Chapter 1

INTRODUCTION

This chapter is devoted to the presentation of basic definitions and notation to be used in this monograph. The definitions fall under three general headings; those related to computable partial functionals and computably enumerable sets, those related to the computably enumerable degrees, and those related to trees.

1.1. Computably Enumerable Sets

Let $\mathbb{N}$ be the natural numbers, i.e., the set of integers $\{0, 1, 2, \dots\}$. We use interval notation on $\mathbb{N}$; thus $[k, m] = \{n : k \leq n \leq m\}$. Open interval notation and half-open interval notation is used in a similar fashion. The direct sum of two subsets A and B of $\mathbb{N}$ is denoted as $A \oplus B$ and is defined as $\{2x : x \in A\} \cup \{2x + 1 : x \in B\}$. $|A|$ will denote the cardinality of the set A.

If $A \subset \mathbb{N}$, $m \in \mathbb{N}$, and Φ is a partial functional, then we write $\Phi(A; m) \downarrow$ if m is in the domain of $\Phi(A)$, and $\Phi(A; m) \uparrow$ otherwise. If Φ and Ψ are partial functionals and A and B are sets, then we write $\Phi(A) \simeq \Psi(B)$ if $\Phi(A)$ and $\Psi(B)$ are *compatible*, i.e., for all x, if $\Phi(A; x) \downarrow$ and $\Psi(B; x) \downarrow$, then $\Phi(A; x) = \Psi(B; x)$; and we write $\Phi(A) = \Psi(B)$ if $\Phi(A)$ and $\Psi(B)$ are *identical*, i.e., $\Phi(A)$ and $\Psi(B)$ are compatible and for all x, $\Phi(A) \downarrow$ iff $\Psi(B) \downarrow$.

Intuitively, a *computable partial functional* Φ is one for which one can write a computer program using an arbitrary oracle A (i.e., which allows instructions of the form: "If $n \in A$ then go to line r") such that for any $m, k \in \mathbb{N}$, $\Phi(A; m) = k$ if and only if the program produces output k when given input m, and $\Phi(A; x) \uparrow$ if the program fails to halt on input m. A *computably enumerable set* is the domain of a computable partial function $\Phi(\emptyset)$, and B is *computably enumerable in* A if there is a computable partial functional Φ such that B is the domain of $\Phi(A)$. If Φ is a computable partial functional and $\Phi(A; m) \downarrow = k$, then the computer program provides a computation for the *axiom* $\Phi(A; m) \downarrow = k$. A is said to be the *oracle*, m the *argument*, and k the *value* for this axiom (or computation). As the computation halts, only finitely many questions of the form "Is $n \in A$?" are asked; the *use* of the axiom (or computation) is the largest such n.

If an oracle has the form $A \oplus B$, then we write $\Phi(A, B)$ in place of $\Phi(A \oplus B)$. In this case, we allow different uses for A and B in an axiom. If only a single use is specified, then we adopt the convention that this is the common use of A and B. $A \upharpoonright m+1$ is the subset of $[0, m]$ which satisfies $(A \upharpoonright m+1)(k) = A(k)$ for all $k \leq m$. Note that if an axiom has oracle A and use m, then it produces the same computation as would be produced from oracle $A \upharpoonright m+1$; thus we may allow finite sets of the form $A \upharpoonright m+1$ as oracles for axioms.

If Φ is a partial functional with oracle A, then we say that $\lim_s \Phi(A; x, s) = m$ if $\Phi(A; x, s) \downarrow = m$ for all sufficiently large s. We say that $\lim_s \Phi(A)$ is *well-defined* if for all x, either there is an m such that $\lim_s \Phi(A; x, s) = m$, or $\Phi(A; x, s) \uparrow$ for all sufficiently large s.

