General methods for analyzing machine learning sample complexity

Christoph Cornelius Michael
College of William & Mary - Arts & Sciences

Follow this and additional works at: https://scholarworks.wm.edu/etd

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation
https://dx.doi.org/doi:10.21220/s2-ngq1-gy69

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
GENERAL METHODS FOR ANALYZING
MACHINE LEARNING SAMPLE COMPLEXITY

A Dissertation
Presented to
The Faculty of the Department of Computer Science
The College of William and Mary in Virginia

In Partial Fulfillment
Of the Requirements for the Degree of
Doctor of Philosophy

by
Christoph Michael
1994
This dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Approved, August 1994

W. Robert Collins
William Bynum
Stefan Frycock
Philip Kearns
Rex Kincaid
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>List of figures</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>viii</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Describing the learning problem</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 What constitutes success in learning?</td>
<td>3</td>
</tr>
<tr>
<td>1.2. Overview</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Notation</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 2: Analyzing Sample Complexity</td>
<td>7</td>
</tr>
<tr>
<td>2.1. Hypothesis classes</td>
<td>7</td>
</tr>
<tr>
<td>2.2. The Vapnik-Chervonenkis Dimension</td>
<td>8</td>
</tr>
<tr>
<td>2.3. Combining (0, 1)-valued hypothesis classes</td>
<td>13</td>
</tr>
<tr>
<td>2.4. Learning rule bases</td>
<td>17</td>
</tr>
<tr>
<td>2.4.1 k-ary conjunction operators</td>
<td>19</td>
</tr>
<tr>
<td>2.5. Discussion</td>
<td>25</td>
</tr>
<tr>
<td>2.6. Bibliographic notes</td>
<td>25</td>
</tr>
<tr>
<td>Appendix A to chapter 2: Proofs of results in Chapter 2</td>
<td>28</td>
</tr>
<tr>
<td>2A.1. Proof of proposition 2.5</td>
<td>28</td>
</tr>
<tr>
<td>2A.2. Proof of proposition 2.7</td>
<td>38</td>
</tr>
<tr>
<td>2A.3. Proof of proposition 2.10</td>
<td>50</td>
</tr>
<tr>
<td>2A.4. Proof of proposition 2.11</td>
<td>57</td>
</tr>
<tr>
<td>2A.5. Proof of proposition 2.14</td>
<td>78</td>
</tr>
<tr>
<td>Chapter 3: Classes of composite functions</td>
<td>86</td>
</tr>
<tr>
<td>3.1. Functional composition</td>
<td>88</td>
</tr>
<tr>
<td>3.2. Analyzing algorithms that learn composite functions</td>
<td>89</td>
</tr>
<tr>
<td>3.3. function networks</td>
<td>92</td>
</tr>
<tr>
<td>3.4. An alternative notation</td>
<td>99</td>
</tr>
<tr>
<td>3.5. Placing constraints on the set of legal conjunctions</td>
<td>103</td>
</tr>
<tr>
<td>3.6. Discussion</td>
<td>106</td>
</tr>
<tr>
<td>3.7. Bibliographic notes</td>
<td>107</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I am indebted to the members of my doctoral committee, and to Stephen Park, for their numerous useful comments, and in particular to Robert Collins, who suggested the proof style used herein.
LIST OF FIGURES

Figure 2.1: A circuit for calculating a $\{0, 1\}$-valued function. 21
Figure 3.1: A simple network of three functions. 88
Figure 3.2: A boolean circuit containing four functions. 90
Figure 3.3: 90
Figure 3.4: 90
Figure 3.5: The boolean circuit of figure 2 represented as a function graph. 94
Figure 3.6: Some members of a class of boolean circuits in which the architecture is not fixed. 96
ABSTRACT

URING the past decade, there has been a resurgence of interest in applying mathematical methods to problems in artificial intelligence. Much work has been done in the field of machine learning, but it is not always clear how the results of this research should be applied to practical problems. Our aim is to help bridge the gap between theory and practice by addressing the question: "If we are given a machine learning algorithm, how should we go about formally analyzing it?" as opposed to the usual question: "how do we write a learning algorithm we can analyze?"

We will consider algorithms that accept randomly drawn training data as input, and produce classification rules as their outputs. For the most part our analyses will be based on the syntactic structure of these classification rules; for example, if we know that the algorithm we want to analyze will only output logical expressions that are conjunctions of variables, we can use this fact to facilitate our analysis.

We use a probabilistic framework for machine learning, often called the pac model. In this framework, one asks whether or not a machine learning algorithm has a high probability of generating classification rules that "usually" make the right classification (pac means probably approximately correct). Research in the pac framework can be divided into two subfields. The first field is concerned with the amount of training data that is needed for successful learning to take place (success being defined in terms of generalization ability); the second field is concerned with the computational complexity of learning once the training data have been selected. Since most existing algorithms use heuristics to deal with the problem of complexity, we are primarily concerned with the amount of training data that algorithms require.
1. Introduction

DURING the past decade, there has been a resurgence of interest in applying mathematical methods to problems in artificial intelligence. Much work has been done in the field of machine learning, but it is not always clear how the results of this research should be applied to practical problems. The aim of this thesis is to bridge the gap between theory and practice in a number of important cases by developing methods for the formal analysis of machine learning algorithms.

We will consider algorithms that accept randomly drawn training data as input, and produce classification rules as their outputs. For the most part our analyses will be based on the syntactic structure of these classification rules; for example, if we know that the algorithm we want to analyze will only output logical expressions that are conjunctions of variables, we can use this fact to facilitate our analysis.

This thesis uses a probabilistic framework for machine learning, often called the pac model. In this framework, we ask whether or not a machine learning algorithm has a high probability of generating classification rules that “usually” make the right classification (pac means probably approximately correct). Research in the pac framework can be divided into two subfields. The first field is concerned with the amount of training data that is needed for successful learning to take place (success is defined in terms of generalization ability, that is, the ability classify arbitrary inputs correctly, whether or not they were used in training); the second field is concerned with the computational complexity of learning once the training data have been selected. Since most existing algorithms use heuristics to deal with the problem of complexity, this thesis is primarily concerned with the amount of training data that algorithms need.

1.1. Describing the learning problem.

This section deals with the formalization of the problem of machine learning. Much of the content of this section was developed in the field of nonparametric pattern recognition (cf [33]).

This thesis is concerned with algorithms that learn from examples. To construct them, we first specify a space \( \mathcal{X} \) of possible inputs, and points in this space are called instances. In our formalism, we assume that there is a
probability measure $P_X$ defined on $X$, and, when a training example is desired, an instance is drawn at random according to this distribution.

We assume that there is some function $F$ which, in some way, classifies the inputs we draw. The goal is to learn this function; by this we mean that the learning algorithm should find an approximation of $F$ that has a high probability of classifying any input correctly, regardless of whether that input was used for training. $F$ is often referred to as the target function in the learning problem.

The goal is for the learning algorithm to learn by observing how the target function behaves at the points in $X$ that are used for training examples. Therefore we do not allow information about the target function to be transmitted by the choice of training examples; we require that $P_X$ does not depend on $F$. It should be pointed out that we are not assuming the independence of an instance $x$ and its mapping $F(x)$; the problem would be quite hopeless if no such dependency existed.

In the literature on supervised learning, an example is defined as an ordered pair

$$(x, F(x)),$$

where $x \in X$ and where $F$ is the target function. In principle we will adopt this convention as well, but we will often find it unnecessary to treat the elements of this pair as if they came from different domains. If $Y$ is the range of $F$, then we will define $X$ as $X \times Y$. We will adopt the convention that for any $x \in X$, where $x$ is the tuple $(v, F(v))$, $x^i$ will refer to $v$, the input that the learning algorithm receives, while $x^o$ will refer to $F(v)$, the desired output. We will assume that there is a probability measure $P_X$ associated with points in $X$, with the property that

$$P_X(v) = P_X((v, F(v))).$$

A sequence of examples that are drawn at random (with replacement) from $X$ according to $P_X$, classified according to $F$, is a training sample, or simply a sample, of $F$. Such samples will constitute the inputs of our learning algorithms.

Using our notation, we will find it convenient to define a learning algorithm as a mapping from $X^t$ to $Y^*$, for some positive integer $t$ and some hypothesis class $H^*$. In other words, a learning algorithm inputs $t$ training examples and outputs a hypothesis. (Ultimately we will assume that the the learning algorithm's argument is a sample drawn from $X$, but we do not require the algorithm to know this.) A possible objection to this scheme is

* We do not assume $F \in H^*$. 
that many real algorithms can use training samples of arbitrary size. However, our analysis is aimed precisely at finding good prior values for $\ell$, so the size of the training sample will be decided by the time the learning algorithm is actually run. We could say that we have a collection of learning algorithms that differ only in the number of training examples they use; our task is to find one whose performance meets our requirements.

1.1.1. What constitutes success in learning?

The quality of a learned hypothesis (and hence the success of the algorithm that learned it) is judged by the expected loss due to misclassification, according to some loss function $Q$, that is incurred when that hypothesis is used for making classifications. If $H \in \mathcal{M}$ is the learned function and $F$ is the target function, then the expected loss is defined as

$$\eta = \int_{\mathcal{X}} Q(F, H, x) \, dP_{\mathcal{X}}.$$  \hfill (1.1)

Some possible choices for $Q(F, H, x)$ are $|H(x) - F(x)|$, $(H(x) - F(x))^2$, or simply $H(x) \oplus F(x)$, where

$$\alpha \oplus \beta = \begin{cases} 1, & \text{if } \alpha \neq \beta, \\ 0, & \text{otherwise.} \end{cases}$$

It is assumed that $Q(\cdot, \cdot, x)$ quantifies the loss that will be incurred by misclassifying $x$; we can also regard it as a measure of the deviation between two functions at $x$. This thesis will use the last of the three functions mentioned above: $Q(F, H, x) = H(x) \oplus F(x)$. Since $Q$ is an indicator function, this makes $\eta$ the probability of obtaining an incorrect classification. (When $H(x) \neq F(x)$ we will say that $H$ misclassifies the example $x$).

We can reformulate (1.1) in terms of $X$:

$$\eta = \int_{\mathcal{X}} Q(H, x) \, dP_X;$$  \hfill (1.2)

here the target function $F$ is implicitly defined when we decide which part of an input $x$ will be $x^\omega$. Thus the three choices for $Q$ can be rewritten $|H(x^\omega) - x^\omega|$, $(H(x^\omega) - x^\omega)^2$, and $H(x^\omega) \oplus x^\omega$.

The quality of the hypothesis depends on the training sample, and therefore we will not be able to compute $\eta$ directly. Instead, we will simply ask for the probability (in the space of possible samples) that $\eta$ will exceed some arbitrarily chosen bound $\epsilon$ ($\epsilon$ is sometimes referred to as the accuracy parameter of the learning algorithm). Our goal is to be able to specify an arbitrary $0 < \epsilon < 1$, and expect the learning algorithm to have an arbitrarily high probability of finding a hypothesis for which $\eta$ is less than $\epsilon$. (What is meant by an "arbitrarily high probability"
is that we also wish to be able to specify what probability the algorithm should have of finding a good hypothesis. However, increasing this probability may cause the algorithm to take more time and require more examples. The same is true if $\epsilon$ is decreased. The probability that the algorithm will not find a good hypothesis is usually denoted by $\delta$, and sometimes referred to as the confidence parameter of the algorithm.

It should be noted that (1.2) is not useful unless $H$ is used to classify inputs drawn according to $P_X$. This is, in some sense, a weakness of this framework, but it is also a limitation of learning in general, for it is always possible to make a good hypothesis bad by changing the input distribution. As an extreme example, one could make a particular point $v_0 \in \mathcal{X}$ an exception to whatever classification rule applied to the rest of the domain. One could choose $P_{X^*}$ so that the probability of drawing $v_0$ was zero during training, but then test the hypothesis in a domain where $v_0$ was drawn with probability 1. It is clear that no learning algorithm can consistently do better than chance in such a situation, and an adversary with knowledge of the algorithm's structure may well be able to force worse-than-chance outcomes ($\eta > 0.5$) by making judicious choices of $F$ and $v_0$.

1.2. Overview.

A machine learning problem can be divided into two parts, namely finding a good training sample and using it. There are several reasons for making this division. First, the problem of finding a good training sample is not particularly amenable to heuristics in the present formalism. More importantly, heuristics cannot easily compensate for training data that contain insufficient information. Thus, if a learning problem requires impractically large amounts of training data, we may well be unable to solve it.

On the other hand, once the training data have been collected, the goal is simply to find a function that makes as few mistakes as possible on those data. This is essentially an optimization problem, and there are many heuristic approaches to it; aside from countless machine learning algorithms there are search strategies such as branch-and-bound or A*, and even more general optimization strategies, the most familiar types being genetic and simulated annealing algorithms. What makes this problem easier is that the optimization (learning) algorithm can test its intermediate results against the training data and obtain some measure of their quality. Quality estimates of this sort are crucial for many heuristic methods, but they can only be applied in this phase of learning.

The problem of generalization ability, however, is often left unanswered when heuristic optimization methods are used for machine learning. This problem has to do with the matter of finding a good training sample because generalization ability depends on whether or not the training data give us a good picture of the function to be learned. When the problem of generalization ability is addressed, it is often dealt with through empirical testing, and, although such tests can provide the sort of statistical information we have described above, they tell us little
about why one learning algorithm generates hypotheses with better generalization ability than those of another learning algorithm.

In this thesis, our approach will be to find prior estimates of the generalization ability of an algorithm, in terms of the size of the training sample that is to be used. This eliminates the element of trial and error that is present in posterior testing, and also gives us quite a bit more information about what makes a learning algorithm successful. This approach has been applied to some extent in the literature on machine learning, but the analysis is usually coupled to particular learning algorithms. In contrast, our approach will be to make our analysis as general as possible with regard to the actual learning algorithm used. Our goal, especially in Chapters 2 and 3, is to provide a set of tools that can be used in the analysis of large classes of learning algorithms. The emphasis is on facilitating the analysis of pre-existing algorithms, rather than algorithms written specifically with the pac model in mind. The issue of analyzing such algorithms seems to have been almost entirely neglected in the past, and by addressing it we begin to bridge an important gap between the theoretical and practical fields of machine learning.

The thesis is organized in the following way: First, chapters 2 and 3 develop methods that can be used to analyze learning algorithms in terms of the syntax of the classification rules they generate. In many cases (especially those where the syntax is simple), existing results can be used to find a bound on the sample complexity of a learning algorithm. We will show how to use these simple cases as building blocks when the output rules have a more complicated syntax.

Chapter 4 extends these results to algorithms that make run-time decisions about the size of the rules they generate.

1.2.1. Notation

In this thesis, the notation $|S|$ denotes the cardinality of the set $S$. If $S$ is a sample space, we will use $P_S$ to denote the probability measure defined on $S$. $R \times S$ will denote the cross product of the sets $R$ and $S$, and $S^m$ will denote the set

$$S \times S \times \cdots \times S$$

of $m$-tuples of elements in $S$. An event is a subset of a sample space, and if $a$ and $b$ are two events, we will sometimes use the notation $a \Rightarrow b$ to denote that $a$ is a subset of $b$. (Note that if $a \Rightarrow b$, the occurrence of the event $a$ implies that $b$ has also occurred. An event $E$ is defined on $S$ if $E$ is an event in $S$.) We will use $\log_2(r)$
to denote the base \( e \) logarithm of \( r \), but we will also use \( \ln(r) \) to denote the natural logarithm of \( r \). \( e \) denotes the base of the natural logarithm.

If \( S \) is a set we say that \( ||S|| \) is the number of bits needed to represent \( S \); that is, \( ||S|| \) is the minimum number of bits that would be needed to give each member of \( S \) a unique index.

\( \mathbb{N} \) and \( \mathbb{R} \) denote the sets of natural and real numbers, respectively. If \( n \) is a number, we will use \( |n| \) to denote the absolute value of \( n \).

The notation \( \mathcal{F} \) indicates a tuple. When it is necessary to refer explicitly to the elements of a tuple \( \mathcal{F} \), we will write it as

\[ \langle x_1, x_2, \ldots, x_m \rangle, \]

if the tuple contains \( m \) elements. We overload the notation for the cardinality of a set and say that \( |\mathcal{F}| \) denotes the number of elements in \( \mathcal{F} \).

If \( F \) is a function then \( F(\cdot) \) will refer to the value of \( F \) when the argument is irrelevant or understood from the context. \( \text{Dom}(F) \) refers to the domain of \( F \). If \( \mathcal{F} \) is a set of functions \( \{F_1, F_2, \ldots, F_t\} \), then \( \text{Dom}(\mathcal{F}) \) will denote

\[ \bigcup_{F_i \in \mathcal{F}} \text{Dom}(F_i). \]

Similarly, \( \text{Ran}(F) \) will refer to the range of \( F \), and \( \text{Ran}(\mathcal{F}) \) will refer to

\[ \bigcup_{F_i \in \mathcal{F}} \text{Ran}(F_i). \]

The symbol \( \Box \) indicates the end of a proof (or the end of a cited theorem or lemma), and \( \bigcirc \) indicates the end of an example.
2. Analyzing Sample Complexity

In the last chapter, we presented a criterion for measuring the quality of a hypothesis $H$ as an approximation of a function $F$ on a sample space $X$. We stated that a good $H$ would be one for which

$$\eta(H) \equiv \int_X H(x) \delta F(x) dP_X$$

was small. An empirical estimate for $\eta(H)$ can be defined on the training sequence that was used to obtain $H$. If $A$ is a learning algorithm and $H = A(F)$, then

$$\hat{\eta}(H) \equiv \frac{1}{|T|} \sum_{t \in T} H(x) \Phi F(x).$$

We can judge the quality of a hypothesis if we can bound $\eta(H)$ in terms of $\hat{\eta}(H)$.

Let $\xi$ be a bound on the permissible gap between the actual and observed rates of misclassification. By "permissible" we mean that we will judge $H$ to be a good hypothesis if and only if

$$|\eta(H) - \hat{\eta}(H)| \leq \xi. \quad (2.1)$$

We wish to know how many training examples will be needed before we can state, with confidence $(1 - \delta)$, that our learning algorithm will generate a hypothesis satisfying (2.1). That question is the focus of this chapter. We will address it by examining several types of functions, and discussing the analysis of learning algorithms that produce such functions as their hypotheses.

2.1. Hypothesis classes

The sample complexity of a learning algorithm is an expression that states the number of examples required by the learning algorithm in terms of $\xi$, $\delta$, and one or more parameters that describe the what kinds of hypotheses can be generated by the learning algorithm we are using. Such parameters are the focus of our discussion in this chapter; we will present ways of determining what they are. Therefore we would like to have a better grasp of the
thing being parameterized, that is, the "set of hypotheses that a learning algorithm can generate." We will refer to this set as the hypothesis class of the learning algorithm.

In our analysis the domains and ranges of learning algorithms are what distinguish them from one another, because our results do not depend on the details of how learning algorithms find their hypotheses. The domain is a collection of $t$-tuples drawn from some set $X$, as we have said already. But $X$ is reflected in the domains and ranges of the hypotheses that the learning algorithm produces as output, so we will only need to know what kinds of hypotheses these are.

Therefore can regard a learning algorithm as an algorithm that chooses from among a number of hypotheses to find the hypothesis that performs best on the training examples. We say that the algorithm has a hypothesis class associated with it; this is just the set of hypotheses that the algorithm chooses among. In practice the hypothesis class is the set of hypotheses that the learning algorithm is able to generate (in other words, if $A$ is a learning algorithm, then $A$'s hypothesis class is simply the image of $X^t$ under $A$.)

We will sometimes wish to speak of a set of functions that is not necessarily connected to a particular learning algorithm; we will call such sets function classes as they are analogous to hypothesis classes.

When we discuss a particular hypothesis class, we will assume that all of the hypotheses in the class have the same domain, and likewise that they have the same range. This will permit us to speak of $\text{Dom}(\mathcal{H})$ and $\text{Ran}(\mathcal{H})$ when $\mathcal{H}$ is a hypothesis class; these two notations simply represent the common domain and range, respectively, of the functions in $\mathcal{H}$.

2.2. The Vapnik-Chervonenkis Dimension.

It is useful to characterize function classes by a combinatorial parameter known as their Vapnik-Chervonenkis Dimension, or simply VC Dimension. Many existing results on sample complexity are stated in terms of this parameter, and this chapter will present methods by which such results can be extended when the hypothesis class consists of syntactically complex hypotheses, such as expert systems.

If $X$ is a set, and $t$ is a non-negative integer, then we call any member of $X^t$ a list. In what follows, we will find a need to deal not only with sets but also with lists; the reason for this is that training examples are drawn with replacement, so that a training list is not generally a set. We will define a number of list operations. First, if $S$ is a list, we will say that $x \in S$ if and only if $x$ is an element of $S$. We also define the intersection of two lists as the set of elements that appear in both of the two lists; likewise we define the intersection of a list and a set as the set of elements that appear in both the list and the set. Formally:
**Definition:** Let $S_1$ and $S_2$ be sequences, and let $E$ be a set. Then

$$S_1 \cap S_2 \equiv \{ x : x \in S_1 \land x \in S_2 \},$$

$$S_1 \cap E \equiv \{ x : x \in S_1 \land x \in E \}.$$

When $S$ is a list we will use $|S|$ to denote the length of $S$.

To define the Vapnik-Chervonenkis Dimension of a hypothesis class, we first consider a set $\mathcal{X}$ (which could be a set of instances) and a set

$$\mathcal{A} = \{ A_1, A_2, A_3, \ldots \}$$

of subsets of $\mathcal{X}$.

Let $S$ be a list of elements in $\mathcal{X}$; for each element $E$ of $\mathcal{A}$, $S$ may be divided into those elements that lie in $S \cap E$ and those that do not.

Each set in $\mathcal{A}$ can be said to induce the subset $S \cap E$ of $S$, but the subsets need not all be unique. We will denote the number of unique subsets that $\mathcal{A}$ can induce in this way by

$$\Pi(\mathcal{A}, S),$$

that is,

$$\Pi(\mathcal{A}, S) = |\{ S \cap E : E \in \mathcal{A} \}|.$$

If every possible subset of $S$ is induced by one of the sets in $\mathcal{A}$, so that $\Pi(\mathcal{A}, S) = 2^{|\{ x : x \in S \}|}$, then $\mathcal{A}$ is said to shatter $S$. (We have used the notation $|\{ x : x \in S \}|$ to denote the number of unique elements in $S$; it is the size of the set of all elements that appear in the list $S$.)

We now define a second function which will be useful later: the maximum of $\Pi(\mathcal{A}, S)$ over all lists $S$ having a certain length $\ell$. If $\ell$ is a non-negative integer we define

$$\Pi(\mathcal{A}, \ell) = \max_{S \in \mathcal{X}^\ell} \Pi(\mathcal{A}, S).$$

Note that if $S$ is a list and $S'$ is the set of unique elements of $S$, then $\Pi(\mathcal{A}, S) = \Pi(\mathcal{A}, S')$.

We have expressed $\Pi$ in terms of lists because we will be interested in the number of subsets that can be induced on a list drawn from $S$. 
We will be interested in the size of the largest subset of $X$ that can be shattered by $\mathcal{F}$, because many results on sample complexity are expressed in terms of this number. It is referred to as the Vapnik-Chervonenkis Dimension of $\mathcal{F}$, and we will denote it by $\nu'(\mathcal{F})$:

$$\nu'(\mathcal{F}) = \max \left( \left\{ |\{x : x \in S\}| : S \subseteq X, \Pi_{\mathcal{F}}(S) = 2^{|\{x \in S\}|} \right\} \right).$$

An equivalent definition is

$$\nu'(\mathcal{F}) = \max \left\{ \ell \in \mathbb{N} : \Pi_{\mathcal{F}}(\ell) = 2^\ell \right\}.$$

Example 2.1: Linear discriminant functions in 2 dimensions

Let $\mathcal{F}$ be the set of functions

$$F(x, y) = \begin{cases} 1, & \text{if } \alpha x + \beta y + \gamma \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

Here $\alpha$, $\beta$, and $\gamma$ distinguish among the different functions in $\mathcal{F}$. Each function can be seen as a characteristic function on $\mathbb{R}^2$, and we will make $\mathcal{S}$ the set of sets that correspond to these functions.

Consider the set of elements

$${\{(0, 0), (0, 1), (1, 0), (1, 1)\}}.$$  

It is well known that $\mathcal{F}$ cannot induce the subsets

$${\{(0, 0), (1, 1)\}}$$

or

$${\{(0, 1), (1, 0)\}}$$

on this set; this is known as the "exclusive or" problem (Cf. Minsky 69). Therefore the VC Dimension of $\mathcal{F}$ is less than 4.

Example 2.2: Closed intervals on the number line.

Let $X$ be the set of real numbers, and let $\mathcal{S}$ be the set of closed intervals on the number line. In this example we show that $\nu'(\mathcal{S}) = 2$ as follows: consider 3 distinct points in $\mathbb{R}$, $a < b < c$. If this set of points is to be shattered by $\mathcal{S}$, then $\mathcal{S}$ must include an interval that includes $a$ and $c$ but not $b$. Clearly there is no such interval, so $\nu'(\mathcal{S}) \leq 2$. 


It is easy to verify that a set of 2 points is, indeed, shattered: assume two distinct points \( a \) and \( b \) such that \( a < b \), and choose three further points \( a_1 < a, a < a_2 < b, \) and \( b < a_3 \). Four distinct subsets of \( \{a, b\} \) are induced by the intervals \([-\infty, a_1], [a_1, a_2], [a_1, a_3], \) and \([a_2, a_3] \). Thus \( \nu(\mathcal{S}) = 2 \).

**Example 2.3: Linear discriminant functions in \( n \) dimensions**

We present the following theorem from [32] without proof:

**Theorem 2.4: [32]:** Let \( \mathcal{H} \) be the set of linear discriminant functions over \( n \) variables. Then \( \nu(\mathcal{H}) = n + 1 \).

It is important to note that if no subset of size \( \ell \) can be shattered, then no subset of size greater than \( \ell \) can be shattered, because such a subset would contain subsets of size \( \ell \), and all of these would be shattered as well.

In this thesis, as in most results pertaining to the VC Dimension of a set, \( X \) is a sample space and \( \mathcal{Y} \) is a set of events defined on that space. We are interested in events that tell us whether or not specific hypotheses err on specific elements of \( X \). Therefore, for each rule \( H \) in the hypothesis class \( \mathcal{H} \), we will define the event \( C(H) \) in \( X \) as:

\[
\{ z \in X : z^* \neq H(z^*) \}.
\]

this is the event in which \( H \) classifies the instance \( z^* \) incorrectly (recall that the correct classification is given by \( z^* \)). For a set of classification rules \( \mathcal{H} \), we will let \( C(\mathcal{H}) \) denote the set of events

\[
\{ C(H) : H \in \mathcal{H} \}.
\]

Several authors have presented results that use the Vapnik-Chervonenkis dimension of \( C(\mathcal{H}) \) to bound (in probability) the actual error of a hypothesis in terms of its empirical error. Such bounds are the cornerstone of our approach, and we state some of them here:
Theorem 2.4: Let $\mathcal{H}$ be a hypothesis class, let $X = \text{Dom}(\mathcal{H}) \times \text{Ran}(\mathcal{H})$, and let $d = V(C(\mathcal{H}))$.

(a) [32]:

$$P_{X} = \left\{ \exists (H \in \mathcal{H}) \text{ s.t. } |\phi_{X}(H) - \eta(H)| \geq \xi \right\}$$

is less than $\delta$ if

$$m \geq \frac{16}{\xi} \left( d \ln \frac{16d}{16} - \ln \frac{\delta}{4} \right).$$

(b) [7]:

$$P_{X} = \left\{ \exists (H \in \mathcal{H}) \text{ s.t. } (\eta(H) \geq \epsilon) \land \left( \frac{\eta(H, \mathcal{F})}{\eta(H)} \geq \xi \right) \right\}$$

is less than $\delta$ if

$$m \geq \max \left( \frac{8}{(1 - \xi)c}, \frac{8}{(1 - \xi)c}, \frac{16d}{(1 - \xi)c} \right).$$

(c) [7]: The probability that a hypothesis in $\mathcal{H}$ whose empirical error is zero will have an actual error greater than $\epsilon$ is bounded by $\delta$, if the training list contains at least

$$\max \left( \frac{4}{e \log_{2} \frac{2}{\delta}}, \frac{8}{e \log_{2} \frac{13}{\delta}} \right)$$

examples.

(d) [19]: If $X$ is countable then the probability that a hypothesis whose empirical error is zero will have an actual error greater than $\epsilon$ is bounded by $\delta$, if the training list contains at least

$$\frac{1}{\epsilon} \left( \log_{2}(|X|) d \ln(2) + \ln \left( \frac{1}{\delta} \right) \right)$$

examples.

(e) [19]: The probability that a hypothesis whose empirical error is zero will have an actual error greater than $\epsilon$ is bounded by $\delta$, if the training list contains at least

$$\frac{8}{\epsilon} \left( 4d \ln \left( \frac{32}{\epsilon} \right) + \ln \left( \frac{2}{\delta} \right) \right)$$

examples.

square
We now turn to the problem that is to be addressed in this chapter. The problem is this: if we have a learning algorithm whose hypothesis class is \( \mathcal{H} \), and we possess a syntactic description of \( \mathcal{H} \) (e.g., "\( \mathcal{H} \) is the set of boolean conjunctions over 12 variables"), how can we upper-bound \( \nu(\mathcal{H}) \), so that we may use theorem 2.4 to analyze the algorithm?

Suppose that \( \mathcal{H}_1 \) and \( \mathcal{H}_2 \) are two simple hypothesis classes whose Vapnik-Chervonenkis Dimensions are known to us. Also suppose that \( \mathcal{H}_3 \) is the set of hypotheses obtained by combining hypothesis from \( \mathcal{H}_1 \) with hypotheses from \( \mathcal{H}_2 \) in a certain way (for example, \( \mathcal{H}_3 \) might be defined as

\[
\{ H_3 : H_3(x) = H_1(x) \land H_2(x), H_1 \in \mathcal{H}_1, H_2 \in \mathcal{H}_2 \}
\]

or

\[
\{ H_3 : H_3(x) = H_1(x) \lor \neg H_2(x), H_1 \in \mathcal{H}_1, H_2 \in \mathcal{H}_2 \},
\]

e tc.) The results presented below will allow us to bound \( \nu(\mathcal{H}_3) \) in terms of \( \nu(\mathcal{H}_1) \) and \( \nu(\mathcal{H}_2) \). (Our results are not restricted to binary combinations, but we will discuss binary combinations first, as an introduction to the more general results that follow.)

### 2.3. Combining \( \{0, 1\} \)-valued hypothesis classes

Consider two \( \{0, 1\} \)-valued functions, \( F_1 \) and \( F_2 \), upon whose values we perform a boolean operation to obtain

\[
F_1(z) \Theta F_2(y),
\]

where \( \Theta \) is some binary boolean operator. If \( F_1 \) is to be chosen from the function class \( \mathcal{F}_1 \), and \( F_2 \) is to be chosen from \( \mathcal{F}_2 \), then the set of composite functions that can be created in this way is fully defined if \( \mathcal{F}_1, \mathcal{F}_2, \) and \( \Theta \) are fully defined. In this section we will show how to bound the VC Dimension of such a composite function class, provided we possess bounds on \( \nu(\mathcal{F}_1) \) and \( \nu(\mathcal{F}_2) \).

Our results apply for all sixteen choices of the boolean operator \( \Theta \). Therefore we will use the combination operator \( \Theta \), which combines \( \{0, 1\} \)-valued functions in the manner described above, as a representation for any arbitrary binary boolean operator. \( \Theta \) can be seen as a set of combination operators, one for each choice of \( \Theta \). Our results apply to all operators in this set. The notion of "combining" \( \{0, 1\} \)-valued functions is formalized as follows:

**Definition:** An anasic operator on the sample spaces \( X \) and \( Y \) is a mapping \( \Theta : 2^X \times 2^Y \rightarrow 2^{X \times Y} \), with the property that for \( L_1 \in X, L_2 \in Y \).

\[
\Theta(L_1, L_2) = \{(x, y) \in X \times Y : (x \in L_1) \Theta (y \in L_2) \}.
\]
where $\Theta$ is any fixed binary boolean operator (thus there are exactly sixteen anasic operators for a given $E_1$ and $E_2$, one for each choice of $\Theta$). If $\mathcal{F}_1$ and $\mathcal{F}_2$ are classes of events that are defined on $X$ and $Y$, respectively, then we use $\odot(\mathcal{F}_1, \mathcal{F}_2)$ to denote the set of events

$$\{\odot(E_1, E_2) : E_1 \in \mathcal{F}_1, E_2 \in \mathcal{F}_2\}$$

in $X \times Y$.

We will sometimes say $E_1 \odot E_2$ instead of $\odot(E_1, E_2)$, and $\mathcal{F}_1 \odot \mathcal{F}_2$ instead of $\odot(\mathcal{F}_1, \mathcal{F}_2)$.

We will informally refer to the the sort of function combinations described above as conjections; for example $E_1 \odot E_2$ is a conjection of $E_1$ and $E_2$.

Our first result is the following:

**Proposition 2.5:** Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two classes of events, and let $\odot$ be an anasic operator. Then, for all $\ell > 0$,

$$\Pi_{\mathcal{F}_1 \odot \mathcal{F}_2}(\ell) \leq \Pi_{\mathcal{F}_1}(\ell)\Pi_{\mathcal{F}_2}(\ell).$$

**Proof:** The proof is given in the appendix to this chapter. \hfill \Box

Anasic conjections therefore have a property that is extremely useful to us: $\Pi_{\mathcal{F}_1 \odot \mathcal{F}_2}(\cdot)$ can be expressed in terms of $\Pi_{\mathcal{F}_1}(\cdot)$ and $\Pi_{\mathcal{F}_2}(\cdot)$. The reason this property is useful is that $\Pi_{\mathcal{F}_1 \odot \mathcal{F}_2}(\cdot)$ can, in turn, be used to bound the Vapnik-Chervonenkis Dimension of $\mathcal{F}_1 \odot \mathcal{F}_2$ by means of the following result:

**Proposition 2.6:** [32] Let $\mathcal{F}$ be some class of events defined on a sample space $X$, and let $d$ denote $\vee(\mathcal{F})$. Then for all $\ell \geq d$, then

$$\Pi_{\mathcal{F}}(\ell) \leq \left(\frac{\ell\ell}{d}\right)^d.$$

\hfill \Box

Recall that the Vapnik-Chervonenkis Dimension of $\mathcal{F}_1 \odot \mathcal{F}_2$ is the largest integer $\ell$ for which

$$2^\ell = \Pi_{\mathcal{F}_1 \odot \mathcal{F}_2}(\ell).$$
By proposition 2.5 we can also say that it is the largest integer for which
\[ 2^t = \Pi_{A_\ell}(t) \Pi_{A_\ell}(t), \]
and, using proposition 2.6, we can say that any \( t \) satisfying
\[ 2^t > \left( \frac{c\ell}{V'(A_1)} \right)^{V'(A_1)} \left( \frac{c\ell}{V'(A_2)} \right)^{V'(A_2)} \]
is an upper bound on \( V'(A_1 \oplus A_2) \). We can use this inequality to obtain the following result:

Proposition 2.7: Consider two classes of events, \( A_1 \) and \( A_2 \), and let the combination operator \( \oplus \) be defined in such a way that
\[ \Pi_{A_1 \oplus A_2}(t) \leq \Pi_{A_1}(t) \Pi_{A_2}(t) \]
for all \( t > 0 \). Then
\[ V'(A_1 \oplus A_2) \leq 4.7(V'(A_1) + V'(A_2)). \]

Proof: The proof is given in the appendix to this chapter. \( \square \)

How can this result be used to bound the sample complexity of a learning algorithm? To answer this question let us first point out that any two-valued function \( H \) induces the subset \( \{ x : H(x) = 1 \} \) on its domain. If \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) are classes of \( \{0, 1\} \)-valued hypotheses, then proposition 2.7 not only holds for the classes of events \( C(\mathcal{M}_1) \) and \( C(\mathcal{M}_2) \), but also for the two classes of events
\[ \{ (x : H_1(x') = 1) : H_1 \in \mathcal{M}_1 \} \]
and
\[ \{ (y : H_2(y') = 1) : H_2 \in \mathcal{M}_2 \}. \]
When we discuss \( \{0, 1\} \)-valued hypotheses we will often treat them as if there were events, with the tacit understanding that the events in question are those induced by the hypotheses, in the sense just described. Thus, if \( \mathcal{M} \) is such a class of hypotheses, we will understand \( \Pi_{\mathcal{M}}(S) \) to be the number of subsets induced on \( S \) by the set of events \( \{ (x : H(x') = 1) : H \in \mathcal{M} \} \), and \( V'(\mathcal{M}) \) to be the Vapnik-Chervonenkis Dimension of the same set of events. Our existing results on \( V'(C(\mathcal{M})) \) also hold for \( V'(\mathcal{M}) \) in those cases, and in fact the following lemma implies that the two numbers are the same:
**Lemma 2.8:** [7]: Consider a set $X$ of examples, where $X$ is defined in such a way that, for all $x \in X$, $x^* \in \{0, 1\}$. For any set $\mathcal{N} \subseteq 2^X$, and all $I \in \Pi$, $\Pi_{\mathcal{N}}(I) = \Pi_{\{C(\mathcal{N}^c)\}}(I)$.

From the definition of the Vapnik-Chervonenkis Dimension, $\Pi_{\mathcal{N}}(I) = \Pi_{\{C(\mathcal{N}^c)\}}(I)$ implies that $V(\mathcal{N}^c) = V(C(\mathcal{N}^c))$.

In light of lemma 2.8, we can extend the definition of an anasic operator as follows:

**Definition:** If $\odot$ is some anasic operator mapping $2^X \times 2^Y$ to $2^{X \times Y}$, and $H_1$ and $H_2$ are $\{0, 1\}$-valued functions, then we define $H_1 \odot H_2$ to be the characteristic function for the set

$$\{x : H_1(x) = 1\} \odot \{y : H_2(y) = 1\}, \quad (x \in X, y \in Y).$$

If $\mathcal{M}_1$ and $\mathcal{M}_2$ are classes of $\{0, 1\}$-valued functions, then $\mathcal{M}_1 \odot \mathcal{M}_2$ is defined as the class of characteristic functions

$$\{H_1 \odot H_2 : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2\}.$$

The intent of this definition is to enable us to treat $\{0, 1\}$-valued functions as if they were events, insofar as it enables us to conjunct them using anasic operators.

We are now in a position to formally define what we mean by conjuction operators.

**Definition:** Let $\odot$ be a $k$-ary function whose domain is

$$\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k \times \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k \quad (k \in \Pi),$$

and whose range is $\mathcal{X}$. Let $H_1, H_2, \ldots, H_k$ be $k$ functions whose respective domains are $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$, and whose respective ranges are $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$. Then

$$\odot(H_1, H_2, \ldots, H_k) : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k \rightarrow \mathcal{X}$$

is defined to be the function which, for all $(x_1, x_2, \ldots, x_k) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k$, takes $(x_1, x_2, \ldots, x_k)$ to $\odot(H_1(x_1), H_2(x_2), \ldots, H_k(x_k), x_1, x_2, \ldots, x_k)$.

Any function that can be constructed in this way is a conjuction of $H_1, H_2, \ldots, H_k$. (The function $\odot$ will be referred to as a conjuction operator in this context.)
A more general definition for functions that combine other functions will also be useful:

**Definition:** Let \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \) be \( k \) functions whose respective domains are \( X_1, X_2, \ldots, X_k \). If

\[
\circledast(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)
\]

is any function whose domain is a superset of \( X_1 \times X_2 \times \cdots \times X_k \), then \( \circledast(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k) \) is a combination of \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \). \( \circledast \) will be referred to as a combination operator in this context.

We will usually use the concept of a combination operator informally, and we will use it with the implication that the value of \( \circledast(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k) \) at any \((x_1, x_2, \ldots, x_k) \in X_1 \times X_2 \times \cdots \times X_k \) is in some way related to the values \( \mathcal{M}_1(x_1), \mathcal{M}_2(x_2), \ldots, \mathcal{M}_k(x_k) \).

### 2.4. Learning rule bases

We consider a **rule base** to be a collection of rules each having the form

\[
F \Rightarrow I,
\]

where \( F \) and \( I \) are functions and "\( \Rightarrow \)" denotes logical implication. We apply a rule base to an input \( x \) as follows: for each rule

\[
F_i \Rightarrow I_i
\]

in the rule base, if the predicate \( F_i \) holds of \( x \), then we conclude that the predicate \( I_i \) also holds of \( x \).

Our approach to "learning" rule bases from examples is the following: for each possible rule

\[
F_i \Rightarrow I_i,
\]

we assume that each training example either disproves the implication \( F_i \Rightarrow I_i \), or fails to disprove it. The learning process will consist of coming up with a set of rules that are not disproved by any of the training examples.

To implement this, we will regard "\( \Rightarrow \)" as a boolean operator with the following truth table:

<table>
<thead>
<tr>
<th>( F_i(x) )</th>
<th>( I_i(x) )</th>
<th>( F_i(x) \Rightarrow I_i(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
When we are finished learning, we ask "what is the probability of drawing an input \( y \) that will falsify one of our rules?" If this probability is low, we are fairly safe in using our rules to perform diagnoses on randomly drawn input data.

