this course.

12 Proposition (Montonicity)**.** *If* $\Gamma \Rightarrow \phi$ *then* $\Gamma, \psi \Rightarrow \phi$ *for any sentence* ψ*.*

Proof. Suppose that $\Gamma \Rightarrow \phi$. We have $\psi \Rightarrow \psi$ by the rule of assumptions; so $\wedge$-intro gives $\Gamma, \psi \Rightarrow \phi \wedge \psi$ and $\wedge$-elim gives $\Gamma, \psi \Rightarrow \phi$. □

In fact, a slightly stronger fact is true: if $\Gamma \Rightarrow \phi$ then $\Gamma \cup \Delta \Rightarrow \phi$ for any finite set Δ of sentences. To see that the stronger claim is true, just replace Δ with the conjunction of all sentences that occur in it.

Exercise 1. Show that if $\Gamma \Rightarrow \bot$, then $\Gamma \Rightarrow \phi$ for any sentence ϕ. This derived rule is called *ex falso quodlibet*.

Exercise 2. Show that if $\Gamma \Rightarrow \phi \vee \psi$ and $\Delta \Rightarrow \neg\phi$ then $\Gamma \cup \Delta \Rightarrow \psi$. This derived rule is called *disjunctive syllogism*.

3 Naive Set Theory

Before we move on to semantics, we need to review some elementary set theory. Here are some of the key notions that we need to understand:

- When is one set "contained" in another?
- When are two sets equal?
- What is a relation on a pair of sets?
- What is a function from one set to another?
- What is the composite of two functions?

The primitive (undefined) relation in set theory is membership: $a \in S$ means that the individual "a" is a member of the set S. Using the relation of membership, we can define relations between sets, for example S is a subset of T, written $S \subseteq T$ just in case

$$\forall x(x \in S \rightarrow x \in T).$$

We can also define operations on sets. For example, if S and T are sets, then we define their intersection, written $S \cap T$ by

$$x \in S \cap T \quad \text{if and only if} \quad x \in S \wedge x \in T.$$

We give a similar definition for $S \cup T$ in terms of the logical connective "or", and we define the relative complement $S - T$ in terms of the logical connective "not". We will also assume that there is a unique empty set $\emptyset$, defined by

$$\forall x(x \notin \emptyset).$$

We will also make liberal use of composition principles, which say that if $\phi(x)$ is a predicate, then we can define a set of things that satisfy $\phi(x)$.[1] We write this set as $\{x : \phi(x)\}$,

[1]Here is where we are most naive. It is well known that the unrestricted use of this principle leads to absurdity. For example, let $\phi(x)$ be the predicate $x \notin x$. Then let $y = \{x : \phi(x)\}$, and ask yourself if $y \in y$.

which is read as "those x such that $\phi(x)$." For example, the following is a true fact of set theory

$$S \cap T = \{x : x \in S \wedge x \in T\}.$$

Here the predicate $\phi(x)$ is $x \in S \wedge x \in T$.

13 Definition. If S and T are sets, then $S = T$ just in case $\forall x(x \in S \leftrightarrow x \in T)$.

14 Definition. Let A and B be sets. We define the *Cartesian product* $A \times B$ to be the set of ordered pairs

$$A \times B = \{\langle a, b\rangle : a \in A \wedge b \in B\}.$$

Here we stipulate that $\langle a, b\rangle = \langle a', b'\rangle$ just in case $a = a'$ and $b = b'$.

15 Example. Let 0 denote the emptyset. Then for any set A, $A \times 0$ consists of pairs $\langle a, b\rangle$ where a is required to be in A, and b is required to be in 0. But no b satisfies this requirement, hence $A \times 0$ is empty.

16 Definition. Given two sets A and B, a *relation* on the pair A, B is a subset of $A \times B$.

Exercise 3. Let 2 be a set with two elements. How many relations are there on the set 2×2?

17 Definition. Given two sets A and B, a *function* f from A to B, written $f : A \to B$ is a relation on A, B subject to the following two constraints:

1. $\forall x(x \in A \to \exists y(y \in B \wedge \langle x, y\rangle \in f)$.
2. $\forall x \forall y \forall z((x \in A \wedge y \in B \wedge z \in B \wedge \langle x, y\rangle \in f \wedge \langle x, z\rangle \in f) \to y = z)$.