It was shown by Kleene [5] that there is an effective enumeration $\{\Phi_i : i \in \mathbb{N}\}$ of all computable partial functionals of a fixed number of integer variables; and there is a computable one-to-one function mapping sequences of integers to $\mathbb{N}$. Thus there is no loss of generality in assuming that each such functional has only one integer argument. (We will, however, use functionals with multiple oracles and arguments, and note that the above enumeration uniformly induces effective enumerations of all computable partial functionals with fixed numbers of set and integer variables.) Similarly, there is a computable enumeration $\{W_i : i \in \mathbb{N}\}$ of all computably enumerable sets. Furthermore, there is a computable sequence of approximations to the computable enumeration of functionals or sets, i.e., an array $\{W_i^s : i, s \in \mathbb{N}\}$ such that for all i, $W_i = \cup\{W_i^s : s \in \mathbb{N}\}$, $\max(W_i^s) = s$, and $\{\langle m, s\rangle : m \in W_i^s\}$ is computable. There is also an array $\{\Phi_i^s : i, s \in \mathbb{N}\}$ such that for all i, A and m, $\Phi_i(A; m) = \lim_s \Phi_i^s(A \upharpoonright s+1; m)$; in fact, if $A = W_j$, then $\Phi_i(W_j; m) = \lim_s \Phi_i^s(W_j^s; m)$, and we can assume that $\{\langle i, j, s, \sigma\rangle : \sigma \subset W_j^s \;\&\; \Phi_i^s(\sigma; m)\downarrow\}$ is computable; without loss of generality, we assume that if $\langle i, j, s, \sigma\rangle$ is in this set, then $i, j \leq s$ and $\sigma \subseteq [0, s]$. The notation $\Phi(A; m)[s]$ will be used for $\Phi_i^s(A; m)$.

1.2. Degrees

Define $A \leq_T B$ if there is a computable partial functional Φ such that $\Phi(B) = A$, and $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. $\equiv_T$ is an equivalence relation known as *Turing equivalence* which captures the notion of information content. The equivalence classes, $\{A : A \equiv_T B\}$ are called *degrees*, and form a partially ordered set (*poset*) with ordering $\leq$ induced by $\leq_T$. Lower-case bold-face letters such as $\boldsymbol{a}$ are used to denote degrees. A degree is *computably enumerable* if it contains a computably enumerable set. $\mathcal{R}$ denotes the poset of computably enumerable degrees, and $\boldsymbol{R}$ denotes the set of computably enumerable degrees.

The degrees support a join operation: Thus if $\boldsymbol{a}$ is the degree of A and $\boldsymbol{b}$ is the degree of B, then $\boldsymbol{a} \cup \boldsymbol{b}$ denotes the degree of $A \oplus B$ which is the least upper bound of $\boldsymbol{a}$ and $\boldsymbol{b}$. Some, but not all pairs of degrees have a meet; when such a meet exists, it is denoted by $\boldsymbol{a} \cap \boldsymbol{b}$. We write $\boldsymbol{a}|\boldsymbol{b}$ if $\boldsymbol{a}$ and $\boldsymbol{b}$ are incomparable.

There is a smallest degree, $\boldsymbol{0}$, the degree of the computable sets. The degrees also support a *jump operator* denoted by $\boldsymbol{a}'$; given a set A of degree $\boldsymbol{a}$, $\boldsymbol{a}'$ is the largest degree of a set that is computably enumerable in A. (We remark that such a set always exists.) Thus $\boldsymbol{0}'$ is the largest computably enumerable degree; we fix a computably enumerable set K of degree $\boldsymbol{0}'$.

The *high/low hierarchy* for the computably enumerable degrees is defined as follows. A computably enumerable degree $\mathbf{a}$ is low_n if $\boldsymbol{a}^{(n)} = \boldsymbol{0}^{(n)}$, and is $high_n$ if $\boldsymbol{a}^{(n)} = \boldsymbol{0}^{(n+1)}$. Computably enumerable degrees $\mathbf{a}$ satisfying

$$\boldsymbol{0}^{(m)} < \boldsymbol{a}^{(m)} < \boldsymbol{0}^{(m+1)}$$

for all m are called *intermediate*. A computably enumerable set A is said to be low_n ($high_n$, *intermediate*, resp.) if its degree is low_n ($high_n$ intermediate, resp.).

We recall the Shoenfield Limit Lemma [23]:

LEMMA 1.2.1 (Limit Lemma). *$A \leq_T K$ iff there is a total computable function f such that for all x, $\lim_s f(x,s) = A(x)$.* ⊣

We note that our definition of limit applies to partial functions as well. It is then easy to verify the following lemma, which we will use as our version of the Limit Lemma:

LEMMA 1.2.2 (Limit Lemma). *$A \leq_T K$ iff there is a computable partial function f such that for all x, $\lim_s f(x,s)\downarrow = A(x)$.*

1.3. Finite Sequences and Trees

Let A be a computable set of symbols, and let $T(A) = A^{<\omega}$, i.e., the set of all finite sequences of symbols from A. $\emptyset$ will denote the sequence of length 0. For $\sigma \in T(A)$, let $|\sigma|$ denote the length of the sequence σ, i.e., the cardinality of the domain of σ. If there is an $n < \min\{|\sigma|, |\tau|\}$ such that $\sigma(n) \neq \tau(n)$, then we write $\sigma|\tau$. For $\sigma, \tau \in T(A)$, we define $\sigma^\frown\tau$ to be the sequence of length $|\sigma| + |\tau|$ consisting of the sequence of symbols σ followed by the sequence of symbols τ.