We are therefore in the same situation that was described at the beginning of this chapter. We have an empirical estimate of the probability with which our rules err, and we wish to know how close the estimate is to the actual probability. If we know the VC Dimension of the set of possible rules, we can use theorem 2.4 to bound the true probability of making an error.

Example 2.9: Simple Rule bases

Suppose that an expert system consists of rules that have the form

\[ F \iff I, \tag{2.4} \]

where \( F \) is a \( \{0, 1\} \)-valued function known to come from the class \( \mathcal{F} \), \( I \) is a \( \{0, 1\} \)-valued function known to come from the class \( \mathcal{F} \), and \( \iff \) denotes logical implication. Since \( \iff \) is an anasic operator in this case, proposition 2.7 tells us that the VC Dimension of the class of rules (2.4) is bounded by

\[ 4.7(\gamma'(\mathcal{F}) + \gamma'(\mathcal{J})). \tag{2.5} \]

In real expert systems, rules often have the form

\[ F_1 \land F_2 \land \cdots \land F_k \iff I, \tag{2.6} \]

so we would like to know the VC Dimension of the set of functions whose form is

\[ F_1 \land F_2 \land \cdots \land F_k, \tag{2.7} \]

assuming \( F_1 \) is in some class \( \mathcal{F}_1 \), \( F_2 \) is in \( \mathcal{F}_2 \), and so on. If we can bound this parameter we can use the result to bound the VC Dimension of the class of rules described in (2.6).

But boolean conjunction is an anasic operator as well. If we conject the functions in (2.7) pairwise we can establish that the Vapnik-Chervonenkis Dimension of the conjunction is bounded as follows:

\[ \gamma'(\mathcal{F}) \leq 4.7\log_k(k) \sum_{i=1}^{k} \gamma'(\mathcal{F}_i); \tag{2.8} \]
combining (2.6) and (2.5) we see that the VC Dimension of the class of rules described by (2.6) is less than

$$4.7^k + 4.7^{|\mathcal{F}|} \sum_{i=1}^k \nu(\mathcal{F}_i).$$

Finally, an expert system usually contains many rules; in this simple example we can say that the system is a disjunction of rules like the one in (2.6). Because disjunction is an anomic operator we can proceed in the same way as before, and obtain a bound on the VC Dimension of a hypothesis class whose members are simple expert systems. * Specifically, if there are to be $n$ rules in the system, then the VC Dimension of the system is bounded by

$$4.7^{|\mathcal{F}|} \left( 4.7^{|\mathcal{F}_1|} \sum_{i=1}^k \nu(\mathcal{F}_i) + \nu(\mathcal{F}) \right).$$

The remainder of this chapter will be devoted to refining this approach and extending it to hypothesis classes more complex than the one just described. We begin by extending proposition 2.7 to $k$-ary conjunction operators.

2.4.1. $k$-ary conjunction operators.

Definition: Consider $k$ sets $X_1, X_2, \cdots, X_k$. An operator $\otimes: 2^{X_1} \times 2^{X_2} \times \cdots \times 2^{X_k} \rightarrow 2^{X_1 \times X_2 \times \cdots \times X_k}$ is said to be anomic if, for any $E_1 \subseteq X_1, E_2 \subseteq X_2, \cdots, E_k \subseteq X_k$,

$$\otimes(E_1, E_2, \cdots, E_k) = \{ (x_1, x_2, \cdots, x_k) \in X_1 \times X_2 \times \cdots \times X_k : \Theta(x_1 \in E_1, x_2 \in E_2, \cdots, x_k \in E_k) \},$$

where $\Theta$ is a $k$-ary $(0,1)$-valued function.

If $H_1, H_2, \cdots, H_k$ are $(0,1)$-valued functions on $X_1, X_2, \cdots, X_k$ respectively, then $\otimes(H_1, H_2, \cdots, H_k)$ is the characteristic function of the set

$$\otimes(\{x_1 : H_1(x_1) = 1\}, \{x_2 : H_2(x_2) = 1\}, \cdots, \{x_k : H_k(x_k) = 1\}).$$

If $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_k$ are classes of $(0,1)$-valued functions, then $\otimes(\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_k)$ is the class of functions $\{ \otimes(H_1, H_2, \cdots, H_k) : H_1 \in \mathcal{H}_1, H_2 \in \mathcal{H}_2, \cdots, H_k \in \mathcal{H}_k \}$. 

* Of course, we ultimately need a numerical value for the VC Dimension of some set of functions. This thesis will not derive any significant new results in that area, but some previous results are given in the appendix.
Here, as before, the intent is to ensure that when we speak of anasic compositions, the classes of \([0,1]\)-valued functions \(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\) and \(\circ(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)\) can be used interchangeably with the classes of events \(\{\{x: H_1(x') = 1\}: H_1 \in \mathcal{M}_1\}, \{\{x: H_2(x') = 1\}: H_2 \in \mathcal{M}_2\}, \ldots, \{\{x: H_k(x') = 1\}: H_k \in \mathcal{M}_k\}\), and

\[
\{\{x_1, x_2, \ldots, x_k\}: \circ(H_1, H_2, \ldots, H_k)(x_1, x_2, \ldots, x_k) = 1\} : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2, \ldots, H_k \in \mathcal{M}_k
\]

respectively.

The results we derive below apply not only to anasic conjections, but to a broader class of combinations which we now define:

**Definition:** Consider a set of function classes \(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\), whose ranges are \(\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_k\) respectively. The combination

\[
\circ(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)
\]

is ricetic if and only if

\[
\Pi_{\circ(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)}(t) \leq \Pi_{\mathcal{M}_1}(t)\Pi_{\mathcal{M}_2}(t)\cdots\Pi_{\mathcal{M}_k}(t)
\]

for all \(t > 0\).

By reasoning similar to that we used for binary anasic operators, we can show that \(k\)-ary anasic operators are ricetic:

**Proposition 2.10:** Consider \(k\) classes of \([0,1]\)-valued functions \(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\). Let \(\circ\) be a \(k\)-ary anasic operator. Then, for all \(t > 0\),

\[
\Pi_{\circ(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)}(t) \leq \Pi_{\mathcal{M}_1}(t)\Pi_{\mathcal{M}_2}(t)\cdots\Pi_{\mathcal{M}_k}(t).
\]
Proof: The proof is given in the appendix to this chapter. □

As was the case with binary anasic operators, this result allows us to bound the Vapnik Chervonenkis Dimension of $k$-ary anasic conjections. In conjunction with proposition 2.6, it tells us that if

$$2^d > \left( \frac{\ell}{\sqrt[\nu]{|\mathcal{A}|}} \right)^{\nu(\mathcal{A})} \left( \frac{\ell}{\sqrt[\nu]{|\mathcal{B}|}} \right)^{\nu(\mathcal{B})} \cdots \left( \frac{\ell}{\sqrt[\nu]{|\mathcal{M}|}} \right)^{\nu(\mathcal{M})},$$

then $\ell$ is an upper bound on the VC Dimension of $\oplus(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{M})$.

In the $k$-ary case, unlike the binary case, our result will be a combination of several bounds.

**Proposition 2.11:** Consider $k$ classes of events, $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$, and let $\oplus$ be a monotonic conjunctive operator.

Then

$$\nu(\oplus(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \leq \Upsilon_k \left( \sum_{i=1}^k \nu(\mathcal{M}_i) \right),$$

where

$$\Upsilon_k(d) \equiv \begin{cases} d, & \text{if } k = 1 \\ 4.7d, & \text{if } k = 2, \\ 2d \log_2(ek), & \text{if } 3 \leq k \leq 7 \text{ or } 11 < k < 13 \\ kd, & \text{if } 8 \leq k < 10 \\ d(1 + \ln(ek) + \ln(1 + \ln(ek)))/\ln(2), & \text{if } k \geq 14. \end{cases}$$

Proof: The proof is given in the appendix to this chapter. □

**Example 2.12:** Circuits that perform boolean operations

In this example, we consider functions like the one depicted in figure 1: there are $k$ input functions $F_1, F_2, \ldots, F_k$, each of which maps its inputs to $\{0, 1\}$. These functions produce $k$ values which are to be conjuncted by a boolean operator $Bf$ to produce the output of the system.

Because we need to know in advance what class of functions is to be learned, we will specify $Bf$ while leaving the choice of $F_1, F_2, \ldots, F_k$ to the learning algorithm, within the restriction that $F_1$ is in the hypothesis class $\mathcal{M}_1$, $F_2$ is in the hypothesis class $\mathcal{M}_2$, and so on. For the sake of completeness suppose $Bf$ is the $k$-ary parity function. Then our learning algorithm chooses from the class of functions

$$\{\text{Parity}(F_1, F_2, \ldots, F_k) : F_1 \in \mathcal{M}_1, F_2 \in \mathcal{M}_2, \ldots, F_k \in \mathcal{M}_k\}.$$
Figure 1: A circuit for calculating a \((0,1)\)-valued function.

Because \(B\)'s fixed, we can regard it as an anasic operator on the input functions. By theorem 2.12, the Vapnik-Chervonenkis Dimension of the circuit is no more than

\[ T(k \left( \sum_{i=1}^{b} f_i(x_i) \right) ) . \]

Suppose that each input is represented as a vector of values,

\[ x = (v_1, v_2, \ldots, v_n) , \]

with each element giving the value of some variable that the hypothesis uses.

Also, suppose that each input function is a linear discriminant function of its inputs, that is, each of \( F_1, F_2, \ldots, F_k \) is a function of the form

\[ F(x) = \begin{cases} 1, & \text{if } a_1 v_1 + a_2 v_2 + \cdots + a_n v_n \geq \theta \\ 0, & \text{otherwise} \end{cases} \]  

(2.9)

where the constants \( a_i \in \mathbb{R} \) are to be determined by the learning algorithm. It is shown in various places (such as [32]) that the VC Dimension of the class (2.9) is \( n + 1 \). Thus the VC-Dimension of the class of circuits

\[ \{ \text{Parity}(F_1, F_2, \ldots, F_k) : F_i \in \mathcal{F}_1, F_2 \in \mathcal{F}_2, \ldots, F_k \in \mathcal{F}_k \} \]

is less than

\[ T(k(n + 1)) . \]
Let us now turn again to rule-based systems. A rule-based system consists of a set of rules, which have the form

\[ F \Rightarrow I, \]

where \( F \) and \( I \) are functions. In this context, \( F \) is sometimes referred to as the rule's *antecedent*, and \( I \) is its *consequent*. The symbol \( \Rightarrow \) informally denotes logical implication, but the exact semantics of the operator varies between rules and systems. Rather than using a different symbol each time, we will use the generic implication arrow \( \rightarrow \), and explain its semantics when we use it.

We have already examined systems where the rules take the form

\[ F_1 \land F_2 \land \ldots \land F_k \Rightarrow I, \]

where \( \Rightarrow \) is analytic.

However, it is often the case that the set of possible diagnoses partitions the input domain, in the sense that the rule base never returns the two diagnoses at the same time. For example, suppose that the three possible diagnoses are *feather_rust_syndrome*, *luggage_warp* and *tunnel_diode_occlusion*. Our assumption is that each possible set of symptoms is associated with at most one of *feather_rust_syndrome*, *luggage_warp* or *tunnel_diode_occlusion*, but that no set of symptoms is associated with more than one diagnosis.

For the sake of compactness, let \( \mathcal{S} = \{ I_1, I_2, \ldots, I_k \} \) be the set of possible diagnoses, and assume they have the property that

\[ \forall (x \in X) : \forall (1 \leq i \leq j) : (I_i(x) = 1 \Rightarrow \exists (i \leq j \leq k, j \neq i) : I_j(x) = 1). \quad (2.10) \]

Now consider any set of two points, \( \{x_1, x_2\} \). If \( x_1 \) and \( x_2 \) imply the same diagnosis (say \( I_i \)), then there is no other diagnosis \( I_j \neq I_i \) with the property that \( I_j(x_1) = 1 \) but \( I_j(x_2) = 0 \). Therefore, \( \mathcal{S} \) does not induce the subset \( \{x_1\} \) on \( \{x_1, x_2\} \).

On the other hand, if \( x_1 \) and \( x_2 \) do *not* imply the same diagnosis, then there is no single diagnosis that can be given for both \( x_1 \) and \( x_2 \). In that case, \( \mathcal{S} \) does not induce the subset \( \{x_1, x_2\} \) on \( \{x_1, x_2\} \).

Recall that \( \mathcal{S} \) is said to shatter a set of points only if it induces all possible subsets on that set. We have just shown that \( \mathcal{S} \) cannot shatter any set of two points. Also recall that the Vapnik-Chervonenkis Dimension of \( \mathcal{S} \) is the size of the largest set that can be shattered by \( \mathcal{S} \). Since no set of size two can be shattered by \( \mathcal{S} \), the VC Dimension of \( \mathcal{S} \) can be no greater than 1.
Consider the class of rules

$$F \Rightarrow I,$$

where $F$ comes from a hypothesis class whose Vapnik-Chervonenkis Dimension $d$, where $I$ comes from a hypothesis class like the one just described, and where $\Rightarrow$ is an operator. By proposition 2.7 the VC Dimension of this class is at most $4.7(d + 1)$.

Example 2.13: Simple Homo clauses in a diagnostic system.

Suppose we have a system whose rules are all of the form

$$F_1 \land F_2 \land \ldots \land F_k \Rightarrow I,$$

where, in each rule, $F_1$ is from the set $\mathcal{F}_1$, $F_2$ is from the set $\mathcal{F}_2$, and so on, and where $I$ is in all cases from the set $\mathcal{I}$ of possible diagnoses. If $\mathcal{I}$ has the property described in (2.10), then the VC Dimension of this class of rules is then less than

$$4.7 \left( 1 + \mathcal{T}_k \left( \sum_{i=1}^{k} \mathcal{V}'(\mathcal{F}_i) \right) \right).$$

(This follows from proposition 2.11.)

Let us generalize slightly and consider an arbitrary class of diagnoses that can induce no more than $w$ subsets on a set of training examples, instead of only two as in the examples above. Any set of $w$ diagnoses fits this description trivially even if two diagnoses can be returned at the same time, since $w$ events cannot induce more than $w$ subsets on any set regardless of any other circumstances. Let $\mathcal{H}_1$ be any hypothesis class, and $\mathcal{H}_2$ be a hypothesis class that can only induce $w$ subsets on any training list. If $\otimes$ is an anasic operator, $d = \mathcal{V}(\mathcal{H}_1)$ and $\ell \geq d$, then $(\mathcal{H}_1 \otimes \mathcal{H}_2)$ can induce only

$$w \left( \frac{\ell^d}{d} \right)$$

subsets on a set of size $\ell$ by proposition 2.5. This is useful because of the following result:

**Proposition 2.14:** Let $\mathcal{H}_3$ be a class of hypotheses with the property that

$$\Pi_{\mathcal{H}_3}(S) \leq w$$

for any $S$ and some $w$, and let $\otimes$ be an anasic operator. For any hypothesis class $\mathcal{H}_4$ having VC Dimension $d$, the Vapnik-Chervonenkis Dimension of $(\mathcal{H}_3 \otimes \mathcal{H}_4)$ is bounded above by

$$\frac{(15.1)d}{\epsilon^{-\ln(w)/d}}.$$
if $\log_2(w)/d \leq 0.5$, and by

$$\frac{8.2}{e} \log_2(w)$$

if $\log_2(w)/d \geq 0.5$.

Proof: The proof is given in the appendix of this chapter. □

2.5. Discussion

This chapter has served as an introduction to the enterprise of analyzing a machine learning algorithm's sample complexity. We have approached this problem by examining the class of hypotheses that the learning algorithm can generate, and by decomposing the members of this class into simpler hypotheses.

The major results of this chapter were proposition 2.5, proposition 2.7, proposition 2.10, proposition 2.11, and proposition 2.14. The second and third of these will be subsumed by more powerful results in the next chapter. They were included to exemplify our approach to sample complexity analysis without using the more complicated notation that will be needed in the next chapter. Proposition 2.5 is included in proposition 2.11, which is the major result of this chapter in the sense that we will continue to use it to bound the VC Dimensions of rich conjunctive hypotheses.

Proposition 2.14, on the other hand, typifies a collection of somewhat ad hoc results that we will continue to see in this thesis. This happens because, (at least to the knowledge of the author) no previous attempt has been made to systematically develop analysis methods for pre-existing learning algorithms under the pac model. The approach is therefore very much in its infancy and many special cases tend to arise during its implementation.

The next chapter will remedy the greatest shortcoming of proposition 2.10, which is that it cannot handle functional compositions. If a learning algorithm outputs hypotheses of the form $F_1(F_2(x))$, then it cannot be analyzed with proposition 2.10, at least not in terms of $F_1$ and $F_2$. However, the next chapter will show that composition is also a rich conjunctive operation. In addition to this we will deal with a number of miscellaneous issues, among these the matter of hypothesis whose ranges are not isomorphic to $\{0,1\}$.

2.6. Bibliographic notes

A paper published in 1984, [30], is often cited as the seminal paper in computational learning theory. This was the paper that proposed analysis of learning algorithms both in terms of generalization ability and computational complexity. The idea of using statistical methods for pattern recognition is older, but the appearance of [30] seems to catalyzed the widespread use of these methods in machine learning.
Overviews of the use of statistical methods in pattern recognition can be found in [9] and [33]. The latter book, in particular, contributed much of the material that was adopted by researchers in machine learning. It particularly emphasizes nonparametric methods, that is, methods that do not make assumptions about probability distributions. The formalism introduced by [30] specified that no such assumptions were to be made.

In the early years of computational learning theory much of the literature was concentrated on the computational complexity of learning algorithms. There were numerous results in which specific learning problems were shown to be computationally intractable, for example [31]. [5], [20], [27], and [6]. The result of [18] is also noteworthy in this context although it did not use the framework of computational learning theory.

The results that are of greater interest to us are those concerning sample complexity, that is, the number of examples needed for learning. A noteworthy early result was [6], in which the sample complexity of a learning algorithm was bounded in terms of the size of the hypotheses in the algorithm's domain. This approach is a straightforward generalization of other results that use the domain size itself to bound an algorithm's sample complexity; some are given in [33]. In general, the framework of [30] requires that a learning algorithm's sample complexity, and not just its computational complexity, grow only polynomially in terms of the various parameters that describe the learning problem. Because of this, positive results that use Valiant's framework usually address the issue of sample complexity. A large number of these present specific algorithms that are subsequently analyzed: [12] gives a neural network algorithm, [31] presents an algorithm for learning disjunctive normal formulae with a bounded number of variable in each minterm, [11] discusses the learning of decision trees whose size is bounded, [29] addresses the learning of decision lists, which are decision trees where each node has only one non-leaf among its descendants. Some results, such as [11], [31], and [6], which discusses the learning of disjunctive normal formulae with a bounded number of minterms, treat the bounds on the sizes of hypotheses as constants (that is, the number of minterms in [6], the number of variables per minterm in [31], and the size of the decision tree in [11]) are considered to be constants). These size-bounds appear as exponents in the expressions describing the learning algorithms' sample complexities, but, by virtue of the fact that the size-bounds are constants rather than variables, the sample complexities are regarded as polynomial expressions.

Other results, such as that of [23], which presented a sample complexity bound for learning disjunctive normal formulae with a bounded number of minterms and a bounded number of variables per minterm, have sample complexities that are polynomial in the size-bounds, and are therefore of a qualitatively different nature than those cited previously.

The results that are of primary interest to us are not those that present analyses of specific algorithms, but rather those that permit large classes of algorithms to be analyzed for sample complexity if they have a certain property.
In [33] and [6], which we have already cited, it is shown that sample complexity grows logarithmically in the size of the learning algorithm’s domain, and linearly in the size of the hypotheses in the domain.

In [32], [33], [7], and [19], among others, the sample complexity of algorithms learning a class of hypotheses is bounded in terms of the Vapnik-Chervonenkis Dimension of the class; this is discussed at length in the body of the thesis and the current appendix. There is an additional important result in this area: [32] shows that the sample complexity of a learning algorithm cannot be bounded in the worst-case framework implied by [30], unless the Vapnik-Chervonenkis Dimension of the algorithm’s hypothesis class is finite. (It should be noted by the interested reader that [32] and [33] do not express their results in the terminology of machine learning. Instead, they simultaneously bound the probabilities of events contained in a given set, in terms of an empirical estimate of those probabilities. In the language of machine learning, the set of events in question contains the events that cause one or more hypotheses in some class to make an error, and the empirical estimate is obtained by counting the number of errors the hypotheses make on the training sample. It is suggested that the reader approach [7] or [19] before [32] or [33].)

Results that use the Vapnik-Chervonenkis Dimension to bound the sample complexity of a learning algorithm make it appear easy, at least in principle, to bound the sample complexity of an arbitrary learning algorithm (if, in fact, it can be bounded). The problem is that we must first have a bound on the VC dimension on the algorithm’s own hypothesis class. The usual approach in the literature on computational learning theory seems to be to present an algorithm and then bound the VC Dimension of it’s hypothesis class; with this approach the analysis of each algorithm is a new research project unless there are uniform methods for finding the VC Dimensions of function classes.
Appendix A to Chapter 2.
Proofs of results in Chapter 2

2A.1. Proof of proposition 2.5.

In what follows the next result will be useful:

Lemma 2A.15: Let \( \mathcal{E} \) be some class of events defined on a sample space \( X \), and let \( d \) denote \( \mathcal{V}(\mathcal{E}) \). If \( S \subseteq X \) and \( |S| \geq d \), then

\[
\Pi_S(S) \leq \left( \frac{\epsilon|S|}{d} \right)^d.
\]

\[\square\]

Corollary 2A.16: Let \( \mathcal{E} \) be some class of events defined on a sample space \( X \), and let \( d \) denote \( \mathcal{V}(\mathcal{E}) \). Then for all \( t \geq d \), then

\[
\Pi_S(t) \leq \left( \frac{\epsilon t}{d} \right)^d.
\]

Proof: This follows because lemma 2A.15 holds for all \( S \subseteq X \) for which \( t \geq d \).

\[\square\]

This result bounds the number of subsets that can be induced on a list whose size is greater than or equal to \( d \) by a class of events whose Vapnik-Chervonenkis Dimension is less than or equal to \( d \).

We now demonstrate how the Vapnik-Chervonenkis Dimension of the class of events \( \mathcal{E}_1 \cup \mathcal{E}_2 \) can be bounded, when \( \mathcal{V}(\mathcal{E}_1) \) and \( \mathcal{V}(\mathcal{E}_2) \) are known. We begin by bounding \( \Pi_{\mathcal{E}_1 \cup \mathcal{E}_2}(S_1 \cup S_2) \) in terms of \( \Pi_{\mathcal{E}_1}(S_1) \) and \( \Pi_{\mathcal{E}_1}(S_2) \). Our first case is a simple one in which all events in \( \mathcal{E}_1 \) induce the same subset on \( S_1 \). In this case, the elements of \( S_2 \) do not provide enough information to distinguish among the members of \( \mathcal{E}_2 \), since every element of \( S_2 \) is either excluded from every event in \( \mathcal{E}_2 \), or included in every event in \( \mathcal{E}_2 \). We expect that \( \Pi_{\mathcal{E}_1 \cup \mathcal{E}_2}(S_1 \cup S_2) \) would only depend on \( \Pi_{\mathcal{E}_1}(S_1) \), since \( \Pi_{\mathcal{E}_1}(S_2) \) is fixed at 1. We will show that this is indeed the case; specifically

\[
\Pi_{\mathcal{E}_1 \cup \mathcal{E}_2}(S_1 \cup S_2) = \Pi_{\mathcal{E}_1}(S_1).
\]
Definition: Let $S_1$ and $S_2$ be two lists such that $|S_1| = |S_2|$, and let $\mapsto: S_1 \rightarrow S_2$ be a bijection. $S_1 \circ \cdot S_2$ denotes the list of tuples 
\[
\{(x_1, \mapsto (x_1)), (x_2, \mapsto (x_2)), \ldots, (x_n, \mapsto (x_n))\}.
\]

The goal in using the $\circ \cdot$ operation is simply to specify some list of tuples 
\[
\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}, \tag{2A.11}
\]
where $x_i \in S_1$ and $y_i \in S_2$, in such a way that the number of elements of the list is the same as $|S_1| = |S_2|$, while at the same time specifying which of the pairs $(x, y) \in S_1 \times S_2$ appear in (2A.11). The exact nature of $\mapsto$ is unimportant, so we will simply use $S_1 \circ \cdot S_2$ to denote (2A.11), leaving $\mapsto$ unspecified.

We now demonstrate how the Vapnik-Chervonenkis Dimension of the class of events $\sigma_1 \oplus \sigma_2$ can be bounded, when $\nu'(\sigma_1)$ and $\nu'(\sigma_2)$ are known. We begin by bounding $\Pi_{\sigma_1 \oplus \sigma_2}(S_1 \circ \cdot S_2)$ in terms of $\Pi_{\sigma_1}(S_1)$ and $\Pi_{\sigma_2}(S_2)$. Our first case is a simple one in which all events in $\sigma_2$ induce the same subset on $S_2$. In this case, the elements of $S_2$ do not provide enough information to distinguish among the members of $\sigma_2$, since every element of $S_2$ is either excluded from every event in $\sigma_2$, or included in every event in $\sigma_2$. We expect that $\Pi_{\sigma_1 \oplus \sigma_2}(S_1 \circ \cdot S_2)$ would only depend on $\Pi_{\sigma_2}(S_1)$, since $\Pi_{\sigma_2}(S_2)$ is fixed at 1. We will show that this is indeed the case; specifically 
\[
\Pi_{\sigma_1 \oplus \sigma_2}(S_1 \circ \cdot S_2) = \Pi_{\sigma_2}(S_1).
\]

We begin with a preliminary lemma.

Lemma 2A.17: Consider two events $E_{1,1}$ and $E_{1,2}$ defined on the sample space $X_1$ and two events $E_{2,1}$ and $E_{2,2}$ defined on the sample space $X_2$. Let $S_1$ be a list of elements of $X_1$ and $S_2$ be a list from $X_2$, such that $|S_1| = |S_2|$, let $\circ \cdot$ be an anasic operator, and assume that 
\[
S_2 \cap E_{2,1} = S_2 \cap E_{2,2},
\]
then 
\[
(S_1 \circ \cdot S_2) \cap E_{1,1} \cap E_{1,2} \neq (S_1 \circ \cdot S_2) \cap E_{1,2} \cap E_{2,1}
\]
implies that 
\[
S_1 \cap E_{1,1} \neq S_1 \cap E_{1,2}.
\]
Proof:

Assume

\[(S_1 \cdot S_2) \cap E_{1,1} \cap E_{2,1} \neq (S_1 \cdot S_2) \cap E_{1,2} \cap E_{2,2}.\]

We will also assume

\[S_1 \cap E_{1,1} = S_1 \cap E_{1,1}\]

and show that a contradiction follows.

\[\exists (x, y) \in S_1 \cdot S_2:
\]

\[\begin{align*}
((x, y) \notin E_{1,1} \cap E_{2,1} & \land (x, y) \notin E_{1,2} \cap E_{2,2}) \\
\lor \\
((x, y) \notin E_{1,1} \cap E_{2,1} & \land (x, y) \notin E_{1,2} \cap E_{2,2})
\end{align*}\]

Proof of 1: [We show this by assuming there is no such point and showing that a contradiction follows.]

\[\forall (x, y) \in S_1 \cdot S_2:
\]

\[\begin{align*}
((x, y) \notin E_{1,1} \cap E_{2,1} & \land (x, y) \notin E_{1,2} \cap E_{2,2}) \\
\land \\
((x, y) \notin E_{1,1} \cap E_{2,1} & \land (x, y) \notin E_{1,2} \cap E_{2,2})
\end{align*}\]

\[\forall (x, y) \in S_1 \cdot S_2:
\]

\[\begin{align*}
((x, y) \notin E_{1,1} \cap E_{2,1} & \rightarrow (x, y) \notin E_{1,2} \cap E_{2,2}) \\
\land \\
((x, y) \notin E_{1,1} \cap E_{2,1} & \rightarrow (x, y) \notin E_{1,2} \cap E_{2,2})
\end{align*}\]

\[\forall (x, y) \in S_1 \cdot S_2:
\]

\[\begin{align*}
((x, y) \notin E_{1,1} \cap E_{2,1} & \rightarrow (x, y) \notin E_{1,2} \cap E_{2,2}) \\
\land \\
((x, y) \notin E_{1,1} \cap E_{2,1} & \rightarrow (x, y) \notin E_{1,2} \cap E_{2,2})
\end{align*}\]

Q.E.D. (1) [1.3] contradicts the original assumption that

\[(S_1 \cdot S_2) \cap E_{1,1} \cap E_{2,1} \neq (S_1 \cdot S_2) \cap E_{1,2} \cap E_{2,2}.\]
[2] Consider a particular point \((x, y) \in S_1 \cap S_2\) such that \\
\((x, y) \in E_{1,1} \cap E_{2,1} \land (x, y) \notin E_{1,2} \cap E_{2,2}\)

[ such a point exists by [1] ]

\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
\n
[3] Case 1: assume

\((x, y) \in E_{1,1} \cap E_{2,1} \land (x, y) \notin E_{1,2} \cap E_{2,2}\)

\[3.1\] 
\((x, y) \in E_{1,1} \cap E_{2,1} \land (x, y) \notin E_{1,2} \cap E_{2,2}) \quad \text{by assumption}

\[3.2\] 
\((x, y) \in E_{1,1} \cap E_{2,1} \rightarrow (x \in E_{1,1}) \cap (y \in E_{2,1}) \quad \text{[this follows from the definition of an analytic operator]}

\[3.3\] 
\((x, y) \notin E_{1,1} \cap E_{2,1} \rightarrow \neg((x \in E_{1,1}) \cap (y \in E_{2,1})) \quad \text{[this follows from the definition of an analytic operator]}

\[3.4\] 
\((x \in E_{1,1}) \cap (y \in E_{2,1}) \land \neg((x \in E_{1,1}) \cap (y \in E_{2,1})) \quad \text{[by [3.1], [3.2], and [3.3]]}

\[3.5\] 
\(S_1 \cap E_{2,1} = S_2 \cap E_{2,2} \quad \text{[by the assumption of the lemma.]}

\[3.6\] 
\(y \in E_{2,1} \leftarrow y \in E_{2,2} \quad \text{[by [3.5], since } y \in S_2 \text{ according to [2]]}

\[3.7\] 
\((x \in E_{1,1} \land x \in E_{1,2}) \rightarrow (x \in E_{1,1}) \cap (y \in E_{2,1}) \land (x \in E_{1,2}) \cap (y \in E_{2,2})\)
Proof of 3.7: We will assume 
\[ (x \in E_{1,1} \land x \in E_{1,2}) \]
and show that under this assumption,
\[ \neg ((x \in E_{1,1}) \lor (y \in E_{2,1}) \land (x \in E_{1,2}) \lor (y \in E_{2,3})) \]
lead to a contradiction.

[3.7.1]
\[ x \in E_{1,1} \land x \in E_{2,1} \quad \text{[by assumption]} \]

[3.7.2]
\[ \neg ((x \in E_{1,1}) \lor (y \in E_{2,1}) \land (x \in E_{1,2}) \lor (y \in E_{2,3})) \quad \text{[by assumption]} \]

[3.7.3]
Case 1: Assume \((x \in E_{2,1})\)

[3.7.3.1]
\[ 1 \odot (y \in E_{2,1}) \quad \text{[by assumption]} \]

[3.7.3.2]
\[ y \in E_{2,1} \iff y \in E_{2,2} \quad \text{[by [3.6]]} \]

[3.7.3.3]
\[ 1 \odot (y \in E_{2,1}) \land 1 \odot (y \in E_{2,2}) \quad \text{[3.7.3.2 and 3.7.3.1]} \]

[3.7.3.4]
\[ x \in E_{1,1} \land x \in E_{2,1} \quad \text{[by [3.7.1]]} \]

[3.7.3.5]
\[ \neg [(1 \odot (y \in E_{2,1}) \land (1 \odot (y \in E_{2,2})) \land (1 \odot (y \in E_{2,3}))] \quad \text{[by [3.7.3.4 and [3.7.2]]} \]

Q.E.D. 3.7.3.3 \[ [3.7.3.5] \text{contradicts [3.7.3.3]} \]
A contradiction follows; the proof is exactly analogous to the proof for case 1.

Case 2: Assume \( \sim(1 \circ (y \in F_{2,1})) \)

Q.E.D. (3.7) \( \text{By } [3.7.4] \text{ and } [3.7.3], [3.7.2] \) contradicts both \( 1 \circ (y \in F_{2,1}) \) and \( \sim(1 \circ (y \in F_{2,1})) \).

\[
\begin{align*}
\{x \notin E_{1,1} \land x \notin E_{1,2}\} \rightarrow \\
(x \in E_{1,1}) \circ (y \in E_{2,1}) \land \\
(x \in E_{1,2}) \circ (y \in E_{2,2})
\end{align*}
\]

Proof of 3.8: We will assume \( \{x \notin E_{1,1} \land x \notin E_{1,2}\} \)

and show that under this assumption,

\[
\sim((x \in E_{1,1}) \circ (y \in E_{2,1}) \land (x \in E_{1,1}) \circ (y \in E_{2,1}))
\]

leads to a contradiction.

Q.E.D. (3.8) \( \text{The proof is exactly analogous to the proof of } [3.7] \).

\[
\begin{align*}
x \in E_{1,1} \rightarrow x \in E_{1,2}
\end{align*}
\]

[This holds for all \( x \in S_{1} \) from the initial assumption of the proof, namely \( S_{1} \cap E_{1,1} = S_{1} \cap E_{1,2} \).]

\[
\begin{align*}
x \in E_{1,1} \land x \in E_{1,2} \
\lor \\
(x \notin E_{1,1} \land x \notin E_{1,2}) \text{ by } [3.9]
\end{align*}
\]

[3.10]

\[
\begin{align*}
x \in E_{1,1} \land x \in E_{1,2} \
\lor \\
x \in E_{2,1} \lor (y \in E_{2,1}) \
\land \\
(x \in E_{1,2}) \circ (y \in E_{2,2}) \text{ by } [3.10], [3.7] \text{ and } [3.8]
\end{align*}
\]

Q.E.D. (3.3) \( [3.11] \) contradicts the assumption in [3].

[4] Case 2: \( \{x, y\} \notin E_{1,1} \circ E_{2,1} \land (x, y) \in E_{1,2} \circ E_{2,2} \)

Q.E.D. (3.3) \( [5] \) contradicts [1].

**Theorem 2A.18:** Consider two classes of events, $\mathcal{A}_1$ and $\mathcal{A}_2$, defined on the sample spaces $X_1$ and $X_2$ respectively. Let $S_1$ be a list of elements of $X_1$ and $S_2$ be a list from $X_2$, such that $|S_1| = |S_2|$, and assume that $\mathcal{A}_2$ has the property that all events in $\mathcal{A}_2$ induce the same subset on $S_2$. Let $\odot$ be an anasic operator. Then

$$\Pi_{\mathcal{A}_1 \odot \mathcal{A}_2}(S_1 \odot S_2) \leq \Pi_{\mathcal{A}_1}(S_1) = \Pi_{\mathcal{A}_1}(S_1) \Pi_{\mathcal{A}_2}(S_2). \quad (2A.12)$$

**Proof:** Construct a mapping

$$M : \{(S_1 \odot S_2) \cap E_1 \odot E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \rightarrow \{S_1 \cap E_1 : E_1 \in \mathcal{A}_1\}$$

as follows: for all $E_1 \in \mathcal{A}_1$, $E_2 \in \mathcal{A}_2$,

$$M((S_1 \odot S_2) \cap E_1 \odot E_2) = S_1 \cap E_1. \quad (2A.13)$$

Either (2A.13) is one-to-one or it is not. If (2A.13) is one-to-one then $\{S_1 \cap E_1 : E_1 \in \mathcal{A}_1\}$ has at least as many members as

$$\{(S_1 \odot S_2) \cap E_1 \odot E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\}$$

and the theorem follows. If (2A.13) is not one-to-one then a contradiction follows, as we now show:

11) $\exists E_1 \in \mathcal{A}_1, E_{2,1}, E_{2,2} \in \mathcal{A}_2 :$

$$((S_1 \odot S_2) \cap E_1 \odot E_{2,1} \neq (S_1 \odot S_2) \cap E_1 \odot E_{2,2})$$
Proof of 1: [We will assume that no such point exists and show that a contradiction follows.]

1.1 \( \forall E_1 \in A_1, E_2, E_2' \in \mathcal{A}_2 : \)

\[
(S_1 \cdot S_2) \cap E_1 \cap E_2' \neq (S_1 \cdot S_2) \cap E_1 \cap E_2,
\]

[by assumption]

(\text{this is the negation of 1.1})

1.2 \( \forall E_1 \in A_1, E_2, E_2' \in \mathcal{A}_2 : \)

\[
(S_1 \cdot S_2) \cap E_1 \cap E_2' \neq (S_1 \cdot S_2) \cap E_1 \cap E_2' \rightarrow
M((S_1 \cdot S_2) \cap E_1 \cap E_2' \neq E_{1,j})
\]

\[
M( (S_1 \cdot S_2) \cap E_1 \cap E_2' ) \neq E_{1,j}
\]

[The implication holds trivially since no \( E_1 \in A_1, E_2, E_2' \in \mathcal{A}_2 \) satisfy the left side.]

1.3 \[ \text{by [1.2] and the definition of a one-to-one mapping} \]

\[ M \text{ is one-to-one.} \]

Q.E.D. (1) \[ [1.3] \text{contradicts the assumption that } M \text{ is not one-to-one.} \]

2 \[ \exists E_1 \in A_1 : \]

\[ S_1 \cap E_1 \neq S_1 \cap E_1' \]

Proof of 2:

2.1 \[ \text{Let } E_1, E_2, \text{ and } E_2' \]

be such that \[ \text{Such an } E_1, E_2, E_2' \text{ exist by [1].} \]

\[
(S_1 \cdot S_2) \cap E_1 \cap E_2' \neq (S_1 \cdot S_2) \cap E_1 \cap E_2',
\]

2.2 \[ S_1 \cap E_1 \neq S_1 \cap E_1' \]

[by lemma 2A.17]

Q.E.D. (2) \[ [by [2.2]] \]

Q.E.D. \[ [2] \text{ is self-contradictory.} \]

Hence (2A.13) must be one-to-one and the theorem follows. \( \square \)
Corollary 2A.19: Consider two classes of events, \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \), defined on the sample spaces \( X_1 \) and \( X_2 \) respectively. Let \( S_1 \) be a list of elements of \( X_1 \) and \( S_2 \) be a list from \( X_2 \), such that \(|S_1| = |S_2|\), and assume that \( \mathcal{A}_1 \) is such that all events in \( \mathcal{A}_1 \) induce the same subset on \( S_1 \). Let \( \otimes \) be an analytic operator. Then

\[
\Pi_{\mathcal{A}_1 \otimes \mathcal{A}_2} (S_1 \cdot S_2) \leq \Pi_{\mathcal{A}_1} (S_2) = \Pi_{\mathcal{A}_1} (S_1) \Pi_{\mathcal{A}_2} (S_2).
\]

Proof: The proof is exactly analogous to the proof of theorem 2A.18. (The corollary differs from theorem 2A.18 in that the roles of \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) are reversed). \( \square \)

Next we consider the case where \( \Pi_{\mathcal{A}_1} (S_2) \geq 1 \):

Lemma 2A.20: Let \( \mathcal{A}_1 \) be a class of events in \( X \), and \( \mathcal{A}_2 \) be a class of events in \( Y \). Let \( S_1 \) be a list of elements \( \{x_1, x_2, \ldots, x_k\} \) from \( X \), \( S_2 \) be a list of elements \( \{y_1, y_2, \ldots, y_k\} \) in \( Y \), and let \( S_1 \cdot S_2 \) be

\[
\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}.
\]

If \( \otimes \) is a analytic operator then

\[
\Pi_{\mathcal{A}_1 \otimes \mathcal{A}_2} (S_1 \cdot S_2) \leq \Pi_{\mathcal{A}_1} (S_1) \Pi_{\mathcal{A}_2} (S_2).
\]

Proof: We partition \( \mathcal{A}_1 \) by grouping together those members that induce the same subset on \( S_1 \). That is, we create partitions

\[
\mathcal{A}_{1,1}, \mathcal{A}_{1,2}, \ldots, \mathcal{A}_{1,k},
\]

where

\[
F_i \in \mathcal{A}_{1,k} \land F_j \in \mathcal{A}_{1,k} \implies F_i \cap S_1 = F_j \cap S_1.
\]
\[ \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} = \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \cup \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_2, E_2 \in \mathcal{A}_1\} \cup \ldots \cup \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \]

[by the definition of a partitioning]

\[ \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \right| \leq \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \right| + \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_2, E_2 \in \mathcal{A}_1\} \right| + \ldots + \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \right| \]

[by \([1]\) and the properties of sets]

\[ \forall \mathcal{A}_i \in \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\} : \]
\[ \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \right| \leq \Pi_{\mathcal{A}}(S_2) \]

[by theorem 2A.18, since each \(\mathcal{A}_i\) satisfies the condition placed on \(\mathcal{A}_1\) in that theorem]

\[ \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \right| \leq \Pi_{\mathcal{A}}(S_1) \]

[by the definition of \(\Pi\)]

\[ [2] \text{ has at most } \Pi_{\mathcal{A}}(S_1) \text{ summands} \]

[by \([4]\)]

\[ \left| \{(S_1 \cdot S_2) \cap E_1 \cap E_2 : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2\} \right| \leq \Pi_{\mathcal{A}}(S_1) \Pi_{\mathcal{A}}(S_2) \]

Q.E.D.

\[ \square \]

Lemma 2A.21: Consider an anasic operator \(\circ\) and two sets of events \(\mathcal{A}_1\) and \(\mathcal{A}_2\), defined on \(X_1\) and \(X_2\) respectively.

