Chapter 5

Grammar

I. Introduction

In the thirties, grammar became a central issue in Wittgenstein's philosophy.[1] Wittgenstein's remarks about grammar from this period are some of most controversial. For example, he wrote that the grammar of some signs completely determines their meaning,

> Zur Grammatik gehört nicht, daß dieser Erfahrungssatz wahr, jener falsch ist. Zu ihr gehören alle Bedingungen (die Methode) des Vergleichs des Satzes mit der Wirklichkeit. Das heißt, alle Bedingungen des Verständnisses (des Sinnes). [PG §45, p. 168]

> What belongs to grammar are the conditions (the method) necessary for comparing the proposition with reality. That is all the conditions necessary for the understanding (of the sense).[PG §45, p. 133]

Wittgenstein also mantained that investigations into the essence of things are grammatical investigations.[2] Most philosophers do not think that Wittgenstein's notion of grammar is the one in common use. The controversial nature of these statements begins with Wittgenstein's notion of grammar. The absence of an explicit definition in his published writings makes it difficult to justify his use of the word 'grammar.' Wittgensein's brief explanation in the *Big Typescript* lacks specificity. The following pages develop a formal definition of grammar provisionally fitting the purposes of this investigation: (*i*) to compare Wittgenstein's notion of grammar with conventional grammars and determine whether Wittgenstein's use of 'grammar' is justified or not, *(ii)* to demonstrate that a grammatical analysis of Wittgenstein's kind can yield mathematical results, (*iii*) to allow for a more

1. Grammar will remain central to Wittgenstein's philosophy beyond the middle period. In the *Philosophical Investigations,* he wrote: "Essence is expressed by grammar" and "Grammar tells what kind of object anything is." PI §371, 373

2. BT 9, 38

precise definition of the grammatical nature of mathematics. This chapter pursues the first goal, while the following two chapters develop the others.

The rest of this chapter compares the analytical capacities of Wittgenstein's grammar and a conventional ones. The first section defines 'language'. The second section formally models the conventional notion of grammar, using basic mathematical and logical tools and the syntax of propositional calculus and English grammar as examples. The third section formalizes Wittgenstein's explicit thoughts about grammar during this period. Finally, the last section compares the analytic capacities of both grammatical notions. It compares their grammatical categories and equivalence relations. The comparison answers two questions, (*i*)? and (*ii*) Does Wittgenstein's approach make finer distinctions than conventional grammar? If it is possible to construct a conventional grammar out of Wittgenstein's categories, Wittgenstein's notion dovetails with conventional ones. If Wittgenstein's approach make finer distinctions than conventional grammar, Wittgenstein's grammar refines the conventional one. Answering these questions will establish if Wittgenstein's notion of grammar covers the same cases than any of the more familiar notions. Their answers might also explain why Wittgenstein created his own approach instead of using a conventional one.

The introduction of these two approaches employs an abstract, rule-based notion of grammar. It models grammar as a formal theory. Grammatical theories are special cases of formal theories. Grammatical theories are first order theories with a concatenation operator and several predicates: one for each grammatical category. The domain of the theory is the set of language expressions, and every proposition is of the form $x_1, x_2, \ldots x_n\ (C_1x_1\ \&\ C_2x_2\ \& \ldots C_nx_n) \quad C_k(\mathsf{C}(x_1, x_2, \ldots x_n))$ where C is a concatenation operator. Logical notions such as satisfaction, truth, model, consistency, completeness, etc. have immediate application.

This formal reconstruction and analysis will ultimately shed light on Wittgenstein's philosophy of mathematics. Even though some of its formal results might well have importance on their own, logic is only a tool for the following philosophical analysis. Accordingly, an intuitive introduction precedes the introduction of every formal element. It assists readers in understanding the issues raised and interpreting the results.[3]

II. A Formal Background for the Discussion of Wittgenstein's Grammar

A. Language

Definition 1.1 [language]: Define a language **L** as the structure < ∑, E, W >, where ∑ is the *alphabet* or the finite, non-empty set of *words*, W and E are sets of finite strings of words, such that (∑   W)   E and every member of E is a substring of some member of W.

---

3. This formal approach to Wittgenstein's grammar is not the first. It is also not the first time that the formalization of Wittgenstein's notion of grammar compares it with linguist's grammar. In 1974, the Research Center for the Language Sciences of Indiana University published, as part of its 'Approach to Semiotics' paperback series, a very interesting book by Cecil H. Brown entitled *Wittgensteinian Linguistics* (The Hague: Mouton, 1974). Brown presented the contemporary linguistic controversy between pure and descriptive semiotics as a dispute between Chomsky's and Wittgenstein's views of language. Brown explicitly recognizes the evolution of Wittgenstein's philosophy of language. When talking about Wittgenstein's views on language, Brown refers to what he calls Wittgenstein's "ordinary language philosophy" (p.13): Wittgenstein's views after 1929 when "after having ignored the philosophy of language for some time, he took it up again." (p.15) "Readers who have encountered the works of both Chomsky and Wittgenstein are no doubt aware of the pronounced difference in the manner in which each explains the essential nature of patterned communication in the modality of natural language. This difference emerges at the most general levels of analysis. Chomsky is concerned with pure semiotics, the development of a language to talk about signs. Wittgenstein emphasizes descriptive semiotics, the study of actual sign use." (p. 13) In Brown's interpretation, Wittgenstein claims that "any language, be it artificial or natural, is understood not in terms of some other language, but in terms of itself, in the manner in which its signs are ordinarily used" (p. 17). Grammatical rules do not hide themselves. They are immediately identifiable in the surface structure of language (p. 90). By contrast, linguistic grammarians – at least of the most common Chomskian sort – locate grammar in the not-so-accessible deep structure of language. For Brown, "the deep structure of language is comparable to the logical systems or artificial languages of logical positivism. The deep structure is a kind of ideal language with which sentences of natural languages can be compared and consequently understood." Except for its pragmatic stress, Brown's formal treatment is very similar to the one this chapter presents.

W is the set of *acceptable* or *well-formed strings*, and E is the set of *expressions*. Every meaningful element of language is an expression.[4] For example, in ordinary English, Σ contains words like 'apple', 'be', 'caring', etc., E contains words like 'dog' and 'caring' and complex expressions like 'my dog' or 'the yellow pencil on my desk.' Finally, W contains all the grammatically correct sentences in English: 'Try to remember my name', 'This is not the end of the line', 'Could you come here for a second?' etc.

B. What is Grammar?

> *Grammar* . . . is felt to be a term with a far wider meaning than that which a considered definition would propose or an elementary text illustrate. . . It is perhaps the vaguest term in the schoolmaster's, if not the scholar's vocabulary.
>
> Ian Michael[5]

For most Wittgenstein's scholars, 'grammar' in Wittgenstein has a ". . . meaning far wider than the ordinary one."[6] In the words of Hans-Johann Glock, Wittgenstein's notion of grammar diverges from ordinary usage only in extension, not in sense.[7] As evidence, Newton Garver quotes one of Wittgenstein's letters to Moore, where he writes to be ". . .using the words 'grammar' and 'grammatical' in their ordinary sense but making them apply to things they do not ordinarily apply to."[8] Calling Wittgenstein's use of the term 'grammar' "liberal", Glock recognizes no significant difference between the ordinary

4. Wittgenstein calls expressions *words*.

5. Ian Michael, "Grammar, Divisions of Grammar and Parts of Discourse" in *English Grammatical Categories and the Tradition to 1800* (Oxford: Cambridge University Press, 1970) 37.

6. Finch, Henry LeRoy, *Wittgenstein: The Later Philosophy* (Atlantic Highlands: Humanities Press, 1997) p.149.

7. Glock, H. G., *A Wittgenstein Dictionary* (Cambridge: Blackwell, 1996) 152.

8. Garver, N., "Philosophy as Grammar" in *This Complicated Form of Life* (Chicago: Open Court, 1994) 150.

sense of grammar and Wittgenstein's. The meaning of 'grammar' covers both Wittgenstein's peculiar use and the ordinary one.[9] In consequence, understanding Wittgenstein's peculiar assessment of grammar requires an investigation into the meaning of the word 'grammar.'

