Chapter 1

Predicate Logic: Syntax

In the last few chapters we developed rules for the syntax and semantics of the language *SL*. We learned how to use two powerful methods, truth-tables and truth-trees, for testing sentences and sets of sentences to analyze the logical properties that we defined for the language. Finally, we developed a system that enables us to derive new sentences of *SL* from given sentences of *SL*.

An important feature of *SL* is that it is *decidable*. Truth-tables and truth-trees are mechanical methods for verifying truth-functional properties like validity, consistency, etc., in the sense that each step of the method is determined by a rule and the previous steps. To say that *SL* is decidable is to say that, for questions about truth-functional concepts, our methods will always give us a definite yes or no answer in a finite number of steps. So, we can, simply by following a set of rules, determine the validity of *any* argument in sentential logic. There will never be an instance for which the validity of an argument in *SL* cannot be determined.

Sentential logic is a powerful tool, but there are times when it fails. Consider this argument:

1. All dogs are mammals.
2. All cats are mammals.
3. Either Lola is a dog or Lola is a cat.

∴ Lola is a mammal.

Symbolizing this argument in *SL* results in something like this:

1. D
2. C
3. $\underline{E \vee F}$

$\therefore$ G

The natural language argument above is obviously valid, but the corresponding argument in *SL* is not. The problem is that *SL* cannot capture the logical relations between “All dogs are mammals”, “Lola is a dog”, and “Lola is a mammal.” Those relations are determined by the internal structures of the sentences, and the internal structures of sentences are invisible to *SL*, because the smallest logical unit in *SL* is an entire sentence.

In this chapter, we will begin to develop a new language, *PL* (for predicate logic), that will allow us to express some of the internal structures of sentences. The argument above has a valid symbolization in *PL*. Unfortunately, this power comes with a cost. *PL* is not a decidable system. There is no method that we can use that is guaranteed to always tell us if an argument in *PL* is valid or a set of sentences in *PL* is consistent.

1.1 Singular Terms and Predicates

Predicate logic identifies three important components of sentences in natural languages: individual constants, predicates, and quantity terms. Individual constants are a type of singular term. A singular term is one that refers to (or designates or denotes), or purports to refer to, some particular object, called the referent of the term. There are two kinds of singular terms, proper names and definite descriptions. Examples of proper names are ‘Aristotle’ and ‘Grace Hopper’. Examples of definite descriptions are ‘the teacher of Alexander the Great’ and ‘the inventor of COBOL’. The referent is often determined by context. For example, there may be several people with the name ‘Grace Hopper’, but in a conversation about the history of computing, it’s clear that the intended referent is Grace Hopper, the inventor of COBOL.

The referent of a pronoun is also determined by context. In the sentence, “Grace Hopper was a naval officer, and she invented COBOL”, ‘Grace Hopper’ and ‘she’ refer to the same person. In this case, ‘she’ is being used as a singular term. There are some tricky cases, though. In the sentence

If someone wins the lottery, then he will be very happy.

Here, ‘he’ isn’t used as a singular term. Substituting some name for ‘he’ results in a non-equivalent sentence:

> If someone wins the lottery, then Joe Biden will be very happy.

Some sentences can contain more than one singular term. For instance,

> Oklahoma City is north of Dallas.

We get predicates by deleting one or more singular terms from a sentence. Form this sentence, we can get the predicates

> _______ is north of Dallas

and

> _______ is north of _______.

So, think of predicates as strings of words with blanks such that when the blanks are filled in with singular terms, we a complete sentences. A predicate with just one blank is a one-place, or monadic, predicate. A predicate with two blanks is a two-place, or dyadic, predicate, and so on. Predicates with more than one place are collectively called polyadic predicates. Instead of using blanks, we’ll use the lowercase letters ‘w’, ‘x’, ‘y’, and ‘z’.

Singular terms and predicates allow us to say things like “Josh is present” and “Sarah doesn’t like root canals.” With these alone, though, we can’t say simple things like “everyone is present” and “no one likes root canals.” For statements like these, we also need quantity terms. These are terms like ‘all’, ‘some’, and ‘none’. It should be clear that quantity terms are not singular terms. ‘Everyone’ does not refer to a particular thing. Instead, the sentence ‘everyone is present’ means that the predicate ‘is present’ is true of everyone, and what ‘everyone’ means will of course be determined by context.