If, for all lists \(S_1\) from \(X_1\), and all lists \(S_2\) from \(X_2\) such that \(|S_1| = |S_2|\),

\[ \Pi_{\mathcal{A} \circ \mathcal{A}}(S_1 \cdot S_2) \leq \Pi_{\mathcal{A}}(S_1) \Pi_{\mathcal{A}}(S_2) \]

then

\[ \Pi_{\mathcal{A} \circ \mathcal{A}}(t) \leq \Pi_{\mathcal{A}}(t) \Pi_{\mathcal{A}}(t) \]

for all \(t > 0\).
Proof:

[1] \( \forall \ell > 0, \exists S_1, S_2 \in X \ni \Pi A \triangle A (S_1 \cdot S_2) = \Pi A \triangle A (\ell) \)  \quad  \text{[definition of } \Pi A (\ell) \text{]}  

[2] Let \( \ell > 0 \)

[3] Let \( S_1, S_2 \) be such that \( \Pi A \triangle A (S_1 \cdot S_2) = \Pi A \triangle A (\ell) \)  \quad  \text{[Such } S_1, S_2 \text{ exist by [1]}  

[4] \( \Pi A \triangle A (\ell) \leq \Pi A (S_1) \Pi A (S_2) \)  \quad  \text{[by lemma 2A.20]}  

[5] \( \Pi A (S_1) \leq \Pi A (\ell) \)  \quad  \text{[by the definition of } \Pi A (\ell) \text{ and the fact that } |S_1| = \ell \]  

[6] \( \Pi A (S_2) \leq \Pi A (\ell) \)  \quad  \text{[by the definition of } \Pi A (\ell) \text{ and the fact that } |S_2| = \ell \]  

[7] \( \Pi A \triangle A (\ell) \leq \Pi A (\ell) \Pi A (\ell) \)  \quad  \text{[by [4], [5], and [6]}  

Q.E.D.

2A.2. Proof of proposition 2.7.

The following lemma will be used in two of our later proofs:

Lemma 2A.22: Let \( C \) be an arbitrary positive constant. Then

\[
\min \left\{ d_1^2 d_2^2 : d_1 + d_2 = C, d_1, d_2 \in \Pi \right\} = \left( \frac{C}{2} \right)^2.
\]

This minimum is unique.
Proof: Let
\[ d_0 = \frac{C}{2}. \]

[1] \[
\min \left\{ d_1^2 d_2^2 : d_1 + d_2 = C, d_1, d_2 \in \mathbb{N} \right\} \leq \min \left\{ (d_0 + x)^{d_0+x} (d_0 - x)^{d_0-x} : x \in \mathbb{N} \right\}.
\]

Proof of 1: (We will assume the existence of an \( x \) that causes [1] to be violated, and show that a contradiction follows.)

[1.1] Let \( g_1^p g_2^p = \min \left\{ d_1^2 d_2^2 : d_1 + d_2 = C, d_1, d_2 \in \mathbb{N} \right\} \)

[1.2] Choose an \( x \) such that \( (d_0 + x)^{d_0+x} (d_0 - x)^{d_0-x} < g_1^p g_2^p \)

[1.3] Let \( h_1 = d_0 + x, h_2 = d_0 - x \)

[1.4] \( h_1^p h_2^p < g_1^p g_2^p \) \[ \text{by [1.2] and [1.3].} \]

[1.5] \( h_1 + h_2 = C \)

Proof of 1.5:

[1.5.1] \( h_1 + h_2 = d_0 + x + d_0 - x \) \[ \text{by the definition of } h_1 \text{ and } h_2. \]

[1.5.2] \( d_0 + x + d_0 - x = d_0 + d_0 \) \[ \text{by arithmetic} \]

[1.5.3] \( d_0 + d_0 = \frac{C}{2} + \frac{C}{2} \) \[ \text{definition of } d_0 \]

[1.5.4] \( \frac{C}{2} + \frac{C}{2} = C \) \[ \text{by algebra} \]

Q.E.D. (1.5) \[ \text{by the chain of equalities [1.5.1]-[1.5.4]} \]
\[ (1) \quad g_1^j, g_2^j \neq \]
\[ \min \left\{ \min \left\{ \begin{array}{c} \min \{ d_1^i, d_1^j : d_1 + d_2 = C, d_1, d_2 \in \mathbb{N} \} \\ \min \{ d_2^i, d_2^j : d_1 + d_2 = C, d_1, d_2 \in \mathbb{N} \} \end{array} \right\} \right\} \]

Q.E.D. \[ (1) \quad \] contradicts \[ (1.1) \quad \]

\[ (2) \quad \min \{ (d_0 + x)^{d_0} (d_0 - x)^{d_0 - x} : x \in \mathbb{N} \} \leq \min \{ d_1^i, d_2^j : d_1 + d_2 = C, d_1, d_2 \in \mathbb{N} \} \]

Proof of 2: We will assume the existence of a \( d_1 \) and a \( d_2 \) that violate \( (2) \), and show that a contradiction follows.

\[ (2.1) \quad \]

Let \( x_0 \) be such that
\[ (d_0 + x_0)^{d_0} (d_0 - x_0)^{d_0 - x_0} = \min \{ (d_0 + x)^{d_0} (d_0 - x)^{d_0 - x} : x \in \mathbb{N} \} \]

\[ (2.2) \quad \]

Let \( g_1 \) and \( g_2 \) be such that
\[ g_1 + g_2 = C \]
and
\[ g_1^{g_2} g_2^{g_1} < (d_0 + x_0)^{d_0} (d_0 - x_0)^{d_0 - x_0} \]

\[ (2.3) \quad \]

Let \( x_1 = g_1 - d_0 \)

\[ (2.4) \quad \]

\[ d_0 + x_1 = g_1 \]

\[ (2.5) \quad \]

\[ d_0 - x_1 = g_2 \]

Proof of 2.5:

\[ (2.5.1) \quad \]

\[ g_1 + g_2 = C \]

by the assumption \( (2.2) \)

\[ (2.5.2) \quad \]

\[ d_0 + d_0 = C \]

by definition of \( d_0 \)

\[ (2.5.3) \quad \]

\[ d_0 + d_0 = g_1 + g_2 \]

by \( (2.5.1) \) and \( (2.5.2) \)
\[ d_0 + d_0 = d_0 + z_1 + z_2 \]  
[ substituting [2.4] into [2.5.3] ]

\[ d_0 - z_1 = z_2 \]  
[ [2.5] by algebra ]

Q.E.D. (2.5)

\[ g_1^T g_2^T \leq (d_0 + x_0)^{d_0 + x_0} \]  
[ by the assumption in [2.2] ]

\[ (d_0 + x_1)^{d_0 + x_1} \]  

\[ (d_0 + x_0)^{d_0 + x_0} \neq \]  
[ 2.7 is a counterexample ]

\[ \min \{(d_0 + z)^{d_0 + z}(d_0 - x)^{d_0 - z} : x \in \mathbb{N}\} \]  
Q.E.D. (2)  
[ [2.8] contradicts the definition of \( x_0 \). ]

\[ \forall (x_0 \geq 0) : \]

\[ (\ln(d_0 + x_0) + 1 - \ln(d_0 - x_0) - 1 = 0) \rightarrow \]

\[ (d_0 + x_0)^{d_0 + x_0}(d_0 - x_0)^{d_0 - x_0} = \]

\[ \min \{(d_0 + z)^{d_0 + z}(d_0 - x)^{d_0 - z} : x \in \mathbb{N}\} \]

Proof of 3: [Let \( x_0 \) be a number satisfying the left side of [3.1]. We will show that \( x_0 \) satisfies the right side.]

\[ (d_0 + x_0)^{d_0 + x_0}(d_0 - x_0)^{d_0 - x_0} = \]

\[ \min \{(d_0 + z)^{d_0 + z}(d_0 - x)^{d_0 - z} : z \in \mathbb{N}\} \]

\[ \forall \]

\[ (d_0 + x_0)^{d_0 + x_0}(d_0 - x_0)^{d_0 - x_0} = \]

\[ \max \{(d_0 + x)^{d_0 + x}(d_0 - x)^{d_0 - x} : x \in \mathbb{N}\} \]

Proof of 3.1: [We prove this be taking the derivative of the logarithm of]

\[ (d_0 + x)^{d_0 + x}(d_0 - x)^{d_0 - x} \]

with respect to \( x \).]
\[3.1.1\]
\[\forall (x): \left( (d_0 + x)^{\delta + \gamma + \epsilon} (d_0 - x)^{\delta - \gamma} \right) = \min \left\{ \left( (d_0 + x)^{\delta + \gamma} (d_0 - x)^{\delta - \gamma} : x \in \mathbb{N} \right) \right\} \]
\[
\ln \left( (d_0 + x)^{\delta + \gamma + \epsilon} (d_0 - x)^{\delta - \gamma} \right) = \min \left\{ \ln \left( (d_0 + x)^{\delta + \gamma} (d_0 - x)^{\delta - \gamma} : x \in \mathbb{N} \right) \right\}
\]
\[
\text{[by the monotonicity of the log function]}
\]

\[3.1.2\]
\[\forall (x): \left( (d_0 + x)^{\delta + \gamma + \epsilon} (d_0 - x)^{\delta - \gamma} \right) = \min \left\{ \left( (d_0 + x)^{\delta + \gamma} (d_0 - x)^{\delta - \gamma} : x \in \mathbb{N} \right) \right\} \]
\[
(d_0 + x) \ln(d_0 + x) (d_0 - x) \ln(d_0 - x) = \min \left\{ (d_0 + x) \ln(d_0 + x) (d_0 - x) \ln(d_0 - x) : x \in \mathbb{N} \right\}
\]
\[
[3.1.1 \text{ and algebra.}]
\]

\[3.1.3\]
\[\forall (x): \left( (d_0 + x)^{\delta + \gamma + \epsilon} (d_0 - x)^{\delta - \gamma} \right) = \max \left\{ \left( (d_0 + x)^{\delta + \gamma} (d_0 - x)^{\delta - \gamma} : x \in \mathbb{N} \right) \right\} \]
\[
(d_0 + x) \ln(d_0 + x) (d_0 - x) \ln(d_0 - x) = \max \left\{ (d_0 + x) \ln(d_0 + x) (d_0 - x) \ln(d_0 - x) : x \in \mathbb{N} \right\}
\]
\[
[\text{The proof is exactly analogous to that of } (3.1.2).]
\]

\[3.1.4\]
\[
\frac{d}{dx} \left[ (d_0 + x) \ln(d_0 + x) + (d_0 - x) \ln(d_0 - x) \right] = \\
\frac{d}{dx} \left( (d_0 + x) \ln(d_0 + x) + (d_0 + x) \frac{d \ln(d_0 + x)}{dx} \right) + \\
\frac{d}{dx} \left( (d_0 - x) \ln(d_0 - x) + (d_0 - x) \frac{d \ln(d_0 - x)}{dx} \right) = \\
\ln(d_0 + x) + \frac{d_0 + x}{d_0 + x} - \ln(d_0 - x) - \frac{d_0 - x}{d_0 - x} = \\
\ln(d_0 + x) + 1 - \ln(d_0 - x) - 1
\]
\[
[\text{by calculus and algebra.}]
\]

\[3.1.5\]
\[
\frac{d}{dx} \left( (d_0 + x) \ln(d_0 + x) + (d_0 - x) \ln(d_0 - x) \right) \text{ evaluated at } x_0 = 0
\]
\[
[\text{By } (3.1.4) \text{ and the assumption in } (3) \text{ that } \ln(d_0 + x_0) + 1 - \ln(d_0 - x_0) - 1 = 0.]
\]

Q.E.D. (3.1) [The claim follows from (3.1.2), (3.1.3), (3.1.5), and the properties of the derivative.]

\[3.2\]
\[
(d_0 + x_0)^{\delta + \gamma + \epsilon} (d_0 - x_0)^{\delta - \gamma} = \\
\min \left\{ \left( (d_0 + x)^{\delta + \gamma} (d_0 - x)^{\delta - \gamma} : x \in \mathbb{N} \right) \right\}
\]
Proof of 3.2: [We prove this by taking the second derivative of the logarithm of
\[(d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z}\]
with respect to \(x\).]

[3.2.1]
\[\forall x : ((d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z} = \min \{(d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z} : x \in \mathbb{I}\}\] \[\text{by the monotonicity of the log function}\]

[3.2.2]
\[\forall x : ((d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z} = \min \{(d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z} : x \in \mathbb{I}\}\] \[\text{by calculus and algebra, as above.}\]

[3.2.3]
\[\forall x : ((d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z} = \max \{(d_0 + x)^{d_0 + x}(d_0 - z)^{d_0 - z} : x \in \mathbb{I}\}\] \[\text{by calculus and algebra, as above.}\]

[3.2.4]
\[\frac{d^2}{dx^2} \left[ (d_0 + x) \log(d_0 + x) + (d_0 - x) \log(d_0 - x) \right] = (d_0 + x)^{-1} + (d_0 - x)^{-1}\]

Proof of 3.2.4:

[3.2.4.1]
\[\frac{d}{dx} \left[ (d_0 + x) \log(d_0 + x) + (d_0 - x) \log(d_0 - x) \right] = \log(d_0 + x) + 1 - \log(d_0 - x) - 1\]
[by calculus and algebra, as above.]

[3.2.4.2]
\[\frac{d}{dx} \log(d_0 + x) + 1 - \log(d_0 - x) - 1\]
\[= \frac{d \log(d_0 + x)}{dx} - \frac{d \log(d_0 - x)}{dx}\]
\[= \frac{1}{d_0 + x} + \frac{1}{d_0 - x}\]
[by calculus]
Q.E.D. (3.2.4)

\[ (d_0 + x)^{-1} + (d_0 - x)^{-1} \geq 0 \quad (d_0 \geq x \geq 0) \]

Proof of 3.2.5: [We will show that the contrary assumption contradicts \( d_0 > x > 0 \).]

- [3.2.5.1] \( d_0 > 0 \) [The definition of \( d_0 \)]
- [3.2.5.2] \( x \geq 0 \) [the assumption in [3.2.5]]
- [3.2.5.3] \( (d_0 + x)^{-1} \geq 0 \) [3.2.5.1 and 3.2.5.2]
- [3.2.5.4] \( ((d_0 + x)^{-1} + (d_0 - x)^{-1} < 0) \rightarrow \)
  \( (d_0 - x)^{-1} < 0 \) [by 3.2.5.3]
- [3.2.5.5] \( (d_0 - x)^{-1} < 0 \) \( \rightarrow \) \( (x > d_0) \)
- [3.2.5.6] \( ((d_0 + x)^{-1} + (d_0 - x)^{-1} < 0) \rightarrow \)
  \( (x > d_0) \) [3.2.5.4 and 3.2.5.5]
- [3.2.5.7] \( x \leq d_0 \) [by assumption]
- [3.2.5.8] \( (d_0 + x)^{-1} + (d_0 - x)^{-1} \geq 0 \) \( \rightarrow \)
  \( (d_0 + x)^{-1} + (d_0 - x)^{-1} \geq 0 \) [3.2.5.7]

Q.E.D. (3.2.5)

- [3.2.6] \( x_0 < d_0 \) [by the definition of \( x_0 \), \( x_0 \) is defined: this requires that \( x_0 < d_0 \)]
- [3.2.7] \( x_0 \geq 0 \) [by the definition of \( x_0 \)]

Q.E.D. (3.2) [by 3.1, 3.2.4, 3.2.5, 3.2.6, 3.2.7, and the properties of the derivative]

Q.E.D. (3) [3.2] is the assertion that was to be proved in step [3]
Proof of 4:

\[ r_0 = 0 \]  
\[ \text{Assumption} \]

\[ \ln(d_0 + x_0) + 1 - \ln(d_0 - x_0) - 1 = \ln(d_0 + 0) + 1 - \ln(d_0 - 0) - 1 \]

\[ \text{By [4.2], substituting 0 for } x. \]

\[ \ln(d_0 + 0) + 1 - \ln(d_0 - 0) - 1 = 0 \]  
\[ \text{algebra} \]

Q.E.D. (4)  
\[ \text{By [4.1] and [4.3].} \]

Proof of 5:

\[ (x_1 \neq 0) \rightarrow (\ln(d_0 + x_1) + 1 - \ln(d_0 - x_1) - 1 \neq 0) \]

\[ \ln(d_0 + x_1) + 1 - \ln(d_0 - x_1) - 1 = \ln(d_0 + x_1) - \ln(d_0 - x_1) \]

\[ \text{algebra} \]

\[ (x_1 \neq 0) \rightarrow (d_0 + x_1 \neq d_0 - x_1) \]

\[ (d_0 + x_1 \neq d_0 - x_1) \rightarrow (\ln(d_0 + x_1) \neq \ln(d_0 - x_1)) \]

\[ \text{monotonicity of the log function} \]

\[ (\ln(d_0 + x_1) \neq \ln(d_0 - x_1)) \rightarrow (\ln(d_0 + x_1) - \ln(d_0 - x_1) \neq 0) \]

\[ (x_1 \neq 0) \rightarrow (\ln(d_0 + x_1) - \ln(d_0 - x_1) \neq 0) \]

\[ [5.2] \text{through [5.4]} \]

Q.E.D. (5)  
\[ [5.5] \text{and [5.1].} \]
The only \( x_0 \) satisfying left side of (3) is 0

\[
\min \left\{ (d_0 + x)^{d_0 - x} : x \in \mathbb{N} \right\}
\]

Q.E.D.  | [7], [8] and the definition of \( d_0 \).

\[\square\]

**Theorem 2A.23:** Consider two classes of events, \( \delta_1 \) and \( \delta_2 \), defined on the sets \( X \) and \( Y \) respectively, and let the combination operator \( \odot \) be defined in such a way that

\[
\Pi_{\delta_1 \odot \delta_2}(\ell) \leq \Pi_{\delta_1}(\ell)\Pi_{\delta_2}(\ell)
\]

for all \( \ell > 0 \). Then

\[
\forall (\delta_1 \odot \delta_2) \leq 4.7(\forall (\delta_1) + \forall (\delta_2)).
\]

**Proof:**

(1) By the definition of the Vapnik-Chervonenkis Dimension.

\[
2^\ell \geq \Pi_{\delta_1 \odot \delta_2}(\ell)
\]

implies that

\[
\ell \geq \forall (\delta_1 \odot \delta_2).
\]

We will show that (2A.14) is satisfied if \( \ell \) is replaced by \( 4.7(\forall (\delta_1) + \forall (\delta_2)) \).

[1] \[
\Pi_{\delta_1 \odot \delta_2}(\ell) \leq \Pi_{\delta_1}(\ell)\Pi_{\delta_2}(\ell) \quad \text{[by assumption]}
\]

[2] \[
\text{Let } h_1 = \forall (\delta_1), h_2 = \forall (\delta_2)
\]

[3] \[
\forall \ell > \max\{h_1, h_2\}:
\]

\[
\Pi_{\delta_1}(\ell)\Pi_{\delta_2}(\ell) \leq \left(\frac{\ell^{h_1}}{h_1}\right)^{h_1}\left(\frac{\ell^{h_2}}{h_2}\right)^{h_2} \quad \text{[by corollary 2A.16]}
\]
∀ ℓ > \max\{h_1, h_2\}:
\Pi_{h_1}(\ell)\Pi_{h_2}(\ell) ≤ \frac{e^{h_1h_1}e^{h_2h_2}}{h_1^{h_1}h_2^{h_2}} \quad \{ \text{[3] and algebra} \}

Let
\[ h_0 = \frac{h_1 + h_2}{2} \]

∀ ℓ,
\[ \frac{e^{h_0h_0}e^{h_0h_0}}{h_0^{h_0}h_0^{h_0}} ≥ \frac{e^{h_1h_1}e^{h_2h_2}}{h_1^{h_1}h_2^{h_2}} \]

Proof of 6:

[6.1]
\[ e^{h_0h_0}e^{h_0h_0} = e^{h_1h_1}e^{h_2h_2} \]

Proof of 6.1:

[5.1.1]
\[ e^{h_1h_1}e^{h_2h_2} = (e^\ell)^{h_1}(e^\ell)^{h_2} = (e^\ell)^{h_1+h_2} = (e^\ell)^{h_0+h_0/2} = (e^\ell)^{h_0}(e^\ell)^{h_0} = e^{h_0h_0}e^{h_0h_0} \]

[6.2]
\[ h_0^{h_0}h_0^{h_0} ≤ h_1^{h_1}h_2^{h_2} \]

Q.E.D. (6.1)

Q.E.D. (6) \{ by [6.1], [6.2], and algebra \}

∀ ℓ > \max\{h_1, h_2\},
\[ \frac{e^{h_0h_0}e^{h_0h_0}}{h_0^{h_0}h_0^{h_0}} ≥ \Pi_{h_1}(\ell)\Pi_{h_2}(\ell) \]

[8]
\[ (\frac{e^\ell}{h_0})^{2h_0} ≥ \Pi_{h_1}(\ell)\Pi_{h_2}(\ell) \]

[ by [7] and algebra ]
\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ \left( \frac{\ell}{h_0} \right)^{2h_0} \geq \Pi_{\delta_1 \cup \delta_2}(\ell) \]  \hfill \{\text{[18] and [1]}\}

\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ 2^\ell > \Pi_{\delta_1 \cup \delta_2}(\ell) \Rightarrow \ell > r(\delta_1 \cup \delta_2) \]  \hfill \{\text{By the definition of } r.\}

\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ \left( 2^\ell \geq \left( \frac{\ell}{h_0} \right)^{2h_0} \right) \Rightarrow \left( \ell \geq r(\delta_1 \cup \delta_2) \right) \]  \hfill \{\text{[19] and [10]}\}

\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ \left( \ell \geq 2h_0 \log_2 \left( \frac{\ell}{h_0} \right) \right) \Rightarrow \left( \ell \geq r(\delta_1 \cup \delta_2) \right) \]  \hfill \{\text{[11], by the monotonicity of the log function}\}

\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ \left( \frac{\ell}{h_0} \geq 9.4 \right) \Rightarrow \left( \ell \geq r(\delta_1 \cup \delta_2) \right) \]  \hfill \{\text{[12] and algebra}\}

Proof of 14:

\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ \left( \frac{\ell}{h_0} \geq 9.4 \right) \Rightarrow \left( \frac{\ell}{2h_0} \geq \log_2 \left( \frac{\ell}{h_0} \right) \right) \]

Proof of 14.1:

\[ \forall \ell > \max\{h_1, h_2\}, \]
\[ \left( \frac{\ell}{h_0} \geq 9.4 \right) \Rightarrow \left( \frac{\ell}{2h_0} \geq \log_2 \left( \frac{\ell}{h_0} \right) \right) \]

Proof of 14.1.1:

\[ \frac{\ell}{h_0} \geq 9.4 \]  \hfill \{\text{Assumption}\}

Proof of 14.1.2:

\[ \frac{9.4}{2} \geq \log_2(9.4 \times) \]  \hfill \{\text{arithmetic}\}
Proof of 14.1.3: [We take the derivative with respect to \( \ell / h \) of the difference of the left and right sides of 14.1.3, and show that it is positive. The result then follows for all \( \ell/h > 9.4 \), since 14.1.1 shows that it holds when \( \ell/h = 9.4 \).]

\[
[14.1.3.1] \quad \frac{d}{d(\ell/h)} \frac{\ell}{2h} = \frac{1}{2} \quad \text{[By calculus]}
\]

\[
[14.1.3.2] \quad \frac{d}{d(\ell/h)} \log_2 \left( \frac{\ell}{h} \right) = \frac{h d \ell - \ell d(h/\ell) h}{h \ell d(\ell/h) h} = \frac{eh}{e\ell} \quad \text{[calculus and algebra]}
\]

\[
[14.1.3.3] \quad \frac{h}{\ell} \leq \frac{1}{9.4} \quad \text{[By 14.1.1].}
\]

\[
[14.1.3.4] \quad \frac{d}{d(\ell/h)} \log_2 \left( \frac{\ell}{h} \right) \leq \frac{1}{9.4} \quad \text{[by 14.1.3.2] and [14.1.3.3].}
\]

\[
[14.1.3.5] \quad \frac{d}{d(\ell/h)} \frac{\ell}{2h} - \frac{d}{d(\ell/h)} \log_2 \left( \frac{\ell}{h} \right) = \frac{1}{2} - \frac{1}{9.4} \quad \text{[subtracting [14.1.3.4] from [14.1.3.1].]}
\]

Q.E.D. (14.1.3) [By 14.1.2], 14.1.3.5, and the properties of derivatives, since the right side of 14.1.3.5 is positive.

Q.E.D. (14.1) [by 14.1.3] and 14.1.1.

Q.E.D. (14) [by 14.1] and (13).

\[
[15] \quad \frac{\ell}{h_0} \geq 9.4 \quad \Rightarrow \quad \ell \geq 9.4 (\delta_1 \cup \delta_2) \quad \text{[14] and [13].}
\]
Corollary 2A.24: Consider two classes of events, \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \), defined on the sets \( X \) and \( Y \) respectively. Let \( \circ \) be an anasic operator. Then

\[
\forall \circ (\mathcal{A}_1 \circ \mathcal{A}_2) \leq 4.7(\forall (\mathcal{A}_1) + \forall (\mathcal{A}_2)).
\]

Proof: This follows from lemma 2A.21 and theorem 2A.23.


We continue now by showing that anasic conjections are ricetic combinations; thus our results for ricetic combinations will apply to anasic conjections.

To inductively derive results about \( k \)-ary anasic operators, we will use the convention that, for any \( k \)-ary anasic operator \( \circ \) and all \( 0 < j < k \), \( \circ_j(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k) \) will denote the function \( \circ(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k) \), where \( \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k \) contain only one event, and hence induce only one subset on any set of elements of the sample space they are defined on.

**Definition:** Let \( S_1, S_2, \ldots, S_k \) be \( k \) lists. Then \( S_1 \circ S_2 \circ \cdots \circ S_k \) denotes \( \cdots (S_1 \circ S_2) \circ S_3 \circ \cdots \circ S_k \).
Theorem 2A.25: Consider \( k > 0 \) sample spaces \( X_1, X_2, \ldots, X_k \) and \( 2k \) events

\[
E_{1,1}, E_{1,2} \in X_1, E_{2,1}, E_{2,2} \in X_2, \ldots, E_{k,1}, E_{k,2} \in X_k.
\]

Let

\[
S_i \in X_i, S_j \in X_j, \ldots, S_k \in X_k
\]

for some \( t > 0 \). Let \( i \) be an integer, \( 0 < i \leq k \), and let \( \odot \) be an anisic operator. Assume that for all \( j \) such that \( i < j \leq k \),

\[
S_{i+1} \cap E_{i+1,1} = S_i \cap E_{i,1}.
\]

Then

\[
(S_1 \odot S_2 \odot \cdots \odot S_k) \cap \odot (E_{1,1}, E_{1,2}, \ldots, E_{k,1}) \neq (S_1 \odot S_2 \odot \cdots \odot S_k) \cap \odot (E_{2,1}, E_{2,2}, \ldots, E_{2,1})
\]

implies that

\[
(S_1 \cap E_{1,1} \neq S_1 \cap E_{1,2}) \lor (S_2 \cap E_{2,1} \neq S_2 \cap E_{2,2}) \lor \ldots \lor (S_k \cap E_{k,1} \neq S_k \cap E_{k,1}).
\]

Proof:

[Assume that]

\[
(S_1 \odot S_2 \odot \cdots \odot S_k) \cap \odot (E_{1,1}, E_{1,2}, \ldots, E_{k,1}) \neq (S_1 \odot S_2 \odot \cdots \odot S_k) \cap \odot (E_{2,1}, E_{2,2}, \ldots, E_{2,1})
\]

but that

\[
(S_1 \cap E_{1,1} = S_1 \cap E_{1,2}) \land (S_2 \cap E_{2,1} = S_2 \cap E_{2,2}) \land \ldots \land (S_k \cap E_{k,1} = S_k \cap E_{k,1}).
\]

We will show that a contradiction follows.]

[1]

\[
\exists \, (x_1, x_2, \ldots, x_k) \in (S_1 \odot S_2 \odot \cdots \odot S_k) : \\
[(x_1, x_2, \ldots, x_k) \in \odot (E_{1,1}, E_{1,2}, \ldots, E_{k,1}) \land (x_1, x_2, \ldots, x_k) \notin \odot (E_{2,1}, E_{2,2}, \ldots, E_{k,1})] \lor \\
[(x_1, x_2, \ldots, x_k) \notin \odot (E_{1,1}, E_{1,2}, \ldots, E_{k,1}) \land (x_1, x_2, \ldots, x_k) \in \odot (E_{2,1}, E_{2,2}, \ldots, E_{k,1})]
\]
Proof of 1: [If this were not the case, we would have]

\[ \forall (x_1, x_2, \ldots , x_k) \in (S_1 \ast S_2 \ast \cdots \ast S_k), \]

\[ (x_1, x_2, \ldots , x_k) \notin \cap (E_{1,1}, E_{2,1}, \ldots , E_{k,1}) \rightarrow (x_1, x_2, \ldots , x_k) \notin \cap (E_{1,2}, E_{2,2}, \ldots , E_{k,2}). \]

This contradicts the assumption that

\[ (S_1 \ast S_2 \ast \cdots \ast S_k) \cap \cap (E_{1,1}, E_{1,2}, \ldots , E_{k,1}) \neq (S_1 \ast S_2 \ast \cdots \ast S_k) \cap \cap (E_{2,1}, E_{2,2}, \ldots , E_{k,2}). \]

Q.E.D. (1)

[2]

Let \((x_1, x_2, \ldots , x_k) \in (S_1 \ast S_2 \ast \cdots \ast S_k)\) be such that

\[ [(x_1, x_2, \ldots , x_k) \notin \cap (E_{1,1}, E_{2,1}, \ldots , E_{k,1}) \land (x_1, x_2, \ldots , x_k) \notin \cap (E_{1,2}, E_{2,2}, \ldots , E_{k,2})] \]

\[ \lor \]

\[ [(x_1, x_2, \ldots , x_k) \notin \cap (E_{1,1}, E_{2,1}, \ldots , E_{k,1}) \land (x_1, x_2, \ldots , x_k) \notin \cap (E_{1,2}, E_{2,2}, \ldots , E_{k,2})] . \]

[Definition. (Such a \((x_1, x_2, \ldots , x_k)\) exists by (1).)]

\[ 3(j, 0 < j < k): S_j \cap E_{1,j} \neq S_j \cap E_{2,j} \]

Proof of 3:

\[ (3.1) \]

\[ (x_1, x_2, \ldots , x_k) \in \cap (E_{1,1}, E_{2,1}, \ldots , E_{k,1}) \land \]

\[ (x_1, x_2, \ldots , x_k) \notin \cap (E_{1,2}, E_{2,2}, \ldots , E_{k,2}) \]

[Assumption]

\[ (3.2) \]

\[ (x_1, x_2, \ldots , x_k) \in \cap (E_{1,1}, E_{2,1}, \ldots , E_{k,1}) \rightarrow \]

\[ \cap (z_1 \in E_{1,1}, z_2 \in E_{2,1}, \ldots , z_k \in E_{k,1}) \]

[By the definition of an analytic operator]

\[ (3.3) \]

\[ (x_1, x_2, \ldots , x_k) \notin \cap (E_{1,2}, E_{2,2}, \ldots , E_{k,2}) \rightarrow \]

\[ \neg \cap (z_1 \in E_{1,2}, z_2 \in E_{2,2}, \ldots , z_k \in E_{k,2}) \]

[By the definition of an analytic operator]

\[ (3.4) \]

\[ \forall (j: i < j \leq k): S_j \cap E_{1,j} = S_j \cap E_{2,j} \]

[By the assumption of the theorem.]

\[ (3.5) \]

\[ \forall (j: i < j \leq k): x_j \in E_{1,j} \rightarrow x_j \in E_{2,j} \]

This would contradict (3.4).
\[ \exists (x_j \in \{x_1, x_2, \cdots, x_i\}) : (x_j \in E_{j,1}) \neq (x_j \in E_{j,2}) \]

Proof of 3.6: Suppose the contrary holds:

\[ \forall (x_j \in \{x_1, x_2, \cdots, x_i\}) : (x_j \in E_{j,1}) = (x_j \in E_{j,2}). \]

If

\[ (x_1, x_2, \cdots, x_i) \in \cap (E_{1,1}, E_{2,1}, \cdots, E_{k,1}), \]

as we assumed in [3.1], then by the definition of an anasic operator it must be that

\[ G(x_1 \in E_{1,1}, x_2 \in E_{2,1}, \cdots x_i \in E_{k,1}) \]

holds. But if that is the case, then by [3.5] it must be that

\[ G(x_1 \in E_{1,2}, x_2 \in E_{2,2}, \cdots x_i \in E_{k,2}). \]

But this contradicts [3.1].

Q.E.D. (3.6)

[3.7]

Let \( x_j \in \{x_1, x_2, \cdots, x_i\} \) be such that \( (x_j \in E_{j,1}) \neq (x_j \in E_{j,2}). \)

[ Such an \( x_j \) exists by [3.6]. ]

[3.8]

\[ x_j \in S_j \]

[ By definition ]

[3.9]

\[ S_j \cap E_{j,1} \neq S_j \cap E_{j,2} \]

Proof of 3.9: It follows from the definition of \( x_j \) in [3.7] that

\[ (x_j \in E_{j,1}) \leftrightarrow (x_j \notin E_{j,2}). \]

But \( x_j \) is in \( S_j \) so we can also say

\[ (x_j \in E_{j,1} \cap S_j) \leftrightarrow (x_j \notin E_{j,2} \cap S_j). \]

But this implies

\[ S_j \cap E_{j,1} \neq S_j \cap E_{j,2}. \]

Q.E.D. (3.9)

Q.E.D. (3) [ We showed that the assumption in [3.1] leads to the desired conclusion, in [3.9]. ]
\[ [(x_1, x_2, \ldots, x_k) \notin \bigcap (E_{1,1}, E_{2,1}, \ldots, E_{k,1}) \land (x_1, x_2, \ldots, x_k) \notin \bigcap (E_{1,2}, E_{2,2}, \ldots, E_{k,2})] \quad \exists (j, 0 < j \leq i) : S_j \cap E_{1,j} \neq S_j \cap E_{2,j} \]

(The proof is exactly analogous to the proof of [3].)

\[ [(x_1, x_2, \ldots, x_k) \notin \bigcap (E_{1,1}, E_{2,1}, \ldots, E_{k,1}) \land (x_1, x_2, \ldots, x_k) \notin \bigcap (E_{1,2}, E_{2,2}, \ldots, E_{k,2})] \lor
[(x_1, x_2, \ldots, x_k) \notin \bigcap (E_{1,1}, E_{2,1}, \ldots, E_{k,1}) \land (x_1, x_2, \ldots, x_k) \in \bigcap (E_{1,2}, E_{2,2}, \ldots, E_{k,2})]. \]

(By the definition of \((x_1, x_2, \ldots, x_k)\) in [2].)

\[ \exists (j, 0 < j \leq i) : S_j \cap E_{1,j} \neq S_j \cap E_{2,j}. \quad \text{[By 5], [3], and [4].} \]

Q.E.D.  \[ [6] \] contradicts the assumption

\[ (S_1 \cap E_{1,1} = S_1 \cap E_{1,2}) \land (S_2 \cap E_{2,1} = S_2 \cap E_{2,2}) \land \cdots \land (S_i \cap E_{i,1} = S_i \cap E_{i,2}). \]

\[ \square \]

Lemma 2A.26: Consider \( k \) classes of events, \( \delta_1, \delta_2, \ldots, \delta_k \) defined on the sample spaces \( X_1, X_2, \ldots, X_k \), respectively. Let \( S_1, S_2, \ldots, S_k \) be lists drawn from \( X_1, X_2, \ldots, X_k \), respectively, and assume \( |S_1| = |S_2| = \cdots = |S_k| \). Assume that for some \( 0 < i \leq k \), all event classes \( \delta_{i+1}, \delta_{i+2}, \ldots, \delta_k \) are such that all the events they induce a common subset on \( S_{i+1}, S_{i+2}, \ldots, S_k \), respectively; in other words, assume

\[ \exists (i : 0 \leq i \leq k); \forall (j : i < j \leq k) : \exists (G \subseteq X_j); \forall (E \in \delta_j) : S_j \cap E = G. \]

Also let \( \otimes \) be a \( k \)-ary anasic operator. Then

\[ \Pi_{0 \leq i \leq k} \delta_i (S_1 \otimes S_2 \otimes \cdots \otimes S_k) \leq \Pi_{0 \leq i \leq k} (S_i \Pi_{0 \leq i \leq k} \delta_i). \]
Proof:

[1] Let $i$ be such that
$$\forall(j : i < j < k) : \exists(G \subseteq X_j) : \forall(E \in \mathcal{A}_j) : S_j \cap E = G.$$ [Such an $i$ exists by assumption.]

[2] Construct the mapping
$$M : \{(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k) : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2, \cdots, E_k \in \mathcal{A}_k\} \rightarrow \{S_1 \cap E_1 : E_1 \in \mathcal{A}_1\} \times \{S_2 \cap E_2 : E_2 \in \mathcal{A}_2\} \times \cdots \times \{S_k \cap E_k : E_k \in \mathcal{A}_k\}$$
as follows:
$$\forall(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k) \in$$
$$\{(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k) : E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2, \cdots, E_k \in \mathcal{A}_k\},$$
$$M((S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k)) = (S_1 \cap E_1, S_2 \cap E_2, \cdots, S_k \cap E_k).$$

[3] $|\text{Ran}(M)| = \prod_{i=1}^k |\mathcal{A}_i(S_1) \cdot \mathcal{A}_i(S_2) \cdots \mathcal{A}_i(S_k)|$ [This follows from the definition of $M$ and the definition of $\Pi$.]


Proof of 4: [We assume $M$ is not $1 : 1$ and derive a contradiction.]

[4.1] $M$ is not $1 : 1$ [Assumption]

[4.2] $\exists(E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2, \cdots, E_k \in \mathcal{A}_k, E_{k+1,1}, E_{k+1,2} \in \mathcal{A}_{k+1})$:
$$E_{i+2,1}, E_{i+2,2} \in \mathcal{A}_{i+2}, \cdots, E_{k,1}, E_{k,2} \in \mathcal{A}_k$$

$$(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k, E_{k+1,1}, E_{k+1,2}) \neq$$
$$(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k, E_{i+1,1}, E_{i+1,2}, E_{i+2,1}, E_{i+2,2})$$

Proof of 4.2: [We assume the contrary and derive a contradiction.]

[4.2.1] $\forall(E_1 \in \mathcal{A}_1, E_2 \in \mathcal{A}_2, \cdots, E_k \in \mathcal{A}_k, E_{k+1,1}, E_{k+1,2} \in \mathcal{A}_{k+1})$:
$$E_{i+2,1}, E_{i+2,2} \in \mathcal{A}_{i+2}, \cdots, E_{k,1}, E_{k,2} \in \mathcal{A}_k$$

$$(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k, E_{k+1,1}, E_{k+1,2}) =$$
$$(S_1 \cdot S_2 \cdots \cdot S_k) \cap \mathcal{C}(E_1, E_2, \cdots, E_k, E_{i+1,1}, E_{i+1,2}, E_{i+2,1}, E_{i+2,2})$$ [Assumption]
[4.2.2] \( \forall (E_1 \in \mathcal{A}, E_2 \in \mathcal{A}, \cdots, E_i \in \mathcal{A}, E_{i+1} \in \mathcal{A}, E_{i+1,j} \in \mathcal{A}) : \\ \quad \quad E_{1+1,1}, E_{i+1,2} \in \mathcal{A}_{1+1}, \cdots E_{i,1}, \mathcal{A}_{i,2} \in \mathcal{A}) : \\ (S_1 \cdot S_2 \cdot \cdots S_k) \cap \bigcirc (E_1, E_2, \cdots, E_i, E_{i+1,1}, E_{i+1,2}, \cdots E_{i,2}) \neq \\ (S_1 \cdot S_2 \cdot \cdots S_k) \cap \bigcirc (E_1, E_2, \cdots, E_i, E_{i+1,1}, E_{i+1,2}, \cdots E_{i,2}) \\ \mapsto \\ M((S_1 \cdot S_2 \cdot \cdots S_k) \cap \bigcirc (E_1, E_2, \cdots, E_i, E_{i+1,1}, E_{i+1,2}, \cdots E_{i,2}))) \neq \\ M((S_1 \cdot S_2 \cdot \cdots S_k) \cap \bigcirc (E_1, E_2, \cdots, E_i, E_{i+1,1}, E_{i+1,2}, \cdots E_{i,2})). \\
| [4.2.1] ensures that the antecedent of the implication is never satisfied, so the implication holds vacuously. |

[4.2.3] \( M \) is 1 : 1. \[ \text{[By 4.2.2] and the definition of a 1 : 1 function} \]

Q.E.D. (4.2) \[ [4.2.3] contradicts the assumption in (4.1). |]

[4.3] \( \exists (E_1 \in \mathcal{A}, E_2 \in \mathcal{A}, \cdots, E_i \in \mathcal{A}) : \\ \exists (j : 0 \leq j \leq i) : \\ S_i \cap E_i \neq S_i \cap E_i \\
Proof of 4.3: [This follows from [4.2] and theorem 2A.25.]