That second condition can be difficult to read. In log-lish, it reads "for any $x \in A$ and $y, z \in B$, if $\langle x, y\rangle \in f$ and $\langle x, z\rangle \in f$, then $y = z$." Thus, the two conditions together are equivalent to saying: for any $x \in A$, there is a unique $y \in B$ such that $\langle x, y\rangle \in f$.

18 Note. Of course, we don't usually write $\langle x, y\rangle \in f$ but instead $f(x) = y$. In some mathematics literature, the set of ordered pairs $\{\langle x, y\rangle : \langle x, y\rangle \in f\}$ is called the *graph* of f. So we have essentially defined a function as *identical* to its graph.

19 Example. There are a couple of natural functions from the Cartesian product $A \times B$ to A and B respectively. Recall that a function from $A \times B$ to A will be a subset of $(A \times B) \times A$. We define $\pi_A : A \times B \to A$ to consist of the triples of the form $\langle x, y, x\rangle$ with $x \in A$ and $y \in B$. (Exercise: check that this relation satisfies the criteria to be a function.) Using the standard function notation, $\pi_A(x, y) = x$, which is called the *projection* onto A. There is a similar projection π_B onto B.

20 Example. Any set A is naturally isomorphic to the set $1 \times A$, where $1 = \{*\}$ is a one-point set. Let B be a subset of A, and consider the projection $\pi : 1 \times A \to 1$ onto the first coordinate. Then the image of B under π,

$$\pi(B) = \{x : x \in 1 \wedge \exists b(b \in B \wedge \pi(b) = x\},$$

is $\{*\}$ if B is non-empty, and is $\emptyset$ if B is empty. This fact is trivial, but it is crucial for understanding how to interpret quantified sentences. See Example 30 below.

21 Example. Let A, B and S be sets, and suppose that $f : S \to A$ and $g : S \to B$ are functions. Then we can define a function $\langle f, g\rangle : S \to A \times B$ by setting

$$\langle f, g\rangle(s) \;=\; \langle f(s), g(s)\rangle,$$

for all $s \in S$. Similarly, if A', B' are sets and $h : A \to A'$ and $g : B \to B'$ are functions, then we can define a function $(h \times k) : A \times B \to A' \times B'$ by setting

$$(h \times k)(a, b) = \langle h(a), k(b)\rangle,$$

for all $a \in A$ and $b \in B$.

22 Definition. Suppose that $f : A \to B$ and $g : B \to C$ are functions. Then we define the composite function $g \circ f : A \to C$ by

$$(g \circ f)(x) \;=\; g(f(x)).$$

23 Example. Let f be a function from $A \times B$ to C, and let $g : A' \to A$ and $h : B' \to B$ be functions. Then $f \circ (g \times h)$ is the function from $A' \times B'$ to C that sends an element $\langle x, y\rangle$ of $A' \times B'$ to the element $f(g(x), h(y))$ of C.

Exercise 4. Let A, B, C, D be sets. Show that

$$(A \times B) \cap (C \times D) \;=\; (A \cap C) \times (B \cap D).$$

Exercise 5. Show that there are sets A, B, C, D such that

$$(A \times B) \cup (C \times D) \;\neq\; (A \cup C) \times (B \cup D).$$

Exercise 6. Let $f : A \to B$ be a function. For a subset S of B, we define the preimage of S

under f by

$$f^{-1}(S) \;=\; \{x : x \in A \wedge f(x) \in S\}.$$

Show that

$$f^{-1}(S \cap T) \;=\; f^{-1}(S) \cap f^{-1}(T).$$

Exercise 7. Let 1 be a one-point set, and let 0 be the emptyset. How many functions are there from 0 to 1? How many functions are there from 1 to 0?

4 Semantics

The central notion in semantics is that of a "structure" for a language. The basic idea — made precise in the following definition — is that a structure maps elements of a language to appropriate sorts of objects in the universe of sets. A structure for L is sometimes also called an "interpretation" of L.

24 Definition. Let L be a first-order language. An L-structure M consists of:

- A non-empty set $|M|$ called the "universe," or the "domain of quantification";
- For each constant symbol c, an element c^M of $|M|$;
- For each n-place relation symbol R, a set R^M of n-tuples of elements of $|M|$;
- For each n-place function symbol f, a function f^M from n-tuples of elements of $|M|$ to elements of $|M|$.