The German word 'Grammatik' – just like the English word 'grammar' – descends from the Greek 'γραμμα' meaning 'letter.' In classical Greek the expression η γραμματικη (τηχνη) had two principal meanings. It addressed the phonetic (accentuation and pronunciation) and metaphysical values of letters. It also referred to the knowledge required to read and write.[10] At the time'grammar' entered the Latin language, its sense had gradually extended to include the general study of literature and language. In medieval usage, 'grammar' referred only to Latin grammar. In the seventeenth century it took on a more general meaning addressing language proficiency in Latin, English, French, etc[11]. Nevertheless, the notion of 'universal grammar' – not the grammatical features of a particular language, but those common to all linguistic usage – did not appear until the work of Port Royal grammarians in the 18th Century. Even though it disappeared again in the middle of the nineteenth century, the work of Noah Chomsky launched a resurgence of universal grammar in the twentieth century.[12]

9. However, few authors venture a detailed characterization of grammar. For LeRoy Finch, for example, grammar is language and the phenomena connected with it in terms of its possibilities. Grammar lays down the limits of sense in language. It draws the line that separates sense from non-sense, expressibility from inexpressibility. Because it helps make sense of the evolution of Wittgenstein's philosophy, LeRoy Finch is not the only scholar to favor this interpretation. In Wittgenstein's middle period, grammar plays a similar role that logic did in his earlier work. During those years, Wittgenstein came to believe that logic was not the philosophical panacea he had mistaken it to be. Instead, logic constitutes a significant part, but not the whole of a larger grammatical philosophy. This dissertation's definition does not diverge far from LeRoy Finch's.

10. (Michael 1970, 24)

11. Jackson, Howard: *Discovering Grammar* (Oxford: Pergamon Institute of English, 1985) 1.

12. Serbat, Guy, *Casos y Funciones* (Madrid: Editorial Gredos, 1988) 74-84.

Today, the term ‘grammar’ has two uses. On the one hand, it refers to the structural features of a language. For example, the ‘Grammar of English’ refers to its structural features, instead of its semantics or pragmatics. On the other hand, ‘grammar’ also refers to the science or art describing (or prescribing) language’s structural traits. One talks about Chomskian or transformational grammars in this sense. Capitalizing the word ‘grammar’ in the first sense avoids confusion. Some authors prefer to mark the difference by calling ‘grammar’ the first one and ‘a grammar’ the second.

These two meanings of grammar have competed with each other since the seventeenth century. For descriptive grammarians, grammar is a science, a study of a set of phenomena. For prescriptive grammarians, it is an art: the skill or technique of using the language well. Ben Jonson, George Kittredge and L. Murray are well-known prescriptive grammarians. In contrast, Francis Bacon was a descriptive grammarian. Today, most consider the prescriptive and descriptive aspects of grammar inseparable. In the introductory pages of his *Discovering Grammar*, Howard Jackson writes,

> In the event, although different basic attitudes prevail, the distinction is probably not so clear cut as the terms ‘descriptive’ and ‘prescriptive’ imply. To be sure, prescriptive grammarians included rules in their grammars, such as “you should not end a sentence with a preposition”; but in so doing they still had to describe what a ‘sentence’ and a ‘preposition’ are. And a descriptive linguist producing a grammar of modern English, for example, has to make a choice of which English usage he is going to describe; and he would usually select the ‘standard’ variety, perhaps even ‘standard educated usage’, and by so doing he would have indulged in an implicit prescription.[13]

Nevertheless, the descriptive/prescriptive dichotomy survives in the current opposition between *school* and *linguistic* grammar. School grammarians stress the prescriptive aspect

13. (Jackson 1985, 2)

of grammar, while linguistic grammarians emphasize the descriptive dimensions of their science.[14]

Sometimes ‘grammar’ refers to the basic structural aspects of a language. Other times, it means only the ‘correct’ or ‘standard’ usage of the language. This makes specifying both the aspect of the language and the kind of grammar referred to vital.

Meaning of Grammar

	Linguist Grammar	School Grammar
Aspect of Language	General Usage	Correct or Standard Usage
Study of Language	Descriptive	Prescriptive

For the purposes of this dissertation, ‘grammar’ describes the structural aspects of language in its general use. For further clarification, it restricts ‘grammar’ to the syntactic structure of sentences. Grammar consists of two sub-components: morphology and syntax. Morphology deals with the form of words, while syntax deals with meaningful word combinations.[15] ‘Syntax’ has its roots in the Greek word for ‘arrangement’. It addresses the possible arrangements, patterns or orders of words as well as the differences in meaning that the various orderings bring out.[16]

14. In *Traditional Grammar* Jewell A. Friend argues against the identification of the prescriptive tradition with schoolroom grammar. At the end of his book’s introduction, he lists seven points of divergence. Amongst them, schoolroom grammar does not distinguish between written and oral forms of language, also ignoring the distinction between lexical and grammatical meaning. (Carbondale: Southern Illinois University Press, 1976) i - xi.

15. Huddleston, Rodney, *English Grammar: An Outline* (Cambridge: Cambridge University Press, 1988) 1.

16. (Jackson 1985, 3)

C. The Conventional Approach

To define the ordinary sense of ‘grammar’, this section synthesizes the essential features of sophisticated linguistic grammars like Chomsky’s and everyday school grammar. All conventional grammars distribute the expressions of the language into several categories, providing explicit rules for combining these expressions in a way that acknowledges the grammatical categories to which they belong. All conventional grammars present this basic feature.

> Reduced to the simplest possible terms, the methods of structural grammarians consist of breaking the flow of spoken language into the smallest possible units, sorting them out, and studying the various ways in which these units are joined in meaningful combinations.[17]

The conventional presentation of syntax for the predicate calculus exemplifies this feature. The basic symbols – broken into categories, and a recursive definition for terms and well-formed formulas – determine the language of predicate calculus. School grammar has a similar presentation.

> Most traditional “school” grammars begin by defining and classifying . . .words into part-of-speech categories, and proceed from there to more inclusive sentence components until they arrive at a discussion of the sentence itself.[18]

The first step in the process of learning the grammar of a language is learning the vocabulary and the grammatical categories. Learning that ‘duck’ is a noun and ‘she’ a pronoun is not sufficient. To learn that ‘duck’ is a singular common noun and that ‘she’ is a singular, feminine, third person, personal pronoun is also necessary. The categories to which an expression belongs exhaust its grammar. The next step is to learn which sequential combinations of categories are grammatically correct and which are not. Determining which

17. Jeanne H. Herndon, *A Survery of Modern Grammars* (New York: Molt, Rinehart & Winston, 1970) 65.

18. Joseph La Palombara, *An Introduction to Grammar* (Cambridge: Winthrop, 1976) 23.

expression sequences are meaningful requires knowing to which categories words belong and which categories combine into grammatically acceptable expressions.

A set **C** of grammatical categories, an interpretation function I and a set **S** of word combination rules constitute an abstract grammar **G** = <**C**, I, **S**>. Besides grammatical categories like ‘noun’ in school grammar, or ‘statement letter’ in the syntax of propositional logic, a grammar also includes an interpretation function that determines which words of the language belong to which categories. This function maps the category ‘noun’ to the set of nouns in the language. Finally, the set of rules **S** sets parameters for the combination of expressions into other expressions. For example, consider SEN as the category ‘sentence’ and CON as the category ‘conjunction’. The rule SEN CON SEN $\rightarrow$ SEN says that the concatenation of a sentence, a conjunction and a sentence creates another sentence.

Definition 1.2.1 [grammatical language]: Let C, C_0, C_1, C_2, . . . and the arrow $\rightarrow$ be the *basic symbols* of grammatical language.

Definition 1.2.1.1 [categorical symbols]: C, C_0, C_1, C_2 are the *categorical* symbols of grammatical language.

Definition 1.2.1 [basic symbols]: Let C, C_0, C_1, C_2, . . . and the arrow $\rightarrow$ be the *basic symbols* of grammatical language.

Definition 1.2.2 [grammatical formulae]: Every sequence of categorical symbols of the form $C_0\ C_1 \ldots C_n \rightarrow C$ is a *well-formed grammatical formula*.