Q.E.D. (4.3)

Q.E.D. (4) \[ [4.3] is self-contradictory. |]

[5] \[ |\text{Ran}(M)| \geq \\ |\{(S_1 \cdot S_2 \cdot \cdots S_k) \cap \bigcirc (E_1, E_2, \cdots E_k) : E_1 \in \mathcal{A}, E_2 \in \mathcal{A}, \cdots E_k \in \mathcal{A})| \\
\text{[By 4], the definition of } M, \text{ and the definition of a 1 : 1 function.} \]

[6] \[ |\text{Ran}(M)| \geq \Pi_{E_{\mathcal{A}}(S_1) \cdot E_{\mathcal{A}}(S_2) \cdot \cdots E_{\mathcal{A}}(S_k)} \text{[By 5] and the definition of } \Pi \]

[7] \[ \Pi_{E_{\mathcal{A}}(S_1)} \Pi_{E_{\mathcal{A}}(S_2)} \cdots \Pi_{E_{\mathcal{A}}(S_k)} \geq \Pi_{E_{\mathcal{A}}(S_1) \cdot E_{\mathcal{A}}(S_2) \cdot \cdots E_{\mathcal{A}}(S_k)} \text{[By 3] and 6]}

Q.E.D.
Corollary 2A.27: Consider $k$ classes of $\{0,1\}$-valued functions $X_1, X_2, \ldots, X_k$, with domains

$$X_1, X_2, \ldots, X_k$$

respectively. Let $S_1, S_2, \ldots, S_k$ be lists drawn from $X_1, X_2, \ldots, X_k$, respectively, and assume $|S_1| = |S_2| = \ldots = |S_k|$. Also let $\otimes$ be a $k$-ary anisic operator. Then

$$\Pi_{\otimes}(S_1 \otimes S_2 \otimes \cdots \otimes S_k) \leq \Pi_{\otimes}(S_1)\Pi_{\otimes}(S_2)\cdots\Pi_{\otimes}(S_k).$$

Proof: This follows from lemma 2A.26 and the definition of $k$-ary anisic operators applied to $\{0,1\}$-valued functions. (The value of $i$ we use in lemma 2A.26 is simply $k$.)

Corollary 2A.28: Consider $k$ classes of $\{0,1\}$-valued functions $X_1, X_2, \ldots, X_k$, with domains

$$X_1, X_2, \ldots, X_k$$

respectively. Let $\otimes$ be a $k$-ary anisic operator. Then, for all $t > 0$,

$$\Pi_{\otimes}(S_1 \otimes S_2 \otimes \cdots \otimes S_k)(t) \leq \Pi_{\otimes}(S_1)(t)\Pi_{\otimes}(S_2)(t)\cdots\Pi_{\otimes}(S_k)(t).$$

Proof: Assume that the lemma does not hold. Then there must be some $S_1 \subseteq A_1, S_2 \subseteq A_2, \ldots, S_k \subseteq A_k$ such that $|S_1| = |S_2| = \ldots = |S_k| = t$ and

$$\Pi_{\otimes}(S_1 \otimes S_2 \otimes \cdots \otimes S_k) > \Pi_{\otimes}(S_1)\Pi_{\otimes}(S_2)\cdots\Pi_{\otimes}(S_k).$$

But by the definition of $\Pi$, the right side of (2A.15) is at least as large as $\Pi_{\otimes}(S_1)\Pi_{\otimes}(S_2)\cdots\Pi_{\otimes}(S_k)$, so that

$$\Pi_{\otimes}(S_1 \otimes S_2 \otimes \cdots \otimes S_k) > \Pi_{\otimes}(S_1)\Pi_{\otimes}(S_2)\cdots\Pi_{\otimes}(S_k).$$

But this contradicts corollary 2A.19.
2A.4. Proof of proposition 2.11.

We begin our series of results on $k$-ary recetic conceptions with the following lemma:

Lemma 2A.29: Let $C$ be a positive constant. Then for any $k > 0$

$$\min \left\{ d_1^{d_1}d_2^{d_2} \cdots d_k^{d_k} : \sum_{i=1}^{k} d_i = C, d_1, d_2, \ldots, d_k \in \mathbb{R}^+ \right\} = \left( \left( \frac{C}{k} \right)^{\frac{1}{k}} \right)^k .$$

Proof:

[1] Let

$$g_1, g_2, \cdots, g_k \in \mathbb{R}^+$$

be such that

$$\sum_{i=1}^{k} g_i = C$$

and

$$g_1^{g_1}g_2^{g_2} \cdots g_k^{g_k} = \min \left\{ d_1^{d_1}d_2^{d_2} \cdots d_k^{d_k} : \sum_{i=1}^{k} d_i = C, d_1, d_2, \ldots, d_k \in \mathbb{R}^+ \right\} .$$

{ Assumption }

[2] $g_1 = g_2 = \cdots = g_k = \frac{C}{k} \rightarrow$

$$g_1^{g_1}g_2^{g_2} \cdots g_k^{g_k} = \left( \left( \frac{C}{k} \right)^{\frac{1}{k}} \right)^k$$

{ If each of $g_1, g_2, \cdots, g_k$ is equal to $C/k$, then the right side of the implication follows by algebra. }

[3]

$g_1 = g_2 = \cdots = g_k \rightarrow$

$$g_1 = g_2 = \cdots = g_k = \frac{C}{k}$$

{ If $g_1, g_2, \cdots, g_k$ were equal, but had a value other than $C/k$, their sum would not equal $C$. This violates the assumption in 111. }

[4]

$g_1 = g_2 = \cdots = g_k$. 
Proof of 4: [We assume the contrary and derive a contradiction.]

[4.1] \( \neg(g_1 = g_2 = \cdots = g_k) \) \hspace{1cm} \{ Assumption \}

[4.2] \[ \exists (g_i, g_j \in \{g_1, g_2, \ldots, g_k\}) : g_i \neq g_j \] \hspace{1cm} \{ This follows from (4.1) \}

[4.3] Let \( g_i, g_j \in \{g_1, g_2, \ldots, g_k\} \) \hspace{1cm} \{ Such a \( g_i, g_j \) exist by [4.2] \}

be such that \( g_i \neq g_j \)

[4.4] Let \( C_0 = \frac{g_i + g_j}{2} \) \hspace{1cm} \{ Definition \}

[4.5] \[
\left( C_0^{C_0} \right)^2 \left( g_i^{g_i} g_j^{g_j} \cdots g_{k-1}^{g_{k-1}} g_{k+1}^{g_{k+1}} \cdots g_k^{g_k} \right) < \left( g_i^{g_i} g_j^{g_j} \cdots g_k^{g_k} \right).
\]

[ This inequality holds because \( \left( C_0^{C_0} \right)^2 \) is less than \( g_i^{g_i} g_j^{g_j} \), by lemma 2A.22. \]

[4.6] \[
\left( g_i^{g_i} g_j^{g_j} \cdots g_{k-1}^{g_{k-1}} C_0^{C_0} g_{k+1}^{g_{k+1}} \cdots g_k^{g_k} \right) < \left( g_i^{g_i} g_j^{g_j} \cdots g_k^{g_k} \right).
\]

[ The inequality is obtained by rearranging the factors in (4.5).]

[4.7] \[ g_1 + g_2 + \cdots + g_{i-1} + C_0 + g_{i+1} + \cdots + g_{j-1} + C_0 + g_{j+1} + \cdots + g_k = C \]

[ Since \( C_0 = (g_i + g_j)/2 \), \( C_0 + C_0 = g_i + g_j \). The equation now follows from the assumption (in [1]) that \( g_1 + g_2 + \cdots + g_k = C \).]

Q.E.D. (4) \hspace{1cm} [4.6] and [4.7] contradict the assumption of [1], which states that

\[
g_i^{g_i} g_j^{g_j} \cdots g_k^{g_k} = \min \left\{ d_1^{d_1} d_2^{d_2} \cdots d_k^{d_k} : \sum_{i=1}^{k} d_i = \zeta, d_1, d_2, \ldots, d_k \in \mathbb{R}^- \right\}.
\]
This lemma permits us to state a theorem that, in principle, tells us how to compute the Vapnik-Chervonenkis Dimension of a class of inductive conjections.

**Theorem 2A.30:** Let \( \delta_1, \delta_2, \ldots, \delta_k \) be \( k > 0 \) classes of events, and let \( d_1 \equiv \nu'(\delta_1), d_2 \equiv \nu'(\delta_2), \ldots, d_k \equiv \nu'(\delta_k) \). Also let

\[
d_0 = \frac{1}{k} \sum_{i=1}^{k} d_i.
\]

and let \( t \) be a positive real number. If \( \circ \) is a inductive operator, then

\[
2^t \geq \left( \frac{et}{d_0} \right)^{d_0} \rightarrow t \geq \nu'(\circ(\delta_1, \delta_2, \ldots, \delta_k)).
\]

**Proof:**

\[
2^t \geq \left( \frac{et}{d_0} \right)^{d_0} \rightarrow 2^t \geq \left( \frac{et}{d_1} \right)^{d_1} \left( \frac{et}{d_2} \right)^{d_2} \cdots \left( \frac{et}{d_k} \right)^{d_k}.
\]

**Proof of 1:**

1.1

\[
\left( \frac{et}{d_1} \right)^{d_1} \left( \frac{et}{d_2} \right)^{d_2} \cdots \left( \frac{et}{d_k} \right)^{d_k} = \frac{(et)^{d_1} (et)^{d_2} \cdots (et)^{d_k}}{d_1^{d_1} d_2^{d_2} \cdots d_k^{d_k}} \]

\[
= \frac{(et)^{d_1 + d_2 + \cdots + d_k}}{d_1^{d_1} d_2^{d_2} \cdots d_k^{d_k}}
\]

[By algebra]

1.2

\[
\sum_{i=0}^{k} d_i = kd_0
\]

[This follows from the definition of \( d_0 \), multiplying both sides of the equality \( d_0 = (1/k) \sum_{i=0}^{k} d_i \) by \( k \).]

1.3

\[
\frac{(et)^{\sum_{i=0}^{k} d_i}}{d_1^{d_1} d_2^{d_2} \cdots d_k^{d_k}} \leq \frac{(et)^{\sum_{i=0}^{k} d_i}}{(d_0)^k}
\]

[This follows from lemma 2A.29, substituting \( kd_0 = \sum_{i=1}^{k} d_i \) for \( c \) in that lemma.]
In the next four theorems we will use theorem 2A.30 to develop closed-form bounds on the VC dimensions of recursive conjunctions.

**Theorem 2A.31:** Let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$ be $k > 0$ classes of events, and let $d_1 = \| \mathcal{E}_1 \|, d_2 = \| \mathcal{E}_2 \|, \ldots, d_k = \| \mathcal{E}_k \|$. Then

\[
\frac{\ell}{\sum_{i=1}^{k} d_i} \geq \log_2 \left( e^{k} \frac{\ell}{\sum_{i=1}^{k} d_i} \right) \rightarrow \ell \geq \nu(\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_k) \cdot \nu(\mathcal{E}_1) \cdot \nu(\mathcal{E}_2) \cdot \ldots \cdot \nu(\mathcal{E}_k).
\]
Proof:

1) \[ \frac{t}{\sum_{i=1}^{k} d_i} \geq \log_2 \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right) \rightarrow 2^{\sum_{i=1}^{k} d_i} \geq \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right) \] [The right side follows from the left by taking antilogarithms.]

2) \[ \frac{t}{\sum_{i=1}^{k} d_i} \geq \log_2 \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right) \rightarrow 2^{\sum_{i=1}^{k} d_i} \geq \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right) \sum_{i=1}^{k} d_i \] [By algebra on the right side of [1]]

3) Let \( d_0 = \frac{1}{k} \sum_{i=1}^{k} d_i \) [Definition]

4) \[ \frac{t}{\sum_{i=1}^{k} d_i} \geq \log_2 \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right) \rightarrow 2^{d_0} \geq \left( \frac{ek \cdot \frac{t}{d_0}}{} \right) \] [Substituting \( d_0 \) into [2]]

5) \[ \frac{t}{\sum_{i=1}^{k} d_i} \geq \log_2 \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right) \rightarrow t \geq \gamma(\otimes(\widetilde{\mathcal{C}}, \mathcal{D} \cdots, \mathcal{D})) \] [Applying theorem 2A.30 to the right side of [4]]

Q.E.D.

We now give three upper bounds on the Vapnik-Chervonenkis dimensions of ricetic conjectures. Our proof method will be to show that the bounds satisfy

\[ \frac{t}{\sum_{i=1}^{k} d_i} \geq \log_2 \left( \frac{ek \cdot \frac{t}{\sum_{i=1}^{k} d_i}}{} \right), \]

and invoke theorem 2A.31 to show that they have the claimed property.

**Theorem 2A.32:** Let \( \otimes \) be a ricetic operator. For any \( k \geq 2 \) classes of sets

\[ \mathcal{C}, \mathcal{D}_1, \cdots, \mathcal{D}_k, \]

\[ \forall(t > 0), t \geq 2 \log_2(ek) \sum_{i=1}^{k} \gamma(d_i) \rightarrow t \geq \gamma(\otimes(\mathcal{C}, \mathcal{D}_1, \cdots, \mathcal{D}_k)). \]
Proof:

[1]
For all $0 < i \leq k$ let $d_i = r(d_i)$.

[2]
$\ell \geq 2 \log_2 (ek) \sum_i r(d_i) \quad \Rightarrow \quad \ell \geq 2 \log_2 (ek) \sum_i d_i$

[3]
\[
\ell \geq 2 \log_2 (ek) \sum_i d_i \quad \Rightarrow \quad \frac{\ell}{\sum_i d_i} \geq 2 \log_2 (ek)
\]

[By the definition of $d_i$ in [1].]

[4]
\[
\frac{\ell}{\sum_i d_i} \geq 2 \log_2 (ek) \quad \Rightarrow \quad \frac{\ell}{\sum_i d_i} \geq \log_2 \left( \frac{ek \ell}{\sum_i d_i} \right)
\]

Proof of 4:

[4.1]
\[
(k = 2 \land \ell = 2 \log_2 (ek)) \quad \Rightarrow \quad \frac{\ell}{\sum_i d_i} \geq \log_2 \left( \frac{ek \ell}{\sum_i d_i} \right)
\]

[The right side of the implication is verified by substituting $2$

for $k$ and $2 \log_2 (ek) = 2 \log_2 (2e)$ for $\ell/\sum_i d_i$.]

[4.2]
\[
\left( k = 2 \land \frac{\ell}{\sum_i d_i} \geq 2 \log_2 (ek) \right) \quad \Rightarrow \quad \frac{\ell}{\sum_i d_i} \geq \log_2 \left( \frac{ek \ell}{\sum_i d_i} \right)
\]

Proof of 4.2: We show that $\ell/\sum_i d_i$ grows more quickly in $\ell/\sum_i d_i$ than

\[
\log_2 \left( \frac{ek \ell}{\sum_i d_i} \right).
\]

The result then follows from [4.1].}

[4.2.1]
Let $x = \frac{\ell}{\sum_i d_i}$.

[4.2.2]
\[
\frac{dx}{dx} = 1
\]

[By calculus.]
[4.2.3] \[ \frac{d}{dx} \log_2(2ex) = \frac{1}{x} \] [ By calculus. ]

[4.2.4] \[ x \geq 1 \]

Proof of 4.2.4: [The claim is equivalent to]

\[ \frac{t}{\sum_i d_i} \geq 1. \]

But we have assumed

\[ \frac{t}{\sum_i d_i} \geq 2 \log_2(ek), \]

so the claim follows if \( 2 \log_2(ek) \geq 1 \). Since we also assumed \( k \geq 2 \) the claim also follows if \( 2 \log_2(2e) \geq 1 \), and the latter inequality can be verified by arithmetic.

Q.E.D. (4.2.4)

[4.2.5] \[ 1 \geq \frac{1}{x} \] [ This follows from 4.2.4. ]

[4.2.6] \[ \frac{dx}{dx} \geq \frac{d}{dx} \log_2(2ex) \]

[ This is obtained by substituting [4.2.2] and [4.2.3] into [4.2.5]. ]

Q.E.D. (4.2) [ The claim follows from [4.2.6], [4.1], and the properties of derivatives. ]

[4.3] \[ \left( k \geq 2 \land \frac{t}{\sum_i d_i} \geq 2 \log_2(ek) \right) \rightarrow \frac{t}{\sum_i d_i} \geq \log_2 \left( ek \frac{t}{\sum_i d_i} \right) \]

Proof of 4.3: [We show that \( t/ \sum_i d_i \) grows more quickly in \( k \) than]

\[ \log_2 \left( ek \frac{t}{\sum_i d_i} \right). \]

The result then follows from [4.2]. ]
Let \( x \geq 0 \) be such that \( \frac{e^x}{\sum_{i=1}^d d_i} = 2\log_2(ek) + z \) [Such an \( x \) exists because we assumed that \( \ell/\sum_{i=1}^d d_i \geq 2\log_2(ek) \).]

\[ \forall x \geq 0, \frac{d}{dk} 2\log_2(ek) + x = \frac{2}{k} \] [By calculus.]

\[ \forall x \geq 0, \]
\[ \frac{d}{dk} \log_2(ek(2\log_2(ek) + x)) = \]
\[ \frac{1}{ek(2\log_2(ek) + x)} \frac{d}{dk} ek(2\log_2(ek) + x) = \]
\[ \frac{1}{ek(2\log_2(ek) + x)} \left[ ek \frac{d}{dk} (2\log_2(ek) + x) + (2\log_2(ek) + x) \frac{d}{dk} ek \right] = \]
\[ \frac{1}{ek(2\log_2(ek) + x)} \left[ ek \left( \frac{2}{k} \right) + x(2\log_2(ek) + x) \right] = \]
\[ \frac{2}{k(2\log_2(ek) + x)} + \frac{1}{k} = \]
\[ \frac{1}{k} \left( 1 + \frac{2}{2\log_2(ek) + x} \right) \]
[By calculus and algebra.]

\[ x \geq 0 \land k \geq 2 \rightarrow \frac{2}{k} \geq \frac{1}{k} \left( 1 + \frac{2}{2\log_2(ek) + x} \right) \]

Proof of 4.3.4:

\[ 4.3.4.1 \]
Let \( y = \frac{x}{2} \) [definition.]

\[ 4.3.4.2 \]
\[ \frac{1}{k} \left( 1 + \frac{2}{2\log_2(ek) + x} \right) = \frac{1}{k} \left( 1 + \frac{2}{2\log_2(ek) + 2y} \right) \]
[By 4.3.4.1 and substitution.]

\[ 4.3.4.3 \]
\[ \frac{1}{k} \left( 1 + \frac{2}{2\log_2(ek) + x} \right) = \frac{1}{k} \left( 1 + \frac{1}{\log_2(ek) + y} \right) \]
[By 4.3.4.2 and algebra]

\[ 4.3.4.4 \]
\[ \log_2(2e) > 1 \] [This can be verified by arithmetic.]
[4.3.4.5] \[ \log_{2}(ke) > 1 \] [ By [4.3.4.4] and the assumption that \( k \geq 2 \). ]

[4.3.4.6] \[ \log_{2}(ke) + y > 1 \] [ By [4.3.4.5], since \( y \geq 0 \). ]

[4.3.4.7] \[ \frac{1}{\log_{2}(ke) + y} < 1 \] [ By [4.3.4.6] and algebra. ]

[4.3.4.8] \[ \frac{1}{\log_{2}(ke) + y} + 1 < 2 \] [ Adding 1 to both sides of [4.3.4.7]. ]

[4.3.4.9] \[ 2 \frac{1}{k} > \left( \frac{1}{\log_{2}(ke) + y} + 1 \right) \] [ Dividing [4.3.4.8] by \( k \). ]

[4.3.4.10] \[ 2 \frac{1}{k} > \left( 1 + \frac{2}{2 \log_{2}(ke) + x} \right) \] [ By [4.3.4.9] and [4.3.4.3]. ]

Q.E.D. (4.3.4)

[4.3.5] \[ x \geq 0 \land k \geq 2 \implies \frac{d}{dk} \log_{2}(ke) + x \geq \frac{d}{dk} \log_{2}(2 \log_{2}(ke) + x) \] [ By substituting the left sides of equations [4.3.2] and [4.3.3] for their right sides in [4.3.4]. ]

[4.3.6] \[ x \geq 0 \land k \geq 2 \land \frac{t}{\sum_i d_i} \geq 2 \log_{2}(ke) + x \implies \frac{d}{dk} \log_{2}\left( \frac{t}{\sum_i d_i} \right) \geq \frac{d}{dk} \log_{2}\left( \frac{t}{\sum_i d_i} \right) \] [ By substituting \( (t/\sum_i d_i) \) for \( (2 \log_{2}(ke) + x) \). ]

[4.3.7] \[ x \geq 0 \land k \geq 2 \land \frac{t}{\sum_i d_i} \geq 2 \log_{2}(ke) + x \implies \frac{t}{\sum_i d_i} \geq \log_{2}\left( \frac{t}{\sum_i d_i} \right) \] [ By [4.3.5], [4.2], and the properties of derivatives. ]

Q.E.D. (4.3) [ By [4.3.7], since \( x > 0 \). ]

Q.E.D. (4) [ By [4.3], since \( k \geq 2 \) is an assumption of the theorem. ]
\[ \ell \geq 2 \log_2(ek) \sum_i d_i \rightarrow \frac{\ell}{\sum_i d_i} \geq \log_2 \left( \frac{ek \ell}{\sum_i d_i} \right) \quad \text{\footnotesize{[By [3], [4], and the transitivity of \( \rightarrow \).]} }
\]

\[ \ell \geq 2 \log_2(ek) \sum_i d_i \rightarrow \ell \geq \nu(\cup(d_1, d_2, \ldots, d_k)) \quad \text{\footnotesize{[By [5] and theorem 2A.31.]} }
\]

Q.E.D.

**Theorem 2A.33:** Let \( \delta \) be a metric operator. For any \( k \geq 8 \) classes of sets

\[ \delta_1, \delta_2, \ldots, \delta_k, \]

\[ \forall (\ell > 0), \quad \ell \geq k \sum_i \nu(d_i) \rightarrow \ell \geq \nu(\cup(d_1, d_2, \ldots, d_k)). \]

**Proof:**

[1] For all \( 0 < i \leq k \) let \( d_i \in \nu(\delta_i) \).

[2] Assume \( \ell \geq k \sum_i d_i \) \quad \text{[This is the assumption of the theorem.]}

[3] \[ \frac{\ell}{\sum_i d_i} \geq k \quad \text{[By [2] and algebra.]} \]

[4] \[ \frac{\ell}{\sum_i d_i} \geq k \rightarrow \frac{\ell}{\sum_i d_i} \geq \log_2 \left( \frac{ek \ell}{\sum_i d_i} \right) \]
Proof of 4:

\[ (k = 8 \land \sum_i d_i = k) \implies \frac{t}{\sum_i d_i} \geq \log_2 \left( e k \frac{t}{\sum_i d_i} \right) \]

This may be verified arithmetically by substituting 8 for \( k \) and \( t/\sum_i d_i \).

\[ (k \geq 8 \land \sum_i d_i = k) \implies \frac{t}{\sum_i d_i} \geq \log_2 \left( e k \frac{t}{\sum_i d_i} \right) \]

Proof of 4.2: We show that \( k \) grows more quickly in \( k \) than \( \log_2 (ek^2) \). The claim then follows because of the assumption that \( \frac{t}{\sum_i d_i} = k \).

\[ \frac{dk}{dk} = 1 \]

By calculus.

\[ \frac{d}{dk} \log_2 (ek^2) = \frac{1}{ek^2} (2k) \]

By calculus.

\[ \frac{1}{ek^2} 2k = \frac{2}{ek} \]

By algebra.

\[ \frac{2}{ek} \leq \frac{2}{8e} \]

By the theorem's assumptions that \( k \geq 8 \).

\[ \frac{2}{8e} < 1 \]

This may be verified by arithmetic.

\[ \frac{d}{dk} \log_2 (ek^2) < 1 \]

By the chain of inequalities \([4.2.2]-[4.2.5]\).

\[ \frac{d}{dk} \log_2 (ek^2) < \frac{dk}{dk} \]

By \([4.2.1]\) and \([4.2.6]\).

\[ (k \geq 8) \implies k \geq \log_2 (ek^2) \]

By \([4.1]\), \([4.2.7]\), and the properties of derivatives.
\[ k \geq 8 \land \sum_{i} d_i = k \Rightarrow \frac{t}{\sum_{i} d_i} \geq \log_2 \left( \frac{ek}{\sum_{i} d_i} \right) \]

This is obtained by substituting \( t/\sum_{i} d_i \) for \( k \) on the left side of [4.2.8], and \( kt/\sum_{i} d_i \) for \( k^2 \) on the right side, according to the assumption that \( t/\sum_{i} d_i = k \).

Q.E.D. (4.2)

\[ k \geq 8 \land \sum_{i} d_i \geq k \Rightarrow \frac{t}{\sum_{i} d_i} \geq \log_2 \left( \frac{ek}{\sum_{i} d_i} \right) \]

Proof of 4.3:

\[ (k \geq 8 \land x \geq k) \Rightarrow x \geq \log_2(e^k x) \]

This may be verified by arithmetic.

\[ \frac{dz}{dx} = ? \]

[By calculus.]

\[ \frac{d}{dx} \log_2(e^k x) = \frac{1}{e^k} \]

[By calculus.]

\[ \frac{1}{e^k} < 1 \]

[This may be verified by arithmetic.]

\[ \frac{dz}{dx} > \frac{d}{dx} \log_2(e^k x). \]

[Substituting [4.3.2] and [4.3.3] into [4.3.4].]

\[ (k \geq 8 \land x \geq k) \Rightarrow x \geq \log_2(e^k x) \]

[Since [4.3.5] holds for all \( k \geq 8 \), this claim follows from [4.3.5] and [4.2].]

\[ \left( k \geq 8 \land \sum_{i} d_i \geq k \right) \Rightarrow \frac{t}{\sum_{i} d_i} \geq \log_2 \left( \frac{ek}{\sum_{i} d_i} \right) \]

[By substituting \( t/\sum_{i} d_i \) for \( x \) in [4.3.6], as per [4.3.1].]

Q.E.D. (4.3)

Q.E.D. (4)
[5] \[ t \geq k \sum_i d_i \quad \Rightarrow \quad \frac{t}{\sum_i d_i} \geq \log_2 \left( e^k \frac{\sum_i d_i}{t} \right) \quad \text{[By algebra on the left side of [4].]} \]

[6] \[ t \geq k \sum_i d_i \quad \Rightarrow \quad t \geq \tau \left( \circ \{ \sigma_1, \sigma_2, \ldots, \sigma_k \} \right) \quad \text{[By [5] and theorem 2A.31.]} \]

Q.E.D.

**Theorem 2A.34:** Let \( \circ \) be a reciter operator. For any \( k \geq 14 \) classes of sets

\[ \sigma_1, \sigma_2, \ldots, \sigma_k. \]

\[
\forall \left( \forall t > 0 \right), \quad t \geq \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \sum_{i=1}^k \psi(\sigma_i) \quad \Rightarrow \quad t \geq \psi \left( \circ \{ \sigma_1, \sigma_2, \ldots, \sigma_k \} \right). \]

**Proof:**

[1] Let \( t \) be any number greater than \[
\frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \sum_{i=1}^k \psi(\sigma_i) \]

[2] For all \( 0 < i \leq k \), let \( d_i = \psi(\sigma_i) \)

[3] \[ t \geq \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \sum_i d_i \quad \text{[By [1] and [2].]} \]

[4] \[ \frac{t}{\sum_i d_i} \geq \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \quad \text{[By [3] and algebra.]} \]

[5] \[
\frac{t}{\sum_i d_i} \geq \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \quad \Rightarrow \quad \frac{t}{\sum_i d_i} \geq \log_2 \left( e^k \frac{\sum_i d_i}{t} \right) \]

\[
\]
Proof of 5:

\[ \begin{align*}
\text{[5.1]} & \quad \left( k = 14 \land \frac{\ell}{\sum_i d_i} = \frac{1 + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) \to \\
& \quad \frac{\ell}{\sum_i d_i} \geq \log_2 \left( e^k \frac{\ell}{\sum_i d_i} \right) \\
& \quad \text{This may be verified arithmetically by substituting 14 for } k \text{ and} \\
& \quad \frac{1 + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \quad = \quad \\
& \quad \frac{1 + \ln(14e) + \ln(1 + \ln(14e))}{\ln(2)} \\
& \quad \text{for } \ell/\sum_i d_i. \end{align*} \]

\[ \begin{align*}
\text{[5.2]} & \quad \left( k \geq 14 \land \frac{\ell}{\sum_i d_i} = \frac{1 + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) \to \\
& \quad \frac{\ell}{\sum_i d_i} \geq \log_2 \left( e^k \frac{\ell}{\sum_i d_i} \right) \\
\end{align*} \]

Proof of 5.2:

\[ \begin{align*}
\text{[5.2.1]} & \quad \frac{d}{dk} \left( \frac{1 + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) = \\
& \quad \frac{1}{\ln(2)} \left( \frac{1 + \frac{1}{k} \left( \frac{1}{\ln(1 + \ln(ek))} \right)}{1 + \ln(1 + \ln(ek))} \right) \quad = \quad \text{[By calculus and algebra]} \\
& \quad \frac{1}{k \ln(2)} \left( 1 + \frac{1}{\ln(1 + \ln(ek))} \right) \\
\end{align*} \]
[5.2.2] 
\[ \frac{d}{dk} \log_2 \left( \frac{e^k + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) = \]
\[ \frac{1}{\ln(2)} \cdot e + \frac{e^k}{\ln(2)} \left( \frac{1}{k} + \frac{1}{\ln(1 + \ln(ek))} \right) \]
\[ \frac{d}{dk} \left( \frac{1}{\ln(1 + \ln(ek))} \right) \]
\[ = \frac{1}{\ln(2)} \cdot e + \frac{e^k}{\ln(2)} \left( \frac{1}{k} + \frac{1}{\ln(1 + \ln(ek))} \right) \]

[5.2.3] 
\[ \frac{d}{dk} \log_2 \left( \frac{e^k + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) = \]
\[ \frac{1}{\ln(2)} \cdot e + \frac{e^k}{\ln(2)} \left( \frac{1}{k} + \frac{1}{\ln(1 + \ln(ek))} \right) \]
\[ \frac{d}{dk} \left( \frac{1}{\ln(1 + \ln(ek))} \right) \]
\[ = \frac{1}{\ln(2)} \cdot e + \frac{e^k}{\ln(2)} \left( \frac{1}{k} + \frac{1}{\ln(1 + \ln(ek))} \right) \]

[By calculus and algebra.]
\[ \frac{d}{dk} \left( \frac{1 + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) \geq \frac{d}{dk} \log_2 \left( \frac{ek + \ln(1 + \ln(ek))}{\ln(2)} \right) \]

Proof of 5.2.4:

\[ \frac{d}{dk} \left( \frac{1 + \ln(ek) + \ln(1 + \ln(ek))}{\ln(2)} \right) = \]

\[ \frac{1}{k \ln(2)} \left(1 + \ln(1 + \ln(ek))\right) = \]

\[ \frac{1}{k \ln(2)} \left(1 + \frac{1 + (\ln(1 + \ln(ek)))^{-1}(1 + \ln(ek))^{-1}k^{-1}}{1 + \ln(ek) + \ln(1 + \ln(ek))}\right) = \]

\[ \frac{1}{k \ln(2)} \left(1 + \frac{1}{\ln(1 + \ln(ek))} - \frac{1}{\ln(2)}\right) \]

\[ \frac{1}{k \ln(2)} \left(1 + \frac{\ln(1 + \ln(ek))^{-1}(1 + \ln(ek))^{-1}k^{-1}}{1 + \ln(ek) + \ln(1 + \ln(ek))}\right) \]

[By [5.2.1], [5.2.3], and algebra.]

\[ \frac{d}{dk} \log_2 \left( \frac{ek + \ln(1 + \ln(ek))}{\ln(2)} \right) = \]

\[ \frac{1}{k \ln(2)} \left(1 - \frac{1}{\ln(2)} + \frac{1}{\ln(1 + \ln(ek))} \left(1 - \frac{1}{\ln(2)} \frac{(1 + \ln(ek))^{-1}k^{-1}}{1 + \ln(ek) + \ln(1 + \ln(ek))}\right)\right) \]

[By [5.2.4.1] and algebra.]

\[ \frac{1}{\ln(2)} < 1 \]

[By arithmetic.]

\[ 1 - \frac{1}{\ln(2)} > 0 \]

[By [5.2.4.3] and algebra.]

\[ \frac{d}{dk} \log_2 \left( \frac{ek + \ln(1 + \ln(ek))}{\ln(2)} \right) \geq \]

\[ \frac{1}{k \ln(2)} \left(\frac{1}{\ln(1 + \ln(ek))} \left(1 - \frac{1}{\ln(2)} \frac{(1 + \ln(ek))^{-1}k^{-1}}{1 + \ln(ek) + \ln(1 + \ln(ek))}\right)\right) \]

[Replacing $1 - 1/\ln(2)$ with 0 in [5.2.4.2], as per [5.2.4.4].]
[5.2.4.6]

\[ 1 + \ln(e^k) + \ln(1 + \ln(e^k)) \geq 1 \]

| By arithmetic, using the assumption that \( k \geq 14 \) |

[5.2.4.7]

\[ (1 + \ln(e^k))^k \geq 1 \]

| By arithmetic, using the assumption that \( k \geq 14 \) |

[5.2.4.8]

\[ \frac{(1 + \ln(e^k))^{-1}k^{-1}}{1 + \ln(e^k) + \ln(1 + \ln(e^k))} < 1 \]

| By [5.2.4.6], [5.2.4.7] and algebra. |

[5.2.4.9]

\[ 1 - \frac{(1 + \ln(e^k))^{-1}k^{-1}}{1 + \ln(e^k) + \ln(1 + \ln(e^k))} \geq 0 \]

| By [5.2.4.8] and algebra. |

[5.2.4.10]

\[ \frac{d}{dk} \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \]

\[ \frac{d}{dk} \log_2 \left( \frac{e^k 1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \right) \geq 0 \]

\[ \frac{1}{k \ln(2)^2} \left( \frac{1}{\ln(1 + \ln(e^k))} \right) \]

| By arithmetic, using the assumption that \( k \geq 14 \) |

[5.2.4.11]

\[ \frac{1}{k \ln(2)^2} \left( \frac{1}{\ln(1 + \ln(e^k))} \right) \geq 0 \]

| By arithmetic, using the assumption that \( k \geq 14 \) |

[5.2.4.12]

\[ \frac{d}{dk} \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \]

\[ \frac{d}{dk} \log_2 \left( \frac{e^k 1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \right) \geq 0 \]

| By [5.2.4.10], [5.2.4.11], and the transitivity of \( \geq \). |

Q.E.D. (5.2.4)

Q.E.D. (5.2)  | By [5.2.4], [5.1], and the properties of derivatives. |
\[ k \geq 14 \land \sum_{i} d_i \geq \frac{1 + \ln(e^k) + \ln(1 + \ln(e^k))}{\ln(2)} \rightarrow \]
\[
\frac{\ell}{\sum_i d_i} \geq \log_2 \left( e^k \frac{\ell}{\sum_i d_i} \right)
\]

Proof of 5.3:

[5.3.1]
Let \( k \geq 14 \) and \( x \equiv \frac{\ell}{\sum_i d_i} \geq k \)

[5.3.2]
\[
\frac{dx}{dx} = 1
\]  [By calculus.]

[5.3.3]
\[
\frac{d}{dx} \log_2(e^kx) = \frac{1}{\ln(2)} \frac{d}{dx} \ln(e^kx) = \frac{1}{\ln(2)} \frac{1}{e^kx} \cdot e^kx = \frac{1}{\ln(2)} \frac{1}{x}
\]  [By calculus and algebra.]

[5.3.4]
\[
\frac{1}{\ln(2)} \leq 1
\]  [By arithmetic, using the assumption that \( x \equiv \frac{\ell}{\sum_i d_i} \geq 14 \).]

[5.3.5]
\[
\frac{d}{dx} \log_2(e^kx) < 1
\]  [By 5.3.3 and 5.3.4.]

[5.3.6]
\[
\frac{dx}{dx} > \frac{d}{dx} \log_2(e^kx)
\]  [By 5.3.2 and 5.3.5.]

Q.E.D. (5.3)  [by 5.2, 5.3.6, and the properties of derivatives.]

Q.E.D. (5)  [By 5.3, since \( k \geq 14 \) is an assumption of the theorem.]
For completeness, we also wish to consider unary conjunctions. We do so in the next theorem.

**Theorem 2A.3: Let** \( \otimes \) \( \) be a unary recotic conjunction operator, and let \( \mathcal{E} \) be a class of events. **Then**

\[
\forall (t > 0), \quad t \geq \mathcal{V}(\mathcal{E}) \quad \Rightarrow \quad t \geq \mathcal{V}(\otimes(\mathcal{E})).
\]

**Proof:**

\begin{enumerate}
  \item \[ \forall (t > 0), \quad \Pi_{\otimes(\mathcal{E})}(t) \leq \Pi_{\mathcal{E}}(t) \quad \text{[By the definition of a recotic operator.]}
  \]
  \item \[ \text{Let } x = \max \{ t \in \mathbb{N} : \Pi_{\mathcal{E}}(t) = 2^t \}
  \]
  \item \[ x \geq \max \{ t \in \mathbb{N} : \Pi_{\otimes(\mathcal{E})}(t) = 2^t \}
  \]
\end{enumerate}
Proof of (3): [We assume the contrary and derive a contradiction]

\[ x < \max \{ \ell \in \mathbb{N} : \Pi_{\mathcal{O}(\mathcal{E})}(\ell) = 2^\ell \} \]  
\[ \text{[Assumption]} \]

\[ x = \max \{ \ell \in \mathbb{N} : \Pi_{\mathcal{E}}(\ell) = 2^\ell \} \]  
\[ \text{[By the definition of } x \text{.]} \]

\[ \Pi_{\mathcal{E}}(x + n) \neq 2^{x+n} \]  
\[ \text{[Such an } n \text{ must exist by [3.1].]} \]

\[ \Pi_{\mathcal{E}}(x + n) = 2^{x+n} \lor \Pi_{\mathcal{E}}(x + n) < 2^{x+n} \]  
\[ \text{[This follows from [3.2] and the fact that } n \text{ is positive.]} \]

\[ \Pi_{\mathcal{E}}(x + n) < 2^{x+n} \]  
\[ \text{[By [3.4] and [3.5].]} \]

\[ \Pi_{\mathcal{E}}(x + n) < \Pi_{\mathcal{O}(\mathcal{E})}(x + n) \]  
\[ \text{[By [3.6] and the definition of } n \text{ in [3.3].]} \]

Q.E.D. (3) [ [3.7] contradicts the assumption that } \mathcal{O} \text{ is ricetic.]}

\[ x = \nu'(\mathcal{E}') \]  
\[ \text{[By [2] and the definition of } \nu' \text{.]} \]

\[ \max \{ \ell \in \mathbb{N} : \Pi_{\mathcal{O}(\mathcal{E}')}(\ell) = 2^\ell \} = \nu'(\mathcal{O}(\mathcal{E}')) \]  
\[ \text{[By the definition of } \nu' \text{.]} \]

\[ \nu'(\mathcal{E}') = \nu'(\mathcal{O}(\mathcal{E}')) \]  

Q.E.D. [ The theorem follows from [6].]

\[ \square \]

Theorem 2A.33 gives better results than theorem 2A.32 when } 8 \leq k \leq 10; \text{ otherwise theorem 2A.32 is a tighter bound. The bound theorem 2A.34 always gives a tighter bound than theorem 2A.32. (Theorem 2A.23 gives a slightly better bound in the case } k = 2, \text{ but its primary purpose is to increase the perspicuity of the results presented here.)}
To make our subsequent results more concise, we will summarize the last several theorems using the function

\[ T_k(d) = \begin{cases} 
  d, & \text{if } k = 1 \\
  4.7d, & \text{if } k = 2 \\
  2d \log_2(rk), & \text{if } 3 \leq k \leq 7 \text{ or } 11 < k < 13 \\
  kd, & \text{if } 8 \leq k \leq 10 \\
  d(1 + \ln(rk) + \ln(1 + \ln(rk)))/\ln(2), & \text{if } k \geq 14.
\]

**Theorem 2A.36**: Consider \( k \) classes of events, \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \), and let \( \odot \) be a ricotic conjunction operator. Then

\[ T'(\odot(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \leq T_k \left( \sum_{i=1}^{k} T'(\mathcal{M}_i) \right). \]

**Proof**: This follows from theorem 2A.35 when \( k = 1 \), from theorem 2A.23 when \( k = 2 \), from theorem 2A.32 when \( 3 \leq k \leq 7 \), from theorem 2A.33 when \( 8 \leq k \leq 10 \), and from theorem 2A.33 when \( k \geq 11 \).