We define a partial order, $\subseteq$, on $T(A)$ by $\sigma \subseteq \rho$ if there is a τ such that $\sigma^\frown\tau = \rho$. $\langle T, \subseteq\rangle$ is a *tree* if T is an initial segment of $T(A)$ for some A, and $\subseteq$ is the restriction of the ordering on $T(A)$ to T. If $\sigma, \tau \in T(A)$, then $\sigma \wedge \tau$ is the longest $\rho \in T(A)$ such that $\rho \subseteq \sigma$ and $\rho \subseteq \tau$; and if $|\sigma| > 0$, then σ^-

is the unique $\rho \subset \sigma$ such that $|\sigma| = 1 + |\rho|$. We also use interval notation for trees; thus $[\sigma, \tau] = \{\rho : \sigma \subseteq \rho \subseteq \tau\}$. Similar notation is used for open and half-open intervals.

A *path* through a tree $T = T(A)$ is an ω-sequence of elements of A; $[T]$ denotes the set of all paths through T. (More generally, *trees* are defined as posets having the property that the set of predecessors of any element is linearly ordered, and a *path* through a tree is a maximal linearly ordered subset of the tree.)

Sequences of trees will be specified; the notation used is $\{T^i : i \leq n\}$. In this context, an element of T^i will be of the form σ^i, i.e., the superscript will identify the tree of the sequence on which σ^i lies. If $i = 0$, then the superscript will be omitted.

We will be using paths through one tree to approximate to paths through another tree through a limit approximation, which is defined as follows. Fix $\tau \in T \cup [T]$ and a function λ from T to another tree T'. Then $\xi \subseteq \lim\{\lambda(\eta) : \eta \subset \tau\}$ iff $\xi \subset \lambda(\eta)$ for a cofinal set of nodes $\eta \subseteq \tau$.

Chapter 2

SYSTEMS OF TREES OF STRATEGIES

Many theorems concerning structural properties of $\mathcal{R}$ are proved by constructing sets satisfying each requirement in an infinite list. Frequently, the action taken to satisfy one requirement will conflict with the preservation of action taken earlier to satisfy another requirement. The priority method was devised to organize the action taken to satisfy requirements so that, at the end of a construction, all requirements are satisfied.

As the logical complexity (measured by the complexity of sentences in a certain language) of the requirements under consideration increases, it becomes more difficult to analyze the conflicts between requirements and to show that all requirements are satisfied. The *systems of trees of strategies* approach is an inductive method which provides such an analysis. The proofs that we present using this approach all follow the same pattern.

We begin with a list of all requirements to be satisfied, and it will usually be clear that the theorem will follow once we show that all requirements are satisfied. We will then introduce *basic modules* for each requirement; these modules will describe, in a high-level language, the manner in which the requirement will be satisfied. The basic modules will be finite binary trees whose edges and non-terminal vertices or *nodes* are labeled with sentences. Each non-terminal node will have a *directing sentence* whose role is to direct the action taken; if the directing sentence is true, then we will follow the instructions of the *validated action* sentence along one edge emanating from that node, and if the sentence is false, then we will follow the instructions of the *activated action* sentence along the other edge emanating from that node. The basic modules will be templates, and the sentences will be allowed to use parameters defined in terms of nodes of the basic module, and will be allowed to restrict quantifiers involving these nodes to lie along the true path through the tree. There will then be instructions on how to implement the basic module on the highest-level tree of strategies T^n, and an observation that the implementation faithfully reflects the basic module and the requirement.

The next step will be the decomposition of requirements, level by level, from T^n to the computable level T^0 at which the construction takes place. Suppose that we have determined what takes place on T^{k+1}, and wish to pass to T^k.