25 Example. Let L be the language with symbols $\{\circ, ^{-1}, e\}$ where the first is a binary function symbol, the second a unary function symbol, and the third a constant symbol. Then $\mathbb{R}$ is an L-structure, where $\circ^{\mathbb{R}}$ is addition of real numbers, $(-1)^{\mathbb{R}}$ is additive inverse, and $e^{\mathbb{R}} = 0$.

Subsequently we will use the same notation for a structure M and its domain $|M|$, when no confusion can result.

26 Proposition. *An L structure M extends to an assignment that takes each term t with free variables among $\vec{x} = x_1, \ldots, x_n$ to a function $t^M : M^n \to M$.*

Proof. The proof of this proposition proceeds by constructing the assignment. To be more precise, we show that for a fixed term t, and for any sequence $\vec{x} = (x_1, \ldots, x_n)$ of variables containing the free variables in t, there is a corresponding function $(\vec{x}.t)^M$ from M^n to M.

– Let t be a constant symbol, say c, and let $\vec{x}$ be a sequence of variables. Define $(\vec{x}.c)^M$ to be the function from M^n to M with constant value c^M.

– Let t be a variable v, and let $\vec{x}$ be a sequence such that $v = x_i$ for some i. We then let $(\vec{x}.t)^M$ be the function from M^n to M that projects out all but the i-th coordinate.

– Suppose that t is of the form $f(t_1, \ldots, t_n)$, and let $\vec{x}$ be a sequence containing all the free variables in $f(t_1, \ldots, t_n)$. Then each term $t_1, \ldots, t_n$ has free variables among $\vec{x}$, and we may assume that $(\vec{x}.t_i)^M$ is defined as a function from M^n to M. We then define $[\vec{x}.f(t_1, \ldots, t_n)]^M$ to be the function obtained by composing f^M with $(\vec{x}.t_i)^M$.

□

27 Example. Let L be the language with a single binary function symbol $\circ$, which we will write in infix notation. Let M be the L structure with $M = \{0, 1\}$ and where $\circ^M = +$ is addition modulo 2. Then $[x.(x \circ x)]^M$ is the function f of one-variable defined by

$$f(a) \;=\; a + a.$$

In contrast, $[x, y.(x \circ x)]^M$ is the function g of two variables defined by

$$g(a, b) \;=\; a + a.$$

Similarly, $[x, y, z.(x \circ (y \circ z))^M$ is the function h of three variables defined by

$$h(a, b, c) \;=\; a + (b + c).$$

28 Proposition. *An L-structure M extends to a map that takes formulas with free variables among $\vec{x} = x_1, \ldots, x_n$ to subsets of M^n.*

Proof. As in the previous proof, we construct for each formula ϕ, and for each sequence $\vec{x}$ containing the free variables of ϕ, a subset $[\vec{x}.\phi]^M$ of M^n.

- Let t_1 and t_2 be terms so that $t_1 = t_2$, and suppose that $\vec{x}$ is a sequence containing the free variables of $t_1 = t_2$. Thus, $\vec{x}$ contains the free variables of t_1 and t_2, and $(\vec{x}.t_1)^M$ and $(\vec{x}.t_2)^M$ are defined as functions from M^n to M. We define $[\vec{x}.(t_1 = t_n)]^M$ to be the subset of M^n consisting of sequences that have the same output under t_1^M and t_2^M.

- Let $t_1, \ldots, t_m$ be terms, and let R be an m-ary relation symbol. Let $\vec{x}$ be a sequence containing the free variables of $R(t_1, \ldots, t_m)$. We define $[\vec{x}.R(t_1, \ldots, t_m)]^M$ to be the

subset of M^n consisting of sequences $\langle a_1, \ldots, a_n \rangle$ such that

$$\langle [\vec{x}.t_1]^M(a_1, \ldots, a_n), \ldots, [\vec{x}.t_m]^M(a_1, \ldots, a_n) \rangle$$

is in R^M.