Since anasic operators were shown to be ricotic we have:

**Corollary 2A.37**: Consider \( k \) classes of events, \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \), and let \( \odot \) be an anasic conjunction operator. Then

\[ T'(\odot(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \leq T_k \left( \sum_{i=1}^{k} T'(\mathcal{M}_i) \right). \]

**Proof**: In the case \( k = 1 \), \( \odot \) is a unary operator. In that case, we define an auxiliary binary anasic operator \( \odot' \) with the property that for any boolean-valued functions \( H_1 \) and \( H_2 \), and for all \( x \in \text{Dom}(H_1), y \in \text{Dom}(H_2) \),

\[ \odot'(H_1, H_2)(x, y) = \odot(H_1)(x). \]

Since \( \odot'(H_1, H_2) \) has the same value as \( \odot(H_1) \) on every element in the domain of \( H_1 \), for any choice of \( H_1 \),

\[ \Pi_{\odot'(\mathcal{M}_1, \mathcal{M}_2)}(\ell) = \Pi_{\odot(\mathcal{M}_1)}(\ell) \]

for all boolean-valued function classes \( \mathcal{M}_2 \), and all \( \ell > 0 \).

Let \( \mathcal{M}_1 \) be a set of functions that only induces on subset on any list from its domain. Then, by theorem 2A.18,

\[ \Pi_{\odot(\mathcal{M}_1)}(\ell) = \Pi_{\odot'(\mathcal{M}_1, \mathcal{M}_2)}(\ell) \leq \Pi_{\mathcal{M}_1}(\ell). \]

Thus \( \odot \) is ricotic (the case \( k > 1 \) was dealt with in lemma 2A.26.) The theorem therefore follows from theorem 2A.36.
2A.5. **Proof of proposition 2.1.4.**

**Theorem 2A.38:** Let $\mathcal{H}$ be a class of hypotheses whose VC Dimension is bounded above by $d$ (for some $d > 0$), and let $\mathcal{H}$ be a hypothesis class with the property that

$$\forall (t \geq 0): \Pi_{\mathcal{H}}(t) \geq w\Pi_{\mathcal{H}}(t),$$

for some $w \geq 0$. Then

$$\nu(\mathcal{H}) \leq \begin{cases} \frac{(15)d^{2}2^{-\frac{\log_{2}(w)}{d}}} {1} \log_{2}(w) / d \geq 0.5, \\ \frac{\log_{2}(d^{2}2^{-\frac{\log_{2}(w)}{d}})} {w} \log_{2}(w) / d \geq 0.5, \end{cases}$$

**Proof:**

[1] \( \Pi_{\mathcal{H}}(t) \geq \left( \frac{e^{f}}{d} \right)^{d} \) [By lemma 2A.15]

[2] \( \Pi_{\mathcal{H}}(t) \geq w\Pi_{\mathcal{H}}(t) \) [By the assumption of the theorem]

[3] \( \Pi_{\mathcal{H}}(t) \geq w\left( \frac{e^{f}}{d} \right)^{d} \) [Substituting the right side of [2] for the left side of [1]]

[4] \( (2^{f} \geq \Pi_{\mathcal{H}}(t)) \rightarrow (t \geq \nu(\mathcal{H})) \) [By the definition of $\nu$]

[5] \( \left( 2^{f} \geq w\left( \frac{e^{f}}{d} \right)^{d} \right) \rightarrow (t \geq \nu(\mathcal{H})) \) [Substituting the right side of [3] for its left side in [4]]

[6] \( \left( 2^{f/d} \geq w^{-d}\frac{e^{f}}{d} \right) \rightarrow (t \geq \nu(\mathcal{H})) \) [Taking the $d$th root of the left side of [5]]

[7] \( \left( \frac{e^{f}}{d} \geq \log_{2}\left( w^{-d}\frac{e^{f}}{d} \right) \right) \rightarrow (t \geq \nu(\mathcal{H})) \) [Taking the log base 2 of the left side of [6]]

[8] \( \left( \frac{e^{f}}{d} \geq \frac{\log_{2}(w)}{d} + \log_{2}\left( \frac{e^{f}}{d} \right) \right) \rightarrow (t \geq \nu(\mathcal{H})) \) [By [7] and algebra]
\[ \left( \frac{\ell}{d} \geq \frac{\log_2(\epsilon)}{d} + \log_2(\epsilon) + \log_2 \left( \frac{\ell}{d} \right) \right) \quad \rightarrow \quad (\ell \geq \gamma(\mathcal{M})) \]

{ By [8] and algebra }

\[ \left( \frac{\log_2(\epsilon)}{d} \leq 0.5 \land \ell \geq \left( \frac{15.1}{\epsilon} \right) 2^{-\log_2(\epsilon)/d} \right) \quad \rightarrow \quad (\ell \geq \gamma(\mathcal{M})) \]

Proof of 10:

10.1: \[ \ell \geq \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \quad [ \text{Assumption. (We will show that this leads to the desired consequence.)} ] \]

10.2: \[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \geq \log_2(\epsilon) + \frac{\log_2(\epsilon)}{d} + \log_2 \left( \frac{\ell}{d} \right) \right) \quad \rightarrow \quad (\ell \geq \gamma(\mathcal{M})) \]

{ Substituting the right side of [10.1] for the first \( \ell \) in [9]. }

10.3: \[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \geq \log_2(\epsilon) + \frac{\log_2(\epsilon)}{d} + \log_2 \left( \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \right)/d \right) \right) \quad \rightarrow \quad (\ell \geq \gamma(\mathcal{M})) \]

{ Substituting the right side of [10.1] for the remaining \( \ell \) in [10.2]. }

10.4: \[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} - \log_2(d) \geq \log_2(\epsilon) + \frac{\log_2(\epsilon)}{d} + \log_2 \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \right) - \log_2(d) \right) \]

\[ (\ell \geq \gamma(\mathcal{M})) \]

{ By algebra on the right side of [10.3]. }

10.5: \[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \geq \log_2(\epsilon) + \frac{\log_2(\epsilon)}{d} + \log_2 \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \right) \right) \]

\[ (\ell \geq \gamma(\mathcal{M})) \]

{ By algebra on the right side of [10.4]. }

10.6: \[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\epsilon)/d} \geq \log_2(\epsilon) + \frac{\log_2(\epsilon)}{d} + \log_2 \left( \frac{15.1}{\epsilon} \right) + \log_2 \left( 2^{-\log_2(\epsilon)/d} \right) \right) \]

\[ (\ell \geq \gamma(\mathcal{M})) \]

{ Expanding the last term on the left side of [10.5]. }
\[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\omega)/d} \geq \log_2(\epsilon) + \frac{\log_2(\omega)}{d} + \log_2 \left( \frac{15.1}{\epsilon} \right) - \frac{\log_2(\omega)}{d} \right) \rightarrow \ell \geq \tau'(\mathcal{M}_f) \]

\[ \left( \frac{15.1}{\epsilon} 2^{-\log_2(\omega)/d} \geq \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \right) \rightarrow \ell \geq \tau'(\mathcal{M}_f) \]

Cancelling \( \log_2(\omega)/d \)'s in (10.7).

\[ \frac{\log_2(\omega)}{d} = 0.5 \rightarrow \left( \frac{15.1}{\epsilon} 2^{-\log_2(\omega)/d} \geq \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \right) \]

By arithmetic: we substitute 0.5 for \( \log_2(\omega)/d \) in (10.8).

\[ \frac{\log_2(\omega)}{d} = 0.5 \rightarrow \left( \frac{15.1}{\epsilon} 2^{-\log_2(\omega)/d} - \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \geq 0 \right) \]

By (10.9) and algebra.

\[ z = 0.5 \rightarrow \left( \frac{15.1}{\epsilon} 2^{-z} - \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \geq 0 \right) \]

For notational convenience we will let \( z \) stand for \( \log_2(\omega)/d \).

\[ \left( \frac{15.1}{\epsilon} 2^{-z} - \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \geq 0 \right) \]

\[ \forall(x \leq 0.5) : \frac{15.1}{\epsilon} 2^{-x} - \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \geq 0 \]

By (10.11) and the fact that

\[ \frac{15.1}{\epsilon} 2^{-x} - \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \]

is a monotone decreasing function of \( x \).

\[ \forall(x \leq 0.5) : \frac{15.1}{\epsilon} 2^{-x} - \log_2(\epsilon) + \log_2 \left( \frac{15.1}{\epsilon} \right) \geq 0 \]

By (10.11) and (10.12).
\[ \forall z \leq 0.5 : \frac{15.1}{e} 2^{-z} \geq \log_2(\varepsilon) + \log_2 \left( \frac{15.1}{e} \right) \]  

(10.14)  

[By (10.13) and algebra.]

\[ \frac{\log_2(u)}{d} \leq 0.5 \rightarrow \frac{15.1}{e} 2^{\log_2(u)/d} \geq \log_2(\varepsilon) + \log_2 \left( \frac{15.1}{e} \right) \]  

(10.15)  

[This holds by (10.14), since \( \log_2(u)/d \) is a number less than or equal to 0.5.]

\[ \frac{\log_2(u)}{d} \leq 0.5 \]  

(10.16)  

[By assumption.]

\[ \frac{15.1}{e} 2^{\log_2(u)/d} \geq \log_2(\varepsilon) + \log_2 \left( \frac{15.1}{e} \right) \]  

(10.17)  

[By (10.15) and (10.16).]

\[ t \geq 1^* \left( \mathcal{X}_1 \right) \]  

(10.18)  

[By (10.17) and (10.8).]

Q.E.D. (10)  

[By (10.11) and (10.18).]

\[ \left( \frac{\log_2(u)}{d} \geq 0.5 \land t \geq \frac{8.2}{e} \log_2(u) \right) \rightarrow \left( t \geq 1^* \left( \mathcal{X}_1 \right) \right) \]  

(11)  

Proof of 11:

\[ t \geq \frac{8.2}{e} \log_2(u) \]  

(11.1)  

[Assumption (we will show that the desired conclusion follows).]

\[ \left( \frac{8.2}{de} \log_2(u) \geq \frac{\log_2(u)}{d} + \log_2 \left( \frac{t}{d} \right) + \log_2(e) \right) \rightarrow t \geq 1^* \left( \mathcal{X}_1 \right) \]  

(11.2)  

[Substituting the right side of (11.1) for the leftmost \( t \) in (9).]

\[ \left( \frac{8.2}{de} \log_2(u) \geq \frac{\log_2(u)}{d} + \log_2 \left( \frac{8.2 \log_2(u)}{de} \right) + \log_2(e) \right) \rightarrow t \geq 1^* \left( \mathcal{X}_1 \right) \]  

(11.3)  

[Substituting the right side of (11.1) for the leftmost \( t \) in (11.2).]
(\left( \frac{8.2}{e} - 1 \right) \frac{\log_2(w)}{d} \geq \log_2 \left( \frac{8.2 \log_2(w)}{d} \right) + \log_2(e) \quad \rightarrow \quad \ell \geq \gamma_1(w) \\

| Collecting the first two \log_2(w)/d terms in [11.3]. |

(\left( \frac{8.2}{e} - 1 \right) \frac{\log_2(w)}{d} \geq \log_2 \left( \frac{8.2 \log_2(w)}{d} \right) - \log_2(e) + \log_2(e) \quad \rightarrow \quad \ell \geq \gamma_1(w) \\

| By [11.4] and algebra |

(\left( \frac{8.2}{e} - 1 \right) \frac{\log_2(w)}{d} \geq \log_2 \left( \frac{8.2 \log_2(w)}{d} \right) \quad \rightarrow \quad \ell \geq \gamma_1(w) \\

| By [11.5] and algebra |

(\log_2(w)/d = 0.5) \rightarrow (\left( \frac{8.2}{e} - 1 \right) \frac{\log_2(w)}{d} \geq \log_2 \left( \frac{8.2 \log_2(w)}{d} \right) \\

| By substituting 0.5 for \log_2(w)/d in [11.6]. |

(\log_2(w)/d \geq 0.5) \rightarrow (\left( \frac{8.2}{e} - 1 \right) \frac{\log_2(w)}{d} \geq \log_2 \left( \frac{8.2 \log_2(w)}{d} \right) \\

Proof of 11.8: [We show by taking derivatives that 

\left( \frac{8.2}{e} - 1 \right) x - \log_2(8.2x) 

is an increasing function of x.]

\frac{d}{dx} \left( \left( \frac{8.2}{e} - 1 \right) x - \log_2(8.2x) \right) = \frac{8.2}{e} - 1 - \frac{1}{x} \\

| By calculus |

x \geq 0.5 \rightarrow \frac{8.2}{e} - 1 - \frac{1}{x} \geq 0 \\

| Since 1/x \leq 2 when x \geq 0.5, while (8.2/e)-1 can be verified arithmetically to be greater than 2. |

x \geq 0.5 \rightarrow \frac{d}{dx} \left( \left( \frac{8.2}{e} - 1 \right) x - \log_2(8.2x) \right) \geq 0 \\

| By [11.8.1] and [11.8.2] |
\[\frac{\log_2(w)}{d} \geq 0.5 \rightarrow \]
\[\left(\frac{8.2}{e} - 1\right) \frac{\log_1(w)}{d} - \log_2 \left(\frac{8.2 \log_2(w)}{d}\right) \geq 0\]
[ By [11.8.3], [11.7], and the properties of the derivative, after substituting \(\log_2(w)/d\) for \(r\). ]

\[\frac{\log_2(w)}{d} \geq 0.5 \rightarrow \]
\[\left(\frac{8.2}{e} - 1\right) \frac{\log_2(w)}{d} \geq \log_2 \left(\frac{8.2 \log_2(w)}{d}\right)\]
[ By [11.8.4] and algebra. ]

Q.E.D. (11.8)

\[\frac{\log_2(w)}{d} \geq 0.5\] [ By the assumption of the present case ]

\[\left(\frac{8.2}{e} - 1\right) \frac{\log_2(w)}{d} \geq \log_2 \left(\frac{8.2 \log_2(w)}{d}\right)\] [ By [11.8] and [11.9]. ]

\[t \geq V(\mathcal{H})\]
[ By [11.10] and [11.6]. ]

Q.E.D. (11)

Q.E.D.

Corollary 2A.39: Let \(\mathcal{H}_1\) be a class of hypotheses with the property that
\[\Pi_{\mathcal{H}_1}(S) \leq u\]
for any \(S\) and some \(u\), and let \(\mathcal{N}\) be an anasic operator. For any hypothesis class \(\mathcal{H}_4\) having VC Dimension \(d\), the Vapnik-Chervonenkis Dimension of \((\mathcal{H}_3 \cup \mathcal{H}_4)\) is bounded above by
\[
\frac{(15.1)d}{e} 2^{-\log_2(u)/d} \]
if \(\log_2(w)/d \leq 0.5\), and by
\[
\frac{8.2}{e} \log_2(w) \]
if \(\log_2(w)/d \geq 0.5\).
Proof: Let $S_3$ and $S_4$ be any two subsets of $\text{Dom}(\mathcal{M}_3)$ and $\text{Dom}(\mathcal{M}_4)$ respectively, with the property that $|S_3| = |S_4|$. Because $\bigcirc$ is an action, corollary 2A.27 allows us to say that

$$\Pi_{\mathcal{M}_3 \circ \mathcal{M}_1}(S_3 \circ S_4) \leq u \cdot \Pi_{\mathcal{M}_4}(S_4).$$  \hfill (2A.17)

The present corollary then follows from theorem 2A.38, using $\mathcal{M}_4$ as the $\mathcal{M}_1$ of theorem 2A.38, and $\mathcal{M}_3 \circ \mathcal{M}_4$ as the $\mathcal{M}_1$ of theorem 2A.38. \hfill \Box
3. Classes of composite functions

Although the results of the last chapter can simplify the analysis of a function class, there are serious restrictions on the kinds of classes that can be so treated. Each function in the class must consist of simpler functions whose values are combined by means of a boolean operator, and that operator must be fixed before learning begins. In this chapter we will treat a much broader topic: functions that can be written as compositions of other functions. We will present a formalism for describing functional composition and use it to show that classes of similarly composed functions are rich. We will then introduce an alternative to the \( \Pi \) notation that we have been using, and use the greater expressiveness of this notation to obtain more precise bounds on the number of partitions a hypothesis class can induce on a training list.

3.1. Functional composition

In the last chapter, we made an explicit distinction between the domains of functions that we conjected. Thus, the function \( \circ (H_1, H_2, \ldots, H_n) \), was said to have \( \text{Dom}(H_1) \times \text{Dom}(H_2) \times \cdots \times \text{Dom}(H_n) \) as its domain. This led to the use of “\( \circ \)” notation in representing a training list; a list of points from \( \text{Dom}(H_1) \times \text{Dom}(H_2) \times \cdots \times \text{Dom}(H_n) \) was represented as \( S_1 \circ S_2 \circ \cdots \circ S_n \). We used this notation in order to make it obvious that our formalism could be applied even when the functions being conjected were quite heterogeneous.

However, we will find this notation cumbersome in what follows. Therefore, when we conject the functions \( H_1, H_2, \ldots, H_n \) we will henceforth treat them as though they had a single domain in common. If it is not clear that the functions being conjected have similar domains, we can regard each point \( x \) in this domain as a list of \( n \) values, and say that \( H_1(x) \) only depends on the first element of the list, \( H_2(x) \) depends only on the second element, and so on.

Definition: Let \( \mathcal{F} = \{ \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n \} \) be a set of function classes, defined so that \( \text{Dom}(\mathcal{F}_1) = \text{Dom}(\mathcal{F}_2) = \cdots = \text{Dom}(\mathcal{F}_n) \). Then we say that \( \text{Dom}(\mathcal{F}) \) is the common domain of \( \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n \).
Definition: Let $X$ and $Y$ be sets, let $H_0$ and $H_1$ be functions mapping $X$ to $Y$, and $f$ be a function mapping $X \times Y$ to $X$. Then, for any point $x \in X$, the composition of $H_0$ and $H_1$ with respect to $f$ is the function that takes $x$ to $H_0(f(x, H_1(x)))$.

We use the $G_f(H_0, H_1)$ to denote the composition of $H_0$ and $H_1$ with respect to $f$.

A composition is obtained by using the value of one function as the argument to another. For example, the class of rules having the form

$$H(a_1, a_2) = \begin{cases} 1 & \text{if } a_1 + a_2 > \beta, \\ 0 & \text{otherwise} \end{cases} \quad (a \in \mathbb{R})$$

(3.1)

can be viewed as a composition of the functions $J(b_1, b_2) = b_1 + \alpha b_2$ and the functions

$$I(a_1) = \begin{cases} 1 & \text{if } a_1 > \beta, \\ 0 & \text{otherwise}. \end{cases}$$

(3.2)

(The composition is $H(a_1, a_2) = I(a_1 + \alpha a_2).$)

In our formalism, we would say that both $I$ and $J$ have the domain $\mathbb{R} \times \mathbb{R}$. For any $(x, y) \in \mathbb{R} \times \mathbb{R}$

$$I((x_0, y_0)) = \begin{cases} 1 & \text{if } x_0 > \beta, \\ 0 & \text{otherwise}, \end{cases}$$

and

$$J((x_1, y_1)) = x_1 + \alpha y_1.$$  

To compose them we could define the function $f$ so that $f((x_2, y_2), z) = (z, x_2)$. Then for all $(x, y)$,

$$f((x, y), J((x, y)))$$

is

$$J((x, y), z) = (x + \alpha y, x),$$

and

$$G_f(I, J) = I(f((x, y), J((x, y)))) = I((x + \alpha y, x)),$$

which is

$$\begin{cases} 1 & \text{if } x + \alpha y > \beta, \\ 0 & \text{otherwise}, \end{cases}$$

as desired.

In the expression $G_f(H_0, H_1)(x)$, we can informally say that $f$ determines which part of $H_0$'s argument comes from $x$, and which part comes from $H_1(x)$. However $f$ can manipulate $x$ and $H_1(x)$ arbitrarily to determine the value it will pass on to $H_0$. Since many functions can serve as $f$, we have many composition operators.
If \( \mathcal{F}_0 \) and \( \mathcal{F}_1 \) are function classes, we will follow our previous convention by saying that

\[
\mathcal{C}_f(\mathcal{F}_0, \mathcal{F}_1) = \{ \mathcal{C}_f(H_0, H_1) : H_0 \in \mathcal{F}_0, H_1 \in \mathcal{F}_1 \}.
\]

Because it seems that the concept of functional composition is often associated with lambda calculus, it may be appropriate to point out that our formalism does not, in fact, describe a lambda calculus. The most important difference is that a lambda-expression can evaluate to another lambda-expression which is then evaluated in its turn. This is what permits recursive functions to be represented.

Our interest is in non-recursive functions that can be built up by using composition; an appropriate example is a multi-layer neural network in which neural units in the first layer provide inputs for units in the second layer, units in the second layer provide inputs for units in the third layer, and so on. We illustrate this in the next example.

Example 3.1: Consider the simple network shown in figure 1. Since the node labeled \( F_1 \) has two inputs, we will find it convenient to say that the function calculated by each of the three nodes has some set of ordered pairs as its domain; let us simply say that the domain of each function is \( \mathbb{R} \times \mathbb{R} \). We will let the range of each function be \( \mathbb{R} \) as well. We will begin by composing \( F_1 \) and \( F_2 \). If we regard this composition as a function if itself, we see that it has two inputs, and that \( F_2 \) receives the first of these while \( F_1 \) receives the second. When this function is evaluated at the point \((x, y)\), we

![Figure 1: A simple network of three functions.](image-url)
first evaluate $F_2(x)$ and then evaluate $F_1(F_2(x), y)$. (In our formalism it is more precise to say that we first evaluate $F_2((x, y))$ and then evaluate $F_1((F_2((x, y)), y))$, while taking note that $F_2((x, y))$ really only depends on $x$.) We write the composition of $F_1$ and $F_2$ as $\mathcal{G}_{f_1}(F_1, F_2)$ where $f_1$ is such that

$$V((x, y) \in \mathbb{R}^2, z \in \mathbb{R}) : f_1((x, y), z) = (z, y).$$

To complete the composition we compose $\mathcal{G}_{f_1}(F_1, F_2)$ with $F_3$, using the value of $F_3$ for both arguments of $\mathcal{G}_{f_1}(F_1, F_2)$. We write this composition as

$$\mathcal{G}_{f_2}(\mathcal{G}_{f_1}(F_1, F_2), F_3),$$

where $f_2$ is such that

$$f_2((x, y), z) = (z, z)$$

for all $(x, y) \in \mathbb{R}^2, z \in \mathbb{R}$.

Note that we did not specify $F_1$, $F_2$, or $F_3$ in the last example. The philosophy behind our construction is that we are only providing a framework, which the learning algorithm will complete by choosing $F_1$, $F_2$, and $F_3$. We have, so to speak, specified the hypothesis architecture that the learning algorithm follows when building its hypotheses.

In the next section we will discuss the analysis of learning algorithms whose hypotheses are composite functions and, by extension, learning algorithms whose hypotheses follow architectures based on functional composition.

### 3.2. Analyzing algorithms that learn composite functions.

Our first result is simply that functional composition is ricitic:

**Proposition 3.2:** Let $X$ be a set and let $\mathcal{M}_0$ and $\mathcal{M}_1$ be two function classes, with $\text{Dom}(\mathcal{M}_0) = \text{Dom}(\mathcal{M}_1) = X$. Let $f : X \times \text{Ran}(\mathcal{M}_1) \to \text{Dom}(\mathcal{M}_0)$. Let $S \in X^t, t \in \mathbb{N}$. Then

$$\Pi_{\mathcal{G}(f, \mathcal{M}_0, \mathcal{M}_1)}(S) \leq \Pi_{\mathcal{M}_0}(t) \Pi_{\mathcal{M}_1}(t).$$
Proof: The proof is given in the appendix to this chapter.


Example 3.3: \( N \)-ary boolean circuits having a fixed architecture

A boolean circuit is an expression consisting of \( \{0, 1\} \)-valued functions whose arguments are either inputs to the circuit (taking on the value 0 or 1), or values of other functions in the circuit, with the understanding that there are to be no cycles. One of the functions is distinguished in that its value is regarded as the value of entire the boolean circuit.

In the present case we will specify the architecture of the circuit before learning begins; that is, we will say in advance which inputs will be supplied by which functions, instead of letting the learning algorithm decide.

![A boolean circuit containing four functions.](image-url)

Our class of boolean circuits can be written as a composition of the function classes \( F_1 \cdots F_n \). As an example, we will construct a class of circuits that have the same form as the circuit in figure 2: if \( F_4 \) is chosen from a class \( F_4 \) of binary functions, and \( F_3 \) is chosen from the class \( F_3 \) of unary functions, then \( C (F_4, F_3) \) is a class of binary \( \{0, 1\} \)-valued functions which can be represented like the circuit consisting of \( F_3 \) and \( F_4 \) in figure 3 (By looking at figure 3 we can see that the functions in the class we have constructed take two arguments. To be formal we could call this class \( C (F_4, F_3) \), where \( f_1 \) is such that \( f_1 (x, y, z) = \langle x, y \rangle \) for all \( \langle x, y \rangle \) in the domain and all \( z \) in the range of \( F_3 \). From figure 2 we can see that the values of \( x \) and \( y \) will ultimately come from a function in \( F_3 \) and \( F_4 \), respectively.)
We can specify that the function we chose from $\mathcal{F}_2$ is to supply the first of $\mathcal{F}_3(\mathcal{F}_4, \mathcal{F}_5)$'s two arguments by
expanding our function class to
\[ \mathcal{C}(\mathcal{G}_d(\mathcal{F}_n, \mathcal{F}_3), \mathcal{F}_2), \]
as in figure 4. The resulting function is binary, with the first argument going to a function in \( \mathcal{F}_3 \) and the second going to a function in \( \mathcal{F}_2 \). We can write the composition as
\[ \mathcal{C}_{f_3}(\mathcal{C}_{f_2}(\mathcal{F}_n, \mathcal{F}_3), \mathcal{F}_2), \]
where
\[ f_2((x, y), z) = (x, z) \]
for all \((x, y)\) in the domain and all \(z\) in the range of \( \mathcal{F}_2 \).

The class of circuits we want to construct (figure 2) is like the one in figure 4 except that both of the inputs come from a function in \( \mathcal{F}_3 \). We can describe the entire class of circuits with the composition
\[ \mathcal{C}_{f_3}(\mathcal{C}_{f_2}(\mathcal{F}_n, \mathcal{F}_3), \mathcal{F}_2), \]
where, for all \((x, y)\) in the domain and all \(z \in \text{Ran}(\mathcal{F}_1)\),
\[ f_3((x, y), z) = (z, z). \]

By proposition 3.2,
\[ \Pi_{d}^{f_3}(\mathcal{C}_{f_2}(\mathcal{F}_n, \mathcal{F}_3), \mathcal{F}_2)(f) \leq \Pi_{d_1}(\ell) \Pi_{d_2}(\ell) \Pi_{d_3}(\ell) \Pi_{d_4}(\ell) \]
for any positive \( \ell \). By appealing to proposition 2.11 we can conclude that the VC Dimension of the function class described by figure 2 is no larger than
\[ \mathcal{V}(\mathcal{F}_1) + \mathcal{V}(\mathcal{F}_2) + \mathcal{V}(\mathcal{F}_3) + \mathcal{V}(\mathcal{F}_4) \]

\section{3.3. function networks}

This section formalizes a different way of conjecting hypotheses. We represent the composite function as a graph, in which the vertices correspond to functions and the edges describe the flow of information between functions, much as in a neural network. We show that our previous formalism for describing compositions is at least as powerful as the function-graph formalism. This section is, in some sense, an extended example, and the reader may omit it if he or she chooses without loss of continuity.
We will use the method we used to construct a circuit class for figure 4 can be used to construct an arbitrary fixed-architecture boolean circuit, and the result will be a ricetic combination of the function classes that are used
to create the circuit class.

To show this we first define a function network as a directed acyclic graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V}$ is a set
of vertices and $\mathcal{E} \subseteq \mathcal{V}^2$. We intend that a vertex in $G$ represents a function whose arity is equal to the number
of edges coming into that vertex. We will say that the graph $G$ represents a function $F_G$, and in the next several
paragraphs we define the value of $F_G$ at an arbitrary point $x$.

We define the relation $\rightarrow$ so that, if $V_i, V_j \in \mathcal{V}$, then $V_i \rightarrow V_j$ if and only if $(V_i, V_j) \in \mathcal{E}$; $V_i \rightarrow V_j$ informally
means that the function at $V_i$ supplies one of the arguments for the function at $V_j$. We also define $\rightarrow^*$ as the
transitive closure of $\rightarrow$. If $n$ is the number of vertices in the graph, we index the vertices with the members of the
set $\{1, 2, \ldots, n\}$ in such a way that each vertex receives a unique index, and that the index of $V_i$ is greater than
the index of $V_j$ if and only if $\neg (V_i \rightarrow V_j)$. (In other words, the inputs to a vertex may only come from vertices
with smaller indices. Note that we are merely formalizing the procedure that we used in example 3.2.) Henceforth
we will let $V_i$ represent the vertex with the index $i$, for all $1 \leq i \leq n$.

Each vertex in $G$ corresponds to a function, and the function that a vertex $V_i$ corresponds to will be represented
by $F_i$.

For all $1 \leq i \leq n$, we let $\mathcal{V}_i$ be the set of vertices indexed from $i$ to $n$, and we define the subgraph $N_i$ as
follows:

$$N_i \equiv \{(V_j, V_k) : V_j, V_k \in \mathcal{V}_i, \rightarrow^*\}$$

For each $1 \leq i \leq n$, let $\mathcal{E}_i$ be the set of edges entering the subgraph $N_i$, that is:

$$\mathcal{E}_i = \{V_h, V_j) \in \mathcal{E} : h < i, j \geq i\}.$$ 

If $m = |\mathcal{E}|$ we will use $E_1, E_2, \ldots, E_m$ to denote the members of $\mathcal{E}$. Let $(v_1, v_2, \ldots, v_m)$ be a list of $m$
values that we wish to use as arguments for $N_i$. We calculate the value of $N_i$ as follows: if $i = n$ then

$N_i((v_1, v_2, \ldots, v_m)) = F_n((v_1, v_2, \ldots, v_m))$. Otherwise,

a. For each incoming edge $E_j \in \{E_1, E_2, \ldots, E_m\}$ define $\text{Val}_i(E_j)$ to be $v_j$.

b. Let $\{E_1, E_2, \ldots, E_m\}$ denote the set $\mathcal{E}_i+1$. For each $E_j \in \{E_1, E_2, \ldots, E_m\}$, define $\text{Val}_{i+1}(E_j)$ as follows:

$$\text{Val}_{i+1}(E_j) = \begin{cases} \text{Val}_i(E_j), & \text{if } E_j \in \mathcal{E}_i; \\ F_i((v_1, v_2, \ldots, v_m)), & \text{otherwise}. \end{cases}$$
Example 3.4: Boolean circuits as function graphs

For this example we will return again to the boolean circuit described in figure 2. However we need to augment this diagram slightly because in our formalism the only subgraphs that can have arguments are those with incoming edges (that is, edges leading from vertices outside the subgraph to vertices inside the subgraph). In figure 5 we have added an extra node to the diagram of figure 2, this node might be viewed as an input node or as an input-providing oracle such as that used by [3] or [30]. (In fact, of course, the extra vertex is only there because it is notationally too cumbersome to specify arguments to a complete graph differently than we specify arguments to a subgraph.)

\[ (V_1, V_2, V_3, V_4, V_5) \]

\[ \{E_1, E_2, E_3, E_4, E_5\} = \{(V_1, V_2), (V_2, V_3), (V_2, V_4), (V_3, V_5), (V_4, V_5)\} \]

The function we want to evaluate is \( N_2 \), since this is the one that corresponds to the circuit in figure 2. It corresponds to the subgraph \( \{E_2, E_3, E_4, E_5\} \). Note that \( \mathcal{E}_2 \) is not \( \{E_2, E_3, E_4, E_5\} \) but rather \( \{(V_2, V_4) \in \mathcal{E} : h < 2, j \geq 2\} \); this set is \( \{E_1\} \). We will evaluate \( N_1 \) at the input...
We first define \( \text{Val}_4(E_1) \) to be \( v \), as in item (a) in the procedure we have given for evaluating the value of \( N_1 \).

Subsequently, according to item (b), we determine which edges are in \( \delta_2 = \{ \{ V_h, V_j \} \in \delta : h < 3, j \geq 3 \} \); these are \( F_2 \) and \( F_3 \). Since neither edge is also in \( \delta_3 \), item (b) says \( \text{Val}_3(E_2) \) and \( \text{Val}_3(E_3) \) are both \( F_2(v) \), where \( F_2 \) is the function associated with \( V_2 \).

Next, according to item (c), we must evaluate \( N_4(\text{Val}_4(E_3), \text{Val}_4(E_4)) = N_4(F_2(F_2(v), F_3(v)), F_2(F_2(v), F_2(v))) \), according to step (e). Again, we find that step (a) was already completed; so, according to step (b), we determine that \( \delta_5 = \{ F_4, E_5 \} \). \( E_5 \) is in \( \delta_3 \) as well, so \( \text{Val}_3(E_5) = \text{Val}_3(E_4) = F_2(F_2(v), F_2(v)) \). \( E_5 \) is not in \( \delta_4 \) so \( \text{Val}_4(E_5) = F_4(F_3(v), F_4(F_3(v), F_2(v))) \).

Next, we proceed to the evaluation of

\[
N_5(\text{Val}_5(E_4), \text{Val}_5(E_3)) = N_5(F_5(F_2(v), F_3(v)), F_4(F_3(v), F_3(v), F_3(F_3(v), F_2(v))))
\]

Since there are only 5 vertices in the function graph, \( N_5(F_3(F_3(v), F_2(v)), F_4(F_3(v), F_3(v), F_3(F_3(v), F_2(v)))) \) is defined to be simply \( F_3(F_3(F_3(v), F_2(v)), F_4(F_3(v), F_3(v), F_3(F_3(v), F_2(v)))) \).

Notice that we could have written

\[
F_3(F_3(F_2(v), F_4(v)))
\]

since \( F_3 \) and \( F_4 \) only have one incoming edge apiece. However, in accordance with our earlier conventions, we say that \( F_3 \) and \( F_4 \) each have two arguments, but that the second argument is ignored in the sense that \( F_3((x, y)) \) only depends on \( x \), for any \( (x, y) \) in \( F_3 \)'s domain, and likewise for \( F_4 \).

Proposition 3.5: Let \( F_1, F_2, \ldots, F_n \) be \( n \) functions with domain \( X \), and let \( G \) be a function graph whose \( n \) vertices correspond to \( F_1, F_2, \ldots, F_n \). Let \( N_i \) be the subgraph consisting of the vertices in \( G \) whose indices are greater than or equal to \( i \), \( 1 \leq i \leq n \), and let \( \overline{v} \) be a list of values whose length is equal to \( |\delta_i| \). Let \( F_N \) denote the function that the subgraph \( N_i \) represents. Then

\[
\exists(f_1, f_{i+1}, \ldots, f_n) : \alpha_{f_1}(\alpha_{f_{i+1}}(\cdots(\alpha_{f_n}(F_n, F_{n-1}), F_{n-2}), \cdots)F_{i+1})(v) = F_N(\overline{v}).
\]
Proof: Our proof is by induction on i. If i = 1 then the claim follows from the definition of the value of $N_n(\vec{v})$ above, that is, $N_n(\vec{v}) = F_n(\vec{v})$.

For the inductive step, let $1 \leq i < n$ and assume we have a set of functions \( \{f_{i+1}, f_{i+2}, \ldots, f_i\} \) such that, for all $\vec{e}_{i+1} \in \text{Dom}(F_{N,i+1})$,

$$F_{N,i+1}(\vec{v}) = \varphi_{f_{i+1}}(\varphi_{f_{i+2}}(\cdots \varphi_{f_{i-k}}(F_{N-i}(F_n, F_{n-1}), F_{n-2}), \cdots), F_{i+2}, F_{i+1})(\vec{v}).$$

The evaluation of $F_N$ for some list of arguments $\vec{w}$, $|\vec{w}| = |\mathcal{S}|$ proceeds as follows: Let $(E_1, E_2, \ldots, E_m)$ be the set of edges in $\mathcal{S}_{i+1}$, and, without loss of generality, assume that

$$E_1, E_2, \ldots, E_k \in \mathcal{S}_{i+1}, E_{k+1}, E_{k+2}, \ldots, E_m \not\in \mathcal{S}_{i+1}$$

for some $1 \leq k \leq m$.

By item (b) in the definition of a function network, $F_N(\vec{w})$ is

$$F_{N,i+1}((w_1, w_2, \ldots, w_k, F_1(\vec{w}), F_2(\vec{w}), \ldots, F_i(\vec{w}))).$$

But this is the same as

$$\varphi_{f_i}(F_{N,i+1}, F_i)(\vec{w}).$$

if, for all $(w_1, w_2, \ldots, w_k)$ and all $z$

$$f_i((w_1, w_2, \ldots, w_k), z) = (w_1, w_2, \ldots, w_k, z, z, \ldots, z).$$

But by the inductive hypothesis $F_{N,i+1}$ calculates the same function as

$$\varphi_{f_{i+1}}(\varphi_{f_{i+2}}(\cdots \varphi_{f_{i-k}}(F_{N-i}(F_n, F_{n-1}), F_{n-2}), \cdots), F_{i+2}, F_{i+1}),$$

so (3.3) is just

$$\varphi_{f_i}(\varphi_{f_{i+1}}(\varphi_{f_{i+2}}(\cdots \varphi_{f_{i-k}}(F_{N-i}(F_n, F_{n-1}), F_{n-2}), \cdots), F_{i+2}, F_{i+1}), F_i)(\vec{w}).$$

Since (3.3) is the same function as $F_N(\vec{w})$, this completes the proof. \qed
Example 3.6: Boolean circuits with a learned architecture

The circuits to be treated in this example are acyclic, as in the last example, and we will continue to stipulate that the function chosen from the class $\mathcal{F}_j$ may receive inputs only from the functions that we chose from $\mathcal{F}_{j-1}, \mathcal{F}_{j-2}, \ldots, \mathcal{F}_1$. However, we will not specify in advance which arguments are to be provided by which functions. Thus, if our class of circuits contains compositions of the functions in $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3,$ and $\mathcal{F}_4$, the class of circuits is to contain all of the circuits in figure 6 and others as well.

This case can be treated as a straightforward generalization of the case treated in the previous example. We apply a simple trick: first, the function will graph contain every edge that may appear in a directed acyclic graph of $n$ vertices. Secondly, we will let the learning algorithm decide which arguments are ignored at each vertex.

We will use our function-graph formalism for this example, so we will have an extra vertex, $V_0$, to provide the input to our composite function as we did in example 3.6. We will the set of edges $\mathcal{E}$ be $\{(V_i, V_j) : V_i, V_j \in \mathcal{V}, i < j \leq n\}$. In other words, vertex $V_j$ has an incoming edge from each vertex $V_i$ when $0 \leq j < i$.

Since there must be some vertex in an acyclic digraph with no incoming edges, and there must be some vertex with at most one incoming edge, etc, this edge set does, in fact, allow us to construct any acyclic digraph with $n$ vertices by simply removing edges (or, in our terminology, by ignoring edges).
Since \( V_j \) has \( j \) incoming edges, we will say that the function corresponding to \( V_j \) ignores all but the first \( j \) inputs it receives. In fact, it may ignore some of the first \( j \) inputs too, for example, in may be a binary boolean operation and only use the first two of its inputs.

Let \( \mathcal{F}_j \) be the function class corresponding to the vertex \( V_j \), and assume that \( \mathcal{F}_j \) contains some function \( F_a \) that uses the inputs associated with first and fourth incoming edges, while ignoring the other inputs. To have a learned architecture we must let the learning algorithm decide where the first and fourth edges come from.

To do this we will add a function to \( \mathcal{F}_j \) that is exactly like \( F_a \), but uses the first and third incoming edges; we will add another that uses the first and second, and so on. Let \( A_j \) denote the number of inputs actually used by \( F_j \). In general, each function \( F_j \in \mathcal{F}_j \) can use up to \( A_j \) incoming edges, so there will be up to \( A_j! \) ways to permute \( F_j \)'s arguments, and up to \( \binom{j}{A_j} \) ways to decide which of the \( j \) potential arguments will be used by \( F_j \) (and have their order permuted).

The new function class we construct will therefore contain \( \binom{j}{A_j} A_j! \) functions for every function in \( \mathcal{F}_j \).

