A Better Statement (hopefully) of My Answer to John Gunde's Question

John G. Asked last time why Carnap's article was titled "The Logical Structure of Science" if all it discussed was syntax and semantics for the three languages. In what follows below, I hope to clarify the answer I was giving in class. Please keep in mind in looking at this that I am (a) dyslexic, (b) about 13 years from having last written-out any of the predicate logic material, and (c) trying to write it out without consulting sources to screw with myself, er, reinvigorate my mind and understanding. That said, I am confident that there will be typos as some mistakes (hopefully few and small). It may not be very clear in points, I didn't really have time to rework it if I was going to send it to you before Monday. But part of the Zeitgeist of this course is development and refinement of ideas. So, we'll figure out better ways to explain things, correct typos, etc. as you need/notice them. I'm hoping Bill will delicately correct my missteps when he notices them. I will also try to show this to Dr. Dieveney and/or Dr. C. Wright and when they stop laughing at my incompetence, they will likely improve the document vastly. I am attaching it as a PDF and I will post it as a link off the syllabus for the course as well.

Best,
ChW

Some general notes about formal languages and particularly first-order predicate logics, and particularly as understood by logicians during the time of the Logical Empiricists. First, one characterizes a formal language as having three components; a **syntax** consisting of primitives and recursive combinatorial rules for combining primitives into well-formed atomic formulas, more complex well-formed formulas, and sentences, a **transformational axiom system** consisting of a set of **syntactically specified transformation rules** (In formal logic, these are inference rules and the axiom system is called a **deductive system**. Throughout the rest of the discussion I'll just refer to the transformational axiom system as the deductive system. **However, we should note that part of the logic of science will include confirmation axioms in addition to deductive axioms.**), and a **semantics** providing a set of recursive combinatorial rules for model-theoretic mappings of the elements of the language to domains. A formal language is specified in a **metalanguage**, that is a language that we use to talk about our formal language. Often times people discuss the two languages using *metalanguage* to refer to the language we use to discuss the formal language and *object language* referring the formal language. So, for people who like this symbolic stuff, creating a formal language requires specifying and ordered-triple, <S,D,I> (i.e., syntax, deductive system, systematic interpretation or model-theoretic mapping function) in a metalanguage.

However, this tripartite characterization is somewhat misleading since the deductive system is a set of axioms the function of which is spelled out in the combination of the syntax, where the axioms are introduced as syntactic strings satisfying allowable syntactic manipulations satisfying the definition of and argument and a derivation, and the semantics, in which the formalized language and deductive axioms are tied to domains and demonstrated to preserve target properties (e.x. truth). It is this dual life of the language and most especially the deductive system in syntax and semantics that allows it to operate as a model-theoretic transformation engine in accordance with some satisfaction function. In the case of deductive logics, it is the dual syntactic definitions of argument/derivation, and the semantic definitions of argument/valid consequence that allows us to use the formal language to make truth-preserving inferences by mastering *content-independent* rules and symbols together with translations (from our ordinary language, like English, into the formal language and vice versa). I realize this probably seems somewhat abstract, so I'll run through a version of the basic syntax, deductive system, and semantics for a simple first-order predicate logic.

Syntax of *L*:

The expressions of *L* are finite strings of symbols from the following set of classes, combined in accordance with the recursive, combinatorial rules for well-formed formulas.

Variables:

The variables of *L* are lowercase italic letters '*u*' through '*z*' with or without subscripts of positive integers (i.e. the set {$u, v, x, y, z, u_1, v_1, \ldots, y_n, z_n$}, where n = any positive integer such that $n < \infty$).

Constants:

The constants of *L* fall into two classes *logical constants* and *non-*

logical constants:

Logical Constants are the following eight symbols:

'¬', 'v', '(', ')', '•', '→', '≡', and '∃'

Non-Logical Constants: The non-logical constants of *L* fall into two classes, *Predicates*, *Individual Constants.*

Individual Constants: The individual constants for *L* consist of lowercase italic letters of the alphabet from '*a*' to '*t*', with or without subscripts of positive integers (i.e. the set {$a, b, c, \ldots, a_1, b_1, c_1, \ldots, a_n, b_n, c_n, \ldots$} where n = any positive integer such that $n < \infty$).

