5.3. Names and Predicates: Formal Semantics

1. Models. The semantics for sentence logic sharpened the intuitive idea of **possible situation** into the technical concept of **valuation**. With the formal language expanded to include names and predicates we expand our semantics as well, by extending valuations into **models**.

For those parts of the formal language inherited from previous chapters, the semantics remains unchanged in models: a model assigns one (and only one) **truth value** to each sentence letter; and molecular sentences follow the familiar semantic rules.

This much semantics addresses A1, along with 2 through 6 of the construction rules.

Revised Construction Rules (*First Draft*)

Atomic Sentences:

A1. Sentence letters are atomic sentence
A2. A predicate letter followed by a name letter is an atomic sentence.

Formal Sentences:

1. Atomic sentences are formal sentences.
2. If ● is a formal sentence, then ~● is a formal sentence.
3. If ● and ▲ are formal sentences, then (● ∧ ▲) is a formal sentence.
4. If ● and ▲ are formal sentences, then (● ∨ ▲) is a formal sentence.
5. If ● and ▲ are formal sentences, then (● → ▲) is a formal sentence.
6. If ● and ▲ are formal sentences, then (● ↔ ▲) is a formal sentence.

Here we provide semantics for the one new type of sentence, introduced by A2: what we'll call a **predicate atom**.

2. Names and Reference. Note that neither an English predicate like "is female" nor a name like "Letitia" is a natural candidate for truth or falsehood. Yet in combination they form something which can be true or false: a sentence such as "Letitia is female". And the same is true of predicate and name letters. So our formal semantics needs to give each predicate letter and name letter a non-1/0 value – yet in a way allowing those values, when combined, to yield a value of 1 or 0 for the whole sentence.

A proper name such as "Letitia" serves to **refer** to some individual – and unlike a short-term, reusable pointer like "it," a proper name always refers to the **same** individual. A name letter, as the formal counterpart to a proper name, likewise refers invariably to an individual.

In order to represent such reference in models, the expanded formal semantics will include a set of **objects** populating a **domain of discourse** – or "domain" for short. A little three-member domain would look like this.

$\mathbb{D}$: {**Neko**, **Letitia**, **Lucretia**}

(Don't be misled by our need to depict things on the printed page with words: the members of this domain aren't three names, but three people.)

And when we don't already have specific individuals (such as Letitia or Lucretia) in mind, but just need some generic objects to populate a model, a quick way to meet that need is to use numerals.

$\mathbb{D}$: {**2, 3, 4**}

(We start with "2" to avoid confusion – because the numerals "0" and "1" are already used in the semantics to represent True and False.)

For the semantics to produce the desired results concerning validity, we insist that **the domain cannot be empty**. Every model must have a domain with at least one object.[1]

[1] On why we don't allow an empty domain, see 5.5.1 Problem E.

Already in the early days of modern logic the logician Augustus De Morgan noted that the domain of discourse – the group of objects under discussion – isn't usually intended to include every object in the universe.

> Thus when we say "All animals require air", or that the name *requiring air* belongs to everything to which the name *animal* belongs, we should understand that we are speaking of things of this earth: the planets, etc., of which we know nothing, not being included.
>
> (De Morgan 1847: 55; cited in Lambert and van Fraassen 1972: 83)

And the domain of discourse can vary as discourse varies. So the sentence "**Everyone** showed up for the exam" will be judged true if all the students in the class (but not necessarily everyone on the planet) showed up for the exam; while "**Everyone** needs oxygen to live" will be judged true if every person on the planet (but not necessarily everyone in the universe) needs oxygen to live. The domain of discourse is usually taken for granted in conversation, and for that reason usually goes unstated. So a certain amount of reflection and reconstruction may be needed to state the domain of discourse for a specific discussion.

With a domain of discourse in hand, the semantics can specify a **referent** – an object referred to – for each name letter, drawing these from the domain of the model. While the semantics of Chapters Two and Three was governed by a single fundamental principle – the **Principle of Bivalence** – the expanded semantics imposes an additional principle of equal importance: the **Principle of Reference**.[2]

> **<u>Principle of Reference</u>: each name letter refers to one and only one object in the domain.**

Just as a valuation assigns exactly one truth value to a sentence letter, a model assigns exactly one referent to each name letter used – as in the following example.

[2] This is sometimes called the Principle of Denotation.

$\mathbb{D}$: {**Neko**, **Letitia**, **Lucretia**, **Jack**}

P: 1	**a**: **Neko**	**d**: **Jack**
Q: 0	**b**: **Letitia**	**e**: **Jack**
R: 0	**c**: **Lucretia**	

Note that the Principle of Reference allows an object to have more than one name within the same model: here both "d" and "e" refer to the same person, Jack. So if, for example, Jack's friends call him "Boots," the names "Jack" and "Boots" refer to the same individual. (Here again the Principle of Reference parallels the Principle of Bivalence: each sentence letter has exactly one truth value, but different sentence letters can have the same truth value.)