Definition 1.2.2.1 [antecedent and resulting categorical symbols]: Given a grammatical formula $C_0\ C_1 \ldots C_n \rightarrow C$, the *antecedent* categorical symbols of the rule are C_0 $C_1 \ldots C_n$, and C is the *resulting* categorical symbol.

Definition 1.2.2.2 [degree of a grammatical formula]: The *degree* a grammatical formula $C_0\ C_1 \ldots C_n \rightarrow C$ is *n*, the number of antecedent categorical symbols.

Definition 1.2 [grammatical theory]: Given a set of categories **C**, a grammatical theory **S** is a set of well-formed expressions of the language — called the *rules* of the grammar— such that, for all C **C**, C occurs in some rule s **S**.

Definition 1.2.3 [interpretation]: Given a language **L** = <Σ, E, W> and a set of categorical symbols **C**, an *interpretation* I is a function from **C** into the power set of the expressions, I: **C** → (E), such that I[**C**]=E.

Definition 1.2.4 [application]: A rule $C_1\ C_2 \ldots C_n \rightarrow C$ *applies* to an n-tuple of expressions $<e_1, e_2, \ldots e_n>$ iff, for all $1 \leq i \leq n$, e_i $I(C_i)$.

Definition 1.2.5 [result of an application]: A concatenation of expressions $e_1 \{ e_2 \{ \ldots \{ e_n$ is *the result of applying* a rule $C_1\ C_2 \ldots C_n \rightarrow C$ to an n-tuple of expres-sions $<e_1, e_2, \ldots e_n>$ iff the rule applies to the n-tuple.

Definition 1.2.6 [satisfaction]: A sequence of expressions $<e_1, e_2, \ldots e_n>$ *satisfies* a rule $C_1\ C_2 \ldots C_n \rightarrow C$ iff, if the rule applies to the n-tuple $<e_1, e_2, \ldots e_n>$, then $e_1 \{ e_2 \{ \ldots \{ e_n$ I(C) where $e_1 \{ e_2 \{ \ldots \{ e_n$ is the result of applying $C_1\ C_2 \ldots C_n \rightarrow C$ to $<e_1, e_2, \ldots e_n>$.

Definition 1.2.7 [grammatical truth]: A rule s of degree n is *true* for a given interpretation I, written $\div_I$s, iff every n-tuple of language expressions satisfies s.

Definition 1.2.8 [model of a theory]: An interpretation I *models* a grammatical theory **S** if every rule in **S** is true for I.

Definition 1.2.9 [consistency]: A grammatical theory **S** is *consistent*, if some interpretation function I models **S**.

Example 1.2.10: Consider the grammatical theory **S** with categories SENT, NOUN and VERB and the single rule s: NOUN VERB $\rightarrow$ SENT. Interpretation I assigns to NOUN the set {Bill} and to VERB {runs}. If I assigns any set of expressions including ‘Bill runs’ to SENT, then s is true for I and I models **S**. However, if another interpretation J agrees with I on NOUN and VERB but assigns to SENT a set of expressions not including ‘Bill runs’, then s is not true for J and J fails to model **S**.

Note 1.2.11: Those familiar with the conventions of Chomskian or generative grammar will recognize that the above notion of abstract grammar is divorced from all questions of computability. It sets no *a priori* limit on the number of rules that may enter into a grammatical theory, except that they cannot comprise a proper class. Indeed, as will appear later, any language **L** whose well-formed expressions make up a set will have a grammar in this sense. The above notion of abstract grammar is ‘purely logical’, yielding a form of decompositional description of a language failing to constrain a finite machine’s ability to recognize or decide appropriate sequences.

Definition 1.3 [conventional equivalence]: Given a set of categories **C** and an interpretation I, define the relation of *conventional equivalence* ~ on E^2 by:

$$e_1 \sim e_2 \text{ iff } \quad C \quad \mathbf{C} \, [\, (e_1 \quad I(C)) \Leftrightarrow (e_2 \quad I(C)) \,].$$

Two expressions are conventionally equivalent if they belong to exactly the same categories.

Definition 1.4 [decomposition]: Let S be a grammatical theory. An expression e *decomposes* into a set of expressions B iff (1) a rule s in S applies to the n-tuple of expressions $<e_1, e_2, ... e_n>$, (2) an expression e_i occurs in the n-tuple $<e_1, e_2, ... e_n>$ if and only if it belongs to B, and (2) the application of rule s to ntuple $<e_1, e_2, ... e_n>$ results in expression e.

An expression decomposes into a set of other expressions if a rule in the grammatical theory explains how to combine those expressions into the original one. Given that other expressions might exist beyond the basic symbols and acceptable strings (E might be larger than $\sum \quad W$), other grammars might decompose a string in different ways. Consider the expression 'My dog is dead.' If a grammatical rule said that combining a singular nominal expression (like 'my dog') with the singular third person indicative present form of the verb to be ('is') and an adjective (like 'dead') resulted in a sentence, then 'My dog is dead' would decompose into the three expressions 'my dog', 'is' and 'dead'. On the other hand, if another rule stated that subjects combined with predicates matching in number form sentences, then 'My dog is dead' would decompose into only two expressions, 'My dog' and 'is dead'. In consequence, the set of expressions into which an expression decomposes depends upon the rules in the grammatical theory.

Definition 1.5 [construction as, code]: Given a grammatical theory **S**, a set of categories **C** and an interpretation function I, a *construction* of an expression e *as* member of category C is a sequence $P = <p_1, p_2, . . . p_n>$ of expressions – said to *occur in* P, such that there is a sequence of categories $<C_1, C_2, . . . C_n>$ —called the *code* of P, where for all $1 \leq i \leq n$,

1. $C_i \quad$ **C**

2. $p_i \quad I(C_i)$

3. If $p_i \quad \sum$, then applying a rule $s = D_1\ D_2\ . . .\ D_k \rightarrow C_i$ in **S** to a k-tuple of expressions $<e_1, e_2, . . . e_k>$, such that for all $1 \leq j \leq k$, $D_j = C_{g<i}$ and $e_j = e_g$, results in p_i. In that case, p_i occurs in P *in virtue of* s

4. If p_i Σ, then $I(C_i)$ Σ

5. $p_n = e$, and

6. $C_n = C$.

Proposition 1.6: Every expression occurring in a construction has a construction itself. ~

Definition 1.7 [proof]: If e W and I(C)=W, then the construction of e as C is a *proof*.

Definition 1.8 [tree]: Given a construction $P = <p_1, p_2, \ldots p_n>$ of code $<C_1, C_2, \ldots C_n>$ for an acceptable string w W in a grammatical theory **S**, a set of categories **C** and an interpretation function I, a *tree* of P is a labeled directed graph <**T**, < > such that:

1. For all p P, p = label(t) for some node t **T**
2. If $t_1 < t_k, t_2 < t_k, \ldots t_{k-1} < t_k$, then $s = D_1, D_2 \ldots D_{k-1} \rightarrow D_k$ is a rule in **S** such that for all $1 \le i \le k$, $\text{label}(t_i) = p_j$ and $D_i = C_j$ for some $1 \le j \le n$
3. For all t **T**, label(t) ≠ w, iff t < u for some u **T**
4. For all t_1, t_2, t_3 **T**, if $t_1 < t_2$ and $t_1 < t_3$, then $t_2 = t_3$
5. For all t **T**, label(t) Σ iff u < t for some u **T**

Definition 1.9 [occurrence of an expression]: Let **T** be the tree of a construction P, then an expression *occurs* in **T** iff it occurs in P.

Definition 1.10 [correspondance]: A proof $P = <p_1, p_2, \ldots p_n>$ *corresponds* to a tree **T** iff for all nodes $t_1 < t_2$ in **T**, there are $1 \le i \le j \le n$ such that $e_i = \text{label}(t_1)$ and $e_j = \text{label}(t_2)$.

Definition 1.11 [theorem]: An expression w is a *theorem* of a grammatical theory **S**, written $\vdash$w, iff w has a proof in **S**.