Predicates: The predicates of *L* consist of individual italic capital letters with superscripts and with or without subscripts. The superscripts of predicates occur immediately before the predicate, and consist of positive natural numbers and indicate degree (i.e., the number of variables and constants of the predicate. Thus, a standard binary predicate, P^2, has two constant or variables. In alternative terminology, it has a degree of 2.) The subscripts of the language *L* occur immediately after the predicate, and consist of positive natural numbers. Thus, the stock of predicates for the language *L* is given in the set {${}^1A, {}^1B, {}^1C, \ldots, {}^1A_1, {}^1B_1, {}^1C_1, \ldots, {}^2A, {}^2B, {}^2C, \ldots {}^2A_2, {}^2B_2, {}^2C_2, \ldots {}^nA_m, {}^nB_m, {}^nC_m, \ldots$} such that n = any positive integer such that $n < \infty$ and m = any positive integer such that $m < \infty$.

The Rules for well-formed formulas in *L:*

We shall define five elements of the language, *L*, *Predicate of degree n*, *Sentential letter, Individual symbol*, and *atomic formula*, and *formula.* We introduce at this time into the metalanguage the symbols, φ, ψ, and α. φ and ψ are elements of the metalanguage that we can use to refer to classes or kinds of elements of the object language.

Predicate of degree *n*: A predicate of degree *n* (or an *n-ary* predicate) is a predicate have a superscript of positive integer *n* with or without a subscript consisting of a positive integer, *m*.

Sentential Letter: A sentential letter is a predicate without a superscript, *n*, with or without a subscript consisting of a positive integer, *m*.

Individual Symbol: An individual symbol of the language *L* consists of either a variable or an individual constant.

Atomic Formula: An atomic formula for the language *L* is a string of elements of the language *L* counting as an expression in *L*. It can consists of either a sentential letter alone or a predicate of degree *n* followed by a string (having *n* elements) of individual symbols.

Formula: A formula (sometimes called and open formula because variables may not be bound, see below) of *L* is an expression of the language *L* is either an atomic formula or else is a string of atomic formulas constructed by a finite number of the applications of the following rules:

Negation: For any string of elements, φ, in *L*, if φ is a formula, then ¬φ is a formula.

Connectives: For any to strings of elements, φ and ψ, in *L*, if φ and ψ are formulas, then (φ v ψ), (φ • ψ), (φ → ψ), and (φ ≡ ψ) are formulas.

Variables: For any string of elements, φ, in *L*, if φ is a formula and α is a variable, then (α)φ is a formula and (∃α)φ is a formula.

Boundedness of variables: An occurrence of a variable α in a formula φ in language *L* is bound if and only if it is within an instance of a formula of the form (α)ψ or (∃α)ψ.

Sentences: A sentence in language *L* is a formula in which no variable occurs free (i.e. all variables must be bound.).

Deductive System:

Argument: An argument in language *L* is an order-pair of sets, $<\Gamma,\varphi>$, where Γ is designated as the set of premises and φ is designated as the conclusion. We formulate the inference rules of *L* as axioms relating permissible transformations for well-formed formulas in *L* such that via additions and subtractions to Γ one may enter a well-formed formula as φ. We call the results of the permissible transformations given by the axioms below *derivations*, and symbolize the relation of derivation as follows: ⊢

Premise Introduction: (or assumption) If φ is a member of Γ, then $\Gamma \vdash \varphi$ (In other words, if one adds φ to Γ, then one can assert the derivation $\Gamma \vdash \varphi$.
Wedge Introduction: If $\Gamma^1 \vdash \varphi$, then $\Gamma^1 \vdash (\varphi \vee \psi)$.
Wedge Extraction: If $\Gamma^1 \vdash (\varphi \vee \psi)$ and $\Gamma^2 \vdash \neg \varphi$, then $\Gamma^1 \cup \Gamma^2 \vdash \psi$.
Dot Introduction: If $\Gamma^1 \vdash \varphi$ and $\Gamma^2 \vdash \psi$, then $\Gamma^1 \cup \Gamma^2 \vdash (\varphi \bullet \psi)$.
Dot Extraction: If $\Gamma^1 \vdash (\varphi \bullet \psi)$, then $\Gamma^1 \vdash \varphi$.
Arrow Introduction: If $\Gamma^1 \cup \varphi \vdash \psi$, then $\Gamma^1 \vdash (\varphi \rightarrow \psi)$.
Arrow Extraction: If $\Gamma^1 \vdash (\varphi \rightarrow \psi)$ and $\Gamma^2 \vdash \varphi$, then $\Gamma^1 \cup \Gamma^2 \vdash \psi$.
Triple Bar Introduction: If $\Gamma^1 \vdash (\varphi \rightarrow \psi)$ and $\Gamma^2 \vdash (\psi \rightarrow \varphi)$, then then $\Gamma^1 \cup \Gamma^2 \vdash (\varphi \equiv \psi)$.
Triple Bar Extraction: If $\Gamma^1 \vdash (\varphi \equiv \psi)$, then $\Gamma^1 \vdash (\varphi \rightarrow \psi)$ or $\Gamma^1 \vdash (\psi \rightarrow \varphi)$.