- Suppose that $\vec{x} = x_1, \ldots, x_n$ is a sequence containing the free variables in $\phi \wedge \psi$. Then $\vec{x}$ contains the free variables in ϕ and ψ, so that $[\vec{x}.\phi]^M$ and $[\vec{x}.\psi]^M$ are subsets of M^n. Then we define:

$$\begin{aligned} [\vec{x}.(\phi \wedge \psi)]^M &= [\vec{x}.\phi]^M \cap [\vec{x}.\psi]^M, \\ [\vec{x}.(\phi \vee \psi)]^M &= [\vec{x}.\phi]^M \cup [\vec{x}.\psi]^M, \\ [\vec{x}.\neg\phi]^M &= M^n - [\vec{x}.\phi]^M, \\ [\vec{x}.(\phi \to \psi)]^M &= (M^n - [\vec{x}.\phi]^M) \cup [\vec{x}.\psi]^M. \end{aligned}$$

- Suppose that $\vec{x}$ is a sequence containing the free variables in $\exists v \phi(\vec{x}, v)$. Then $\vec{x}, v$ contains the free variables in $\phi(\vec{x}, v)$, and $[\vec{x}, v.\phi(\vec{x}, v)]^M$ is a subset of M^{n+1}. We let $[\vec{x}.\exists v \phi(\vec{x}, v)]^M$ consist of those sequences $\langle a_1, \ldots, a_n \rangle$ in M^n such that there is some $b \in M$ with $\langle a_1, \ldots, a_n, b \rangle$ in $[\vec{x}, v.\phi(\vec{x}, v)]^M$. (Another way to think of it: $[\vec{x}.\exists v \phi(\vec{x}, v)]^M$ is the projection of the subset $[\vec{x}, v.\phi(\vec{x}, v)]^M$ of $M^n \times M$ onto M^n.) For the special case where $\vec{x}$ is the empty sequence, we let $1 = \{*\}$ denote an arbitrary singleton set, and we let 0 denote the empty set. Then $[v.\phi(v)]^M$ is a subset of M, and we define $[\exists v \phi(v)]^M = 1$ iff $[v.\phi(v)]^M$ is non-empty.

- Suppose that $\forall v \phi(\vec{x}, v)$ has free variables among $\vec{x}$. As above, we let $[\vec{x}.\forall v \phi(\vec{x}, v)]^M$ consist of those sequences $\langle a_1, \ldots, a_n \rangle$ in M^n such that for all $b \in M$, $\langle a_1, \ldots, a_n, b \rangle$ is in $[\vec{x}, v.\phi(\vec{x}, v)]^M$.

□

29 Note. The previous definition includes the following special case: for any formula ϕ with no free variables (i.e., ϕ is a sentence), $[\phi]^M$ is a subset of M^0, which is a one-point set $1 = \{*\}$. There are two possibilities: $[\phi]^M = 1$ or $[\phi]^M = 0$, where 0 is the empty set. We say that ϕ is *true* in the structure M just in case $[\phi]^M = 1$, and *false* just in case $[\phi]^M = 0$. We will see later that $[\phi]^M = 1$ iff for any sequence $\vec{x} = x_1, \ldots, x_n$ of variables, $[\vec{x}.\phi]^M = M^n$, including the special case $M^0 = 1$. Indeed, for any formula ψ, if the variable v is not free in ψ, then $[\vec{x}, v.\psi]^M = [\vec{x}.\psi]^M \times M$.

30 Example. Let L be the language with a single unary predicate symbol P. Let $M = \{0, 1, 2\}$, and let $P^M = \{0, 1\}$. Then $[x, y.Px]^M$ consists of pairs of elements of M such that

the first element is in P^M, i.e. the set

$$P^M \times M \;=\; \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 0,2\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,2\rangle\}.$$

Consider now $[x, y.(Px \vee Py)]^M$. Here the definition tells us that

$$[x, y.(Px \vee Py)]^M \;=\; [x, y.Px]^M \cup [x, y.Py]^M.$$

Furthermore, $[x, y.Px]^M = P^M \times M$ and $[x, y.Py]^M = M \times P^M$, from which it follows that

$$[x, y.(Px \vee Py)]^M \;=\; (P^M \times M) \cup (M \times P^M).$$

As these examples indicate, $[\vec{x}.\phi(\vec{x})]^M$ can be read as, "those n-tuples from M that satisfy the formula ϕ."