Definition 1.12 [completeness]: A grammatical theory **S**, together with a set of categories **C** and an interpretation I, is *complete for* a language **L** if all acceptable strings of **L** are theorems of **S** and conversely. In other words, **S** is complete whenever w W iff |w.

Definition 1.13 [conventional grammar]: Given a language **L**, a *conventional grammar* for **L** is a triple <**S**, I, **C**> where I models **S**, and **S,** together with **C** and I, is complete for **L**.

D. Example: The Language of Propositional Calculus

Any conventional presentation of propositional calculus syntax fits the previous definition of a conventional grammar. Take, for example, Elliot Mendelson's presentation of propositional calculus in the third edition of his *Introduction to Mathematical Logic*.[19] Displaying the syntax of propositional calculus as a language in the aforementioned form <∑, E, W> and reconstructing Mendelson's recursive definition of well-formed formula as a conventional grammar <**S**, I, **C**> is easy. Nevertheless, besides showing that the grammar corresponds to the syntax, it also illustrates the concepts defining the notion of conventional grammar.

1. The Language of Propositional Calculus

The Language of Propositional Calculus **L** is the structure <∑, E, W> where ∑ is the set of basic symbols { ¬, , (,), A_1, A_2, A_3, . . . }, W is the set of well-formed formulas of propositional calculus and E contains both the basic symbols and well-formed formulas, so that E = ∑ W.

2. The Grammar of Propositional Calculus

The following is Mendelson's definition of a well-formed formula [wff]:

[19]. Elliot Mendelson, *Introduction to Mathematical Logic* (Monterrey: Wadsworth & Brooks/Cole Advanced Books & Software, 1987)

1. The vocabulary ∑ of language **L** contains ¬, , (,), and the letters A_i with positive integers i as subscripts: A_1, A_2, A_3, ... The symbols ¬ and are *primitive connectives*, and the letters A_i are *statement letters*.

2. (a) All statement letters are well-formed formulas (wfs).

   (b) If *A* and *B* are wfs, so are (¬*A*) and (*A* *B*).

An expression is a wf only if it can be shown to be a wf on the basis of clauses (a) and (b). It is possible to reconstruct this definition as a conventional grammar the following way.

P = <**S**, I, **C**>

C = {*neg*, *arr*, *lpar*, *rpar*, *sl*, *wff*}

I(*neg*) = { ¬ }

I(*arr*) = { }

I(*lpar*) = { (}

I(*rpar*) = {) }

I(*sl*) = { A_1, A_2, A_3, . . . }

I(*wff*) = well-formed formulas

The syntax of propositional calculus contains six grammatical categories: corner (*neg*), right arrow (*arr*), open parenthesis (*lpar*), closed parenthesis (*rpar*), statement letters (*sl*) and well-formed formulas (*wff*). Also, it contains three formation rules.

S = {s_1, s_2, s_3}

s_1 = *sl* → *wff*

First, all statement letters are well-formed formulas. If a is an expression of category *sl* (a statement letter), then it is an expression of category *wff*, i.e. a well-formed formula.

s_2 = *lpar wff arr wff' rpar* → *wff*

Second, if *A* and *B* are arbitrary wfs, so is (*A* *B*). If (*i*) c and d are well-formed formulas – expressions of category *wff* – (*iii*) a is an expression of category *lpar* (an open parenthesis), (*iii*) e is an expression of category *rpar* (a closed parenthesis), and (*iv*) c is an expression of category *arr* (a right arrow), then the concatenation a{b{c{d{e is a well-formed formula itself.

s_3 = *lpar neg wff rpar* → *wff*

Third, if *A* is a wf, so is (¬*A*). If (*i*) d is an expressions of category *wff* (a well-formed formula), (*ii*) a is an expression of category *lpar* (left parenthesis), (*iii*) d is an expression of category *rpar* (right parenthesis), and (*iv*) b is an expression of category *neg* (corner), then the concatenation a{b{c{d,is an expression of category *wff*, that is, a well-formed formula itself.

Proposition 1.14: I models **S**.

Proof: Assume not. Then, a rule s **S** is not true for I. Hence, a sequence of expressions $<e_1, e_2, \ldots e_n, \ldots>$ does not satisfy s. Since **S** = $\{s_1, s_2, s_3\}$, three cases must be considered.

Case 1. s = s_1. $<e_1, e_2, \ldots e_n, \ldots>$ does not satisfy *sl* → *wff*. Hence, *sl* → *wff* applies to $<e_1>$, but the result of applying *sl* → *wff* to $<e_1>$ does not belong to I(*wff*). Since *sl* → *wff* applies to $<e_1>$, e_1 I(*sl*). I(*sl*) = $\{ A_1, A_2, A_3, \ldots A_n \}$. Therefore, e_1 is a statement letter. In other words, $e_1=A_i$ for some $1 \le i \le n$. Also, it is the result of applying *sl* → *wff* to $<e_1>$. In consequence, A_i does not belong to I(*wff*). But I(*wff*) is the set of well-formed formulas. This would mean that the statement letter A_i is not a well-formed formula, which is false.

Case 2. s = s_2. <e_1, e_2, . . . e_n, . . . > does not satisfy *lpar wff arr wff' rpar* → *wff*. Hence, *lpar wff arr wff' rpar* → *wff* applies to <e_1, e_2, e_3, e_4, e_5>, but the result of applying *lpar wff arr wff' rpar* → *wff* to <e_1, e_2, e_3, e_4, e_5> does not belong to I(*wff*). Since *lpar wff arr wff' rpar* → *wff* applies to <e_1, e_2, e_3, e_4, e_5>, e_1 I(*lpar*), e_2 I(*wff*), e_3 I(*arr*), e_4 I(*wff*) and e_5 I(*rpar*). Therefore, e_1 is an open parenthesis, e_2 is a well-formed formula, e_3 is the right arrow, e_4 is also a well-formed formula and e_5 is the closed parenthesis. The result of applying *lpar wff arr wff' rpar* → *wff* to < e_1, e_2, e_3, e_4, e_5> is a sequence of the form (A B) where A and B are wfs. Hence, it does not belong to I(*wff*). But I(*wff*) is the set of well-formed formulas. This would mean that (A B) is not a well-formed formula, which is false.

Case 3. s = s_3. <e_1, e_2, . . . e_n, . . . > does not satisfy *lpar neg wff rpar* → *wff*. In consequence, *lpar neg wff rpar* → *wff* applies to <e_1, e_2, e_3, e_4>, but the result of *lpar neg wff rpar* → *wff* to <e_1, e_2, e_3, e_4> does not belong to I(*wff*). Since *lpar neg wff rpar* → *wff* applies to <e_1, e_2, e_3, e_4>, e_1 I(*lpar*), e_2 I(*neg*), e_3 I(*wff*) and e_4 I(*rpar*). Hence, e_1 is an open parenthesis, e_2 is the corner, e_3 a well-formed formula and e_4 is the closed parenthesis. The result of applying *lpar neg wff rpar* → *wff* to < e_1, e_2, e_3, e_4> is a sequence of the form (¬A) where A is a wf. Thus, it does not belong to I(*wff*). But I(*wff*) is the set of well-formed formulas. This would mean that (¬A) is not a well-formed formula, which is false.

Therefore, I models **S** for the language of propositional calculus **L**. ~

Proposition 1.15: All acceptable strings w W have a proof.

Proof: Let w W.

Induction on the complexity of w.

Base: w is a statement letter.

w $\sum$ and w I(*sl*). Hence, <w> is the construction of w with code <*sl*>. Let w be a wf of complexity n.

Inductive hypothesis: all wfs in **L** of complexity m<n have a construction in **P**.

Case 1. w is of the form ¬A.

Since A's complexity is less than n, the inductive induction applies to it. There is a proof $P_A = < e_1, e_2, e_3, \ldots e_n >$ of A with code $C_A = < C_1, C_2, C_3, \ldots C_n, >$.

Claim: $P_w = < e_1, e_2, e_3, \ldots e_n, \neg, w >$ is a proof of w with code $C_w = < C_1, C_2, C_3, \ldots C_n,$ *neg, wff* >.

Proof: Let e_i be an expression in $P_{w.}$

Case 1: $i \leq$ n. e_i P_A. Hence, either e_i $\sum$ or e_i decomposes into some of the previous expressions in the sequence.