1.2 PL: Singular Terms and Predicates

The formal language *PL* contains the sentential connectives of *SL*, along with components that correspond to the singular terms, predicates, and quantity terms of English and other natural languages. For singular terms, *PL* uses individual constants, which are lowercase letters of the Roman alphabet ‘a’ through ‘v’, with subscripts if necessary. The predicates of *PL* are uppercase letters of the Roman alphabet. As in English, predicates in *PL* have blanks that are filled in with individual constants to form sentences. A sentence is formed by writing the predicate letter first, followed by the singular terms in the order given by the translation key. A translation key will specify a universe of discourse (UD), which is the set of things that we’re talking about. It will also assign individual constants to some, or all, of the members of the universe of discourse. Finally, it will assign predicate letters to the relevant natural language predicates. Here is an example of a translation key:

UD: The Smith family pets

Cx: x is a cat

Dx: x is a dog

Lxy: x likes y

a: Abby

b: Bailey

c: Cookie

Using this translation key, we can translate these English sentences

1. Abby is a cat.
2. Bailey and Cookie are dogs.
3. Bailey likes Abby.
4. If Abby is a cat, then Cookie doesn’t like Abby.

like this in *PL*:

1. Ca
2. Db & Da
3. Lba
4. Ca ⊃ ¬Lca

We could also use the same translation key to go from *PL* to English. Translating these sentences in *PL*

1. Bc

2. $\mathsf{Bd} \vee \neg \mathsf{Lcb}$
3. $\mathsf{Lab}\ \&\ \neg \mathsf{Laa}$
4. $(\mathsf{Lab}\ \&\ \mathsf{Lac} \supset) \neg (\mathsf{Db} \vee \mathsf{Dc})$

like this into English:

1. Either Bailey is a dog or Cookie doesn't like Bailey.
2. Abby likes Bailey, but doesn't like herself.
3. Abby likes both Bailey and Cookie only if neither Bailey nor Cookie are dogs.

1.3 Quantifiers

When we establish a universe of discourse, we are essentially pretending, for the sake of discussion, that those are the only things that exist. Using the above translation key, we could translate

Bailey likes everything

as

$(\mathsf{Lba}\ \&\ \mathsf{Lbc})\ \&\ \mathsf{Lbb}$

Note that since Bailey is one of the three things that are in the universe of discourse, then saying that Bailey likes everything is to imply that Bailey likes himself. Likewise, we could translate

Bailey likes something

as

$(\mathsf{Lba} \vee \mathsf{Lbc}) \vee \mathsf{Lbb}$

This is only practical, however, when the universe of discourse is *very* small. Even with a small universe of discourse, things can quickly get out of hand. For example, to translate

Something likes something

we would have to do something like this:

$\{[(\mathsf{Laa} \vee \mathsf{Lab}) \vee (\mathsf{Lac} \vee \mathsf{Lba})] \vee [(\mathsf{Lbb} \vee \mathsf{Lbc}) \vee (\mathsf{Lca} \vee \mathsf{Lcb})]\} \vee \mathsf{Lcc}$

Imagine what translating 'Someone likes someone else' would require if the universe of discourse were all of the students at the university!

This is why we need something analogous to the quantity terms that was discussed earlier. For this, we will use the quantifier symbols '$\forall$', usually read as "for all", and '$\exists$', usually read as "there exists" or "there is". A quantifier consists of one of these symbols followed by a variable, which in *PL* are the lowercase Roman letters 'w' through 'z'.

Let's us the same translation key that we used in the preceding section and look at a few examples. To say that everything in the universe of discourse is a dog, we simply write

$\forall$xDx

This is how we would symbolize 'There is at least one cat':

$\exists$xCx

We can symbolize 'Bailey likes everything' as

$\forall$xLbx

and 'Bailey likes something' would be

$\exists$xLbx

We can think of quantifiers as providing a way of interpreting variables in formulas of *PL*. In '$\forall$xDx', the universal quantifier tells us that the variable in "Dx" stands for every object in the universe of discourse, while an existential quantifier would tell us that the variable stands for at least one object in the universe of discourse.