Each node $\eta^{k+1} \in T^{k+1}$ will have potentially infinitely many *derivatives* on T^k; these will be nodes working to ensure that the instructions for η^{k+1} are carried out. Thus we will need an algorithm that determines the node of T^{k+1} for which a given node of T^k is working, and a definition of the path through T^{k+1} computed, through a limit approximation, by a given path through T^k. The trees of strategies will need to be defined carefully in order to reflect this computation. We will then need to describe how the directing sentence and action sentences for η^{k+1} are decomposed to produce the sentences for the derivative $\eta^k \in T^k$ of η^{k+1}. This will usually be done by bounding some quantifiers, and we will again allow nodes of T^k to be used as parameters in the bounding process. We will have to show that this process is a faithful reflection of the requirement, namely, that if we follow the instructions of the sentences for the derivatives of η^{k+1} along a path Λ^k through T^k and Λ^k computes the path Λ^{k+1} through T^{k+1} along which η^{k+1} lies, then we will have satisfied the instructions of the sentences for η^{k+1}.

When we reach T^0, then we will have instructions for an effective construction. However, we will need to show that these instructions can be implemented. Thus we cannot have instructions telling us to withhhold a number x from a computably enumerable set A if x has already been placed into A, nor can we have instructions to declare an axiom $\Phi^s(\sigma; x) = m$ when we have already declared an axiom $\Phi^t(\tau; x) = k$ for some $t < s$, $\tau \subseteq \sigma$ and $k \neq m$. Similarly, we cannot have such conflicting instructions at a given stage s. Once this is shown, we implement the construction and the theorem then follows. There may be nodes along the true path through T^0 whose instructions we choose to ignore, but in all such cases, we will show that these instructions are derived from a node that does not lie on the true path through its tree.

As is evident from the above description, there are many things that need to be checked or proved for a given theorem. If separate proofs were required for each theorem, there would be no advantage to this approach, but this is not the case. Many of the steps to be carried out will be the implementation of one of several fixed algorithms used over and over, and we will have theorems showing that if an algorithm has certain easily checked properties, then it accomplishes its task. Thus the proof of each theorem will consist of checking, usually by inspection, that certain properties hold, and quoting some theorems about the framework. These theorems will apply to many constructions, and thus prevent duplication of work in proofs. Thus once we have the needed theorems describing properties of the framework, the hard part in proving a theorem will consist of designing basic modules, and sometimes revising fixed algorithms, so that the necessary properties are satisfied.

In Section 2.1, we present an overview of this chapter, and use the proof of the Friedberg–Mučnik Theorem to motivate the trees of strategies approach. Trees of strategies are introduced in Section 2.2. The properties to be obeyed by basic modules are presented in Section 2.3. The λ *function*, defined in Section 2.4, will provide a computable limit approximation of a path Λ^{k+1} through T^{k+1} from a path Λ^k through T^k. The *links* that are defined in Section 2.5 will be used to keep track of nodes $\eta^{k+1} \in T^{k+1}$ that are free to be *switched* by λ, i.e., nodes for which it is safe to have $\lambda(\eta^k)|\lambda((\eta^k)^-)$ with $\eta^{k+1} = \lambda(\eta^k) \wedge \lambda((\eta^k)^-)$. In Section 2.6, we define the nodes that are eligible to be *antiderivatives* of η^k, i.e., from which η^k is allowed to be *derived*. Each construction will have to describe the way the antiderivative of η^k is chosen from among these eligible nodes. Antiderivatives for nodes of T^k will be chosen within *blocks* of nodes, i.e., segments of T^k containing no infinite paths. Properties of the block formation process are specified in Section 2.7, and the assignment of requirements to nodes of trees and the decomposition of requirements will also be discussed in that section. The *weight function*, introduced in Section 2.8, will supply bounds for the quantifiers and will determine arguments for functionals.

Directing sentences and action are discussed in Sections 2.9 and 2.10. The first of these sections deals primarily with an example, and the second section with more general properties of these sentences. The Framework Theorem is presented in Section 2.11. This section does not require a deep understanding of the framework, and can be read without understanding many of the details of prior sections. We present two simple proofs of lemmas about the framework in Section 2.12.

2.1. An Overview

This chapter is devoted to the development of the trees of strategies framework. Much of this development is inductive in nature, and terminology may be encountered that has not yet been defined. One of the purposes of this section is to give an overview of the development of the framework, and in particular, to introduce the concepts and terminology used and the intuition behind the concepts. In order to make the intuition more concrete, we will interleave a description of the proof of the Friedberg–Mučnik Theorem within the framework. We begin with an introduction to this theorem.