Case 2. e_i = ¬. Hence, e_i $\sum$ and e_i *neg.*

Case 3. e_i = w. Since P_A is a proof of A, e_n=A. From s_3, w decomposes into ¬ and A, both occurring before w in P_w. Therefore, P_W is a proof of w.

Case 2. w is of the form A B.

Since A and B are of complexity less than n, the inductive induction applies to them. There is a proof $P_A = < e_1, e_2, e_3, \ldots e_n >$ of A with code $C_A = < C_1, C_2, C_3, \ldots C_n, >$ and a proof $P_B = < e_{n+1}, e_{n+2}, e_{n+3}, \ldots e_{n+m} >$ of B with code $C_B = < C_{n+1}, C_{n+2}, C_{n+3}, \ldots C_{n+m}, >$.

Claim: $P_w = < e_1, e_2, e_3, \ldots e_n, e_{n+1}, \ldots e_{n+m},$, w > is a proof of w with code $C_w = < C_1, C_2, C_3, \ldots C_n, C_{n+1}, \ldots C_{n+m},$ *arr, wff* >.

Proof: Let e_i be an expression in $P_{w.}$

Case 1: $i \leq n$. e_i P_A. Hence, either e_i $\sum$ or e_i decomposes into some of the previous expressions in the sequence.

Case 2: $n < i \leq n+m$. e_i P_B. Hence, either e_i $\sum$ or e_i decomposes into some of the previous expressions in the sequence.

Case 3. e_i = . Hence, e_i $\sum$ and e_i *arr*.

Case 4. e_i = w. Since P_A is a proof of A, e_n= A. Since P_B is a proof of B, e_{n+m}= B. From s_2, w decomposes into ¬, A and B, occurring before w in P_w. Therefore, P_W is a proof of w.

~

Proposition 1.16: The structure **P** = <**S**, I, **C**> is a conventional grammar.

Proof: Directly from the previous two propositions 1.14 and 1.15. ~

E. Strong, Redundant and Trivial Grammars

Section C presented the minimal requirements for an abstract grammar. This section explores two special kinds of abstract grammars: strong and trivial. A strong grammar contains constructions for every expression as every category it belongs to. The syntax of first order logic is a clear example of a strong grammar, because its only categories are ‘well-formed formula’ and ‘basic symbols’.

The grammatical categories of a trivial grammar are all singletons of expressions. As their names suggest, trivial and strong grammars represent the two extremes of grammatical expressibility. A strong grammar’s categorical distinctions are the finest possible, while those of a trivial grammar are the weakest. Any grammatical distinction not in a strong grammar is superfluous. Any grammar containing superfluous distinctions is redundant.

1. Strong Grammars

Definition 1.17 [strong grammar]: A conventional grammar <**S**, I, **C**> is *strong* if, for every expression e E and every category C **C**, if e I(C), then e has a construction as C.

Example 1.17.1: The above grammar **P** of the syntax of propositional calculus is strong.

Proof: Let e E. Since E=W ∑, either e W or e ∑. In the first case, where e ∑, if e C <e> constructs e as C with code <C>. In the second case, where e W, from proposition 1.13, e has a proof P_e. Also, P_e constructs e as *wff.* Except for statement letters, which belong to ∑, no wff belongs to any other category except *wff.* This proves that every expression in **L** has a construction in **P**. In other words, **P** is strong.

Proposition 1.18: For every language such that E = W ∑, every conventional grammar is strong. ~

Recognizing which grammatical distinctions give rise to a language's significant syntactic features is critical. Imagine a grammar for the syntax of propositional calculus containing, instead of a category for statement letters SLET = { A_i | i N }, two categories ODSL = {A_i | i N and i is odd } and EVSL = {A_i | i N and i is even }. It is easy to imagine how, instead of a rule saying all propositional symbols are well-formed formulas, the grammar had two such rules: one for each of the categories ODSL and EVSL. To an extent, this distinction is superfluous, if compared with the distinction between the negation and open parenthesis symbols. Consider an English grammar which not only distinguishes between adjectives and adverbs but also between nouns which contain more than three occurrences of the letter 'e' and nouns which do not. Clearly, this latter distinction does not have the grammatical significance of the former. The grammar resulting from the collapse of two categories with superfluous distinctions neither stops being a grammar nor loses its strength.

2. Redundant Grammars

Definition 1.19 [redundant grammar]: Let <**S**, I, **C**> be a conventional grammar for language **L** such that A, B **C**. Let AB be a categorial symbol not in **C**. Let **C*** be (**C** - {A, B}) {AB}. Let **S*** be the grammatical theory resulting from substituting every occurrence of A or B by AB, and let I* be the function such that for all C **C**, I*(C) = I(A) I(B) if C = AB, and I*(C) = I(C), otherwise. Grammar <**S**, I, **C**> is *redundant.* if the following to conditions hold: (*i*) <**S***, I*, C*> is also a conventional grammar for language **L**, and (*ii*) if <**S**, I, **C**> is strong, so is <**S***, I*, **C***>.

Example 1.19.1: The above grammar **P** for the syntax of propositional calculus is not redundant.

Proof: Proving the non-redundancy of **P** requires demonstrating that the collapse of any two categories in **C** affects the conventionality or strength of **P**. This seems false, because it is possible to take a well-formed-formula and substitute one of its component expressions for an expression not of the same category, to obtain a new-well-formed formula. This is possible by substituting molecular subformulas for atomic ones, like substituting 'A_3' for '(A_1 A_2)' in '¬ (A_1 A_2)' to obtain '¬ A_3' (claim 1). This is the only collapse of categories that respects truth (claim 3). However, this substitution does not respect the completeness of **P** (claim 2). **P*** is not complete for **L**, because it lacks proofs for every well-fromed formula. In particular, it offers no proof for single letters being well-formed. Therefore, **P*** is not a conventional grammar for **L**.

Slorwff is a categorical symbol not in **C**. Let **C*** be (**C** - {*sl*, *wff*}) {*slorwff*}. Let **S*** be the grammatical theory resulting from substituting every occurrence of *sl* or *wff* by *slorwff*, and let I* be the function such that for all C   **C**, I*(C) = I(*sl*)   I(*wff*) if C = *slorwff*, and I*(C) = I(C), otherwise.

Claim 1. **P*** = <**S***, I*, **C***> models the language of propositional calculus **L**. In other words, every rule in **S*** is true for **L**.

Proof: Every statement letter by itself is also a well-formed formula: I(*sl*)   I(*wff*). Hence, I(*slorwff*) = I(*wff*)   I(*sl*) = I(*wff*). Hence, the substitution of *wff* for *slorwff* does not affect the interpretation of the rule. Similarly, every occurrence of *sl* or *wff* in S could stand for *slorwff* without affecting the truth of the theory. First, $s_1 = sl \rightarrow wff$ is the only rule in which *sl* occurs. Hence, it is the only rule in **S** that is significantly transformed in **S***. Substituting *slorwff* from s_1 for every occurrence of *sl* results in the rule *slorwff* → *slorwff*, but obviously, $\div_{I}\, wff \rightarrow wff$. Therefore, $\div_{I^*}\, slorwff \rightarrow slorwff$. End of proof for claim 1.

Claim 2. The grammar **P*** = <**S***, I*, **C***> resulting from collapsing the categories *wff* and *sl* into *slorwff*, is not complete.

Proof: No proof for statement letters as well-formed formulas would exist. Remember that this definition of proof includes the condition that every basic symbol in a proof must belongc to a category whose interpretation includes only basic symbols. The syntax of propositional calculus satisfies this condition, precisely because a separate category exists for well-formed formulas which are also basic symbols – statement letters. In other words, I(*sl*)   ∑. However, without *sl*, this is no longer true. End of proof of claim 2.

Claim 3. Let $C_1 or C_2$ be the category resulting from collapsing different categories C_1 and C_2 in **C** such that C_1 is neither *sll* nor *wff* , and let **P*** = <**S***, I*, **C***> be the grammar constructed according to definition 1.19. At least one rule s* in S is false for **L**.