Let this new function class be denoted \( \mathcal{F}_j' \). We can think of this as a collection of up to \( \binom{j}{A_j} A_j! \) smaller function classes which are like \( \mathcal{F}_j \) except that they use different arguments. Each of these smaller classes can induce up to \( \Pi_{\mathcal{F}_j}(\ell) \) subsets on a list of length \( \ell \), so the collection all the smaller classes induces up to \( \binom{j}{A_j} A_j! \Pi_{\mathcal{F}_j}(\ell) \) subsets on such a list. In other words,

\[
\Pi_{\mathcal{F}_j'}(\ell) \leq \binom{j}{A_j} A_j! \Pi_{\mathcal{F}_j}(\ell).
\]

By proposition 2.14 this implies that

\[
\ell'(\mathcal{F}_j') \leq \begin{cases} 
(151)^{\ell'}(\mathcal{F}_j) 2^{-\log_2(w)/\ell'}(\mathcal{F}_j) & \text{if } \log_2(w)/\ell'(\mathcal{F}_j) \leq 0.5, \\
\frac{8.2}{\epsilon} \log_2(w) & \text{if } \log_2(w)/\ell'(\mathcal{F}_j) \geq 0.5.
\end{cases}
\]

(3.4)
where
\[ w = \left( \frac{j}{A_j} \right) A_j !. \]

By proposition 2.11 and the fact that our class of boolean circuits is rich (as was shown in the previous example), the VC Dimension of this class of circuits is no greater than
\[ T_t \left( \sum_{i=1}^{t} r(\mathcal{H}_i) \right), \]
where the \( r(\mathcal{H}_i) \) terms are provided by (3.4).

### 3.4. An alternative notation

In this section we introduce an alternative notation to \( \Pi(S) \) and \( \Pi(\ell) \) that allows us to prove a slightly more expressive result than proposition 3.2. Recall the definition of \( \Pi(C)(S) \): it is the size of the set
\[ \{F \cap S : F \in \mathcal{F} \}. \tag{3.5} \]
We will use \( \Gamma(\mathcal{F}, S) \) to denote this set. We will also define
\[ \Delta(\mathcal{F}, S) = \{ \{F : F \in \mathcal{F}, F \cap S = \emptyset \} : v \in \Gamma(\mathcal{F}, S) \}. \]
Therefore \( \Delta(\mathcal{F}, S) \) is a set of sets whose union is \( \mathcal{F} \), and each set in \( \Delta(\mathcal{F}, S) \) consists of hypotheses that induce the same subset on \( S \).

Finally, for any list
\[ S = (x_1, x_2, \ldots, x_t), \]
and any hypothesis class \( \mathcal{F} \) with the property that \( \Pi(\mathcal{F})(S) = |\Gamma(\mathcal{F}, S)| = 1 \), we define \( U(f, H, S) \) as the list
\[ (u(f, H, x_1), u(f, H, x_2), \ldots, u(f, H, x_t)) \]
for any function \( H \) whose domain contains the members of \( S \), and define \( U(f, \mathcal{F}, S) \) as the list
\[ (u(f, \mathcal{F}, x_1), u(f, \mathcal{F}, x_2), \ldots, u(f, \mathcal{F}, x_t)), \]
where
\[ u(f, \mathcal{F}, x) \equiv f(x, c), \quad \{c = \max \{I(x) : l \in \mathcal{F}\}. \tag{3.6} \]
Note that, since \( \Pi(\mathcal{F})(S) = |\Gamma(\mathcal{F}, S)| = 1 \) the set \( \{I(x) : l \in \mathcal{F}\} \) only has one member (that is, all \( l \in \mathcal{F} \) have the same value at \( x \)).

We show the following in the appendix:
Proposition 3.7: Let $X$ be a set and let $\mathcal{X}_0$ and $\mathcal{X}_1$ be two function classes, with $\text{Dom}(\mathcal{X}_0) = \text{Dom}(\mathcal{X}_1) = X$. Let $f : X \times \text{Ran}(\mathcal{X}_1) \to \text{Dom}(\mathcal{X}_0)$. Let $S \in X^t, t \in \mathbb{N}$. Then
\[ \Pi_{f, \mathcal{X}_1, \mathcal{X}_1}(S) = \sum_{f \in \Delta(\mathcal{X}, S)} \Pi_{\mathcal{X}_1}(U(f, \mathcal{X}, S)). \]

Proof: The proof is given in the appendix to this chapter. \[\square\]

This leads easily to the next result:

Proposition 3.8: Let $X$ be a set and let $\mathcal{X}_0$, $\mathcal{X}_1$, and $\mathcal{X}_2$ be three function classes, with $\text{Dom}(\mathcal{X}_0) = \text{Dom}(\mathcal{X}_1) = \text{Dom}(\mathcal{X}_2) = X$. Let $f_1 : X \times \text{Ran}(\mathcal{X}_1) \to \text{Dom}(\mathcal{X}_0)$ and $f_2 : X \times \text{Ran}(\mathcal{X}_2) \to \text{Dom}(\mathcal{X}_1)$. Let $S \in X^t, t \in \mathbb{N}$. Then
\[ \Pi_{\mathcal{X}_1}(f_1, (\mathcal{X}_1, \mathcal{X}_1))(S) = \sum_{f_1 \in \Delta(\mathcal{X}_1, S)} \sum_{f_2 \in \Delta(\mathcal{X}_2, U(f_1, \mathcal{X}, S))} \Pi_{\mathcal{X}_1}(U(f_2, \mathcal{X}_1, U(f_1, \mathcal{X}, S))). \]

Proof: By proposition 3.7,
\[ \Pi_{\mathcal{X}_1}(f_1, (\mathcal{X}_1, \mathcal{X}_1))(S) = \sum_{f_1 \in \Delta(\mathcal{X}_1, S)} \Pi_{\mathcal{X}_1}(U(f_1, \mathcal{X}, S)). \] (3.7)

But, again by proposition 3.7,
\[ \Pi_{\mathcal{X}_1}(f_1, (\mathcal{X}_1, \mathcal{X}_1))(U(f_2, \mathcal{X}_1, S)) = \sum_{f_2 \in \Delta(\mathcal{X}_2, U(f_1, \mathcal{X}, S))} \Pi_{\mathcal{X}_1}(U(f_2, \mathcal{X}_1, U(f_1, \mathcal{X}, S))) \] (3.8)
for all $f_1 \in \Delta(\mathcal{X}_1, S)$. The claim follows from (3.7) and (3.8). \[\square\]

Clearly, we can use the method of proposition 3.7 again on compositions of four functions, then on compositions of five functions, and so on indefinitely. But the nested $U$ functions would soon become too cumbersome to write, so we would like to have a more compact notation. Let $\hat{A}$ abstractly represent an expression, say
\[ \mathcal{G}_F(\mathcal{G}_F(\cdots, F_2, F_1)), \]
and assume that all functions have a unique index. Assume that the argument received by the function $F_1$ in $\hat{A}$ depends on the values of $F_2, F_3, \cdots, F_n$. Then we will say that
\[ \text{Imp}_1(\hat{A}, (F_1, F_2, \cdots, F_n), S) \]
is the list of arguments that $F_1$ sees when $\hat{A}$ sees the list $S$. (The $\text{Imp}_1$ function is defined formally in the appendix.)
Example 3.9:

Consider the composition we arrived at in example 3.3:

\[ \mathcal{G}_{F_1}(\mathcal{G}_{F_2}(F_3, F_4), F_5) \]  

Let us choose a specific \( F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2, F_3 \in \mathcal{F}_3, F_4 \in \mathcal{F}_4, \) and let \( \mathcal{A} \) refer to the composition

\[ \mathcal{G}_{F_1}(\mathcal{G}_{F_2}(F_3, F_4), F_5) \]

Since the argument to \( F_1 \) does not come from any other function in \( \mathcal{A} \), we say that \( \text{Imp}_1(\mathcal{A}, \{\}, S) = S \). The argument to \( F_2 \), however, depends on \( F_1 \); according to our formalism for composition the argument of \( F_2 \) is the same as the argument of \( \mathcal{G}_{F_1}(F_3, F_4), F_5 \). When the composition in (3.9) is evaluated for each member of the list \( S \), then (as we have already stated),

\[ \mathcal{G}_{F_1}(\mathcal{G}_{F_2}(F_3, F_4), F_5) \]

is evaluated for each member in the list \( U(f_3, F_1, S) \). Therefore we must also evaluate \( F_2 \) for each member of the list \( U(f_3, F_1, S) \), and \( \text{Imp}_2(\mathcal{A}, \{F_1\}, S) \) is therefore \( U(f_3, F_1, S) \).

In a similar manner, we see that the input list for \( F_3 \) is the same as the input list for \( \mathcal{G}_{F_2}(F_3, F_5) \), and this is just \( U(f_2, F_2, U(f_3, F_1, S)) \). Therefore

\[ \text{Imp}_3(\mathcal{A}, \{F_2, F_1\}, S) = U(f_3, F_2, U(f_3, F_1, S)). \]

Finally, if the input list for \( \mathcal{G}_{F_1}(F_3, F_5) \) is \( U(f_2, F_2, U(f_3, F_1, S)) \), then the definition of \( U \) states that the input list for \( F_4 \) is

\[ U(f_1, F_3, U(f_2, F_2, U(f_3, F_1, S))). \]

Thus

\[ \text{Imp}_4(\mathcal{A}, \{F_3, F_2, F_1\}, S) = U(f_2, F_3, U(f_2, F_3, U(f_3, F_1, S))). \]

Using our notation, we can extend proposition 3.9 in the following way:
Proposition 3.10: Let \( F_1, F_2, \ldots, F_n \) be a set of function classes, and let \( \mathcal{H} = \{ F_1, F_2, \ldots, F_n \} \) be a class of conject hypotheses consisting of one or more compositions. Then for any \( S \in \text{Dom} (\mathcal{F})^t, t \in \mathbb{N} \)

\[
\left| \Delta (\oplus (F_1, F_2, \ldots, F_n), S) \right| = \\
\sum_{\mathcal{H} \in \Delta (F_1, S)} \sum_{\mathcal{H} \in \Delta (F_2, \mathcal{H}_2(\mathcal{H}_1), S)} \cdots \sum_{\mathcal{H}_n \in \Delta (F_n, \mathcal{H}_n(\mathcal{H}_{n-1}), S)} \Delta (\oplus (F_1, F_2, \ldots, F_n), S).
\]

Proof: The claim is stated formally and then proved in the appendix. \( \square \)

We also show the following in the appendix:

Proposition 3.11: Let \( F_1, F_2, \ldots, F_n \) be a set of function classes, and let \( \mathcal{H} = \{ F_1, F_2, \ldots, F_n \} \) be a class of conject hypotheses consisting of one or more compositions. Then for any \( S \in \text{Dom} (\mathcal{F})^t, t \in \mathbb{N} \) and

\[
\left( \mathcal{H} \in \Delta \left( F_1, \text{Imp}_1(\mathcal{H}, \{\}), S \right) \right), \\
\left( \mathcal{H} \in \Delta \left( F_2, \text{Imp}_2(\mathcal{H}, \{\mathcal{H}_1\}), S \right) \right), \\
\vdots \\
\left( \mathcal{H} \in \Delta \left( F_n, \text{Imp}_n \left( \mathcal{H}, \{F_1, F_2, \ldots, F_{n-1}\}, S \right) \right) \right)
\]

\[
\left| \Delta (\oplus (F_1, F_2, \ldots, F_n), S) \right| = 1.
\]

Proof: The claim is stated formally and proved in the appendix. \( \square \)

This proposition implies that

\[
\left| \Delta (\oplus (F_1, F_2, \ldots, F_n)) \right|
\]

is equal to the number of terms in the summation of proposition 3.9, since the terms being summed are all 1. By the definition of \( \Pi \) this implies that for all \( S \in \text{Dom} (\mathcal{F})^t, t \in \mathbb{N} \),

\[
\Pi (\oplus (F_1, F_2, \ldots, F_n))(S) = \\
\sum_{\mathcal{H} \in \Delta (F_1, S)} \sum_{\mathcal{H} \in \Delta (F_2, \mathcal{H}_2(\mathcal{H}_1), S)} \cdots \sum_{\mathcal{H}_n \in \Delta (F_n, \mathcal{H}_n(\mathcal{H}_{n-1}), S)} 1.
\]
3.5. Placing constraints on the set of legal conjunctions.

Consider again a class of function classes

\[ \{F_1, F_2, \ldots, F_n\}. \]

For each function class \( F_i \in \{F_1, F_2, \ldots, F_n\} \) we define a constraint function

\[ K_i : 2^{F_i} \times 2^{F_i} \times \cdots \times 2^{F_i} \rightarrow 2^{F_i}, \]

and a finite set of sets

\[ \{ F_i, F_j, \ldots, F_k \subseteq 2^{F_i}, F_i, F_j, \ldots, F_k \subseteq 2^{F_i} \}. \]

For any given \( F_1, F_2, \ldots, F_k \) and any given \( F_i, F_j, \ldots, F_k \) let

\[ \mathcal{S}(F_1, F_2, \ldots, F_k) = \bigcup \left\{ K_i(J_{i,1}, J_{i,2}, \ldots, J_{i,k}) : (J_{i,1} \in F_i \wedge F_i \in F_i), (J_{i,2} \in F_i \wedge F_2 \in J_{i,2}), \ldots, (J_{i,k} \in F_i \wedge F_k \in J_{i,k}) \right\}. \quad (3.10) \]

We will call \( \mathcal{S}, \mathcal{S}_2, \ldots, \mathcal{S}_n \) a constraint system for \( F_1, F_2, \ldots, F_n \).

We will use the constraint functions to restrict the hypotheses that may take part in the conjunction. Instead of considering the class of objections

\[ \mathcal{S}(F_1, F_2, \ldots, F_k) \equiv \{F(F_1, F_2, \ldots, F_k) : F_1 \in F_1, F_2 \in F_2, \ldots, F_k \in F_k \} \]

we will only consider members of

\[ \{F(F_1, F_2, \ldots, F_k) : F_1 \in \mathcal{S}_1(F_1, F_2, \ldots, F_k), F_2 \in \mathcal{S}_2(F_1, F_2, \ldots, F_k), \ldots, F_k \in \mathcal{S}_n(F_1, F_2, \ldots, F_k) \}. \quad (3.11) \]

This should be understood as follows: Each combination of function classes

\[ J_{i,1} \in F_i, J_{i,2} \in F_i, \ldots, J_{i,n} \in F_i \]

determines a further function class via \( K_i \).

Suppose we are learning a conjunction of the function classes \( F_1, F_2, \ldots, F_n \), and we have already chosen to use

\[ F_1 \in F_1, F_2 \in F_2, \ldots, F_{i-1} \in F_{i-1}, F_{i+1} \in F_{i+1}, \ldots, F_n \in F_n. \]
It remains to choose and \( F_i \) from \( \mathcal{F}_i \), but we must obey a restriction: \( F_i \) must lie in \( K_i(\mathcal{J}_{i,1}, \mathcal{J}_{i,2}, \ldots, \mathcal{J}_{i,n}) \) for some

\[
J_{i,1} \in \mathcal{J}_{i,1}, J_{i,2} \in \mathcal{J}_{i,2}, \ldots, J_{i,n} \in \mathcal{J}_{i,n}
\]

which are such that

\[
F_1 \in J_{i,1}, F_2 \in J_{i,2}, \ldots, F_n \in J_{i,n}.
\]

In other words, \( F_i \) must be a member of the class

\[
\mathcal{A}(F_1, F_2, \ldots, F_n)
\]

Thus, when the learning algorithm makes its choice for \( F_1 \), that choice is constrained by the choices for \( F_2, F_3, \ldots, F_n \). In a similar manner, all the choices the algorithm makes are constrained by its other choices.

It is possible that

\[
F_1, F_2, \ldots, F_{i-1}, F_{i+1}, \ldots, F_n
\]

may also be subject to constraints involving \( F_i \), and it may be nontrivial to find an \( F_1, F_2, \ldots, F_n \) that satisfy these constraints. However we are not concerned with this at the moment, because the question we are asking now is how to analyze existing learning algorithms. We are concerned with discovering the constraint system (if any) used by the learning algorithm we are analyzing.

Example 3.12:

To give a simple example, suppose a learning algorithm has the hypothesis class

\[
\mathcal{A}(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{12}, \ldots, \mathcal{F}_{23})
\]

but with the added constraint that \( \mathcal{F}_1 = \mathcal{F}_{12} \) and the hypothesis chosen from \( \mathcal{F}_1 \) must always be the same as the hypothesis chosen from \( \mathcal{F}_{12} \). We can formalize this restriction by letting \( \mathcal{J}_{12,1} \) be the set of singleton subsets of \( \mathcal{J}_1 \), and, for all \( J_{12,1}, J_{12,2}, \cdots, J_{12,23} \) letting

\[
K_{12}(J_{12,1}, J_{12,2}, \cdots, J_{12,23}) = J_{12,1}.
\]

With this restriction, we see that (3.10) for \( \mathcal{F}_{12} \) becomes

\[
\mathcal{A}_{12}(F_1, F_2, \cdots, F_{23}) = \bigcup \{ J_{12,1} : J_{12} \in \mathcal{J}_{12,1} \land F_1 \in J_{12,1} \} = \bigcup \{ J_{12,1} \in \mathcal{J}_{12,1} : F_1 \in J_{12,1} \}.
\]
Since each member of $\mathcal{J}_{12,1}$ contains only a single member of $\mathcal{J}_1$, this constraint is the same as

$$\mathcal{G}_{12}(F_1, F_2, \cdots, F_{23}) = \bigcup \{ J_{12,1} \in \mathcal{J}_{12,1} : J_{12,1} = \{ F_i \} \}$$

or simply

$$\mathcal{G}_{12}(F_1, F_2, \cdots, F_{23}) = \{ F_1 \}.$$  

Then, by (3.11), the class of hypotheses under consideration is

$$\left\{ \oplus(F_1, F_2, \cdots, F_{12}, \cdots, F_{23}) : F_1 \in \mathcal{G}_1(F_1, F_2, \cdots, F_{23}), F_2 \in \mathcal{G}_2(F_1, F_2, \cdots, F_{23}), \cdots, F_{23} \in \mathcal{G}_{23}(F_1, F_2, \cdots, F_{23}) \right\}.$$  

Thus the only acceptable choice for $F_{12}$ is $F_1$.

The requirement for consistency also does not appear explicitly in our analysis. We are only concerned with the number of ways in which the learning algorithm can choose constraint functions (that is, the $K_i$ functions described earlier); we do not care that these functions are chosen implicitly via the choices of $F_1, F_2, \cdots, F_n$, rather than being chosen explicitly as $F_1, F_2, \cdots, F_n$ themselves are.

Because of this it is convenient to describe the possible constraint functions for $F_1$ as a class. We define

$$\mathcal{K}_1 = \bigcup \{ \mathcal{K}_1(H_1, H_2, \cdots, H_n) : H_1 \in \mathcal{K}_1, H_2 \in \mathcal{K}_2, \cdots, H_n \in \mathcal{K}_n \}.$$  

Our final result in this chapter is the following:

**Proposition 3.13:** Let $\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_n$ be $n$ hypothesis classes, and let $l$ be a positive integer. Let $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n$ be a constraint system for $\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_n$ and, for each $\mathcal{A}_i \in \{ \mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n \}$, let

$$\mathcal{K}_i = \bigcup \{ \mathcal{K}_i(H_1, H_2, \cdots, H_n) : H_1 \in \mathcal{K}_1, H_2 \in \mathcal{K}_2, \cdots, H_n \in \mathcal{K}_n \}.$$  

Let $\oplus$ be an $n$-ary conjunct operator with the property that for any $\{ \mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_n \} \in \text{Dom}(\oplus)$ and for any $S \in \text{Dom}(\mathcal{F})^l$, $l \in \mathbb{N}$

$$| \Delta(\oplus(\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_n), S) | =$$

$$\sum_{\mathcal{A}_1 \in \Delta(\mathcal{F}_1, S)} \sum_{\mathcal{A}_2 \in \Delta(\mathcal{F}_2, \mathcal{A}_1, S)} \cdots \sum_{\mathcal{A}_n \in \Delta(\mathcal{F}_n, \mathcal{A}_{n-1}, S)} \Delta(\oplus(\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n), S).$$  

Let $\mathcal{F}$ be the class

$$\{ \oplus(F_1, F_2, \cdots, F_n) : F_1 \in \mathcal{G}_1(F_1, F_2, \cdots, F_n), F_2 \in \mathcal{G}_2(F_1, F_2, \cdots, F_n), \cdots, F_n \in \mathcal{G}_n(F_1, F_2, \cdots, F_n) \}.$$
Then, for any list \( S \in \text{Dom}(\mathcal{F}) \),

\[
\Pi_{\mathcal{F}}(S) = \sum_{K_1 \in K_1} \sum_{K_2 \in K_2} \ldots \sum_{K_n \in K_n} \sum_{F \in \Delta(K_1, S)} \sum_{F \in \Delta(K_2, \mathcal{F}((S, K_1), S))} \ldots \sum_{F \in \Delta(K_n, \mathcal{F}((S, K_1, \ldots, K_{n-1}), S))} |\Delta((S, K_1, \ldots, K_n), S)|.
\]

Proof: The proof is given in the appendix. \( \square \)

3.6. Discussion

In this chapter we attempted to create a fairly comprehensive framework for the analysis of sample complexity in learning algorithms.

In the latter part of this chapter we developed several methods for evaluating \( \Pi_{\mathcal{F}}(S) \) in composite functions. An important caveat is that \( \Pi_{\mathcal{F}}(S) \) cannot be used directly to determine the VC Dimension of \( \mathcal{F} \); we need \( \Pi_{\mathcal{F}}(f) \).

This chapter, more than the others in this thesis, presented a snapshot of ongoing research. This research has two goals: first, to provide methods for analyzing algorithms whose hypotheses are not \( \{0,1\} \)-valued, and second, to provide tighter bounds on the sample sizes required to achieve good generalization ability. Our \( \Delta \) and \( f \) notation facilitates the first of these goals by divorcing our formalism somewhat from the terminology of \( \{0,1\} \)-valued functions, and the second, by allowing the interaction of the training sample and the hypothesis class to be described more precisely.

The two goals above are fairly general, so it is not surprising that machine learning researchers have already pursued them to some extent. [19] devotes two chapters to the analysis of algorithms whose hypotheses are real-valued, and [13] contains many results as well. But the issue that is most germane in the context of this thesis is the analysis of hypothesis classes whose ranges are finite and small, not those whose ranges are infinite. This is because the thesis is aimed at the analysis of existing learning algorithms, and many of these use hypotheses with small ranges. The author is not aware of previous results addressing this issue.

In the area of obtaining tighter bounds on sample complexity, authors have incorporated distributional assumptions in order to derive average-case results (C.f. [34]), or at least assumed that certain distributions are known.
as in [14]. But these results are limited in their practical applicability since the distributions in question may not be known. The hope in developing proposition 3.12 was to develop a more precise way of describing the behavior of learning algorithms, and ultimately to incorporate distributional information gradually, so that approximate or incomplete information about these distributions can be used to provide at least some improvement in sample complexity bounds.

3.7. Bibliographic notes

In the area of bounding VC Dimensions for arbitrary hypothesis classes, there are two important results. The first is mentioned in [7], [19], and elsewhere: for any finite hypothesis class \( \mathcal{H} \),

\[
\gamma'(\mathcal{H}) \leq \log_2(|\mathcal{H}|).
\]

This is a trivial consequence of the definition of the VC Dimension. The second result is presented in [4]; this paper shows (in the language of the present thesis) that the combination of functions in a fixed network architecture is a ricet operation, and gives a weaker version of proposition 2.11. We showed in section 3.3 that our result is at least as powerful as that of [4], in that it applies to function networks. It is not clear that proposition 3.2 is more powerful than [4]'s result. However, our notation is more precise, and, in my opinion, makes it easier to determine whether an algorithm can be analyzed with the help of proposition 3.2.

(The reason that this thesis re-derived such a large part of [4]'s result, instead of merely citing the paper, is that the proof presented there is vague and apparently intended for application to the analysis of neural networks. To show formally that Baum and Haussler's result could be applied to the problem at hand would, in my opinion, have required nearly as much work as the approach that was finally used.)

I am not familiar with any previous attempts to introduce mutual constraints on the functions that may take part in a conjecture.
Appendix A to Chapter 3.
A result on learning non-\{0, 1\}-valued functions

In this appendix we will extend the above results to conjunctions having the form \(\oplus(H_1, H_2, \ldots, H_k)\) where \(H_1, H_2, \ldots, H_k\) are not \{0, 1\}-valued functions.

To extend our earlier results to these conjunctions, we must express \(\mathcal{F}(\mathcal{C}(\mathcal{H}))\) in terms of \(\mathcal{F}(\mathcal{C}(\mathcal{H}))\), \(\mathcal{F}(\mathcal{C}(\mathcal{H}))\), \(\mathcal{F}(\mathcal{C}(\mathcal{H}))\) etc. (recall that \(\mathcal{C}(\mathcal{H}) = \{\star : f : \mathcal{H} \rightarrow \{0, 1\} : f \in \mathcal{H}\}\).)

It is easy enough to say what \(\mathcal{C}(\mathcal{H})\) is: we know that there is some particular function \(F\) that we want to the learned conjunction to approximate, and so, for any conjunction \(\oplus(H_1, H_2, \ldots, H_k)\) in \(\ominus(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k)\), and any instance \(x^v\) in its domain, we can define \(x^v\) to be \(F(x^v)\) and so determine the value of \(\mathcal{C}(\mathcal{H})\) at the instance in question.

But it is difficult to define \(\mathcal{C}(\mathcal{H})\) if \(\mathcal{H}\) is merely one of the hypothesis classes taking part in the conjunction \(\ominus(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k)\). This is because we may not know how to construct an \(x^v\) for some or all \(x^v \in \text{Down}(\mathcal{H})\); the paradigm of supervised learning only guarantees that we will know the desired values of the function being learned, and it doesn’t have anything to say about individual parts of such a function.

It seems that we are faced with what, in (25), was called a credit assignment problem: if

\[\oplus(H_1, H_2, \ldots, H_k)(x_1, x_2, \ldots, x_k)\]

is not what we want it to be, which of the functions \(H_1, H_2, \ldots, H_k\) will we hold responsible for the error?

Fortunately, in developing ways to analyze the performance of learning algorithms, we will find that this problem is not as difficult as it would be if we were developing the algorithms themselves. We will be able to assign credit (or blame) in a somewhat arbitrary way. This section will show that it is sufficient, for our purposes, to have an acceptable credit-assignment function, which we define as follows:

**Definition:** Consider \(k\) classes of functions \(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k\) and a corresponding class of conjunctions

\[\oplus(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k)\].

The functions \(F_1, F_2, \ldots, F_k\) are acceptable target functions for \(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k\) with respect to \(\oplus\) and

\[C(\oplus(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k))\]
if and only if

\[ C(\hat{\beta}(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \subseteq \{\{(x_1, x_2, \ldots, x_k) : \oplus((F_1(x_1) \neq H_1(x_1)), (F_2(x_2) \neq H_2(x_2)) \ldots) : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2, \ldots\} \],

where \( \hat{\beta} \) is some ricetic combination operator. Let

\[ F_1, F_2, \ldots, F_k \]

be acceptable target functions for

\[ \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \]

with respect to \( \oplus \) and \( C(\hat{\beta}(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \), and let the set of functions \( C_1(\mathcal{M}_1) \) be

\[ \{\{x_1 : H_1(x_1) \neq F_1(x_1) : H_1 \in \mathcal{M}_1\} : \}

likewise let \( C_2(\mathcal{M}_2) \) be

\[ \{\{x_2 : H_2(x_2) \neq F_2(x_2) : H_2 \in \mathcal{M}_2\} : \}

and so on. Then \( C_1, C_2, \ldots, C_k \) are acceptable credit assignment functions for \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \) with respect to \( \oplus \) and \( C(\hat{\beta}(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \).

The nature of acceptable target functions can be illustrated best by an example: suppose we know in advance that there is some \( F_1, F_2, \ldots, F_k \) such that \( C(\hat{\beta}(F_1, F_2, \ldots, F_k)) = \emptyset \) (that is, \( \hat{\beta}(F_1, F_2, \ldots, F_k) \) is correct). Then the conject function \( \hat{\beta}(H_1, H_2, \ldots, H_k) \) can err only on those inputs \( (x_1, x_2, \ldots, x_k) \) where \( H_1(x_1) \neq F_1(x_1) \) or \( H_2(x_2) \neq F_2(x_2) \) or \( H_3(x_3) \neq F_3(x_3) \) etc. Thus

\[ C(\hat{\beta}(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \subseteq \{\{(x_1, x_2, \ldots, x_k) : (F_1(x_1) \neq H_1(x_1)) \lor (F_2(x_2) \neq H_2(x_2)) \lor \ldots : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2, \ldots\} \].

By definition, \( F_1, F_2, \ldots, F_k \) are acceptable target functions for \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \) with respect to \( \oplus \) and \( C(\hat{\beta}(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \).

But since \( F_1 \) is the correct choice for \( H_1 \), \( F_2 \) is the correct choice for \( H_2 \), and so on, we can simply say that the individual target functions for \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \) are also acceptable target functions for \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k \) with respect to \( \oplus \) and \( C(\hat{\beta}(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)) \). (Note that we need not know what \( F_1, F_2, \ldots, F_k \) are; we must only assume that they exist. We will make this assumption implicitly henceforth, but it is fairly unrestricted for the combination operators we will discuss.)

The following lemma is preliminary to our extension of proposition 2.11 to non \( \{0, 1\} \)-valued functions:
Lemma 3A.14: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two sets of events defined on a set $\mathcal{S}$. If $\mathcal{A}_1 \subseteq \mathcal{A}_2$ then $\nu(\mathcal{A}_1) \leq \nu(\mathcal{A}_2)$.

Proof: The lemma follows because any set of elements of $\mathcal{S}$ that is shattered by $\mathcal{A}_1$ must also be shattered by $\mathcal{A}_2$. For any $\ell$, if $\nu(\mathcal{A}_1) \geq \ell$ then there is some set of $\ell$ elements of $\mathcal{S}$ that is shattered by $\mathcal{A}_1$. But this set of $\ell$ elements must also be shattered by $\mathcal{A}_1$, so $\nu(\mathcal{A}_2)$ must be at least $\ell$ by the definition of the Vapnik-Chervonenkis Dimension. $\square$

The next claim follows without much difficulty:

Proposition 3A.15: Consider a set of $k$ hypothesis classes

$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$$

and their conjunction

$$\Theta(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k)$$

under some conjunction operator $\Theta$. Let $C_1, C_2, \ldots, C_k$ be acceptable credit-assignment functions for $H_1, H_2, \ldots, H_k$ with respect to $\Theta$ and $C(\Theta(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k))$. Then

$$\nu(C(\Theta(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k))) \leq \nu_k \left( \sum_{i=1}^{k} \nu(C_i(\mathcal{M}_i)) \right). \quad (3A.12)$$

Proof: By the definition of an acceptable credit-assignment function, the set $C(\Theta(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k))$ is a subset of

$$\Theta(C_1(\mathcal{M}_1), C_2(\mathcal{M}_2), \ldots, C_k(\mathcal{M}_k)), \quad (3A.13)$$

for some ricetic operator $\Theta$. By proposition 2.11 the VC Dimension of (3A.13) is bounded above by

$$\nu_k \left( \sum_{i=1}^{k} \nu(C_i(\mathcal{M}_i)) \right),$$

and by lemma 3A.14 the VC Dimension of $C(\Theta(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k))$ is bounded above by the VC Dimension of (3A.13). $\square$
Appendix B to Chapter 3.
Proofs of results in chapter 3

3B.1. Proof of proposition 3.2.

It is convenient to express a composition in a way that makes it explicit that one of the functions uses the value of the other function as its argument. To do this we will use the following definitions:

Definition. Let $H$ and $f$ be functions, with $\text{Dom}(f) = \text{Ran}(H) \times \text{Dom}(H)$. For any point $x \in \text{Dom}(H)$ we define $u(f, H, x)$ to be $f(x, H(x))$. If $\mathcal{H}$ is a hypothesis class we define $u(f, \mathcal{H}, x)$ to be $f(x, c)$, where $c = \max \{H(x) : H \in \mathcal{H}\}$.

Theorem 3B.16: For any $\{0, 1\}$-valued function class $\mathcal{H}$, and any two integers $0 < m < n$,

$$\Pi_\mathcal{H}(n) \geq \Pi_\mathcal{H}(m)$$

Proof:

1. $\exists(S_1 \in \text{Dom}(\mathcal{H})^m) : \Pi_\mathcal{H}(S_1) = \Pi_\mathcal{H}(m)$ \ [Such an $S_1$ exists by the definition of $\Pi_\mathcal{H}(m)$.]  

2. Let $S_2 \in \text{Dom}(\mathcal{H})^n$ be such that $\forall(x : x \in S_1) : x \in S_2$ \ [Such an $S_2$ is just an arbitrary list of length $n > m$ that contains all the elements in $S_1$.]  

3. $\Pi_\mathcal{H}(S_1) \leq \Pi_\mathcal{H}(S_2)$ \ [By 2]  

4. $\Pi_\mathcal{H}(S_1) \leq \Pi_\mathcal{H}(n)$ \ [By the definition of $\Pi$, since $|S_2| = n$.]  

5. $\Pi_\mathcal{H}(S_1) = \Pi_\mathcal{H}(m)$ \ [By the definition of $S_1$ in 1].
112

\[ \Pi_{\mathcal{M}}(S_1) \leq \Pi_{\mathcal{M}}(S_2) \]  
\hspace{1cm} \text{[Substituting the right side of [5] into [3].]}

\[ \Pi_{\mathcal{M}}(m) \leq \Pi_{\mathcal{M}}(n) \]  
\hspace{1cm} \text{[By [4], [6], and the transitivity of \( \leq \).]}

Q.E.D.

\[ \square \]

Lemma 3B.17: Let \( \mathcal{M}_0 \) and \( \mathcal{M}_1 \) be two function classes, and let \( X \) be a sample space. Let \( f : X \times \text{Ran}(\mathcal{M}_1) \to \text{Dom}(\mathcal{M}_0) \). Assume \( \Pi_{\mathcal{M}_1}(1) = 1 \). Let \( S \in X^t \), where \( t \in \mathbb{N} \). Then

\[ |\{ \{ x \in S : u(f, \mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{M}_0 \}| \leq |\{ \{ u(f, \mathcal{M}_1, x) : x \in S \} \cap H_0 : H_0 \in \mathcal{M}_0 \}| \]

Proof:

1. \( |\{ \{ x \in S : u(f, \mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{M}_0 \}| = 0 \) \rightarrow

\[ |\{ \{ x \in S : u(f, \mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{M}_0 \}| \leq |\{ \{ u(f, \mathcal{M}_1, x) : x \in S \} \cap H_0 : H_0 \in \mathcal{M}_0 \}| \]

\hspace{1cm} \text{(Since the size of a set cannot be negative, the inequality on the right side of the implication holds trivially.)}

2. \( |\{ \{ x \in S : u(f, \mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{M}_0 \}| = 1 \) \rightarrow

\[ |\{ \{ x \in S : u(f, \mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{M}_0 \}| \leq |\{ \{ u(f, \mathcal{M}_1, x) : x \in S \} \cap H_0 : H_0 \in \mathcal{M}_0 \}| \]
Proof of 2:

[2.1] \[|\{\{z \in S : u(f, \mathcal{M}, z) \in H_0 \} : H_0 \in \mathcal{M}\}| = 1\]  
[ Assumption. ]

[2.2] \[\exists (H_0 \in \mathcal{M})\]  
[ If no such \(H_0\) existed, then the set in [2.1] would be empty, violating the assumption in [2.1]. ]

[2.3] Let \(H_0\) be a member of \(\mathcal{M}\)  
[ Such an \(H_0\) exists by [2.2]. ]

[2.4] \[\{u(f, \mathcal{M}, x) : x \in S\} \cap H_0 \in \{\{u(f, \mathcal{M}, x) : x \in S\} \cap H_0 : H_0 \in \mathcal{M}\}\]  
[ By the definition of \(H_0\) in [2.3]. ]

[2.5] \[|\{\{u(f, \mathcal{M}, x) : x \in S\} \cap H_0 : H_0 \in \mathcal{M}\}| \geq 1\]  
[ By [2.4], the set has at least one member. ]

Q.E.D. (2)  
[ By [2.1] and [2.5]. ]

[3] \[\{\{z \in S : u(f, \mathcal{M}, x) \in H_0 \} : H_0 \in \mathcal{M}\}\]  
\[\geq 1\]  
\[-\]
[\[\{\{z \in S : u(f, \mathcal{M}, x) \in H_0 \} : H_0 \in \mathcal{M}\}\]  
\[\leq\]
[\[\{\{u(f, \mathcal{M}, x) : x \in S\} \cap H_0 : H_0 \in \mathcal{M}\}\]  

Proof of 3:

[3.1]
\[ \| \{ x \in S : u(\mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{K}_0 \| > 1 \]

[ Assumption. ]

[3.2]
define
\[ M : \{ \{ x \in S : u(\mathcal{M}_1, x) \in H_0 \} : H_0 \in \mathcal{K}_0 \} \rightarrow \{ u(f, \mathcal{M}_1, x) : x \in S \cap H_0 : H_0 \in \mathcal{K}_0 \} \]

so that
\[ M(\{ x \in S : u(\mathcal{M}_1, x) \in H_0 \}) = \{ u(f, \mathcal{M}_1, x) : x \in S \cap H_0 : H_0 \}

[ Definition. ]

[3.3]
M is 1:1

Proof of 3.3:

[3.3.1]
Let \( H_a, H_b \in \mathcal{K}_0 \) be such that
\[ \{ x \in S : u(\mathcal{M}_1, x) \in H_a \} \neq \emptyset \]
\[ \{ x \in S : u(\mathcal{M}_1, x) \in H_b \} \neq \emptyset \]

[ \( H_a \) and \( H_b \) must exist by [3.1]. ]

[3.3.2]
\[ \exists x \in S : \]
\[ u(\mathcal{M}_1, x) \in H_a \quad \neq \]
\[ u(\mathcal{M}_1, x) \in H_b \]

[ By (3.3.1). ]

[3.3.3]
Let \( x_0 \in S \) be such that
\[ u(\mathcal{M}_1, x_0) \in H_a \neq \]
\[ u(\mathcal{M}_1, x_0) \in H_b \]

[ Such an \( x_0 \) exists by [3.3.2]. ]
\[\{u(f, \mathcal{M}, z) : z \in S \} \cap H_a \neq \{u(f, \mathcal{M}, z) : z \in S \} \cap H_b\]

By [3.3.3], \(u(f, \mathcal{M}, z_0)\) is in one set and not the other.

Q.E.D. (3.3)

\[|\text{Dom}(M)| \leq |\text{Ran}(M)|\] [By (3.3)]

\[|\{x \in S : u(f, \mathcal{M}, x) \in H_0 : H_0 \in \mathcal{M}_0\}| \leq\]
\[|\{u(f, \mathcal{M}, z) : z \in S \} \cap H_0 : H_0 \in \mathcal{M}_0\}|\]

By the definition of \(M\) in [3.2] and [3.4].

Q.E.D. (3)

Q.E.D. By (1), (2), and (3), the claim holds for all possible sizes of the set

\[\{\{x \in S : u(f, \mathcal{M}, x) \in H_0 : H_0 \in \mathcal{M}_0\}\}.

\[\square\]

**Lemma 3D.18:** Let \(\mathcal{M}_0\) and \(\mathcal{M}_1\) be two function classes, and let \(X\) be a sample space. Let \(f : X \times \text{Ran}(\mathcal{M}_1) \to \text{Dom}(\mathcal{M}_0)\). Let \(S \in X^l, l \in \mathbb{N}\). Then

\[\{\mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{M}_0, H_1 \in \mathcal{M}_1\} =\]
\[\{\mathcal{G}_f(H_0, H_1) \cap S : \mathcal{G}_f(H_0, H_1) \in \mathcal{F}_f(\mathcal{M}_0, \mathcal{M}_1)\}\]

**Proof:**
Proof of 1:

1.1 Let $w$ be a member of
\[ \{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{X}_0, H_1 \in \mathcal{X}_1 \} \]
[ Definition. ]

1.2 \[ \exists (H_0 \in \mathcal{X}_0, H_1 \in \mathcal{X}_1) : \]
\[ w = \mathcal{G}_f(H_0, H_1) \cap S \]
[ By the definition of $w$ in 1.1. ]

1.3 Let $H_0 \in \mathcal{X}_0, H_1 \in \mathcal{X}_1$ be such that
\[ w = \mathcal{G}_f(H_0, H_1) \cap S \]
[ Such $H_0, H_1$ exist by 1.2. ]

1.4 \[ \mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{X}_0, \mathcal{X}_1) \]
[ By the definition of $\mathcal{G}_f(\mathcal{X}_0, \mathcal{X}_1)$ and the fact (from 1.3) that $H_0 \in \mathcal{X}_0$ and $H_1 \in \mathcal{X}_1$. ]

1.5 \[ w \in \{ \mathcal{G}_f(H_0, H_1) \cap S : \]
\[ \mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{X}_0, \mathcal{X}_1) \}
[ By 1.3 and 1.4. ]

Q.E.D. (1) [ By 1.1 and 1.5. ]
\[ \forall w \in \mathcal{G}_f(H_0, H_1) \cap S : \\
\mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_1) : \\
w \in \{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{M}_0, H_1 \in \mathcal{M}_1 \} \]

**Proof of 2:**

1. \[2.1\] Let \( w \) be a member of
\[ \{ \mathcal{G}_f(H_0, H_1) \cap S : \\
\mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_1) \} \]
\[ \text{[Definition.]} \]

2. \[2.2\] \[ \exists \mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_1) : \\
w = \mathcal{G}_f(H_0, H_1) \cap S \\
\{ \text{By the definition of } w \text{ in } [2.1]. \} \]

3. \[2.3\] Let \( \mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_1) \)
be such that
\[ w = \mathcal{G}_f(H_0, H_1) \cap S \\
\{ \text{Such a } \mathcal{G}_f(H_0, H_1) \text{ exists by } [2.2]. \} \]

4. \[2.4\] \[ \mathcal{G}_f(H_0, H_1) \text{ is such that} \\
H_0 \in \mathcal{M}_0, H_1 \in \mathcal{M}_1 \\
\{ \text{This is a necessary condition for the membership of} \\
\mathcal{G}_f(H_0, H_1) \text{ in } \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_1); \text{ this membership was stipulated in } [2.3]. \} \]

5. \[2.5\] \[ w \in \{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{M}_0, H_1 \in \mathcal{M}_1 \} \\
\{ \text{By } [2.4]. \} \]
Q.E.D. (2) | By [2.1] and [2.5].