The Semantics of *L*:

I will first give you a truth-functional semantics for L, then I will give you a general satisfaction schema copied from MathWorld. I'm sorry about the last, but this took a lot longer than I thought and I'm falling behind in my work for this class. ;-).

We shall define truth for sentences in *L* under a given interpretation*, I.* Note that the definition of a sentence in *L* (above) includes sentential letters, formulas with only constants, and formulas in which all the variables in the formula are bound. We define truth for sentences in our language *L* relative any interpretation*, I,* using the recursive rules as follows:

For any interpretation, *I*, of language *L,* any sentences, φ, ψ, and μ in *L,* any variable α, and β, where β is the first individual constant not occurring in φ when the set of constants given above, {$a, b, c,\ldots,a_1, b_1, c_1,\ldots,a_n, b_n, c_n,\ldots$} where *n* = any positive integer such that $n < \infty$ is, in fact, an ordered set <$a, b, c,\ldots,a_1, b_1, c_1,\ldots,a_n, b_n, c_n,\ldots$>, where *n* = any positive integer such that $n < \infty$ and subscripts progress in ordinal value [i.e., 1, 2, 3, 4, 5,....]. This weird and seemingly pointless specification of β is actually necessary technically to pick out the first constant in the language that is not used in a given quantified statement as we'll see in (8) and (9).

(1) If φ is a sentential letter, then φ is true under interpretation *I* (Henceforth I will use T_{LI} to express "is a sentential letter, formula, or sentence in language *L* that is true under interpretation, *I*) if and only if *I* assigns T to φ.
(2) If φ is an atomic formula and is not a sentential letter, then φ is T_{LI} if and only if the domain objects assigned to the constants in φ by *I* are related in the domain in accordance with the order in which the corresponding constants occur in φ. [This clause seems abstract and complex, but it might make sense to consider the alternative way that one often constructs interpretations for specific languages and domains: ^{1}Pa is true under *I* if and only iff ^{1}Pa ∈ of the set assigned as follows: ^{1}Px : x is a horse. Or, alternatively ^{1}Px : {^{1}Pa, ^{1}Pd, ^{1}Ph, ^{1}Pj}. Similarly for a binary predicate ^{2}Rxy as follows: ^{2}Rxy : x is taller than y. Or, alternatively, ^{2}Rxy: {^{2}Rab, ^{2}Rbd, ^{2}Rdl, ^{2}Rad, ^{2}Ral, ^{2}Rbl}. In short the interpretation maps constants to objects that preserves the structured ordering of the predicate.]
(3) If φ = ¬ψ, then φ is T_{LI} if and only if ψ is not T_{LI}.
(4) If φ = (μ v ψ), then φ is T_{LI} if and only if either μ is T_{LI} or ψ is T_{LI} or both ψ

and μ are T_{LI}.
(5) If φ = (μ • ψ), then φ is T_{LI} if and only if μ is T_{LI} and ψ is T_{LI}.
(6) If φ = (μ → ψ), then φ is T_{LI} if and only if either μ is not T_{LI} or both ψ and μ are T_{LI}.
(7) If φ = (μ ≡ ψ), then φ is T_{LI} if and only if both ψ and μ are T_{LI} or both ψ and μ are not T_{LI} with mutual exclusivity.
Before continuing we must define the term *β-variant*. Let *I* and *I'* be two different interpretations of language *L.* Let β be an individual constant in *L.*
***β*-variant:** *I* is a β-variant of *I'* if and only if *I* = *I'* or *I* and *I'* differ only in their assignments of elements of their domain to the individual constant β. Note this restricts β-variants to interpretations mapping elements of language *L* to the same domains.
(8) If φ = (α)ψ, then φ is T_{LI} if and only if ψ α/β (α/β is just short-hand for a substitution instance of a quantified statement where the variable, α, is replaced in the sentence with the constant β; for instance Fb is a substitution instance of (x)Fx.) under every β-variant of *I.*
(9) If φ = (∃α)ψ, then φ is T_{LI} if and only if ψ α/β (again, α/β is just short-hand for a substitution instance of a quantified statement where the variable, α, is replaced in the sentence with the constant β; for instance Fb is a substitution instance of (x)Fx.) under at least one β-variant of *I.*