Proof: Case 1. C_2 is *sl*. Since s_1 = *sl* → *wff* S, s_1* = C_1*orsl* → *wff* S*. Since C_1 is not *wff*, s* is false for **L**. In the vocabulary of propositional calculus, only single letters are well-formed formulas. Case 2. C_2 is *wff*. Since s_1 = *sl* → *wff* S, s_1* = *sl* → C_1*orwff* S*. Since C_1 is not *sl*, s* is false for **L**. Substituting a single letter for another symbol in the vocabulary of propositional calculus in a well-formed formula results in a non-well-formed expression. Case 3. C_1 and C_2 are *neg* and *arrow*. Since s_2 = *lpar wff arr wff' rpar* → *wff* S, s_2* = *lpar wff anegorrr wff' rpar* → *wff* S*, but s_2* is false for **L**. Since negation and arrow have a different n-arity, they cannot substitute for each other in a well-formed formula. Notice that if the language included, besides negation and implication, symbols for other propositional operators, it would not be necessary to include a new grammatical category for each one. They would be grouped by their n-arity. For example, in the syntax of such an extended language for propositional grammar, ‘ ’ $\sim_G$ ‘ ’ if G is not redundant. Symbols of the same n-arity can substitute for each other, without affecting their truth or construction. Finally, case 4, C_1 is *lpar* or *rpar*. Since s_2 = *lpar wff arr wff' rpar* → *wff* S, s_2* = C_1*orlpar wff anegorrr wff' rpar* → *wff* S*, but s_2* is false for **L**. Also, since s_2 = *lpar wff arr wff' rpar* → *wff* S, s_2* = *lpar wff anegorrr wff'* C_1*orlrar* → *wff* S*, but s_2* is false for **L**. End of claim 3. ~

3. Trivial Grammars

Definition 1.21 [trivial grammar]: Let **S** = <S, I, C> be a conventional grammar. If e, f E ($e \sim_s f$) (e = f), then **S** is *trivial*.

Theorem 1.23: Every language **L** has a trivial grammar.

Proof: To easily create a trivial grammar for a language, construct for every expression in the language a unique category whose interpretation is its singleton. This specifies that for every expression e E, [e] = {e}. For the grammatical rules, constructing a rule for the decomposition of every expression into its basic words is sufficient. For example, for the expression ‘second world war’, construct the rule SECOND WORLD WAR → SECOND-WORLD-WAR, where the only expression in the category SECOND is ‘second’, the only expression in WORLD is ‘world’, the only expression in the category WAR is ‘war’ and the only expression in the category SECOND-WORLD-WAR is ‘second world war’. The only category which may include more than one expression is the category of acceptable string. Hence, the acceptable string ‘The Ocean is Deep’ only needs the inclusion in the grammatical theory of the rule THE OCEAN IS DEEP → WFF. It is straightforward to see that this method has application in any language. ~

III. Wittgenstein’s Approach

In Wittgenstein’s grammatical method, categories do not depend directly on the rules for building acceptable strings, but proceed from given acceptable strings through allowable substitutions. Two expressions belong to the same grammatical category if they can substitute for each other without affecting the grammar of the expression. Nevertheless, Wittgenstein’s writing is ambiguous as to whether the substitution of expressions belonging to the same category must respect acceptability or all the grammatical categories. The following pages explore the first interpretation, where the grammar of a word determines the words which can substitute for it preserving acceptability. For the rest of this chapter, two expressions are Wittgenstein-equivalent if the substitution for each other preserves grammatical correctness in any context.[20] Substituting some or all of the occurrences of an expression in an acceptable string by another one with the same grammar results in an acceptable expression.

Definition 2.1 [context]: Given a language <Σ, E, W>, let expression e occur in acceptable string w. Omitting an occurrence of e from w and leaving blanks in its place produces an incomplete string called a *context*. A context is not a complete expression, because it includes blank spaces.

Definition 2.2 [Wittgenstein category]: Let C be a context, and let function A assigns to each expression e E the string resulting from placing e in the blanks of C. Define the associated *Wittgenstein category* of the context as B = { e E | A(e) W }, also expressed as B = λ*x* A(*x*).

Definition 2.3 [Wittgenstein-grammatical equivalence]: Let **A** be the set of Wittgenstein categories of the language, containing every category associated with any context in the language (resulting from W and E). Given **A**, for all e, f E, (e $\sim_w$ f) iff {C **A** | e C} = {C **A** | f C}. This defines the relation of *Wittgenstein-grammatical equivalence* among expressions. Thus, the above relation's equivalence classes are the Wittgenstein-grammatical categories.

$$[e]_A = \{ A \quad \mathbf{A} \mid e \quad A \}$$

Proposition 2.4: For all e, f E, {(e$\sim_A$ f) iff A **A** [*A*(e) W iff *A*(f) W]}. ~

20. BT §9, p.34.

Proposition 2.5: $\sim_A$ is an equivalence relation on E. ~

IV. A Comparison between the two Approaches

After defining both approaches, it is crucial to determine whether they yield different grammatical categories. This section shows that it is possible to construct a conventional grammar out of Wittgenstein's categories. The rest of this section compares this sort of grammar with other more traditional ones. Most of all, it shows how the distinctions resulting from Wittgenstein's approach to grammar are less fine than the conventional ones.

A. Wittgenstein Grammar

Definition 3.1 [Wittgenstein Grammar and Pure Wittgenstein Grammar]: Let **G** = <**C**, **I**, **S**> be a conventional grammar for a language **L** = <Σ, I, C>such that for every pair of expressions e, f E, $e\sim_G f$ iff $e\sim_A f$, where **A** is the set of Wittgenstein categories for **L**. Call **G** a *Wittgenstein* grammar for **L**. Let **S** be a conventional grammar for language **L** such that I[**C**]=**A**, where **A** is the set of Wittgenstein categories for **L**, then **G** is a *pure Wittgenstein grammar* for **L**.

A conventional grammar bears Wittgenstein's name if it respects the notion of Wittgenstein equivalence. It is a *pure Wittgenstein grammar* if the categories interpretations are the language's Wittgenstein categories.

Proposition 3.2: Every pure Wittgenstein grammar is a Wittgenstein grammar. ~

Proposition 3.3: A conventional grammar **G** is a Wittgenstein grammar iff e, f E, $e\sim_G f$ iff $e\sim_w f$, where **W** is the set of Wittgenstein categories. ~

Theorem 3.4: Not every language has a pure Wittgenstein grammar.

Proof: Consider the following language **L**:

$\Sigma = \{a, b, c\}$

$W = \{ab, ac, ba, bc, ca, cb\}$

$E = \Sigma \quad W$

The contexts of the language, with its corresponding Wittgenstein categories, are

A = {b, c} corresponding to contexts $\lambda x\ (x\mathrm{a})$ and $\lambda x\ (\mathrm{a}x)$,

B = {a, c} corresponding to contexts $\lambda x\ (x\mathrm{b})$ and $\lambda x\ (\mathrm{b}x)$,

C = {a, b} corresponding to contexts $\lambda x\ (x\mathrm{c})$ and $\lambda x\ (\mathrm{c}x)$,

and W.

Assume towards a contradiction, that **L** has a pure Wittgenstein grammar <**S**, I, **C**>. $A_1\ A_2 \rightarrow W$ is a rule s **S**, where A_1, A_2 and W **A**. Unfortunately, every combination of two categories in **A** yields a string of symbols not in W. A and B produce ‘cc’, A and C yield ‘bb’ and B and C make ‘aa’. Hence, it is impossible that s be true in **L**. ~

The method for producing trivial grammars from Theorem 1.23 generates Wittgenstein grammars for any language. Considering Wittgenstein equivalence classes as grammatical categories guarantees that for every expression e E, $e \sim_G f$ iff $e \sim_w f$. Constructing a rule for every decomposition of every acceptable string produces a true grammatical theory. For example, if the grammar contains the acceptable string ‘rabbits run wild’, introduce RABBITS, RUN and WILD as categories and RABBITS RUN WILD $\rightarrow$ WFF as a rule. Assign the interpretation $[\text{rabbit}]_w$ to category RABBITS, the interpretation $[\text{run}]_w$ to RUN, $[\text{wild}]_w$ to WILD, and the set of acceptable strings in the language to WFF.