Let's compute the truth-value of the sentence $\exists x Px$ in the structure M. In other words, let's compute the set $[\exists x Px]^M$. Recall that, in general, $[\vec{x}.\exists v\phi(\vec{x}, v)]^M$ results from projecting out the last coordinate of $[\vec{x}, v.\phi(\vec{x}, v)]^M$. How do we apply this rule to the special case of $[\exists v\phi(v)]^M$, i.e. where $\vec{x}$ is the empty sequence? Here we use the fact that each $a \in M$ can be paired with a unique $\langle *, a\rangle \in 1 \times M$, where $1 = \{*\}$ is a one-point set. So, $[v.\phi(v)]^M$ is a subset of $M = 1 \times M$, and $[\exists v\phi(v)]^M$ is the projection of $[v.\phi(v)]^M$ onto 1. In fact, $[\exists v\phi(v)]^M = 1$ precisely when there is some $a \in M$ such that $\langle *, a\rangle \in [v.\phi(v)]^M$, i.e. precisely when $[v.\phi(v)]^M$ is non-empty.

31 Example. Let L be the language with a single binary relation symbol R. Let $M = \{0, 1, 2\}$ and interpret R as "less than." Then $[y.\exists x Rxy]^M$ is defined to be the projection of $[x, y.Rxy]^M$ onto the second coordinate. In other words, $[y.\exists x Rxy]^M$ consists of those $b \in M$ such that there is a $a \in M$ with $\langle a, b\rangle \in R^M$, that is, $a < b$. Similarly, $[y.\forall x Rxy]^M$ consists of those $b \in M$ such that for every $a \in M$, $\langle a, b\rangle \in [x, y.Rxy]^M$.

32 Definition. For a formula ϕ whose free variables occur in $\vec{x}$, we write $M \models_{\vec{x}} \phi(\vec{a})$ just in case $\vec{a}$ is in $[\vec{x}.\phi]^M$. For the special case where ϕ is a sentence, we write $M \models \phi$ when $[\phi]^M = 1$.

In the following proposition, we spell out some useful consequences of our definition of $[\vec{x}.\phi(\vec{x})]^M$ for an L structure M. It is these consequences — rather than the definition itself — that will be most useful for proving things about L structures.

33 Proposition. *If M is an L structure, then:*

1. $M \models \phi(c)$ *iff* $M \models_x \phi(c^M)$.

2. $M \models \phi \wedge \psi$ *iff* $M \models \phi$ *and* $M \models \psi$.

3. $M \models \phi \vee \psi$ *iff* $M \models \phi$ *or* $M \models \psi$.

4. $M \models \neg\phi$ *iff* $M \not\models \phi$.

5. $M \models \phi \to \psi$ *iff* $M \models \neg\phi$ *or* $M \models \psi$.

6. $M \models \exists x\phi(x)$ *iff there is an* $a \in M$ *such that* $M \models_x \phi(a)$.

7. $M \models \forall x\phi(x)$ *iff for all* $a \in M$, $M \models_x \phi(a)$.

Proof. For (1), we have $M \models \phi(c)$ just in case $[\phi(c)]^M = 1$ which means that $c^M \in [x.\phi(x)]^M$, which means that $M \models_x \phi(c^M)$.

(2)–(5) follow from the definition of the set-theoretic operations $\cap, \cup$, and $-$. Recall that for a sentence ϕ, $[\phi]^M$ is a subset of the one-point set 1. So, if we use 0 to denote the empty set, then $[\phi]^M$ is either 0 or 1. Thus,

$$\begin{array}{lll} M \models \phi \wedge \psi & \text{iff} & [\phi \wedge \psi]^M = 1, \\ & \text{iff} & [\phi]^M \cap [\psi]^M = 1, \\ & \text{iff} & [\phi]^M = 1 \text{ and } [\psi]^M = 1, \\ & \text{iff} & M \models \phi \text{ and } M \models \psi. \end{array}$$

For (6), we first show that there is some $a \in [x.\phi(x)]^M$ iff $[\exists x\phi(x)]^M = 1$. Note that for any function $f : A \to B$ and subset S of A, there is some $b \in f(S)$ iff there is some $a \in S$. Thus,

$$\begin{array}{lll} \text{There is an } a \text{ in } [x.\phi(x)]^M & \text{iff} & \text{there is an } \langle a, * \rangle \text{ in } [x.\phi(x)]^M \times 1, \\ & \text{iff} & \text{there is a } b \text{ in } \pi_1([x.\phi(x)]^M \times 1), \\ & \text{iff} & \text{there is a } b \text{ in } [\exists x\phi(x)]^M, \\ & \text{iff} & [\exists x\phi(x)]^M = 1. \end{array}$$

We then have:

$$\begin{array}{lll} M \models \exists x\phi(x) & \text{iff} & [\exists x\phi(x)]^M = 1, \\ & \text{iff} & \text{there is an } a \text{ in } [x.\phi(x)]^M, \\ & \text{iff} & \text{there is an } a \in M \text{ such that } M \models_x \phi(a). \end{array}$$

□

Exercise 8. Complete the remaining steps of the previous proof.