Quantifiers are like negations in two important senses. First, remember that a negation operates only on the sentence that immediately follows it. In '$\neg$P $\supset$ Q', the tilde negates only the sentence 'P', while in '$\neg$(P $\supset$ Q)', the tilde negates the entire conditional 'P $\supset$ Q'. In the same way, quantifiers interpret only the formulas that they immediately precede. In the sentence

$\forall$xDx & Cx

the universal quantifier only provides an interpretation of the formula 'Dx', but in

$\forall$x(Dx & Cx)

it allows us to interpret both variables in the conjunction ‘$Dx \,\&\, Cx$’.

Second, just like ‘$\neg(P \lor Q)$’ is a negation, while ‘$\neg P \lor \neg Q$’ is a disjunction of two negations, ‘$\forall x(Dx \lor Cx)$’ is a universal quantification, while ‘$\forall xDx \lor \forall xCx$’ is a disjunction of two universal quantifications.

We’ve reserved the letters ‘w’ through ‘z’ for variables. For the most part, which letter you choose to use does not matter. For example,

$$\forall xLxb$$

and

$$\forall yLyb$$

are the same sentence. They are both perfectly good symbolizations of ‘Everything likes Bailey’. Although we used ‘x’ for the translation key, we don’t have to use the same variable when doing the actual translations. Note that we do have to use the same variable for the quantifier and the quantified formula. For example, this isn’t allowed:

$$\forall xDy$$

The ‘y’ isn’t in the scope of any quantifier. There are times when it’s helpful, although not necessary, to use different variables. For example, here’s a translation of ‘There is at least one dog, and there is at least one cat.’

$$\exists xDx \,\&\, \exists xCx$$

Although, this is a perfectly good translation, using different variables would help to prevent the sentence from being read as this:

$$\exists x(Dx \,\&\, Cx) \text{ (Mistake!)}$$

I’ll leave it up to you to decide what would be wrong with that. So, here’s a good way to translate the sentence:

$$\exists xDx \,\&\, \exists yCy$$

Which quantifier needs to be used is usually fairly easy to determine. Words like ‘some’, ‘something’, or ‘someone’ will use the existential quantifier, while words like ‘each’, ‘every’, ‘any’, and ‘all’ will ordinarily use the universal quantifier. There are times when this can get a bit tricky, however. For example, ‘Bailey likes everything’ and ‘Bailey likes anything’ mean the same thing, and are both translated this way.

$\forall xLbx$

Notice the difference between ‘Bailey doesn’t like everything’ and ‘Bailey doesn’t like anything’, however. Those sentence don’t mean the same thing, because they have different truth conditions. The first is false only if Bailey likes everything, while the second only requires that there be one thing that Bailey likes to be false. So, we can translate the first as a negated universal quantification and the second as a negated existential quantification.

Actually, anything that we can translate with a existential quantifier could be translated with a universal quantifier, and vice versa. That is, we only really need one quantifier. Remember that we didn’t really need both of ‘&’ and ‘∨’ because

$A \& B$

is equivalent to

$\neg(\neg A \lor \neg B)$

In the same way, ‘Everything is F’ has exactly the same truth conditions as ‘It’s not the case that there is something that is not F.’ So,

$\forall xFx$

is equivalent to

$\neg\exists x\neg Fx$

Just keep in mind that as a negation is moved across a quantifier, the quantifier changes.

1.4 Formal Syntax of PL

Now that we have a basic understanding of the features of *PL*, we're in a better position to understand its syntax. The vocabulary of *PL* consists of:

Predicates: Capital Roman letters with or without positive integer subscripts followed by zero or more primes. (Exactly *n* primes indicate an *n*-place predicate. A zero-place predicate is equivalent to a sentence letter of *SL*.)

Individual constants: Lowercase Roman letters 'a' through 'v' with or without positive integer subscripts.

Individual variables: Lowercase Roman letters 'w' through 'z' with or without positive integer subscripts.

Individual term: An individual constant or individual variable.

Truth-functional connectives: $\neg$ & $\vee$ $\supset$ $\equiv$

Quantifier symbols: $\forall$ $\exists$

Punctuation marks: ( )

In practice, we will omit the primes from the predicate letters. Adding zero-place predicates means that every sentence that is symbolized in *SL* can also count as a symbolization in *PL*.[1]

Expression of *PL*: a sequence of elements of the vocabulary of *PL*.