When using this method to construct a trivial Wittgenstein grammar, including decomposition into basic words, as well as into other complex expressions, is essential. For example, if the language includes ‘run wild’ as a complex expression, add to the grammatical theory the rule RABBITS RUN-WILD → WFF, where the equivalence class $[\text{run wild}]_w$ interprets the category RUN-WILD. Otherwise, the rules might not use all the Wittgenstein categories.

Clearly, this method applies to any language. However, proving that every language has a Wittgenstein grammar requires the following lemma:

Lemma 3.5: Given a language **L**, let e and f be a pair of expressions of the language such that f occurs in e. Let g be the string resulting from substituting every occurrence of f in e by h. For any strong grammar **S** for the language **L**, if $f \sim_s h$, then $e \sim_s g$.

Proof: Assume, towards a contradiction, that $f \sim_s h$, but not $e \sim_s g$. Then, e   I(C) and g   I(C) for some conventional grammatical category C   **C**. Since **S** is strong, there is a construction P of e as C. Let P = $<p_1, p_2, \ldots p_n>$ and let $<C_1, C_2, \ldots C_n>$ be a code for it. For every expression p_i occurring in P, construct q_i substituting all occurrences (if any) of f in p_i by h.

Claim: For all i≤n, q_i $I(C_i)$.

Proof of claim by induction on i.

Base: i=1. Hence, p_1 ∑. Since no word in the vocabulary occurs in another simple word, either p_1 = f or not. In the first case, if $p_1 \neq f$, then $p_1 = q_1$. Since p_1 $I(C_1)$, then q_1 $I(C_1)$. In the second case, if p_1 = f, then q_1 = g. Also, C_i = C. Therefore, since $f \sim_s h$, q_1 $I(C_i)$. In either case, q_1 $I(C_i)$.

Inductive hypothesis: Assume that, for all j<i, q_j $I(C_j)$, to prove that q_i $I(C_i)$.

Either p_i $\sum$ or not. In the first case, $p_i = f \sim_s h = q_i$, so it reduces to the base case. In the second case, some rule $s = D_1\ D_2\ .\ .\ .\ D_m \rightarrow C_1$ in **S** decomposes p_1 into some previous expressions in P. In other words, p_1 is the result of applying s to a sequence of expressions $<e_1, e_2, .\ .\ .\ e_m>$ such that for all $1 \leq g \leq m$, $e_g = p_k$ and $D_g = C_k$ for some k<i . This also means that every expression e_g in $<e_1, e_2, .\ .\ .\ e_m>$ occurs in P somewhere before p_i. Because of this, the inductive hypothesis applies to them. In consequence, for all $1 \leq g \leq m$, q_k $I(C_k)$ for some k<j . In this case, let $e_g' = q_k$. Since C_k is the category of term D_g in s, rule s applies to $<e_1', e_2', .\ .\ .\ e_m'>$. Since the only difference between e_g and e_g' is the substitution of f for h, q_i is the result of applying s to $<e_1', e_2', .\ .\ .\ e_m'>$. Finally, since C_i is the resulting category of s, q_i also belongs to the interpretation of C_i. End of proof of claim.

From the claim, for all i≤n, q_i $I(C_i)$. In particular, p_n $I(C_n)$. However, $p_n = g$ and $C_n = C$, so g $I(C)$, which contradicts the hypothesis. ~

Corollary 3.6: Let $e = e_1 \{ e_2 \{ .\ .\ . \{ e_n$, and $e' = e_1' \{ e_2' \{ .\ .\ . \{ e_n'$. Given any strong grammar **S**, if, for all 1≤i≤n $e_i \sim_s e_i'$ then $e \sim_s e'$. ~

Corollary 3.7: Let e be an acceptable string of a language **L** such that another expression f occurs in e. Let g be the string resulting from substituting every occurrence of f in e by h. For any conventional grammar **S** for the language **L**, if $f \sim_s h$, then g is acceptable too. ~

Corollary 3.8: Given a language < ∑, E, W >, let e = $e_1\{e_2\{. . .\{e_n$ W, and e'= $e_1'\{e_2'\{. . .\{e_n'$ E. Given any conventional grammar **S**, if, for all 1≤i≤n $e_i \sim_s e_i'$ then e'   W. ~

Theorem 3.9: Every language **L** has a Wittgenstein grammar.

Proof: Let **L** = <Σ, E, W> be a language. Let **C** be the set of Wittgenstein-equivalence classes. **C** = { $[e]_w$ | e   E }. Let I be the identity function. Let **D** be the set of tuples of (more than one) expressions whose concatenation is also an expression of the language. In other words, for all $e_1, e_2, e_3, . . . e_n$ E; <$e_1, e_2, e_3, . . . e_n$>   **D** iff $e_1\{e_2\{e_3\{ . . . \{e_n$ E, where n>1. Let **S** = { $[e_1]_w [e_2]_w [e_3]_w . . . [e_n]_w \rightarrow [e_1\{e_2\{e_3\{ . . . \{e_n]_w$ | <$e_1, e_2, e_3, . . . e_n$>   D}.

Claim 1:   e   E, $e \sim_s f$ iff $e \sim_w f$.

Proof: Assume $e \sim_s f$ to prove $e \sim_w f$. Since $e \sim_S e$, f must also belong to $[e]_w$. This means that $e \sim_w f$. For the converse, assume e is not grammatically equivalent in **S** to f to show that they are not Wittgenstein equivalent either. Without loosing generality, f $[e]_w$. But, in any case, f $[f]_w$. Hence, $[e]_w \neq [f]_w$. In other words, it is false that $e \sim_w f$.

Claim 2: <**S**, I, **C**> is a grammar for **L**.

Proof: Assume not. <**S**, I, **C**> may not be a grammar only if I does not model **S**. Then, some rule s = $[e_1]_w [e_2]_w [e_3]_w . . . [e_n]_w \rightarrow [e_1\{e_2\{e_3\{ . . . \{e_n]_w$ **S** is not true for the language **L**. An n-tuple of language expressions <$e_1', e_2', e_3', . . . e_n'$> does not satisfy s. Even though, for all i≤n, e_i' $[e_i]_w$, $e_1'\{e_2'\{e_3'\{ . . . \{e_n'$ $[e_1\{e_2\{e_3\{ . . . \{e_n]_w$.

Nevertheless, from Corollary 3.8, since $e_1' \{ e_2' \{ e_3' \{ \ . \ . \ . \ \{ e_n'$ W and for all i≤n, $e_i' \sim_w e_i$, it must also be the case that $e_1 \{ e_2 \{ e_3 \{ \ . \ . \ . \ \{ e_n$ W. This contradiction proves that <**S**, I, **C**> is a grammar for **L**. Since e E, $e \sim_s f$ iff $e \sim_w f$, <**S**, I, **C**> is also a Wittgenstein grammar. ~

B. Formal Requirements for a Wittgenstein Grammar

The previous section demonstrated the existence of Wittgenstein grammars. This section displays the formal constrains on Wittgenstein grammars. It compares traditional grammatical categories with Wittgenstein's. It reveals the sort of grammars for which Wittgenstein's approach produces finer (or at least as fine) categorical distinctions. The traditional relation of grammatical equivalence does not always match that of Wittgenstein equivalence. It is false that, for every language and every grammar $e \sim_s f$ iff $e \sim_w f$. This sections shows the structural features of the grammar falsifying the double implication. It describes the sort of grammars where Wittgenstein equivalence implies grammatical equivalence, and vice versa.

Definition 3.10 [normal grammar]: A conventional grammar **<S**, I, **C**> is *normal* for a language <∑, E, W> iff, for every expression e E and every acceptable string w W, if e occurs in w, e occurs in a tree for w in <**S**, I, **C**> as many times as e occurs in w.

Lemma 3.11: Let A be a context in a normal language **L** = <∑, E, W>. Given a strong conventional grammar <**S**, I, **C**>, for every pair of acceptable strings *A*(e) and *A*(f), there is a pair of trees $\mathbf{T}_e$ and $\mathbf{T}_f$ isomorphic down to a node t_e such that for all t ≥ e $\mathbf{T}_e$ there is a t' $\mathbf{T}_f$ such that (i) label(t) results from label(t') when substituting f for e once at most such that (ii) the category of label(t) in the proof corresponding to $\mathbf{T}_e$ is the same as the category of label(t') in the proof corresponding to $\mathbf{T}_f$.