34 Definition. If Γ is a set of sentences and ϕ is a sentence, we write $\Gamma \models \phi$ as shorthand for: for any L structure M, if $M \models \psi$ for all $\psi \in \Gamma$, then $M \models \phi$. If $\Gamma \models \phi$ we say that Γ *semantically entails* ϕ.

Using the definition of $\models$, we can now check that our derivation rules are *sound*. For example:

35 Proposition. *The rule $\to$-intro is sound.*

Proof. The $\to$-intro rule says that if $\Gamma, \phi \Rightarrow \psi$ then $\Gamma \Rightarrow \phi \to \psi$. To show that this rule is sound we must show that if $\Gamma, \phi \models \psi$ then $\Gamma \models \phi \to \psi$. Suppose then that $\Gamma, \phi \models \psi$, and that M is an arbitrary model of Γ. Then either $M \models \phi$ or $M \not\models \phi$. In the former case, $M \models \phi$ and so $M \models \Gamma \cup \{\phi\}$. Since $\Gamma \cup \{\phi\} \models \psi$, it follows that $M \models \psi$ and thus that $M \models \phi \to \psi$. In the latter case, since $M \not\models \phi$, it follows that $M \models \neg\phi$ and hence that $M \models \phi \to \psi$. In either case, $M \models \phi \to \psi$. Since M was an arbitrary model of Γ, it follows that $\Gamma \models \phi \to \psi$. □

36 Proposition. *The rule $\exists$-intro is sound.*

Proof. The $\exists$-intro rule allows us to infer $\Gamma \Rightarrow \exists\phi(x)$ from $\Gamma \Rightarrow \phi(c)$. To say that the rule is sound means that if $\Gamma \models \phi(c)$ then $\Gamma \models \exists x\phi(x)$. So suppose that $\Gamma \models \phi(c)$, and let M be an arbitrary model of Γ (i.e. an L structure in which all sentences in Γ are true). Since $\Gamma \models \phi(c)$, we have $M \models \phi(c)$. By two applications of Proposition 33, we have $M \models_x \phi(c^M)$ and hence $M \models \exists x\phi(x)$. Since M was an arbitrary model of Γ, we conclude that $\Gamma \models \exists x\phi(x)$. □

37 Definition. An *L-theory* is a set of sentences of L. If T is an L-theory, an L-structure M is called a *model* of T just in case $M \models \phi$ for all ϕ in T.

5 Normal forms and other important equivalences

We will prove in the coming weeks that our definition of provability $\Rightarrow$ matches exactly our definition of semantic entailment $\models$. But for now we will simply assume that they coincide. The question of this section is which sentences are (semantically/provably) equivalent? For example, the sentences $\neg\exists xPx$ and $\forall x\neg Px$ are equivalent, as can easily be seen either by using the definition of $\Rightarrow$, or by using the definition of $\models$. Now I will just list a few important equivalences:

Basic Propositional Equivalences

$$\begin{array}{rcl}
\phi \wedge \phi & \Leftrightarrow & \phi \\
\phi \vee \phi & \Leftrightarrow & \phi \\
\phi \to \psi & \Leftrightarrow & \neg\phi \vee \psi \\
\neg(\phi \to \psi) & \Leftrightarrow & \phi \wedge \neg\psi \\
\neg(\phi \wedge \psi) & \Leftrightarrow & \neg\phi \vee \neg\psi \\
\neg(\phi \vee \psi) & \Leftrightarrow & \neg\phi \wedge \neg\psi \\
\phi \wedge (\psi \vee \chi) & \Leftrightarrow & (\phi \wedge \psi) \vee (\phi \wedge \chi) \\
\phi \vee (\psi \wedge \chi) & \Leftrightarrow & (\phi \vee \psi) \wedge (\phi \vee \chi)
\end{array}$$

In addition to these, it can also be shown that $\wedge$ and $\vee$ are commutative and associative. In contrast, $\to$ is neither commutative nor associative. For example, $\phi \to \psi$ does not imply $\psi \to \phi$; and $\phi \to (\psi \to \chi)$ does not imply $(\phi \to \psi) \to \chi$.