The study of $\mathcal{R}$ traces its history back to Post's [20] fundamental paper of 1944. Post focused on the computably enumerable sets, sets that can be enumerated by a computer, and began the study of the information content of such sets by trying to determine properties of $\mathcal{R}$. There are two special computably enumerable degrees, $\boldsymbol{0}$, the degree of the computable sets which is the smallest degree in this poset, and $\boldsymbol{0}'$, the largest computably enumerable degree. Post asked whether there were other computably enumerable degrees, a question that became known as *Post's Problem*.

Solutions to Post's Problem were found independently by Friedberg [4] and Mučnik [19] more than a decade later through the construction of a pair of computably enumerable sets whose degrees are incomparable. These solutions introduced a new technique, the *priority method*, that is the subject of this monograph. As the proof of this result is one of the simplest applications of the priority method, it will be used to motivate the systems of trees of strategies approach.

THEOREM 2.1.1 (Friedberg, Mučnik). *There are computably enumerable sets A and B such that $A \not\leq_T B$ and $B \not\leq_T A$.*

Each construction will have an associated level, and a level n construction will take place on the sequence of trees $T^0, \dots, T^n$. The description of the construction will begin on T^n, and will descend, one level at a time, until we reach the computable level T^0. The levels go hand-in-hand with the arithmetical hierarchy, as for all k, an oracle of degree $\boldsymbol{0}^{(k)}$ will be able to determine the action of the construction on T^k. For each $k \leq n$, a portion of the construction will be assigned to each node η^k of T^k, and if $k > 0$, then η^k will divide its assignment among many nodes of T^{k-1}. Furthermore, each node of T^{k-1} will receive an assignment from at most one node of T^k. If η^k delegates part of its assignment to $\eta^{k-1} \in T^{k-1}$, then we will call η^{k-1} a *derivative* of η^k, and will call η^k the *antiderivative* of η^{k-1}. We will also introduce notation to track this relationship; we say that $\text{up}(\eta^{k-1}) = \eta^k$ if η^k is the antiderivative of η^{k-1}.

A typical construction will have to satisfy each requirement in a given sequence of requirements. In a level n construction, each requirement will have a level, and that level will be $\leq n$.

EXAMPLE 2.1.2. In the proof of the Friedberg–Mučnik Theorem, we will construct two computably enumerable sets, A and B. The construction will aim to satisfy the following requirements, for each computable partial functional Φ.

$$P_\Phi : \Phi(A) \neq B.$$

$$Q_\Phi : \Phi(B) \neq A.$$

The manner in which requirements are to be satisfied will be presented through a labeled finite binary tree called a *basic module*. A *directing sentence* (in prenex normal form) will be assigned to each non-terminal node of the basic module, and each such node will have a level. If the level of the node is k, then the directing sentence will be a Σ^0_k sentence if k is odd, and a Π^0_k sentence if k is even. *Action sentences* will be assigned to each of the two edges emanating from a non-terminal node. Suppose that we are given two successive nodes, α and β, of a basic module, and that $\beta = \alpha^\frown\langle\gamma\rangle$; then we call γ an *outcome* of α. If the directing sentence assigned to α is Σ^0_k (Π^0_k, resp.), then we refer to γ as a Σ-*outcome* (Π-*outcome*, resp.) and say that α has

Σ *outcome along* β (Π *outcome along* β, resp.) if β guesses that the directing sentence is true, and we refer to γ as a Π-*outcome* (Σ-*outcome*, resp.) and say that α has Π *outcome along* β (Σ *outcome along* β, resp.) if β guesses that this directing sentence is false. We also want terminology that keeps track of the truth values of directing sentences in a level-independent manner. Thus we refer to γ as a *validated outcome* (*activated outcome*, resp.) if β guesses that the directing sentence assigned to α is true (false, resp.), and say that α is *validated along* β (*activated along* β, resp.). Note that the identification of the activated vs. validated designation with the Σ vs. Π designation alternates, level by level. We will later describe how the nodes of the basic module are identified with nodes of T^n.