Proof: Since **L** is a normal language, there is a tree $\mathbf{T}_e$ of *A*(e) such that e occurs as many times in $\mathbf{T}_e$ as e occurs in *A*(e). Let P_e be the proof corresponding to tree $\mathbf{T}_e$. First, define the function Rule : $\mathbf{T}_e \rightarrow \mathbf{S}$ mapping every node t in $\mathbf{T}_e$ to the rule in the grammatical theory **S** such that label(t) occurs in P_e in virtue of s and the function Cat : $\mathbf{T}_e \rightarrow \mathbf{S}$ maps every node t in $\mathbf{T}_e$ to the corresponding category in the code of P_e.

Let t_e be a node in $\mathbf{T}_e$ such that label(t_e) = e. Let P_f be the construction of f as Cat(t_e), and $\mathbf{T'}_f$ be its tree, with $<'_f$ as its ordering relation and label' as its labeling function. Since <**S**, I, **C**> is strong, this construction exists. Let $\mathbf{T}_f = \mathbf{T'}_f \quad \mathbf{T}_e$. Define function f:$\mathbf{T}_f \rightarrow$ E the following way: (i) f(t) = label'(t) if t $\quad \mathbf{T'}_f$, (ii) f(t) = label(t) if t does not belong to $\mathbf{T'}_f$ and e does not occur in label(t), (iii) otherwise, recursively define $f(t_e)$ = f and for all t > t_e let t be such node in $\mathbf{T}_e$ that e occurs in label(t) and let u be the least unique node in $\mathbf{T}_e$ such that t < u. Label(u) is the result of applying Rule(u) to an n-tuple of expressions $<e_1, e_2, e_3, \ldots e_n>$, such that, for every expression e' $\quad$ E, e' = label(t') for some t' in $\mathbf{T}_e$ and u is the least node such that t' < u iff e' = e_i for some i ≤ n. Since u is the least node such that t < u, label(t) occurs in the n-tuple $<e_1, e_2, \ldots$ label(t), $\ldots e_n>$. However, label(t) may well occur more than once in the n-tuple. Since Rule(u) is of the form $C_1\ C_2 \ldots C_n \rightarrow C$, there is an i ≤ n such that C_i = Cat(t) and e_i = label(t). Actually, more than one may satisfy these two conditions. Which one the proof uses makes no difference.

Now, since f I(Cat(t)) = I(C_i), Rule(u) applies to the n-tuple <e_1, e_2, . . . e_{i-1}, *f*(label(t)), e_{i+1}, . . . e_n> resulting from substituting label(t) by *f*(label(t)) in the ith place. Hence $e_1 \{ e_2 \{ . . . e_{i-1} \{$ *f*(label(t))$\{ e_{i+1}, . . . \{ e_n$ I(Cat(u)). Then, let *f*($e_1 \{ e_2 \{ . . . e_{i-1} \{$ label(t)$\{ e_{i+1}, . . . \{ e_n$) = $e_1 \{ e_2 \{ . . . e_{i-1} \{$*f*(label(t)) $\{ e_{i+1}, . . . \{ e_{n.}$ Since label(t) and *f*(label(t))differ only in one substitution of e by f, so do ($e_1 \{ e_2 \{ . . . e_{i-1} \{$ label(t) $\{ e_{i+1}, . . . \{ e_n$) and *f*($e_1 \{ e_2 \{ . . . e_{i-1} \{$ label(t) $\{ e_{i+1}, . . . \{ e_n$).

Finally, consider the tree < $\mathbf{T}_f$, $<_f$ > where $<_f$ = $<'_f$ < |~(t < t_e) {(top($\mathbf{T}'_f$), t_e)} is the ordering relation, and *f*(label(t)) labels every node t in $\mathbf{T}_e$. Since *f*(label(t)) I(Cat(t)) for all t in $\mathbf{T}_e$, the result of re-labeling $\mathbf{T}_e$ is also a tree. It is a tree proving that *f*(*A*(e)) is acceptable. Since *f*(*A*(e)) differs from *A*(e) in one substitution of f for e, *A*(e) = *A*'(e) and *f*(*A*(e)) = *A*'(f) for some context A'. The posibilities of repeating this process is the same as the number of times e occurs in *A*(e). Each one provides a different context A'. Also, since erasing one time e occurs in *A*(e) produces every context, this is the same number of contexts A' for which *A'*(e) = *A*(e). In particular, A' must equal A. ~

Theorem 3.12: If a strong conventional grammar **G** is normal for a language **L** = <∑, E, W>, then for all e, f E, (e $\sim_G$ f) (e $\sim_w$ f).

Proof: Let **G** = <**S**, I, **C**> be a strong conventional grammar normal for language <∑, E, W>. Assume, towards a contradiction, that e $\sim_G$ f but not e $\sim_w$ f. This implies that e and f belong to all the same categories in **C**, but not in **A**. Without loss of generality, e A, and f A for some Wittgenstein category A **A**. In consequence, *A*(e) W but *A*(f) W. Now, since *A*(e) W and **G** is normal, e occurs in the tree T of a proof P of *A*(e) as many times as e occurs in *A*(e) itself. Furthermore, by lemma 3.11, it is possible to substitute e for f only the e in *A*(e) yielding context A. This strategy creates a tree and proof of *A*(f). Nevertheless, this would mean that *A*(f) is an acceptable string, contradicting the hypothesis that *A*(f)   W. ~

However, an ambiguity remains in Wittgenstein's thesis saying two expressions belong to the same grammatical category, if they can stand in place of each other in some context without affecting the grammar of the original expression. The preceding section dealt only with one possible interpretation of this thesis. In such an interpretation, contexts result from well-formed sentences. Accordingly, the substitution of grammatically equivalent expressions preserves the acceptability of statements. Still, another interpretation is possible. A stronger relationship of grammatical equivalence results from allowing the production of contexts from any grammatical phrase, instead of only full sentences. The resulting relation of equivalence is stronger, because it respects all the grammatical categories. In contrast, this chapter's approach respected only the category of well-formed sentence. Under this alternative interpretation, two expressions would be grammatically equivalent if they could substitute for each other within any expression without affecting the expression's grammatical categories. Substituting some or all the times an expression occurs in any well-formed expression – not necessarily a complete sentence – with a synonymous expression must result in an expression belonging to exactly the same categories as the original. The results from this sections are bound by considering only the first interpretation. Generalizing these results would require performing a more thorough investigation into the relation between these grammars and natural language.

V. Conclusion

Two final conclusions summarize the results of this chapter. First, it is not possible to construct conventional grammars for every language out of the categories resulting from Wittgenstein's analysis. Nevertheless, it is always possible to construct a 'Wittgenstein grammar' where grammatical equivalence corresponds to Wittgenstein equivalence. This justifies calling 'grammar' whatever results from the kind of analysis of substitutions Wittgenstein proposes. On the other hand, this sort of grammar does more than satisfy the minimal expectations for a grammar. Wittgenstein grammars are not always strong. They do not provide enough information to construct all acceptable expressions out of the basic words in the language. In this sense, they are weaker than other conventional grammars. Nevertheless, they are not the weakest. In general, Wittgenstein's grammatical distinctions are neither the most specific nor the most general.

Wittgenstein's philosophy provides no straightforward interpretation of these formal results. It is tempting to dismiss Wittgenstein's notion of grammar as too weak to play the central role he expects it to.

This chapter formalized some of Wittgenstein's intuitions about grammar and drew several philosophical conclusions from them. Now, it is time to place them in the bigger picture of Wittgenstein's philosophy of mathematics. The following chapter builds on the results of this analysis. It applies the previous formal reconstruction to a portion of language containing numerical expressions. It proves that the grammatical theory resulting from Wittgenstein's approach contains expressions whose natural interpretation is mathematical. It shows that arithmetical rules regulate the use of numerical expressions in natural language.