For purposes of convenience we require further that **every object in the domain have at least one name**. There's no deep metaphysical point to this stipulation; the requirement is instead just a time-saving measure useful later, in the semantics of quantifiers. (While we could develop quantifier semantics without this assumption, it would only end up being a more complicated way of achieving the same results).

3. Predicates and Extensions. When we turn to predicate letters we better appreciate the merits of insisting that every object be named. For while predicates likewise aren't true or false, they are, in logical parlance, said to be "**true of**" something – that is, to make a true claim about that object. Of course the predicate alone doesn't make a claim; but the predicate can yield a claim about an object by having the object's name fill the blank in that predicate. So in a domain consisting of just Neko, Lucretia, and Letitia, the predicate "is a student" is **true of** two individuals in the domain, and not true of one of them – because the sentences "Lucretia is a student" and "Letitia is a student" are **true**, while "Neko is a student" is false.

The things in the domain a predicate 'holds true of' form the **extension** of that predicate. In the last example, the extension of "is a student" was Letitia and Lucretia.

In formal models the extension of a predicate letter is likewise the family of objects in the model's domain which that predicate letter 'holds true of'. The semantics thus sets out an extension for every predicate letter listed in the

model – each such extension populated by objects drawn from the domain of that model.[3] We extend our earlier example to include extensions for predicate letters "G" through "J".

$\mathbb{D}$: {**Neko**, **Letitia**, **Lucretia**, **Jack**}

P: 1	**a**: **Neko**	**d**: **Jack**
Q: 0	**b**: **Letitia**	**e**: **Jack**
R: 0	**c**: **Lucretia**	

G: {**Letitia**, **Lucretia**} **H**: {**Neko**, **Letitia**, **Lucretia**}
I: {**Neko**, **Jack**} **J**: {**Neko**, **Letitia**, **Lucretia**, **Jack**}
K: { }

For instance, "G" might stand for "is a student"; "H" for "is female"; "I" for "is a cat; "J" for "is a mammal"; and "K" for "is a unicorn". Then our model works out sensibly enough: in this little four-member domain Letitia and Lucretia are students; Neko and Jack are cats; Neko, Letitia, and Lucretia are female; everyone here is a mammal; and nobody is a unicorn.

Note that while every proper name must refer to an object, a predicate letter isn't required to have any objects in its extension. In this model the extension of "J" is empty. Exactly right: reading "J" as "is a unicorn," in a situation involving Neko, Letitia, and Lucretia, and Jack, that predicate should indeed fail to apply to anything.

But we do require that **each predicate letter** have **only one extension** in a given model, so that name and predicate semantics in combination yield truth or falsehood without violating the Principle of Bivalence.

Securing truth or falsehood for a predicate atom (a predicate-letter-plus-name-letter) is then straightforward: the sentence is true exactly when the object referred to by the name letter is in the extension of the predicate letter.

In this model, name letter "a" refers to Neko, who is indeed in the extension of "I" ("is a cat"). So the sentence "Ia" ("Neko is a cat") is **true** in this model.

[3] The extension of each predicate letter will thus be some (proper or improper) subset of the domain.

But Neko's not in the extension of "G" ("is a student") or "K" ("is a unicorn"), so "Ga" ("Neko is a student") and "Ka" ("Neko is a unicorn") are **false**.

Once the model assigns truth values to such atomic sentences, the semantic rules for negation, conjunction, disjunction, conditional, and biconditional assign values to **molecular sentences** built out of these atoms – as the following examples illustrate.

$\mathbb{D}$: {**Neko**, **Letitia**, **Lucretia**, **Jack**}

P: 1	**a**: **Neko**	**d**: **Jack**
Q: 0	**b**: **Letitia**	**e**: **Jack**
R: 0	**c**: **Lucretia**	

G: {**Letitia**, **Lucretia**} **H**: {**Neko**, **Letitia**, **Lucretia**}
I: {**Neko**, **Jack**} **J**: {**Neko**, **Letitia**, **Lucretia**, **Jack**}
K: { }

Ga: 0	**Ha**: 1	**Ia**: 1	**Ja**: 1	**Ka**: 0
Gb: 1	**Hb**: 1	**Ib**: 0	**Jb**: 1	**Kb**: 0
Gc: 1	**Hc**: 1	**Ic**: 0	**Jc**: 1	**Kc**: 0
Gd: 0	**Hd**: 0	**Id**: 1	**Jd**: 1	**Kd**: 0
Ge: 0	**He**: 0	**Ie**: 1	**Je**: 1	**Ke**: 0

~Ga: 1	**(Hb ∧ P)**: 1
(Ha ∧ Ia): 1	**(Ge ∨ Q)**: 0
(Hc ∧ Ic): 0	**(Hb ↔ Hc)**: 1
(Gb ∧ Gc): 1	**(Ga ↔ Jb)**: 0
(Ka ∨ Ha): 1	**(Hc ↔ Gc)**: 1
(Kd → Jd): 1	**(Je ↔ He)**: 0
(Jd → Kd): 0	**(Ke ↔ He)**: 1