EXAMPLE 2.1.3. We describe the basic module for the requirement $\Phi(A) \neq B$ of the Friedberg–Mučnik Theorem. The module will consist of a single non-terminal node α having level 1, with two terminal successors α_0 and α_1. α will have activated and Π outcome along α_0, and will have validated and Σ outcome along α_1. The directing sentence will be

$$\exists u \exists s(\Phi(A \restriction u; x)[s] = 0)$$

where x will be a parameter determined by the framework from the node of T^1 to which α is assigned. The activated action sentence will be

$$\forall t(x \notin B^t),$$

and the validated action sentence will be

$$\exists r \forall t \geq s(x \in B^r \;\&\; A^t \restriction u = A^s \restriction u).$$

(In actuality, we will restrict the range of the variable s, but we need not be concerned with that aspect here.)

While a computable construction is to be carried out along T^0, the action sentences are Π^0_1 sentences. (In Example 2.1.3 the action sentences presented are those for T^1; when we pass to T^0, parameters will have been chosen to replace r, s and u, and the quantifier $\exists r$ will disappear.) These are to be interpreted as instructions to the construction at later stages. Thus we will carry out the instructions of the action sentences for all t that are less than or equal to the weight of nodes ρ^0 along the true path through T^0 which extend the node $\eta^0 \in T^0$ to which the requirement is assigned. We will not necessarily implement the action at all such nodes ρ^0, but it will suffice for us to implement action at infinitely many such nodes, as long as it seems that η^0 is a node for which the construction requires action to be carried out.

Let $\eta^{k-1} \in T^{k-1}$ be a derivative of $\eta^k \in T^k$, and suppose that S is the directing sentence assigned to η^k. For concreteness, we assume that k is odd; a similar pattern is observed when k is even. We observe from the above comments that S is a Σ^0_k sentence. If S is not properly Σ^0_k (i.e., if S is Π^0_{k-1}

or Σ^0_{k-1}), then S is also the directing sentence assigned to η^{k-1}. Otherwise, the directing sentence $\widetilde{S}$ for η^{k-1} is obtained by bounding each quantifier in the block of leading existential quantifiers in S; thus $\widetilde{S}$ will be a Π^0_{k-1} sentence. (This is why validated outcomes switch from Σ outcomes on T^k to Π outcomes on T^{k-1}.) The particular bounds are frequently generated by the *weight function* wt, a function whose domain consists of the nodes of the trees of strategies, and which, on each tree, is monotonically increasing with the length of the node.

EXAMPLE 2.1.4. We again analyze the sentences used in the proof of the Friedberg–Mučnik Theorem. Suppose that the directing sentence of Example 2.1.3 is assigned to $\eta^1 \in T^1$, and let $\eta^0 \in T^0$ be a derivative of η^1. The directing sentence for η^0 will have the form

$$\exists u \leq p \exists s \leq q(\Phi(A \restriction u; x)[s] = 0),$$

where p and q are number parameters generated by the weight function evaluated on η^0 and nodes generated from η^0. The activated action sentence for η^0 will be

$$\forall t \leq w(x \notin B^t),$$

and the validated action sentence for η^0 will be

$$\forall t \in [s, w](x \in B^r \;\&\; A^t \restriction u = A^s \restriction u),$$

where r and w are other parameters generated by the nodes that are currently being visited. The parameter p is the weight of the longest node on T^1 lying along the current path through T^1 computed by η^0, the parameter q is the weight of η^0 itself, the parameter r will be the weight of the successor node to η^0 on the path determined by the construction, and the parameter w will be the weight of the current path through T^0. The reason for these choices will become apparent when we see what is needed to make the proof succeed; for the time being, it is only important to note the bounding of quantifiers.

For $k \in (0, n]$, the correspondence between the outcome of a node $\eta^k \in T^k$ along a path $\Lambda^k \in [T^k]$ and the outcomes of its derivatives along a path $\Lambda^{k-1} \in [T^{k-1}]$ which generates Λ^k must preserve predicted truth values of directing sentences. We make this concept more precise with the consideration of the form of directing sentences. Suppose that the directing sentence S for η^k is a Σ^0_k sentence (a similar analysis can be carried out for Π^0_k sentences).

First suppose that Λ^k predicts that S is true, i.e., that S is validated along Λ^k. Then Λ^k predicts that we will find a witness for each quantifier in the leading block of existential quantifiers of S. This prediction must be borne out along Λ^{k-1}; thus there must be a derivative η^{k-1} of η^k along Λ^{k-1} at which these witnesses are found. This, in turn, requires the bounds that η^{k-1} places on the quantifiers just mentioned to be sufficiently large to bound the witnesses