Q.E.D. | The result follows from [1] and [2].

Lemma 3B.19: Let $\mathcal{K}_0$ and $\mathcal{K}_1$ be two function classes, and let $X$ be a sample space. Let $f : X \times \text{Ran}(\mathcal{K}_1) \rightarrow \text{Dom}(\mathcal{K}_0)$. Assume $\Pi_{\mathcal{K}_0}(t) = 1$. Let $S \in X^t, t \in \Pi$. Then

$$
\forall (x \in S) : \forall (H_1 \in \mathcal{K}_1) : u(f, H_1, x) = u(f, H_1, x).
$$

Proof:

[1] \[ \forall (x \in S) : \forall (H, H' \in \mathcal{K}_1) : \]

$$
\forall (x \in S) : \forall (H_1, H'_1 \in \mathcal{K}_1) : u(f, H_1, x) \neq u(f, H'_1, x)
$$

[ Assumption. ]

[1.1] \[ \exists (x \in S) : \exists (H_1, H'_1 \in \mathcal{K}_1) : \]

$$
\exists (x \in S) : \exists (H_1, H'_1 \in \mathcal{K}_1) : u(f, H_1, x) \neq u(f, H'_1, x)
$$

[ Such an $x$ exists by [1.1]. ]

[1.2] \[ \text{Let } x \in S \text{ be such that} \]

$$
\exists (H_1, H'_1 \in \mathcal{K}_1) : u(f, H_1, x) \neq u(f, H'_1, x)
$$

[ Such an $x$ exists by [1.1]. ]

[1.3] \[ \text{Let } H, H' \in \mathcal{K}_1 \text{ be such that} \]

$$
\exists (H_1, H'_1 \in \mathcal{K}_1) : u(f, H_1, x) \neq u(f, H'_1, x)
$$

[ Such $H, H'$ exist by [1.2]. ]
Proof of 1.4: [We assume the contrary and derive a contradiction.]

[1.4.1]
\[ H(x) = H'(x) \quad \text{[Assumption.]} \]

[1.4.2]
\[ f(x, H(x)) = f(x, H'(x)) \quad \text{[The right side of the equation is the same as the left side but with } H' \text{ substituted for } H. \text{ The equation therefore follows from [1.4.1].]} \]

[1.4.3]
\[ u(f, H, x) = u(f, H', x) \quad \text{[By [1.4.2] and the definition of } u. \text{]} \]

Q.E.D. (1.4) \{ [1.4.3] contradicts the definition of } H \text{ and } H' \text{ in [1.3]. \}

[1.5]
\[ |\{ H \cap \{x\} : H \in \mathcal{F} \}| \geq 2 \quad \text{[According to [1.4], } H \cap \{x\} \neq H' \cap \{x\}. \]

[1.6]
\[ \Pi_{\mathcal{F}}(1) \geq 2 \quad \text{[By [1.5] and the definition of } \Pi. \text{]} \]

[1.7]
\[ \Pi_{\mathcal{F}}(\ell) \geq 2 \quad \text{[By [1.6] and theorem 3B.16.]} \]

Q.E.D. (1) \{ [1.7] contradicts the assumption that } \Pi_{\mathcal{F}}(\ell) = 1. \}

Q.E.D. \quad \text{[The result follows from [1] and the definition of } u(f, \mathcal{F}, x). \}

\[ \square \]

Theorem 3B.20: Let \( \mathcal{K}_0 \) and \( \mathcal{K}_1 \) be two function classes, and let \( X \) be a sample space. Let \( f : X \times \text{Ran}(\mathcal{K}_1) \rightarrow \text{Dom}(\mathcal{K}_0) \). Assume \( \Pi_{\mathcal{K}_0}(\ell) = 1 \). Let \( S \in X' \). \( \ell \in \mathbb{N} \). Then

\[ \Pi_{\mathcal{K}_0 \cup \mathcal{K}_1}(S) \leq \Pi_{\mathcal{K}_0}(\{u(f, \mathcal{K}_1, x) : x \in S\}). \]
Proof:

1. \( \forall (z \in S) : \forall (H_1 \in \mathcal{K}) : \)
   
   \[ u(f, H_1, z) = u(f, \mathcal{K}, z) \]
   
   [By lemma 3B.19.]

2. \( \forall (H_0 \in \mathcal{K}) : \forall (H_1 \in \mathcal{K}) : \forall (z \in S) : \)
   
   \[ H_0(u(f, \mathcal{K}, z)) = \mathcal{C}_f(H_0, H_1)(z) \]

Proof of 2:

2.1. \( \forall (H_1 \in \mathcal{K}) : \forall (z \in S) : \)
   
   \[ u(f, H_1, z) = u(f, \mathcal{K}, z) \]
   
   [By [1].]

2.2. \( \forall (H_0 \in \mathcal{K}) : \forall (H_1 \in \mathcal{K}) : \forall (z \in S) : \)
   
   \[ H_0(u(f, \mathcal{K}, z)) = H_0(u(f, H_1, z)) \]
   
   [By [2.1].]

2.3. \( \forall (H_0 \in \mathcal{K}) : \forall (H_1 \in \mathcal{K}) : \forall (z \in S) : \)
   
   \[ H_0(u(f, \mathcal{K}, z)) = \mathcal{C}_f(H_0, H_1)(z) \]
   
   [By [2.2], the definition of \( f \), and the definition of \( \mathcal{C} \).]

Q.E.D. (2)

3. \( \forall (H_0 \in \mathcal{K}) : \forall (H_1 \in \mathcal{K}) : \forall (z \in S) : \)
   
   \[ (u(f, \mathcal{K}, z) \in H_0) \rightarrow (x \in \mathcal{C}_f(H_0, H_1)) \]
   
   [This is just [2] rewritten in set notation.]
\[4\] \(\forall (H_0 \in \mathcal{H}_0) : \forall (H_1 \in \mathcal{H}_1) :\]
\[\{ x \in S : u(f, \mathcal{M}, x) \in H_0 \} = \{ x \in S : x \in \mathcal{G}_f(H_0, H_1) \}\]

[This follows from [3].]

\[5\] \(\forall (H_0 \in \mathcal{H}_0) : \forall (H_1 \in \mathcal{H}_1) :\]
\[\{ x \in S : u(f, \mathcal{M}, x) \in H_0 \} = \mathcal{G}_f(H_0, H_1) \cap S\]

[By [4] and the definition of set intersection.]

\[6\] \(\forall (H_1 \in \mathcal{H}_1) :\]
\[\{ \{ x \in S : u(f, \mathcal{M}, x) \in H_0 \} : H_0 \in \mathcal{H}_0 \} = \{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{H}_0 \}\]

[This follows from [5].]

\[7\] \(\{ \{ x \in S : u(f, \mathcal{M}, x) \in H_0 \} : H_0 \in \mathcal{H}_0 \} = \{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{H}_0, H_1 \in \mathcal{H}_1 \}\)

[By [6].]

\[8\] \(|\{ x \in S : u(f, \mathcal{M}, x) \in H_0 \} : H_0 \in \mathcal{H}_0 \}| \leq |\{ (u(f, \mathcal{M}, x) : x \in S) \cap H_0 : H_0 \in \mathcal{H}_0 \}|\]

[By lemma 3B.17.]

\[9\] \(|\{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{H}_0, H_1 \in \mathcal{H}_1 \}| \leq |\{ (u(f, \mathcal{M}, x) : x \in S) \cap H_0 : H_0 \in \mathcal{H}_0 \}|\]

[By [8] and [7].]

\[10\] \(\{ \mathcal{G}_f(H_0, H_1) \cap S : H_0 \in \mathcal{H}_0, H_1 \in \mathcal{H}_1 \} = \{ \mathcal{G}_f(H_0, H_1) \cap S : \mathcal{G}_f(H_0, H_1) \in \mathcal{G}_f(\mathcal{H}_0, \mathcal{H}_1) \}\)

[By lemma 3B.18.]
Corollary 3B.21: Let $\mathcal{M}_0$ and $\mathcal{M}_1$ be two function classes, and let $X$ be a sample space. Let $f : X \times \text{Ran}(\mathcal{M}_1) \rightarrow \text{Dom}(\mathcal{M}_0)$. Assume $\Pi_{\mathcal{M}_1}(\ell) = 1$. Let $S \in X^\ell$, $\ell \in \mathbb{N}$. Then

$$
\Pi_{\sigma_{f,\mathcal{M}_0,\mathcal{M}_1}}(S) \leq \Pi_{\mathcal{M}_0}(\{u(f, \mathcal{M}_1, x) : x \in S\})
$$

[By theorem 3B.20.]

Proof:

[1]

$$
\Pi_{\sigma_{f,\mathcal{M}_0,\mathcal{M}_1}}(S) \leq \Pi_{\mathcal{M}_0}(\{u(f, \mathcal{M}_1, x) : x \in S\})
$$

[By theorem 3B.20.]

[2]

$$
\{\{u(f, \mathcal{M}_1, x) : x \in S\} \leq |S|
$$
Proof of 2:

[2.1] define

\[ M : \{ u(f, \mathcal{I}, z) : z \in S \} \rightarrow \{ z \in S \} \]

so that

\[ M(u(f, \mathcal{I}, z)) = z \]

[Definition.]

[2.2] \( M \) is 1:1

Proof of 2.2:

[2.2.1] \( \{ u(f, \mathcal{I}, z) : z \in S \} = \emptyset \rightarrow M \) is 1:1

[The antecedent implies that there are no elements in the
domain of \( M \) (according to the definition of \( M \) in [2.1]).
so the consequent holds trivially.]

[2.2.2] \(|\{ u(f, \mathcal{I}, z) : z \in S \}| = 1 \rightarrow M \) is 1:1

[The implication holds since each element in the domain of
\( M \) has a unique mapping under \( M \).]

[2.2.3] \(|\{ u(f, \mathcal{I}, z) : z \in S \}| > 1 \rightarrow M \) is 1:1

Proof of 2.2.3:

[2.2.3.1] \(|\{ u(f, \mathcal{I}, z) : \mathcal{I} \in S_{0}, \mathcal{I} \in S_{1} \}| > 1 \]

[Assumption.]

[2.2.3.2] Let \( x_{0}, x_{1} \in S \) be such that

\[ u(f, \mathcal{I}, x_{0}) \neq u(f, \mathcal{I}, x_{1}) \]

[Such \( x_{0}, x_{1} \) exist by [2.2.3.1].]

[2.2.3.3] \( x_{0} \neq x_{1} \)
Proof of 2.2.3.3: [We assume the contrary and derive a contradiction.]

[2.2.3.3.1] \[ x_0 = x_1 \]

[ Assumption. ]

[2.2.3.3.2] \[ c = \max\{H_i(x_0) : H_i \in \mathcal{H}\} = \max\{H_i(x_1) : H_i \in \mathcal{H}\} \]

[ Such a \( c \) exists by [2.2.3.3.0]. ]

[2.2.3.3.3] \[ f(x, c) = u(f, \mathcal{H}, x_0) \]

[ By [2.2.3.3.1], [2.2.3.3.2], and the definition of \( u \). ]

[2.2.3.3.4] \[ f(x, c) = u(f, \mathcal{H}, x_1) \]

[ By [2.2.3.3.1], [2.2.3.3.2], and the definition of \( u \). ]

[2.2.3.3.5] \[ u(f, \mathcal{H}, x_0) = u(f, \mathcal{H}, x_1) \]

[ By [2.2.3.3.3], [2.2.3.3.4], and the transitivity of \( '=' \). ]

Q.E.D. (2.2.3.4) [ [2.2.3.3.5] contradicts the definition of \( x_0 \) and \( x_1 \) in [2.2.3.2]. ]

Q.E.D. (2.2.3) [ The claim follows from [2.2.3.1] and [2.2.3.3]. ]

Q.E.D. (2.2) All possible cases are covered by [2.2.3.1], [2.2.2], and [2.2.3]; in each case the claim was shown to follow.

[2.3] \[ |\text{Dom}(M)| \leq |\text{Ran}(M)| \]

[ By [2.2] and the properties of 1:1 functions. ]

[2.4] \[ |\{u(f, \mathcal{H}, x) : x \in S\}| \leq |\{x \in S\}| \]

[ By [2.3] and the definition of \( M \) in [2.1]. ]

[2.5] \[ |\{x \in S\}| \leq |S| \]

[ By the definition of lists. ]
Theorem 3D.22: Let \( \mathcal{M}_0 \) and \( \mathcal{M}_1 \) be two function classes, and let \( X \) be a sample space. Let \( f : X \times \text{Ran}(\mathcal{M}_1) \to \text{Dom}(\mathcal{M}_0) \). Assume \( \Pi_{\mathcal{M}_1}(\ell) = 1 \). Let \( \ell \in \Pi_1 \). Assume \( \Pi_{\mathcal{M}_1}(\ell) = 1 \). Then

\[
\max \{ \Pi_{\sigma_f(\mathcal{M}_0,\mathcal{M}_1)}(S) : S \in X' \} \\
= \Pi_{\sigma_f(\mathcal{M}_0,\mathcal{M}_1)}(\ell) \\
\leq \Pi_{\mathcal{M}_1}(\ell).
\]
Proof:

1. \[ \exists (S \in X^f) : \quad \Pi_{\mathcal{G}_f(X_0, \mathcal{M}_0)}(S) > \Pi_{\mathcal{M}_0}(t) \quad \text{[Assumption.]} \]

2. Let \( S \in X^f \) be such that \( \Pi_{\mathcal{G}_f(X_0, \mathcal{M}_0)}(S) > \Pi_{\mathcal{M}_0}(t) \)
   \[ \text{[Such an } S \text{ exists by [1].]} \]

3. \( \Pi_{\mathcal{G}_f(X_0, \mathcal{M}_0)}(S) \leq \Pi_{\mathcal{M}_0}(t) \)
   \[ \text{[By theorem 3B.20.]} \]

Q.E.D. \([3]\) contradicts \([1]\). \(\square\)

Lemma 3B.23: Let \( X \) be a sample space, and let \( \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k \) be \( k + 1 \) function classes (\( k > 0 \)), defined so that \( \text{Dom}(\mathcal{M}_1) = \text{Dom}(\mathcal{M}_2) = \ldots = \text{Dom}(\mathcal{M}_k) = X \), and let \( \mathcal{F} \equiv \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_k \). Let \( f : X \times \text{Ran}(\mathcal{M}_1) \to \text{Dom}(\mathcal{M}_0) \). Let \( S \in X^f \), \( \ell \in \mathbb{N} \). Assume that \( \Pi_{\mathcal{M}_1}(S) = \Pi_{\mathcal{M}_2}(S) = \ldots = \Pi_{\mathcal{M}_k}(S) = 1 \). Then

\[
[\mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{F})] \\
= \bigcup \{ \mathcal{G}_f(H_0, H_i) \cap S : \mathcal{G}_f(H_0, H_i) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_i) \}.
\]

Proof:

1. \[ \mathcal{G}_f(\mathcal{M}_0, \mathcal{F}) = \bigcup_i \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_i) \]
Proof of 1:

[1.1] \( \mathcal{G}_f(\mathcal{X}_0, \mathcal{F}) = \{ \mathcal{G}_f(H_0, F) : H_0 \in \mathcal{X}_0, F \in \mathcal{F} \} \)

[ By the definition of \( \mathcal{G}_f(\mathcal{X}_0, \mathcal{F}) \). ]

[1.2] \( \mathcal{G}_f(\mathcal{X}_0, \mathcal{F}) = \{ \mathcal{G}_f(H_0, F) : H_0 \in \mathcal{X}_0, F \in \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_n \} \)

[ By (1.1) and the definition of \( \mathcal{F} \) in the statement of the theorem. ]

[1.3] \( \mathcal{G}_f(\mathcal{X}_0, \mathcal{F}) = \bigcup_i \{ \mathcal{G}_f(H_0, F) : H_0 \in \mathcal{X}_0, F \in \mathcal{X}_i \} \)

[ By (1.2) and the properties of sets. ]

[1.4] \( \mathcal{G}_f(\mathcal{X}_0, \mathcal{F}) = \bigcup_i \mathcal{G}_f(\mathcal{X}_0, \mathcal{X}_i) \)

[ Rewriting the right side of (1.4) using the definition of \( \mathcal{G}_f(\mathcal{X}_0, \mathcal{X}_i) \). ]

Q.E.D. (1)

[2] \( \{ \mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{X}_0, \mathcal{F}) \} \)

= \( \bigcup_i \{ \mathcal{G}_f(H_0, H_i) \cap S : \mathcal{G}_f(H_0, H_i) \in \mathcal{G}_f(\mathcal{X}_0, \mathcal{X}_i) \} \)
Proof of 2:

[2.1]

\[ \forall (w \in \mathcal{F}_f(H_0, F) \cap S) : \]

\[ \mathcal{F}_f(H_0, F) \in \mathcal{F}_f(\mathcal{K}_0, \mathcal{P}) : \]

\[ w \in \bigcup_i \{ \mathcal{F}_f(H_0, H_i) \cap S : \mathcal{F}_f(H_0, H_i) \in \mathcal{F}_f(\mathcal{K}_0, \mathcal{P}) \} \]

Proof of 2.1:

[2.1.1]

Let \( w \in \{ \mathcal{F}_f(H_0, F) \cap S : \mathcal{F}_f(H_0, F) \in \mathcal{F}_f(\mathcal{K}_0, \mathcal{P}) \} \)

[ Definition ]

[2.1.2]

\[ \exists (H_0 \in \mathcal{K}_0, F \in \mathcal{P}) : \]

\[ w = \mathcal{F}_f(H_0, F) \cap S \]

[ By [2.1.1]. ]

[2.1.3]

Let \( H_0 \in \mathcal{K}_0, F \in \mathcal{P} \) be such that

\[ w = \mathcal{F}_f(H_0, F) \cap S \]

[ Such \( H_0, F \) exist by [2.1.3]. ]

[2.1.4]

\[ \mathcal{F}_f(H_0, F) \in \mathcal{F}_f(\mathcal{K}_0, \mathcal{P}) \]

[ By the definition of \( \mathcal{F}_f(\mathcal{K}_0, \mathcal{P}) \), because \( H_0 \in \mathcal{K}_0 \) and \( F \in \mathcal{P} \) by [2.1.3]. ]

[2.1.5]

\[ \exists (\mathcal{K}_i, 0 < i \leq k) : \]

\[ \mathcal{F}_f(H_0, F) \in \mathcal{F}_f(\mathcal{K}_0, \mathcal{K}_i) \]

[ By [2.1.4] and [1]. ]

[2.1.6]

Let \( \mathcal{K}_i, 0 < i \leq k \) be such that

\[ \mathcal{F}_f(H_0, F) \in \mathcal{F}_f(\mathcal{K}_0, \mathcal{K}_i) \]

[ Such a \( \mathcal{K}_i \) exists by [2.1.5]. ]
[2.1.7] \( w \in \{ \mathcal{G}_f(H_0, H_i) \cap S : \mathcal{G}_f(H_0, H_i) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{K}_i) \} \)

[ By [2.1.3] and [2.1.6]. ]

[2.1.8] \( \forall (w \in \mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{F})) : \)

\( w \in \bigcup_i \{ \mathcal{G}_f(H_0, H_i) \cap S : \mathcal{G}_f(H_0, H_i) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{K}_i) \} \)

[ By [2.1.7] and the assumption (in [1]) that \( \mathcal{K}_i \subseteq \mathcal{F} \). ]

Q.E.D. (2.1)

[2.2] \( \forall \left( w \in \bigcup_i \{ \mathcal{G}_f(H_0, H_i) \cap S : \mathcal{G}_f(H_0, H_i) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{K}_i) \} \right) : \)

\( w \in \{ \mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{F}) \} \)

Proof of 2.2:

[2.2.1] Let \( w \in \bigcup_i \{ \mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{K}_i) \} \)

[2.2.2] \( \exists (\mathcal{K}_i) : \)

\( w \in \{ \mathcal{G}_f(H_0, F) : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{K}_i) \} \)

[ By [2.2.1]. ]

[2.2.3] Let \( \mathcal{K}_i \) be such that \( w \in \{ \mathcal{G}_f(H_0, F) : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{K}_0, \mathcal{K}_i) \} \)

[ Such a \( \mathcal{K}_i \) exists by [2.2.2]. ]
\[2.2.4]\]
\[\exists (\mathcal{F}_f(H_0, F) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_i)) : \]
\[w = \mathcal{G}_f(H_0, F) \cap S\]
[By 2.2.3].

\[2.2.5]\]
Let \( \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_i) \)
be such that
\[w = \mathcal{G}_f(H_0, F) \cap S\]
[Such a \( \mathcal{G}_f(H_0, F) \) exists by 2.2.4].

\[2.2.6]\]
\[\mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{F})\]
[By 2.2.5] and [1].

\[2.2.7]\]
\[w \in \{\mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{F})\}\]
[By 2.2.5] and [2.2.6].

Q.E.D. (2.2)

Q.E.D. (2) [By 2.1] and (2.2).

Q.E.D. [2] was to be proven.

□

Lemma 3B.24: Let \( X \) be a sample space, and let \( \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_i \) be \( k + 1 \) function classes (\( k > 0 \)),
defined so that \( \text{Dom}(\mathcal{M}_1) = \text{Dom}(\mathcal{M}_2) = \cdots = \text{Dom}(\mathcal{M}_i) = X \), and let \( \mathcal{F} \in \mathcal{M}_0 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_i \). Let
\[f : X \times \text{Ran}(\mathcal{M}_1) \to \text{Dom}(\mathcal{M}_0)\]. Let \( S \in X^t, t \in \mathbb{N} \). Assume that \( \Pi_{\mathcal{M}_1}(S) = \Pi_{\mathcal{M}_2}(S) = \cdots = \Pi_{\mathcal{M}_i}(S) = 1 \).
Then

\[\left| \{\mathcal{G}_f(H_0, F) \cap S : \mathcal{G}_f(H_0, F) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{F})\} \right|\]
\[= \sum_i \left| \{\mathcal{G}_f(H_0, H_i) \cap S : \mathcal{G}_f(H_0, H_i) \in \mathcal{G}_f(\mathcal{M}_0, \mathcal{M}_i)\} \right|.\]
Proof:

1. \[
\{ \mathcal{G}_j(H_0, F) \cap S : \mathcal{G}_j(H_0, F) \in \mathcal{G}_j(\mathcal{A}, \mathcal{F}) \} \\
= \bigcup \{ \mathcal{G}_j(H_0, H_i) \cap S : \mathcal{G}_j(H_0, H_i) \in \mathcal{G}_j(\mathcal{A}, \mathcal{A}_i) \} \\
\]
   [By lemma 3B.23.]

2. \[
\left| \{ \mathcal{G}_j(H_0, F) \cap S : \mathcal{G}_j(H_0, F) \in \mathcal{G}_j(\mathcal{A}, \mathcal{A}_i) \} \right| \\
= \sum_{i=1}^k \left| \{ \mathcal{G}_j(H_0, H_i) \cap S : \mathcal{G}_j(H_0, H_i) \in \mathcal{G}_j(\mathcal{A}, \mathcal{A}_i) \} \right| \\
\]
   [By 1 and the properties of sets.]

Q.E.D.

Lemma 3B.25: Let \( X \) be a sample space, and let \( \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k \) be \( k+1 \) function classes \((k > 0)\), defined so that \( \text{Dom}(\mathcal{A}_1) = \text{Dom}(\mathcal{A}_2) = \cdots = \text{Dom}(\mathcal{A}_k) = X \), and let \( \mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \cdots \cup \mathcal{A}_k \). Let \( f : X \times \text{Ran}(\mathcal{A}_1) \to \text{Dom}(\mathcal{A}_0) \). Let \( S \in X^t, t \in \mathbb{N} \). Assume that \( \Pi_{\mathcal{A}_0}(S) = \Pi_{\mathcal{A}_1}(S) = \cdots = \Pi_{\mathcal{A}_k}(S) = 1 \). Then

\[
\Pi_{\mathcal{A}_0}(S) \leq \sum_{i=1}^k \Pi_{\mathcal{A}_i}(U(f, \mathcal{A}_i, S)).
\]

Proof:

1. \[
\left| \{ \mathcal{G}_j(H_0, F) \cap S : \mathcal{G}_j(H_0, F) \in \mathcal{G}_j(\mathcal{A}_0, \mathcal{F}) \} \right| \\
= \sum_{i=1}^k \left| \{ \mathcal{G}_j(H_0, H_i) \cap S : \mathcal{G}_j(H_0, H_i) \in \mathcal{G}_j(\mathcal{A}_0, \mathcal{A}_i) \} \right| \\
\]
   [By lemma 3B.24.]
[[\mathcal{G}_f (H_0, \mathcal{F}) \cap \mathcal{S} : \mathcal{G}_f (H_0, \mathcal{F}) \in \mathcal{G}_f (\mathcal{F}_0, \mathcal{F})]]

= \sum_{i=1}^{k} |\Pi_{\mathcal{F}_i, \mathcal{F}_0, \mathcal{F}}(S)|

[ By [1] and the definition of \Pi. ]

[[\mathcal{G}_f (H_0, \mathcal{F}) \cap \mathcal{S} : \mathcal{G}_f (H_0, \mathcal{F}) \in \mathcal{G}_f (\mathcal{F}_0, \mathcal{F})]]

\leq \sum_{i=1}^{k} \Pi_{\mathcal{F}_i}(\{u(f, \mathcal{N}_i, x) : x \in S\})

[ By [2] and theorem 3B.20. ]

[[\mathcal{G}_f (H_0, \mathcal{F}) \cap \mathcal{S} : \mathcal{G}_f (H_0, \mathcal{F}) \in \mathcal{G}_f (\mathcal{F}_0, \mathcal{F})]]

\leq \sum_{i=1}^{k} \Pi_{\mathcal{F}_i}(U(f, \mathcal{N}_i, S))

[ By [3] and the definition of U. ]

\Pi_{\mathcal{F}_i, \mathcal{F}_0, \mathcal{F}}(S) \leq \sum_{i=1}^{k} \Pi_{\mathcal{F}_i}(U(f, \mathcal{N}_i, S))

[ By [4] and the definition of \Pi. ]

Q.E.D.

[Theorem 3B.26: Let \( X \) be a sample space, and let \( \mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_k \) be \( k+1 \) function classes \( (k > 0) \), defined so that \( \text{Dom} (\mathcal{F}_i) = \text{Dom} (\mathcal{F}_j) = \cdots = \text{Dom} (\mathcal{F}_k) = X \), and let \( \mathcal{F} \equiv \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_k \). Let \( f : X \times \text{Ran} (\mathcal{F}_i) \rightarrow \text{Dom} (\mathcal{F}_0) \). Let \( S \in X', t \in \mathbb{R} \). Assume that \( \Pi_{\mathcal{F}_i}(S) = \Pi_{\mathcal{F}_i}(S) = \cdots = \Pi_{\mathcal{F}_i}(S) = 1 \). Then

\[ \Pi_{\mathcal{F}_i, \mathcal{F}_0, \mathcal{F}}(S) \leq k \Pi_{\mathcal{F}_i}(S). \] ]
Proof:

1) $$\left| \left( \mathcal{G}_f(H_0,F) \cap S : \mathcal{G}_f(H_0,F) \in \mathcal{G}_f(\mathcal{A},\mathcal{P}) \right) \right|$$

   $$= \sum_{i=1}^{n} \left| \left( \mathcal{G}_f(H_0,H_i) \cap S : \mathcal{G}_f(H_0,H_i) \in \mathcal{G}_f(\mathcal{A},\mathcal{P}) \right) \right|$$

   [ By lemma 3B.24. ]

2) $$\Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S) \leq \sum_{i=1}^{n} \Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S)$$

   [ Rewriting 1 using the definition of \( \Pi \). ]

3) $$\forall (0 < i \leq k) :$$

   $$\Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S) \leq \Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S)$$

   [ By corollary 3B.21. ]

4) $$\Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S) \leq \sum_{i=1}^{n} \Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S)$$

   [ By 3 and 2. ]

5) $$\Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S) \leq k \Pi_{\mathcal{G}_f(\mathcal{A},\mathcal{P})}(S)$$

   [ By 4 and algebra. ]

Q.E.D.

The results that follow make use of the \( \Delta \) and \( \Gamma \) notation introduced in section 3.4. Although this notation is only introduced after proposition 3.2 in the body of the chapter, the results that use \( \Gamma \) and \( \Delta \) are slightly more general in that proposition 3.2 follows from them, and not vice-versa. Therefore we find it more convenient to develop these results in parallel with the proof of proposition 3.2 than to follow the format of the chapter.
Lemma 3B.27: Let $\mathcal{F}$ be a function class, and let $S \in \text{Dom}(\mathcal{F})^t$ for some $t > 0$. Then

$$|\Delta(\mathcal{F}, S)| \leq |\Gamma(\mathcal{F}, S)|.$$ 

Proof:

[1] Define $M : \{(F: F \in \mathcal{F}, F \cap S = v) : v \in \Gamma(\mathcal{F}, S)\} \rightarrow \Gamma(\mathcal{F}, S)$ so that $\forall \{(F: F \in \mathcal{F}, F \cap S = v) \in \{(F: F \in \mathcal{F}, F \cap S = v) : v \in \Gamma(\mathcal{F}, S)\} : M(\{F: F \in \mathcal{F}, F \cap S = v\}) = v$  

[ Definition ]

[2] $M$ is 1:1  

Proof of 2: [We assume the contrary and derive a contradiction.]

[2.1] $M$ is not 1:1  

[ Assumption ]

[2.2] $\exists \left( \{F: F \in \mathcal{F}, F \cap S = v\}, \{F: F \in \mathcal{F}, F \cap S = v'\} \right) \in \{(F: F \in \mathcal{F}, F \cap S = v) : v \in \Gamma(\mathcal{F}, S)\}$  

$\{(F: F \in \mathcal{F}, F \cap S = v) \neq (F: F \in \mathcal{F}, F \cap S = v')\} \wedge (v = v'\}$  

[ By [2.1], [1], and the definition of a 1:1 function. ]

[2.3] Let $\{F: F \in \mathcal{F}, F \cap S = v\}, \{F: F \in \mathcal{F}, F \cap S = v'\}$  

$\in \{(F: F \in \mathcal{F}, F \cap S = v) : v \in \Gamma(\mathcal{F}, S)\}$  

be such that  

$\{(F: F \in \mathcal{F}, F \cap S = v) \neq (F: F \in \mathcal{F}, F \cap S = v')\} \wedge (v = v'\}$  

[ Such $\{F: F \in \mathcal{F}, F \cap S = v\}, \{F: F \in \mathcal{F}, F \cap S = v'\}$ exist by [2.2]. ]

[2.4] $v = v'$  

[ By [2.3]. ]
[2.5] 
\{F : F \in \mathcal{F}, F \cap S = v\} = \{F : F \in \mathcal{F}, F \cap S = v'\} 
[By 2.4, ]

Q.E.D. (2) \[2.5\text{ contradicts } 2.3.\]

[3] \[\text{By } [2]\text{ and the properties of } 1:1 \text{ functions}\]

\[|\text{Dom}(M)| \leq |\text{Range}(M)|\]

[4] \[\{\{F : F \in \mathcal{F}, F \cap S = v\} : v \in \Gamma(\mathcal{F}, S)\} \leq |\Gamma(\mathcal{F}, S)|\]
[By 3 and the definition of M in [1].]

Q.E.D.

Lemma 3B.28: Let \(\mathcal{F}\) be a function class and let \(S \in \text{Dom}(\mathcal{F})^t\) for some \(t \in \mathbb{N}\). Let \(\Delta(\mathcal{F}, S) = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\}\). Then

\[\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_n = \mathcal{F}.\]

Proof:

[1] \[\{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\} = \{(F : F \in \mathcal{F}, F \cap S = v) : v \in \Gamma(\mathcal{F}, S)\}\]
[By the definition of \(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\) in the statement of the lemma, and the definition of \(\Delta(\mathcal{F}, S)\).]

[2] \[\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_n \neq \mathcal{F} \]
[Because each of \(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\) consists only of elements of \(\mathcal{F}\), by [1].]

[3] \[\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_n \subseteq \mathcal{F}\]
Proof of 3: [We assume the contrary and derive a contradiction.]

\[3.1\]
\[\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n \subset \mathcal{F}\]  
\hspace{1cm} \text{[Assumption.]} \\

\[3.2\]
\[\exists (F \in \mathcal{F}):
\hspace{1cm} F \notin \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n\]  
\hspace{1cm} \text{[By [3.1].]} \\

\[3.3\]
Let \( F_0 \in \mathcal{F} \) be such that
\[F \notin \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n\]  
\hspace{1cm} \text{[Such an \( F \) exists by [3.2].]} \\

\[3.4\]
\[\forall (\mathcal{F}_i \in \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n): \hspace{1cm} F_0 \notin \mathcal{F}_i\]  
\hspace{1cm} \text{[By [3.3].]} \\

\[3.5\]
\[\forall \{(F: F \in \mathcal{F}, F \cap S = \nu_i) \in \{(F: F \in \mathcal{F}, F \cap S = \nu_i) : \nu_i \in \Gamma(\mathcal{F}, S)\} : \hspace{1cm} F_0 \notin \{F: F \in \mathcal{F}, F \cap S = \nu_i\}\]  
\hspace{1cm} \text{[By [3.4] and the definition of \( \Delta \).]} \\

\[3.6\]
\[\forall (\nu_i \in \Gamma(\mathcal{F}, S)):
\hspace{1cm} F_0 \cap S = \nu_i\]  
\hspace{1cm} \text{[By [3.5].]} \\

\[3.7\]
\[F_0 \cap S \notin \Gamma(\mathcal{F}, S)\]  
\hspace{1cm} \text{[By [3.6].]} \\

\[3.8\]
\[\forall \{F \cap S : F \in \mathcal{F}\}\]  
\hspace{1cm} \text{[By [3.7] and the definition of \( \Gamma(\mathcal{F}, S) \).]} \\

\[3.9\]
\[F_0 \notin \mathcal{F}\]  
\hspace{1cm} \text{[This follows from [3.8].]} \\

Q.E.D. (3) \hspace{1cm} \text{[The assumption in [3.1] leads to a contradiction between [3.9] and the definition of \( F_0 \) in [3.3].]} \\

Q.E.D. \hspace{1cm} \text{[The claim follows from [2] and [3].]} \\

\[\Box\]

\textit{Lemma 3B.29:} Let \( \mathcal{F} \) be a function class and let \( S \in \text{Dom}(\mathcal{F})^l \) for some \( l \in \mathbb{N} \). Let \( \Delta(\mathcal{F}, S) = \)
\{J_1, J_2, \ldots, J_n\}. Then

\[ \forall (J \in \Delta(J, S)) : \Pi_J(S) = 1. \]

Proof:

[1] Let \( J \) be an arbitrary element of \( \Delta(J, S) \). \[ \text{[Definition.]} \]

[2] \( \exists (v \in \Gamma(J, S)) : \]

\[ \begin{align*}
J &= \{ F : F \in J, F \cap S = v \} \\
\text{[By the definition of} \ \Delta(J, S) \].
\end{align*} \]

[3] Let \( v \in \Gamma(J, S) \) be such that

\[ \begin{align*}
J &= \{ F : F \in J, F \cap S = v \} \\
\text{[Such a} \ v \text{exists by}[2].
\end{align*} \]

[4] \( \forall (F \in J) : \]

\[ 
F \cap S = v \quad \text{[By}[3].
\]

[5] \[ \left| \{ F \cap S : F \in J \} \right| = 1 \quad \text{[By}[4].
\]

[6] \[ \Pi_J(S) = 1 \quad \text{[By}[5] \text{and the definition of} \ \Pi(S). \]

Q.E.D.

Theorem 3.13.10: Let \( X \) be a set and \( \mathcal{K}_0 \) and \( \mathcal{F} \) be two function classes, with \( \text{Dom}(\mathcal{K}_0) = \text{Dom}(\mathcal{K}_1) = X. \)

Let \( f : X \times \text{Ran}(\mathcal{K}_1) \to \text{Dom}(\mathcal{K}_0) \). Let \( S \in X^1, \ell \in \mathbb{N}. \) Then

\[ \Pi_{\mathcal{K}_0, \mathcal{F}, \ell}(S) \geq \sum_{\mathcal{K} \in \Delta(J, S)} \Pi_{\mathcal{K}_0}(U(f, J, S)). \]
Proof:

[1] \[ |\Delta(\mathcal{F}, S)| \leq |\Gamma(\mathcal{F}, S)| \] \[ \text{[By lemma 3B.27.]} \]

[2] \[ \forall (\mathcal{G} \in \Delta(\mathcal{F}, S)) : \quad P_{\mathcal{F}}(S) = 1 \] \[ \text{[By lemma 3B.29.]} \]

[3] \[ \text{Let } \{v_1, v_2, \ldots, v_n\} = \Gamma(\mathcal{F}, S) \] \[ \text{[Definition (we are naming the elements of } \Gamma(\mathcal{F}, S)\text{).]} \]

[4] \[ \text{Let } \mathcal{F}_1 = \{F : F \in \mathcal{F}, F \cap S = v_1\}, \] \[ \mathcal{F}_2 = \{F : F \in \mathcal{F}, F \cap S = v_2\}, \] \[ \vdots \] \[ \mathcal{F}_n = \{F : F \in \mathcal{F}, F \cap S = v_n\} \] \[ \text{[Definition.]} \]

[5] \[ \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_n = \mathcal{F} \] \[ \text{[By lemma 3B.28.]} \]

[6] \[ \Pi_{\mathcal{G}_1(\mathcal{X}, \mathcal{F})}(S) \leq \sum_{\mathcal{F}_i \in \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\}} \Pi_{\mathcal{F}_1}(U(f, \mathcal{F}_1, S)) \] \[ \text{[By \[2\], \[3\], and lemma 3B.25.]} \]

[7] \[ \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\} = \Delta(\mathcal{F}, S) \] \[ \text{[4], and the definition of } \Delta(\mathcal{F}, S). \]

[8] \[ \Pi_{\mathcal{G}_1(\mathcal{X}, \mathcal{F})}(S) \leq \sum_{\mathcal{F}_i \in \Delta(\mathcal{F}, S)} \Pi_{\mathcal{F}_1}(U(f, \mathcal{F}_1, S)) \] \[ \text{[By \[6\] and \[7\].]} \]

Q.E.D. \[ \square \]

Corollary 3B.31: Let \( X \) be a sample space and let \( \mathcal{X}_0 \) and \( \mathcal{F} \) be two function classes, with \( \text{Dom}(\mathcal{X}_0) = \text{Dom}(\mathcal{X}_1) = X \). Let \( f : X \times \text{Rat}(\mathcal{X}_1) \rightarrow \text{Dom}(\mathcal{X}_0) \). Let \( S \in X^t, t \in \mathbb{N} \).

\[ \Pi_{\mathcal{G}_1(\mathcal{X}, \mathcal{F})}(S) \leq \Pi_{\mathcal{X}_1}(S) \Pi_{\mathcal{F}_1}(t). \]
Proof:

[1] \[ \Pi_{\mathcal{J}, \mathcal{K}_S}(S) \leq \sum_{\mathcal{J}_i \in \Delta(\mathcal{J}, S)} \Pi_{\mathcal{J}_i}(U(f, \mathcal{J}_i, S)) \]
[ By theorem 3B.30. ]

[2] \forall (\mathcal{J}_i \in \Delta(\mathcal{J}, S)):
\[ |U(f, \mathcal{J}_i, S)| \leq \ell \]

Proof of 2:

[2.1] \forall (\mathcal{J}_i \in \Delta(\mathcal{J}, S)):
\[ U(f, \mathcal{J}_i, S) = \{ w(f, \mathcal{J}_i, x) : x \in S \} \]
[ By the definition of \( U(f, \mathcal{J}_i, S) \). ]

[2.2] \forall (\mathcal{J}_i \in \Delta(\mathcal{J}, S)):
\[ |\{ w(f, \mathcal{J}_i, x) : x \in S \}| = \ell \]
[ Since \( S \) has \( \ell \) elements. ]

Q.E.D. (2) [ The claim follows from [2.1] and [2.2]. ]

[3] \[ \Pi_{\mathcal{J}, \mathcal{K}_S}(S) \leq \sum_{\mathcal{J}_i \in \Delta(\mathcal{J}, S)} \Pi_{\mathcal{J}_i}(\ell) \]
[ By [1], [2], and the definition of \( \Pi_{\mathcal{J}_i}(\ell) \). ]

[4] \[ \Pi_{\mathcal{J}, \mathcal{K}_S}(S) \leq \Pi_{\mathcal{K}_S}(\ell) \sum_{\mathcal{J}_i \in \Delta(\mathcal{J}, S)} 1 \]
[ Factoring \( \Pi_{\mathcal{K}_S}(\ell) \) out of the summation on the right side of [3]. ]
\[ |\Delta(\mathcal{F}, S)| \leq |\Gamma(\mathcal{F}, S)| \] [By lemma 3B.27.]

\[ |\Delta(\mathcal{F}, S)| \leq |\{F \cap S : F \in \mathcal{F}\}| \] [By 5 and the definition of \( \Gamma(\mathcal{F}, S) \).]

\[ |\Delta(\mathcal{F}, S)| \leq \Pi_{\mathcal{F}}(S) \] [By 6 and the definition of \( \Pi_{\mathcal{F}}(S) \).]

\[ |\Delta(\mathcal{F}, S)| \leq \Pi_{\mathcal{F}}(\ell) \] [By 7 and the definition of \( \Pi_{\mathcal{F}}(\ell) \), since \( |S| = \ell \) by assumption.]

\[ \Pi_{\mathcal{F}}(\mathcal{M}, \mathcal{F})(S) \leq \Pi_{\mathcal{M}}(\ell)\Pi_{\mathcal{F}}(\ell) \] [By 4 and 8.]