Quantifier Negation Equivalences

$$\begin{array}{rcl}
\neg\forall x\phi(x) & \Leftrightarrow & \exists x\neg\phi(x) \\
\neg\exists x\phi(x) & \Leftrightarrow & \forall x\neg\phi(x)
\end{array}$$

Swoosh Equivalences

$$\begin{array}{rcl}
\forall x(\phi(x) \wedge \psi(x)) & \Leftrightarrow & \forall x\phi(x) \wedge \forall x\psi(x) \\
\exists x(\phi(x) \vee \psi(x)) & \Leftrightarrow & \exists x\phi(x) \vee \exists x\psi(x)
\end{array}$$

For the following equivalences, we assume that ψ is a formula in which the variable x is not free.

$$\begin{array}{rcl}
\forall x(\phi(x) \vee \psi) & \Leftrightarrow & \forall x\phi(x) \vee \psi \\
\exists x(\phi(x) \wedge \psi) & \Leftrightarrow & \exists x\phi(x) \wedge \psi \\
\forall x(\psi \to \phi(x)) & \Leftrightarrow & \psi \to \forall x\phi(x) \\
\forall x(\phi(x) \to \psi) & \Leftrightarrow & \exists x\phi(x) \to \psi
\end{array}$$

Now we turn to some "normal forms." For starters, let's consider sentences from *propositional* logic; i.e. in a language L that has only zero place predicate symbols.

38 Definition. A sentence ϕ is said to be a *literal* just in case it is either a zero place predicate symbol, or the negation of a zero place predicate symbol. A sentence ϕ is said to be a *simple conjunction* just in case it is a conjunction of literals. Similarly, a sentence ϕ is said to be a *simple disjunction* just in case it's a disjunction of literals. A sentence ϕ is said

to be in *disjunctive normal form* if it is a disjuntion of simple conjunctions. A sentence ϕ is said to be in *conjunctive normal form* if it is a conjunction of simple disjunctions.

39 Fact. *For every sentence ϕ in propositional logic, there is a sentence ϕ^c in conjunctive normal form and a sentence ϕ^d in disjunctive normal form such that $\phi \Leftrightarrow \phi^c$ and $\phi \Leftrightarrow \phi^d$.*

For now, you can take my word that this fact is true; we'll prove it before too long.

6 Decision procedures

40 Definition. Let ϕ be a sentence in the language L. We say that ϕ is *satisfiable* just in case there is an L-structure M such that $M \models \phi$.

Consider the following question:

> (SAT) *Is ϕ satisfiable? In other words, is there an L structure M such that $M \models \phi$?*

In general, SAT may be very difficult to answer, since it would involve a search through the infinite space of L structures. And yet, in some cases, there is a straightforward and finitary procedure that yields a definitive answer to SAT. I will give two examples of sentences for which there is a "decision procedure" for SAT.

41 Example. Propositional logic is that fragment of first-order logic that drops the use of the equality sign, of function signs, and of relation symbols of arity greater than zero. In other words, a propositional language only permits 0-place relation symbols. We need to be clear about what an L-structure looks like for a propositional language. Let P be a zero-place relation symbol. So what should a L-structure assign to p? For a $n > 0$ place relation symbol R, R^M is required to be some subset of the set M^n of ordered n-tuples of elements of n. So by analogy, we should require that p^M be a subset of the set M^0 of 0-tuples of elements of M. But what *is* the set of 0-tuples? In fact, M^0 is a set with one element, call it 1. (There is a good reason for saying that $M^0 = 1$. Indeed, M^0 should be a set with the property that for any other set S, there is a unique function from S into M^0. It's easy to see that such a set must contain precisely one element.) There are precisely two subsets of 1, the empty set 0, and 1 itself. Thus, the structure M assigns p either to 1 or to 0. We can think of 1 as "true" and 0 as "false", so that an L-structure assigns every propositional letter p a truth-value.

Now the truth value of a complex sentence is just a function of the truth-value of its component sentences. Indeed, each propositional logic sentence ϕ corresponds to a unique

function f_ϕ that takes n Boolean inputs (where n is the number of distinct propositional symbols in ϕ) and has one Boolean output. So the question (SAT) of whether ϕ is satisfiable amounts to the question of whether f_ϕ ever takes the value 1. And that question can be answered by enumerating the 2^n possible input values and computing f_ϕ in each case. That procedure might take a long time if n is large, but it will always eventually yield the answer to (SAT).