These last two clauses for quantification are often conceptually obtuse to people. (8) simply says that a given universal statement is true if and only if it is true under all substitution instances for it's bound variable for the first constant not in that sentence under all interpretations mapping *L* to a given domain. Of course, for multiple variables one must use the definition recursively. (9) simply says that an existentially quantified statement is true if and only if it is true for at least one substitution instance for it's bound variable for the first constant not occurring in the sentence given the interpretations mapping *L* to a given domain, i.e. there is something that is an instance of the variable.

Finally, in the case of first-order predicate logic the standard semantic properties one establishes for the deductive system are *soundness* and *completeness*. Basically, soundness says that all derivations consistent with the axioms of the deductive system preserve truth. More formally: For any given well-formed formula, Φ, and set of well-formed formulas, Γ, and any deductive system, *D*, if $\Gamma \vdash_D \Phi$ (i.e., you can derive Φ from Γ using the axioms in *D*), then $\Gamma \vDash \Phi$ (i.e., the well-formed formula Φ is a valid consequence of the set of well-formed formulas Γ. That is, if the set of well-formed formulas Γ is true under any interpretation, then Φ is true under that interpretation. Sometimes this is called semantic consequence, logical consequence, model-theoretic consequence, or satisfaction preserving consequence).

Completeness says that all valid arguments are derivable with the deductive system, *D*. More formally: For any given set of well-formed formulas, Γ, if Γ is consistent (i.e., for the set of well-formed formulas Γ it is not the case that there is a well-formed formula Φ such that $\Phi \in \Gamma$ and $\Gamma \vdash_D \Phi$ and $\Gamma \vdash_D \neg\Phi$), then Γ is satisfiable (i.e., for now, it is true).

By establishing soundness and completeness you map the syntactic notions of argument/derivation to the semantic notions of argument/valid consequence. Thus, you establish that your syntactic engine is a truth-preserving semantic engine.

Satisfaction Sidebar

Let ***A*** be a relational system, and let *L* be a language which is appropriate for ***A***. Let φ be a well-formed formula of *L*, and let *s* be a valuation in ***A***. Then $\boldsymbol{A}\vDash_s\varphi$ is written provided that one of the following holds:

1.φ is of the form x = y, for some variables x and y of *L*, and *s* maps x and y to the same element of the structure ***A***.

2. φ is of the form $\boldsymbol{R}x_1,...x_n$, for some n-ary predicate symbol ***R*** of the language *L*, and some variables $x_1,...x_n$ of *L*, and $\{s(x_1),...,s(x_n)\}$ is a member of $R^{\boldsymbol{A}}$.

3.φ is of the form (ψ ∧ γ), for some formulas ψ and γ of *L* such that $\boldsymbol{A}\vDash_s \psi$ and $A\vDash_s \gamma$.

4. φ is of the form ((∃ x)ψ), and there is an element α of ***A*** such that $\boldsymbol{A}\vDash_{s(x|\alpha)}$ ψ.

In this case, ***A*** is said to satisfy φ with the valuation *s*.

Insall, Matt. "Satisfaction." From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein.
http://mathworld.wolfram.com/Satisfaction.html

A relational system is a structure $\boldsymbol{R} = (\boldsymbol{S},\{P_i : i \in I\},\{f_j : j \in J\})$ consisting of a set $\boldsymbol{S}$, a collection of relations $P_i (i \in I)$ on $\boldsymbol{S}$, and a collection of functions $f_j (j \in J)$ on $\boldsymbol{S}$.

Bengtsson, Viktor. "Relational System." From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein.
http://mathworld.wolfram.com/RelationalSystem.html