Quantifier of *PL*: An expression of *PL* of the form ∀x (a universal quantifier) or ∃x (an existential quantifier). A quantifier is said to *contain* a variable. For example, '∀x' contains the variable 'x' and '∃z' contains the variable 'z'. We'll call a quantifier containing 'x' an 'x quantifier', a quantifier containing 'y' a 'y quantifier', and so on.

Atomic formula of *PL*: An expression of *PL* that is an *n*-place predicate of *PL* followed by *n* individual terms of *PL*.

[1] In this course, however, we won't be using sentence letters in *PL*.

Now, we can give a recursive definition of ‘formula of *PL*’, using ϕ, ψ, and χ as metavariables for expressions of *PL* and α as a metavariable for individual variables of *PL*:

1. Every atomic formula of *PL* is a formula of *PL*.
2. If ϕ is a formula of *PL*, then so is $\neg\phi$.
3. If ϕ and ψ are formulas of *PL*, then so are $(\phi \,\&\, \psi)$, $(\phi \vee \psi)$, $(\phi \supset \psi)$, and $(\phi \equiv \psi)$.
4. If ϕ is a formula of *PL* that contains at least one occurrence of α and no α-quantifier, then $\forall\alpha\phi$ and $\exists\alpha\phi$ are both formulas or *PL*.
5. Nothing is a formula of *PL* unless it can be formed by repeated applications of 1–4.

Logical operator of *PL*: An expression of *PL* that is either a quantifier or a truth-functional connective.

Now, we need the concept of a subformula and a main logical operator:

1. If ϕ is an atomic formula of *PL*, then ϕ contains no logical operator, and ϕ is the only subformula of ϕ.
2. If ϕ is a formula of *PL* of the form $\neg\psi$, then $\neg$ is the main logical operator of ϕ and ψ is the immediate subformula of ϕ.
3. If ϕ is a formula of *PL* of the form $(\psi\&\chi)$, $(\psi\vee\chi)$, $(\psi \supset \chi)$, or $(\psi \equiv \chi)$ then the binary connective between ψ and χ is the main logical operator of ϕ and ψ and χ are the immediate subformulas of ϕ.
4. If ϕ is a formula of *PL* of the form $\forall\alpha\psi$ or $\exists\alpha\psi$, then the quantifier that occurs before ψ is the main logical operator of ϕ and ψ is the immediate subformula of ϕ.
5. If ϕ is a formula of *PL*, then every subformula of a subformula of ϕ is a subformula of ϕ, and ϕ is a subformula of itself.

Scope of a quantifier: The scope of a quantifier in a formula ϕ of *PL* is the subformula ψ of which that quantifier is the main logical operator.

Bound variable: An occurrence of a variable α in a formula ϕ of *PL* is bound if and only if that occurrence is within the scope of an α-variable.

Free variable: An occurrence of a variable α in a formula ϕ of *PL* is free if and only if it is not bound.

Sentence of *PL*: A formula ϕ of *PL* is a sentence of *PL* if and only if contains no free variables.

1.5 PL: Translations

In **??**, we encountered the A, E, I, and O statements of categorical logic:

A: All S are P.

E: No S are P.

I: Some S are P.

O: Some S are not P.

We are able to express all of these claims, and more, in *PL*. Let's look at some examples using this translations scheme:

UD: living things
Sx: x is a snake.
Rx: x is a reptile.
Px: x is poisonous.
Wx: x is warm-blooded.

Now, consider the sentence:

All snakes are reptiles.

How should we translate that in *PL*? An initial inclination might be to translate it this way:

$\forall x Sx \ \& \ \forall x Rx$

This, though, is clearly wrong, since it means that everything in the universe of discourse is a snake *and* everything in the universe of discourse is a reptile. That would be true only if every living thing were a snake. We can begin to translate it correctly by first paraphrasing it in a way that captures the truth conditions of the sentence. Basically, it is saying, for each living thing, if that thing is snake, then it is also a reptile. So, an equivalent sentence of *PL* is

$\forall x(Sx \supset Rx)$

The sentence ‘No snakes are warm-blooded’ asserts that everything in the universe of discourse that is a snake is not warm-blooded, so this is an appropriate translation:

$\forall x(Sx \supset \neg Wx)$

Of course, when one says that no snakes are warm-blooded, one is essentially denying that there are any warm-blooded snakes. So, another perfectly good translation would be this:

$\neg\exists x(Sx \,\&\, Wx)$

The claim that some snakes are poisonous is not a claim about everything, so we shouldn’t use the universal quantifier. ‘Some snakes are poisonous’ is true just in case there is at least one thing in the universe of discourse that is both a snake and is poisonous. So, this I sentence is naturally translated as:

$\exists x(Sx \,\&\, Px)$

O sentences are translated the same way, with the addition of a negation. ‘Some snakes are not poisonous’ is

$\exists x(Sx \,\&\, \neg Px)$

Keep in mind that for sentences of more than the simplest degree of complexity, there will always be more than one acceptable translation. Any universal quantification can be rephrased as a negated existential quantification, every existential quantification is equivalent to a negated universal quantification, every conjunction can be expressed as the negation of a disjunction, and so on. Even so, there is a natural way to translate the categorical statements, using the universal quantifier for the universal A and E sentences, and an existential quantifier for the particular I and O sentences. If ϕ and ψ are formulas with the variable x, then

A: $\forall x(\phi \supset \psi)$

E: $\forall x(\phi \supset \neg\psi)$

I: $\exists x(\phi \,\&\, \psi)$

O: $\exists x(\phi \,\&\, \neg\psi)$

Let's practice with some sentences using this translation scheme:

UD: All birds

Fx: x can fly

Rx: x is a raptor

Hx: x is a hawk

Px: x is a penguin

Sx: x can swim

Vx: x is a vertebrate

Wx: x is warm-blooded

1. All raptors can fly.
2. No penguins can fly.
3. Some raptors are hawks.
4. Penguins can swim but can't fly.
5. If any penguin swims, then all raptors are warm-blooded.
6. If any penguin swims, then it is warm blooded.
7. All birds are vertebrates.

The first three are simple:

1. $\forall x(Rx \supset Fx)$
2. $\forall x(Px \supset \neg Fx)$
3. $\exists x(Rx \,\&\, Hx)$

One way to understand the fourth would be as a conjunction: 'All penguins swim and no penguins fly.' So, the translation could be

4. $\forall x(Px \supset Sx) \,\&\, \forall y(Py \supset \neg Fy)$

Another way to paraphrase this would 'All penguins both swim and not fly'. This means that the third sentence can also be translated this way:

4. $\forall x[Px \supset (Sx \,\&\, \neg Fx)]$

This simpler translation should be preferred.

The fifth sentence says that if there is a penguin that swims, then all raptors are warm-blooded. This is a conditional sentence that has a existential quantification for an antecedent and a universal quantification for a consequent.

5. $\exists$x(Px & Sx) $\supset$ $\forall$y(Rx $\supset$ Wx)

The antecedent and the consequent of the sixth sentence share the same subject. It says that for any penguin, if that penguin swims then it is warm-blooded. This means the same as saying that all penguins that swim are warm blooded. So, we should translate it like this:

6. $\forall$x[(Px & Sx) $\supset$ Wx]

The key difference between the fifth and sixth sentences is that the fifth contains two quantity terms, 'any' and 'all', and the sixth has only one quantity term. So, the fifth sentence should require two quantifiers, but only one quantifier for the sixth.

Since the universe of discourse is all birds, the seventh simply says that everything is a vertebrate, or

7. $\forall$xVx

Had the universe of discourse been broader, then it would need to be translated something like

$\forall$x(Bx $\supset$ Vx)

For the rest of the examples in this section, we will pick predicate letters and individual constants that make sense for the sentence to be translated.

PL can be used to translate much more complex sentences than just the four standard sentence types in categorical logic.

Assuming a universe of discourse as all people, then

Everyone who likes Alfred also likes Beth and Carlos.

This is paraphrased as, 'For all people, if a person likes Alfred, then they also like Beth and like Carlos.' It's then translated as

$\forall x[Lxa \supset (Lxb \,\&\, Lxc)]$

Now consider

Beth likes anything that Alfred and Carlos like.

This can be paraphrased as “Everything is such that, if both Alfred and Carlos like it, then Beth does too.” So, a natural translation would be

$\forall x[(Lax \,\&\, Lcx) \supset Lbx]$

1.6 Multiple Quantifiers

1.7 Identity