42 Example. Now for a less trivial example. Consider a language L with only 0-place and 1-place relation symbols — no function symbols or constants, and no equality symbol. I will sketch a procedure for answering (SAT) for a sentence ϕ of L — and, if ϕ is satisfiable, this procedure will return a structure M in which ϕ is true. This procedure consists of three nested algorithms.

Algorithm A: The first algorithm works for a sentence ϕ that is built out of monadic predicate symbols, but with no quantifiers (and no equality symbol, no relation symbols, no function symbols).

1. Use the truth-table method to determine if there is an assignment of 0 and 1 to the atomic sentences in ϕ that would make ϕ come out as true. If there is no such truth assignment, then return "unsatisfiable." If there is a truth assignment j, then proceed to Step 2.

2. Put the constant symbols that occur in ϕ in a list $a_1, \ldots, a_n$. Let the domain of M be $\{1, \ldots, n\}$. For each predicate letter P, and for each $i = 1, \ldots, n$, let $i \in P^M$ just in case $j(Pa_i) = 1$. Return "satisfiable" and the L-structure M.

Algorithm B: The second algorithm works for simple quantified sentences. A *simple monadic sentence* is one that contains no relation symbols, and a single quantifier whose scope is the entire sentence. For example, $\forall x((Fx \wedge Gx) \rightarrow \neg Hx)$ is a simple monadic sentence. If $\phi(x)$ is a simple monadic sentence, we let $\phi(a)$ denote the instance of $\phi(x)$ that results from taking off the initial quantifier and replacing all instances of the variable x with the constant symbol a. For example, if $\phi(x)$ is the sentence $\exists x(Fx \wedge Gx)$ then $\phi(a)$ is the sentence $Fa \wedge Ga$.

We now show how to determine if a collection $\phi_1, \ldots, \phi_n$ of simple monadic sentences is consistent.

1. Reorder the sentences if necessary so that the first m sentences begin with an existential quantifier, and the remaining sentences begin with a universal quantifier. Proceed to Step 2.

2. If none of the sentences begins with an existential quantifier, then choose one arbitrary constant symbol a. Now put the sentences $\phi_1(a), \ldots, \phi_n(a)$ into Algorithm A. Otherwise proceed to Step 3.

3. Choose m constant symbols $a_1, \ldots, a_m$ (one per existential sentence). Now put the following sentences into Algorithm A:

$$\begin{array}{ll} \phi_1(a_1), \ldots, \phi_m(a_m) & \text{(instances of existential sentences)} \\ \\ \phi_{m+1}(a_1), \ldots, \phi_{m+1}(a_m) & \\ \vdots & \text{(instances of universal sentences)} \\ \phi_n(a_1), \ldots, \phi_n(a_m) & \end{array}$$

That last step could use some explanation: for each distinct existential sentence ϕ_i, we choose a distinct name a_i and construct the corresponding instance $\phi_i(a_i)$. Then for each universal sentence ϕ_j, we construct m instances $\phi_j(a_1), \ldots, \phi_j(a_m)$, one for each name that was introduced for an existential sentence.

Algorithm C: The third algorithm is a simple extension of Algorithm B; it works for truth functional combinations of simple monadic sentences. A *pure monadic sentence* is a sentence that is a truth-functional combination of simple monadic sentences. For example,

$$\exists x Fx \forall y (Gy \rightarrow Fy),$$

is a pure monadic sentence. On the other hand,

$$\forall x (Fx \rightarrow \exists y Fy)$$

is *not* simple monadic, since it has nested quantifiers.

1. Transform $\phi_1 \wedge \cdots \wedge \phi_n$ into disjunctive normal form. The result will be of the form $\psi_1 \vee \cdots \vee \psi_m$, where each ψ_i is a conjunction of simple monadic sentences and negated simple monadic sentences.

2. Within each disjunct $\psi_1, \ldots, \psi_n$, change each negated simple monadic sentence into an equivalent simple monadic sentence using the quantifier-negation equivalences.

3. For $i = 1, \ldots, n$ put the modified conjuncts in ψ_i into Algorithm B. If Algorithm B returns "unsatisfiable" for all disjuncts, then return "unsatisfiable." If Algorithm B returns "satisfiable" and a structure M for any disjunct, then return "satisfiable" and the structure M.