Q.E.D. \[ \square \]

Theorem 3B.32: Let \( X \) be a set, and let \( \mathcal{H}_0 \) and \( \mathcal{H}_1 \) be two hypothesis classes whose domain is \( X \). Let \( S \in X^\ell, \ell \in \mathbb{N} \). Then

\[ \Gamma(\mathcal{F}(\mathcal{H}_0, \mathcal{H}_1), S) = \bigcup_{\mathcal{H}_0 \in \mathcal{H}(\mathcal{H}_0, \mathcal{H}_1)} \Gamma(\mathcal{H}_0, U(f_1, \mathcal{H}_0, S)). \]

Proof:

\[ \Gamma(\mathcal{F}(\mathcal{H}_0, \mathcal{H}_1), S) = \{C \cap S : C \in \mathcal{F}(\mathcal{H}_0, \mathcal{H}_1)\} \] [By the definition of \( \Gamma(\cdot, \cdot, \cdot) \).]

\[ \{C \cap S : C \in \mathcal{F}(\mathcal{H}_0, \mathcal{H}_1)\} = \{\mathcal{F}(H_0, H_1) \cap S : H_0 \in \mathcal{H}_0, H_1 \in \mathcal{H}_1\} \] [By lemma 3B.18.]

\[ \Gamma(\mathcal{F}(\mathcal{H}_0, \mathcal{H}_1), S) = \{\mathcal{F}(H_0, H_1) \cap S : H_0 \in \mathcal{H}_0, H_1 \in \mathcal{H}_1\} \] [By [1] and [2].]
Proof of 4:

[4.1] \[ \forall (H_0 \in \mathcal{H}_0, H_1 \in \mathcal{H}_1) : \]

\[ \exists (\mathcal{F}_1 \in \Delta (\mathcal{H}_1, S)) : \]

\[ \mathcal{F}_{f_1}(H_0, H_1) \cap S = H_0 \cap U(f_2, \mathcal{F}_1, H_1) \]

[4.1] If such \( H_0, H_1 \) do not exist the claim holds vacuously.

[4.2] \[ \exists (\mathcal{F}_1 \in \Delta (\mathcal{H}_1, S)) : \]

\[ H_1 \in \mathcal{F}_1 \]

[4.2] This follows from lemma 3B.28 and the fact that \( H_1 \in \mathcal{H}_1 \).

[4.3] Let \( \mathcal{F}_1 \in \Delta (\mathcal{H}_1, S) \)

be such that \( H_1 \in \mathcal{F}_1 \)

[4.3] Such \( \mathcal{F}_1 \) exists by [4.2].

[4.4] \[ \forall (x \in S) : \]

\[ \mathcal{F}_{f_1}(H_0, H_1)(x) = H_0(u(f_2, H_1, x)) \]

[4.4] By the definition of \( u(\cdot, H_1, \cdot) \).

[4.5] \[ \forall (x \in S) : \]

\[ \mathcal{F}_{f_1}(H_0, H_1)(x) = H_0(u(f_2, \mathcal{F}_1, x)) \]

[4.5] By theorem 3B.20 and the assumption (in [4.3]) that \( H_1 \in \mathcal{F}_1 \).

[4.6] \[ \forall (x \in S) : \]

\[ \{ x \in \mathcal{F}_{f_1}(H_0, H_1) \} \leftrightarrow \{ u(f_2, \mathcal{F}_1, x) \in H_0 \} \]

[4.6] This is just [4.5] rewritten in set notation.

[4.7] \[ \{ x : x \in S \cap \mathcal{F}_{f_1}(H_0, H_1) = \{ u(f_2, \mathcal{F}_1, x) : x \in S \} \cap H_0 \]

[4.7] [By [4.6].]
\[ S \cap \mathcal{G}_{f_1}(H_0, H_1) = \{ w(f_2, \mathcal{J}_1, x) : x \in S \} \cap H_0 \]

[4.8] 

By [4.7] and the definition of set operations on lists (in this case the set operation in question is the intersection on the left side of the equation). 

\[ S \cap \mathcal{G}_{f_1}(H_0, H_1) = U(f_1, \mathcal{J}_1, S) \cap H_0 \]

[4.9] 

By [4.8] and the definition of \( U(\cdot, \cdot, \cdot) \). 

Q.E.D. (4) The claim follows from [4.1], [4.3], and [4.9], since the \( H_0 \) and \( H_1 \) of [4.1] are arbitrary members of \( \mathcal{H}_0 \) and \( \mathcal{H}_1 \). 

[5] 

\[ \Gamma(\mathcal{G}_{f_1}(\mathcal{H}_0, \mathcal{H}_1), S) = \{ H_0 \cap U(f_2, \mathcal{J}_1, S) : H_0 \in \mathcal{H}_0, \mathcal{J}_1 \in \Delta(\mathcal{H}_1, S) \} \]

[6] 

By [5] and [4]. 

\[ \Gamma(\mathcal{G}_{f_1}(\mathcal{H}_0, \mathcal{H}_1), S) = \bigcup_{\mathcal{J}_1 \in \Delta(\mathcal{H}_1, S)} (H_0 \cap U(f_2, \mathcal{J}_1, S) : H_0 \in \mathcal{H}_0) \]

By [5]. 

Q.E.D. 

Theorem 3D.33: Let \( X \) be a set, and let \( \mathcal{H}_0 \) and \( \mathcal{H}_1 \) be two hypothesis classes whose domain is \( X \). Let \( S \in X^t, t \in N \). Then

\[ \Delta(\mathcal{G}_{f_1}(\mathcal{H}_0, \mathcal{H}_1), S) = \bigcup_{\mathcal{J}_1 \in \Delta(\mathcal{H}_1, S)} \Delta(\mathcal{G}_{f_1}(\mathcal{H}_0, \mathcal{H}_1), S). \]
Proof:

[1] \[ \Gamma(G_f(\mathcal{F}_0, \mathcal{F}_1), S) = \{ C \cap S : C \in G_f(\mathcal{F}_0, \mathcal{F}_1) \} \]
[By the definition of \( \Gamma(\cdot, \cdot, \cdot) \).]

[2] \[ \{ C \cap S : C \in G_f(\mathcal{F}_0, \mathcal{F}_1) \} = \{ G_f(H_0, H_1) \cap S : H_0 \in \mathcal{F}_0, H_1 \in \mathcal{F}_1 \} \]
[By lemma 3B.18.]

[3] \[ \Gamma(G_f(\mathcal{F}_0, \mathcal{F}_1), S) = \{ G_f(H_0, H_1) \cap S : H_0 \in \mathcal{F}_0, H_1 \in \mathcal{F}_1 \} \]
[By [1] and [2].]

[4] \[ \Gamma(G_f(\mathcal{F}_0, \mathcal{F}_1), S) = \bigcup_{\mathcal{F} \in A(\mathcal{F}_0, \mathcal{F}_1)} \Gamma(\mathcal{F}_0, U_f(\mathcal{F}_1, S)) \]
[By theorem 3B.32.]

[5] \[ \Gamma(G_f(\mathcal{F}_0, \mathcal{F}_1), S) = \bigcup_{\mathcal{F} \in A(\mathcal{F}_0, \mathcal{F}_1)} \{ H_0 \cap U_f(\mathcal{F}_1, S) : H_0 \in \mathcal{F}_0 \} \]
[By [4] and the definition of \( \Gamma \).]

[6] \[ \forall (v \in \Gamma(G_f(\mathcal{F}_0, \mathcal{F}_1), S)) : \]
\[ \{ C : C \cap S = v, C \in G_f(\mathcal{F}_0, \mathcal{F}_1) \} = \]
\[ \{ G_f(H_0, H_1) : G_f(H_0, H_1) \cap S = v, H_0 \in \mathcal{F}_0, H_1 \in \mathcal{F}_1 \} \]
[By [3] and the definition of \( \Gamma \).]

[7] \[ \forall (v \in \Gamma(G_f(\mathcal{F}_0, \mathcal{F}_1), S)) : \]
\[ \{ C : C \cap S = v, C \in G_f(\mathcal{F}_0, \mathcal{F}_1) \} = \]
\[ \bigcup_{\mathcal{F} \in A(\mathcal{F}_0, \mathcal{F}_1)} \{ G_f(H_0, H_1) : G_f(H_0, H_1) \cap S = v, H_0 \in \mathcal{F}_0, H_1 \in \mathcal{F}_1 \} \]
[By [6] and lemma 3B.28.]
\[ (C : C \cap S = v, C \in \mathcal{P}_1(\mathcal{K}_0, \mathcal{K}_1)) : v \in \Gamma(\mathcal{P}_1(\mathcal{K}_0, \mathcal{K}_1), S) \]
\[ = \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ \mathcal{P}_1(H_0, H_1) : H_0 \cap U(f_1, f_2, S) = v, H_0 \in \mathcal{K}_0, H_1 \in \mathcal{F}_1 \} : v \in \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ H_0 \cap U(f_1, f_2, S) : H_0 \in \mathcal{K}_0 \} \]
\[ \text{[By [5] and [7].]} \]

\[ (C : C \cap S = v, C \in \mathcal{P}_1(\mathcal{K}_0, \mathcal{K}_1)) : v \in \Gamma(\mathcal{P}_1(\mathcal{K}_0, \mathcal{K}_1), S) \]
\[ = \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ \mathcal{P}_1(H_0, H_1) : H_0 \cap U(f_1, f_2, S) = v, H_0 \in \mathcal{K}_0, H_1 \in \mathcal{F}_1 \} : v \in \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ H_0 \cap U(f_1, f_2, S) : H_0 \in \mathcal{K}_0 \} \]
\[ \text{[By [8] lemma 3B.19.]} \]

\[ \forall (f_1, f_2 \in \Delta(\mathcal{K}_1, S)) : \]
\[ \forall \left( w \in \left\{ \begin{array}{l}
(\mathcal{P}_1(H_0, H_1) : H_0 \cap U(f_1, f_2, S) = v, H_0 \in \mathcal{K}_0, H_1 \in \mathcal{F}_1) : v \in \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ H_0 \cap U(f_1, f_2, S) : H_0 \in \mathcal{K}_0 \}
\end{array} \right\} : w \right) = \left\{ \begin{array}{l}
(\mathcal{P}_1(H_0, H_1) : H_0 \cap U(f_1, f_2, S) = v, H_0 \in \mathcal{K}_0, H_1 \in \mathcal{F}_1) : v \in \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ H_0 \cap U(f_1, f_2, S) : H_0 \in \mathcal{K}_0 \}
\end{array} \right\} \]
\[ \text{Proof of 10:} \]

\[ \text{[10.1]} \]
Let \( \mathcal{F}_1, \mathcal{F}_2 \in \Delta(\mathcal{K}_1, S) \)

be such that
\[ \exists w \in \left\{ \begin{array}{l}
(\mathcal{P}_1(H_0, H_1) : H_0 \cap U(f_1, f_2, S) = v, H_0 \in \mathcal{K}_0, H_1 \in \mathcal{F}_1) : v \in \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ H_0 \cap U(f_1, f_2, S) : H_0 \in \mathcal{K}_0 \}
\end{array} \right\} : w \]
\[ \text{[If no such } \mathcal{F}_1, \mathcal{F}_2 \text{ exist, then the claim follows trivially.]}

\[ \text{[10.2]} \]
Let
\[ w \in \left\{ \begin{array}{l}
(\mathcal{P}_1(H_0, H_1) : H_0 \cap U(f_1, f_2, S) = v, H_0 \in \mathcal{K}_0, H_1 \in \mathcal{F}_1) : v \in \bigcup_{\mathcal{A} \in \Delta(\mathcal{K}_1, S)} \{ H_0 \cap U(f_1, f_2, S) : H_0 \in \mathcal{K}_0 \}
\end{array} \right\} : w \]
\[ \text{[Such a } w \text{ exists by [10.1].]}

[10.3] Let \( H_0 \in \mathcal{H}_0, v \in \{ H_0 \cap U(f_1, \mathcal{A}, S) : H_2 \in \mathcal{H}_0 \} \)

be such that

\[ w = \{ \mathcal{G}_f(H_0, H_1) : H_0 \cap U(f_1, \mathcal{A}, S) = v, H_0 \in \mathcal{H}_0, H_1 \in \mathcal{A} \} \]

[ Such \( H_0, v \) exist by [10.2]. ]

\[ H_0 \cap U(f_1, \mathcal{A}, S) = v \quad \text{[ By [10.3].]} \]

[10.5] \( v \in \{ H_0 \cap U(f_1, \mathcal{A}, S) : H_0 \in \mathcal{H}_0 \} \)

[ By [10.4], since \( H_0 \in \mathcal{H}_0 \) by [10.3]. ]

[10.6] \( w \in \{ \mathcal{G}_f(H_0, H_1) : H_0 \cap U(f_1, \mathcal{A}, S) = v, H_0 \in \mathcal{H}_0, H_1 \in \mathcal{A} \} \)

\[ v \in \{ H_0 \cap U(f_1, \mathcal{A}, S) : H_0 \in \mathcal{H}_0 \} \]

[ By [10.3], [10.4], and [10.5]. ]

Q.E.D. (10) [ The claim follows from [10.3] and [10.6], since \( w \) is an arbitrary member of the set

\[ \{ \mathcal{G}_f(H_0, H_1) : H_0 \cap U(f_1, \mathcal{A}, S) = v, H_0 \in \mathcal{H}_0, H_1 \in \mathcal{A} \} : v \in \{ H_0 \cap U(f_1, \mathcal{A}, S) : H_0 \in \mathcal{H}_0 \} \} \]

[ By [9] and [10]. ]

[11] \( \{ C : C \cap S = v, C \in \mathcal{G}_f(\mathcal{H}_0, \mathcal{A}) \} : v \in \Gamma(\mathcal{G}_f(\mathcal{H}_0, \mathcal{A}), S) \} = \)

\[ \bigcup_{\mathcal{A} \in \Delta(\mathcal{A}, S)} \{ \mathcal{G}_f(H_0, H_1) : H_0 \cap U(f_1, \mathcal{A}, S) = v, H_0 \in \mathcal{H}_0, H_1 \in \mathcal{A} \} : v \in \{ H_0 \cap U(f_1, \mathcal{A}, S) : H_0 \in \mathcal{H}_0 \} \}

[ By [9] and [10]. ]

[12] \( \{ C : C \cap S = v, C \in \mathcal{G}_f(\mathcal{H}_0, \mathcal{A}) \} : v \in \Gamma(\mathcal{G}_f(\mathcal{H}_0, \mathcal{A}), S) \} = \)

\[ \bigcup_{\mathcal{A} \in \Delta(\mathcal{A}, S)} \{ \mathcal{G}_f(H_0, H_1) : H_0 \cap U(f_1, \mathcal{A}, S) = v, H_0 \in \mathcal{H}_0, H_1 \in \mathcal{A} \} : v \in \Gamma(\mathcal{H}_0, U(f_1, \mathcal{A}, S)) \}

[ By [11] and the definition of \( \Gamma(\cdot, \cdot) \). ]
3B.2. The \( \text{inp}_i \) function.

In this section, we develop a notation that allows us to speak of the functions that appear in a composition (or any other conjecture) in purely syntactic terms. The problem with our current notation is we have no concise way to refer to "the first of the two things being conjectured in the expression \( \mathcal{G}_f(F_1, F_2) \)." When we speak of \( F_1 \) or \( F_2 \) in reference to this composition, we are speaking of functions, when in fact we would also like to speak of the positions of things within a composition. We would like a notation analogous to the one we used in example 3B.32, where the relationships between functions could be expressed in terms of the edges and vertices in a graph.

For this reason, we introduce the idea of a conjecture specification.

By [13] and the definition of \( U(\cdot, \cdot, \cdot) \).

By [14] and lemma 3B.18.

By [15] and the definition of \( \Delta(\cdot, \cdot) \).

By [16] and the definition of \( \Delta(\cdot, \cdot) \).

Q.E.D. Δ
Definition: Let $\mathcal{D}$ be a set of function classes, with the property that

$$\forall (\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{D}) \left( \text{Dom}(\mathcal{F}_1) = \text{Dom}(\mathcal{F}_2) \land \text{Ran}(\mathcal{F}_1) = \text{Ran}(\mathcal{F}_2) \right).$$

Let $\mathcal{D}$ be a set, and let $W$ be a function

$$W : \mathcal{D} \to \mathcal{D}. $$

If $\phi \in \mathcal{D}$ we say that $\phi$ is a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$, and, for all $p \in \mathcal{D}$, we define

$$\text{Func}(\mathcal{D}, \mathcal{D})(\phi)$$

to be $W(\phi)$. If $\psi$ is an $n$-ary conjectural operator, and $\phi_1, \phi_2, \ldots, \phi_n$ are conjecture specifications with respect to $\mathcal{D}$ and $\mathcal{D}$, then

$$\psi(\phi_1, \phi_2, \ldots, \phi_n)$$

is a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$, and we define

$$\text{Func}(\mathcal{D}, \mathcal{D})(\psi(\phi_1, \phi_2, \ldots, \phi_n))$$

to be

$$\psi(\text{Func}(\mathcal{D}, \mathcal{D})(\phi_1), \text{Func}(\mathcal{D}, \mathcal{D})(\phi_2), \ldots, \text{Func}(\mathcal{D}, \mathcal{D})(\phi_n)).$$

Example 38.34: A conjunction specification for figure 3

Consider again the boolean circuit in example 3.3. We showed that the circuit in that example could be written as a class of compositions in the following way:

$$\mathcal{G}_f_1(\mathcal{G}_f_1(\mathcal{G}_f_1(\mathcal{F}_1, \mathcal{F}_2), \mathcal{F}_3), \mathcal{F}_4), \mathcal{F}_5).$$

Let $\mathcal{D}$ be the set $\{\phi_1, \phi_2, \phi_3, \phi_4\}$, let $\mathcal{D} = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$, and define $W(\phi_1)$ to be $\mathcal{F}_1$, $W(\phi_2)$ to be $\mathcal{F}_2$, $W(\phi_3)$ to be $\mathcal{F}_3$, and $W(\phi_4)$ to be $\mathcal{F}_4$.

$\phi_1$ and $\phi_2$ are conjecture specifications with respect to $\mathcal{D}$ and $\mathcal{D}$, and, since $\mathcal{G}_f_1$ is a conjectural operator, $\mathcal{G}_f_1(\phi_1, \phi_2)$ is a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$. Since $\phi_2$ is also a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$, and since $\mathcal{G}_f_2$ is a conjectural operator,

$$\mathcal{G}_f_2(\mathcal{G}_f_1(\phi_1, \phi_2), \phi_3)$$

is a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$. Finally, since $\phi_3$ is a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$ and $\mathcal{G}_f_3$ is a conjectural operator,

$$\mathcal{G}_f_3(\mathcal{G}_f_2(\mathcal{G}_f_1(\phi_1, \phi_2), \phi_3), \phi_4)$$

is a conjecture specification with respect to $\mathcal{D}$ and $\mathcal{D}$.
is a conjunction operator with respect to \( \mathcal{D} \) and \( \mathcal{F} \).

By the definition of \( \text{Func}(\mathcal{D}, \mathcal{F}) \) we have

\[
\text{Func}(\mathcal{D}, \mathcal{F})(\mathcal{F}_1(f_1, (\mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4), \mathcal{F}_5)) =
\]

\[
\mathcal{F}_2(\text{Func}(\mathcal{D}, \mathcal{F})(\mathcal{F}_3(f_2, (\mathcal{F}_4, \mathcal{F}_5)), \mathcal{F}_6)) =
\]

\[
\mathcal{F}_3(\text{Func}(\mathcal{D}, \mathcal{F})(\mathcal{F}_4(f_4, (\mathcal{F}_5, \mathcal{F}_6)), \mathcal{F}_7)) =
\]

\[
\mathcal{F}_4(\text{Func}(\mathcal{D}, \mathcal{F})(\mathcal{F}_5(f_5, (\mathcal{F}_6, \mathcal{F}_7)), \mathcal{F}_8)) =
\]

\[
\mathcal{F}_5(\text{Func}(\mathcal{D}, \mathcal{F})(\mathcal{F}_6(f_6, (\mathcal{F}_7, \mathcal{F}_8)), \mathcal{F}_9)).
\]

The point of having conjunction specifications is to allow us to denote complex specifications without writing them out. We will often use a kind of shorthand (as in fact we already have) such as using the specification \( \mathcal{F}_g \) to refer to the specification \( \mathcal{F}_f(\mathcal{F}_g, \mathcal{F}_h) \) where \( \mathcal{F}_g \) and \( \mathcal{F}_h \) may stand for conjunction specifications as well. Because of this it will be useful to define a containment relationship for conjunction specifications. The idea is that \( \mathcal{F}_g \subset \mathcal{F}_h \) iff the specification \( \mathcal{F}_g \) is part of the specification \( \mathcal{F}_h \). Formally:

**Definition:** If \( \mathcal{D} \) is a set of function classes, \( \mathcal{F} \) is a set, and \( \mathcal{F}(\mathcal{D}, \mathcal{F}) \) is a conjunction specification with respect to \( \mathcal{D} \) and \( \mathcal{F} \), then we say that

\[
\mathcal{F}_g \subset \mathcal{F}_h \text{ if } \mathcal{F}_g \subset \mathcal{F}_h.
\]

(If we use \( \mathcal{F}_g \) to denote \( \mathcal{F}(\mathcal{D}, \mathcal{F}) \), then \( \mathcal{F}_g \subset \mathcal{F}_h \) if \( \mathcal{F}_g \subset \mathcal{F}_h \).

If we use \( \mathcal{F}_g \) to denote \( \mathcal{F}(\mathcal{D}, \mathcal{F}) \), then \( \mathcal{F}_g \subset \mathcal{F}_h \) and \( \mathcal{F}_g \subset \mathcal{F}_h \).

Additional:

(a) for all conjunction specifications \( \mathcal{F}_g \),

\[
\mathcal{F}_g \subset \mathcal{F}_h.
\]

(b) for all conjunction specifications \( \mathcal{F}_g, \mathcal{F}_h \),

\[
(\mathcal{F}_g \neq \mathcal{F}_h) \land (\mathcal{F}_g \subset \mathcal{F}_h) \rightarrow \mathcal{F}_g \not\subset \mathcal{F}_h.
\]
(c) for all conjecture specifications \( \hat{s}_i, \hat{s}_j, \hat{s}_k \).

\[
(\hat{s}_k \in \hat{s}_j) \land (\hat{s}_j \in \hat{s}_i) \rightarrow \hat{s}_k \in \hat{s}_i.
\]

Item (b) in this definition requires a comment: we have not defined conjecture specifications in such a way that (b) is guaranteed to hold. For example, we could have the specification \( \hat{s}_i = \mathcal{C}_f(\hat{s}_j, \hat{s}_k) \) with \( \hat{s}_k = \mathcal{C}_f(\hat{s}_i, \hat{s}_k) \). However, conjecture specifications that violate (b) defeat the purpose of having conjecture specifications, which is to have unique names for all the elements of a conjecture. Hence:

Definition: A legal conjecture specification is one that causes item (b) in the definition of \( \hat{s}_i \) to be satisfied.

Henceforth we will assume that all our conjecture specifications are legal.

We are specifically interested conjecture specifications that consist entirely of compositions. For this reason we define a composition specification as follows:

Definition: Let \( \hat{s}_i \) be a conjecture specification with respect to some set \( \mathcal{D} \) and some set \( \mathcal{E} \). We say that \( \hat{s}_i \) is a composition specification if and only if, for all \( \hat{s}_j \in \hat{s}_i \), either \( \hat{s}_j \in \mathcal{D} \) or \( \hat{s}_j \) has the form \( \mathcal{C}_f(\hat{s}_k, \hat{s}_h) \), where \( \mathcal{C}_f \) is a composition operator and \( \hat{s}_k \) and \( \hat{s}_h \) are conjecture specifications with respect to \( \mathcal{D} \) and \( \mathcal{E} \).

We will need the following in what follows:

Definition: Let \( S_1 \) and \( S_2 \) be two lists. Then \( S_1 + S_2 \) denotes the result of catenating \( S_1 \) and \( S_2 \).

The following concept is also useful:

Definition: Let \( \hat{s}_i \) be a conjecture specification with respect to some set \( \mathcal{D} \) of function classes and some set \( \mathcal{E} \). Then \( \mathcal{E}(\hat{s}_i) \) is defined to be the set

\[
\{ \hat{q} \in \mathcal{E} : \hat{q} \in \hat{s}_i \}.
\]
If \( \mathcal{D}(i_i) = \{i_1, i_2, \ldots, i_n \} \) and

\[
\mathcal{F}_1 \in \mathcal{W}(i_1), \\
\mathcal{F}_2 \in \mathcal{W}(i_2), \\
\vdots \\
\mathcal{F}_n \in \mathcal{W}(i_n),
\]

then we say that

\[
i_i \left[ i_1 \setminus \mathcal{F}_1 \setminus i_2 \setminus \mathcal{F}_2 \setminus \cdots \setminus i_n \setminus \mathcal{F}_n \right]
\]

is the class that would be specified by \( \hat{i}_i \) if we defined \( \text{Func}(s, \mathcal{D})(i_j) \) to be \( \mathcal{F}_j \) instead of \( \mathcal{W}(i_j) \) for all \( i_j \in \mathcal{D}(i_i) \).

Usually we will use

\[
i_i \left[ \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n \right]
\]

as shorthand notation for

\[
i_i \left[ i_1 \setminus \mathcal{F}_1 \setminus i_2 \setminus \mathcal{F}_2 \setminus \cdots \setminus i_n \setminus \mathcal{F}_n \right].
\]

Note that if \( i_i \) a composition specification and \( |\mathcal{D}(i_i)| = 1 \), then \( \hat{i}_i \) cannot be a composition, because a composition must contain at least two elements of \( \mathcal{D} \) (because the recursive definition of a conjunction specification is only anchored when the specification is a member of \( \mathcal{D} \), and a composition contains at least two nested specifications). Therefore \( \hat{i}_i \) must refer directly to a member of \( \mathcal{D} \). If \( \mathcal{D}(i_i) = \{i_1\} \) we can therefore conclude that \( \hat{i}_i = i_1 \).

We also make the following definitions:

**Definition:** Let \( i_h \) be a composition specification, then:

(a) If \( i_h \) contains the specification \( \mathcal{F}_j(i_i, i_j) \) for any composition operator \( \mathcal{F}_j \), and any specifications \( i_i, i_j \), we say that \( i_j \succ i_i \).

(b) If \( i_h \) contains the specification \( \mathcal{F}_j(i_i, \mathcal{F}_j(i_j, i_k)) \) for any operators \( \mathcal{F}_j \), and \( \mathcal{F}_j \), and any specifications \( i_i, i_j, i_k \), we say that \( i_j \succ i_i \).

(c) If \( i_h \) contains the specification \( \mathcal{F}_j(i_i, \mathcal{F}_j(i_j, \mathcal{F}_j(i_j, \mathcal{F}_j(i_k, \mathcal{F}_k)))) \) for any operators \( \mathcal{F}_j \), and \( \mathcal{F}_j \), and any specifications \( i_i, i_j, i_k \), then \( i_k \succ i_j \).

(d) If \( i_h \) contains the specification \( i_h = \mathcal{F}_j(i_i, \mathcal{F}_j(i_j, \mathcal{F}_j(i_j, \mathcal{F}_k))) \) for any operators \( \mathcal{F}_j \), and \( \mathcal{F}_j \), and any specifications \( i_i, i_j, i_k \), then, for any specification \( i_k \in i_h \), if \( i_1 \succ \hat{i}_h \) then \( \hat{i}_1 \succ \hat{i}_h \).
We define $\succ^*$ to be the transitive closure of $\succ$.

The intent of the $\succ$ relation is to let us specify what functions in a composition supply arguments to what other functions.

Example 3B.35:

Consider the conjunction specification

$$A = \mathcal{G}_k(\mathcal{G}_{k_1}(\mathcal{G}_{k_2}(\mathcal{G}_{k_3}(\mathcal{G}_{k_4}(\mathcal{G}_{k_5}(\mathcal{G}_{k_6}(\mathcal{G}_{k_7}(\mathcal{G}_{k_8}(\mathcal{G}_{k_9}(1, q_1), q_2), q_3), q_4), q_5), q_6), q_7), q_8), q_9))$$

from example 3B.34. $A$ contains the specifications

$\dot{q}_3$,  
$\dot{q}_4$,  
$\mathcal{G}_{k_1}(\dot{q}_3, \dot{q}_4)$,  
$\dot{q}_2$,  
$\mathcal{G}_{k_1}(\mathcal{G}_{k_1}(\dot{q}_3, \dot{q}_4), \dot{q}_2)$,  
$\dot{q}_1$,  
$\mathcal{G}_{k_1}(\mathcal{G}_{k_1}(\mathcal{G}_{k_1}(\dot{q}_3, \dot{q}_4), \dot{q}_2), \dot{q}_1)$.

By our definition we have

$\dot{q}_1 \succ \dot{q}_2$ by part (c)

$\dot{q}_1 \succ \mathcal{G}_{k_1}(\mathcal{G}_{k_1}(\dot{q}_3, \dot{q}_4), \dot{q}_2)$ by part (a)

$\dot{q}_2 \succ \dot{q}_4$ by part (c)

$\dot{q}_2 \succ \mathcal{G}_{k_1}(\dot{q}_3, \dot{q}_4)$ by part (a)

$\dot{q}_4 \succ \dot{q}_5$ by part (a)

From this we can see that

$\dot{q}_1 \succ \dot{q}_2 \succ \dot{q}_3 \succ \dot{q}_4$. 

\[\]
Example 3B.36:

Consider a class of functions $G_f(\mathcal{F}, \mathcal{F})$ in which members of $\mathcal{F}$ are composed with members of $\mathcal{F}$. We wish to specify in an abstract manner that, in any $G_f(F_1, F_2) \in G_f(\mathcal{F}, \mathcal{F})$, $F_1$ does not supply arguments to $F_2$. However, a statement like $\mathcal{F} \supset \mathcal{F}$ cannot express this.

But let $\mathcal{S}$ be the set $\{q_1, q_2\}$, and let $W$ be defined so that $W(q_1) = W(q_2) = \mathcal{F}$. Then

$$G_f(q_1, q_2)$$

is a conjunction specification with respect to $\{\mathcal{F}\}$ and $\mathcal{S}$, and, by the definition of $Func(\mathcal{F}, \mathcal{F})$,

$$Func(\mathcal{F}, \mathcal{F})(G_f(q_1, q_2)) =$$

$$G_f(Func(\mathcal{F}, \mathcal{F})(q_1), Func(\mathcal{F}, \mathcal{F})(q_2)) =$$

$$G_f(W(q_1), W(q_2)) =$$

$$G_f(\mathcal{F}, \mathcal{F})$$.

But we can state that the first member of the conjunction does not supply arguments to the second by saying that $q_2 \supset q_1$.

Our notation allows us to conveniently collapse the $U$ notation of proposition 3B.36 as follows:

Definition: Let $\mathcal{S}$ be a set, let $\mathcal{D}$ be a set of function classes, and let $W$ be a function mapping $\mathcal{S}$ to $\mathcal{D}$. Let $S \in \text{Dom}(\mathcal{D})^t$ for some $t > 0$. We make the following definitions:

(a) If $i_s$ is a composition specification with respect to $\mathcal{D}$ and $\mathcal{S}$, and $\bar{S}(i_s) = \{q_s, q_{s+1}, \ldots, q_m\}$, then

$$\text{Impl}(i_s, \{S\}) \equiv S.$$

(b) If $\bar{G}_f(i_s, i_j)$ is a composition specification with respect to $\mathcal{D}$ and $\mathcal{S}$, and $\bar{S}(i_j) = \{q_s, q_{s+1}, \ldots, q_m\}$,

$$\bar{S}(i_s) = \{q_s, q_{s+1}, \ldots, q_n\},$$

where

$$q_s \supset q_{s+1} \supset \ldots \supset q_m$$

and

$$q_s \supset q_{s+1} \supset \ldots \supset q_n.$$
Then

\[ V \left( \dot{q}_t \in \mathcal{D}(\dot{x}_1) : \right) \]

\[ \begin{align*}
\mathcal{I}_0 & \in \Delta \left( \text{Func}(\varphi, x)(\dot{q}_0), \text{Imp}_n(\dot{x}_j, \varphi, S) \right), \\
\mathcal{I}_{n+1} & \in \Delta \left( \text{Func}(\varphi, x)(\dot{q}_{n+1}), \text{Imp}_n(\dot{x}_j, \varphi, S) \right), \\
& \vdots \\
\mathcal{I}_n & \in \Delta \left( \text{Func}(\varphi, x)(\dot{q}_n), \text{Imp}_n(\dot{x}_j, \varphi, S) \right)
\end{align*} \]

if \( U \left( f, \dot{x}_j [\mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m], S \right) \) is defined, then

\[ \begin{align*}
\mathcal{I}_0 & \in \Delta \left( \text{Func}(\varphi, x)(\dot{q}_0) \right), \\
\mathcal{I}_n & \in \Delta \left( \text{Imp}_n(\dot{x}_j, \varphi, U \left( f, \dot{x}_j [\mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m], S \right)) \right) \\
\mathcal{I}_{n+1} & \in \Delta \left( \text{Imp}_n(\dot{x}_j, \varphi, U \left( f, \dot{x}_j [\mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m], S \right)) \right) \\
& \vdots \\
\mathcal{I}_n & \in \Delta \left( \text{Imp}_n(\dot{x}_j, \varphi, U \left( f, \dot{x}_j [\mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m], S \right)) \right).
\]

Informally, \( \text{Imp}_n(\dot{x}_j, \varphi, U \left( f, \dot{x}_j [\mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m], S \right)) \) is the list of inputs that \( \dot{q}_t \in \mathcal{D}(\dot{x}_1) \) sees when it appears in \( \dot{x}_j [\mathcal{I}_0 \setminus \mathcal{I}_0, \mathcal{I}_{n+1} \setminus \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m \setminus \mathcal{I}_m] \), and when \( \dot{x}_j [\mathcal{I}_0 \setminus \mathcal{I}_0, \mathcal{I}_{n+1} \setminus \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m \setminus \mathcal{I}_m] \) sees the list \( S \). However, since the input for \( \dot{q}_k \) does not depend on \( \dot{q}_{k+1}, \dot{q}_{k+2}, \ldots, \dot{q}_m \) if \( \dot{q}_k > \dot{q}_{k+1} > \dot{q}_{k+2} > \cdots > \dot{q}_m \), \( \text{Imp}_n(\dot{x}_j, \varphi, U \left( f, \dot{x}_j [\mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m], S \right)) \) does not specify \( \dot{q}_{k+1}, \dot{q}_{k+2}, \ldots, \dot{q}_m \).

Part (b) of the definition of \( \text{Imp}_n \) requires that \( U(\dot{x}_j, \varphi, \mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m, S) \) be defined. The issue is that this function is only defined when

\[ \left[ \Gamma \left( \text{Func}(\varphi, x)(\dot{x}_j, \varphi, \mathcal{I}_0, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_m) \right), S \right] = 1, \tag{3B.14} \]

but that it will be more convenient to have the \( \text{Imp}_n \) function defined already when we show that (3B.13) holds.
Example 30.37:

Consider the conjunction specification

\[ \hat{A} = \hat{a}(q_1, q_2, \ldots, q_n) = \hat{a}_{\mathcal{F}_1}(\hat{a}_{\mathcal{F}_1}(q_3, q_2), q_1) \]

from example 30.34. We will designate the elements of the conjunction specification as follows:

\[ i_1 \equiv q_1, \]
\[ i_2 \equiv \hat{a}_{\mathcal{F}_1}(\hat{a}_{\mathcal{F}_1}(q_2, q_3), q_2), \]
\[ i_3 \equiv q_2, \]
\[ i_4 \equiv \hat{a}_{\mathcal{F}_1}(q_4, q_3), \]
\[ i_5 \equiv q_3, \]
\[ i_6 \equiv q_4. \]

Since we showed in example 30.35 that \( q_3 \triangleright q_2 \). Hence we can say that for any \( \mathcal{F}_1 \in \Delta \left( \text{Func}(\mathcal{F}, A_1)(q_1), S \right) \):

\[ \text{Inp}_2 \left( \hat{A}, (\mathcal{F}_1), S \right) = \]
\[ \text{Inp}_1 \left( i_2(), U(f, i_1, (q_1), S) \right) = \]
\[ \text{Inp}_2 \left( i_2(), U(f, q_1, S) \right) = \]
\[ U(f, q_1, S). \]

The first equality comes from part (b) of the definition of Inp, the second follows from the definition of \( i_1(\ldots) \), and the third comes from part (a) of the definition of Inp.

For the input to \( q_3 \) we have:

\[ \forall \left( \mathcal{F}_1 \in \Delta \left( \text{Func}(\mathcal{F}, A_2)(q_1), S \right), \mathcal{F}_2 \in \Delta \left( \text{Func}(\mathcal{F}, A)(q_2) \right) \right) \text{Inp}_3 \left( \hat{A}, (\mathcal{F}_1, \mathcal{F}_2), S \right) \]
\[ = \text{Inp}_3 \left( i_5(), U(f_1, i_1[\mathcal{F}_1], S) \right) \text{ By part (b)} \]
\[ = \text{Inp}_3 \left( i_4(), U(f_2, i_5[\mathcal{F}_2], U(f_1, i_1[\mathcal{F}_1], S)) \right) \text{ By part (b)} \]
\[ = U(f_1, i_1[\mathcal{F}_1], S) \text{ By part (a)} \]

We see that in these two examples the formal definition of Inp conforms to our intuitive description of what Inp should represent, that is, the list of inputs seen by \( q_3 \) in the first case and \( q_2 \) in the second case.
3B.3. Proof of proposition 3.11.

Theorem 3B.38: Let \( \mathcal{G} \) be a set of function classes, and let \( \mathcal{Y} \) be a set \( \{q_1, q_2, \ldots, q_n\} \). Assume that \( \hat{k} \) is a composition specification with respect to \( \mathcal{G} \) and \( \mathcal{D} \) and that \( \mathcal{D}(\hat{k}) = \{q_0, q_1, \ldots, q_n\} \). Let \( S \in \text{Dom}(\mathcal{D}) \), \( l \in I \).

Then

\[
\left\{ \begin{array}{l}
\mathcal{I}_1 \in \Delta (\text{Func}(\mathcal{G}, \mathcal{A})(q_1), \text{Inp}_a(\hat{k}_1, (\hat{k}_1), S)) \\
\mathcal{I}_{l+1} \in \Delta (\text{Func}(\mathcal{G}, \mathcal{A})(q_{l+1}), \text{Inp}_a(\hat{k}_1, (S), S)) \\
\vdots \\
\mathcal{I}_m \in \Delta (\text{Func}(\mathcal{G}, \mathcal{A})(q_m), \text{Inp}_m (\hat{k}_1, (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{m-1}), S)) \\
\end{array} \right.
\]

\[
|\Delta (\text{Func}(\mathcal{G}, \mathcal{A})(\hat{k}_1 [\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m]))| = 1.
\]

Proof:

1. \( \forall (i_1 : |\mathcal{D}(\hat{k}_1)| = 1) : \)

\[
\forall (i_1 \in \Delta (\text{Func}(\mathcal{G}, \mathcal{A})(q_{l+1}), \text{Inp}_a(\hat{k}_1, (), S))) : \\
|\Delta (\text{Func}(\mathcal{G}, \mathcal{A})(\hat{k}_1 [\mathcal{I}_1]))| = 1 \text{ where } \mathcal{D}(\hat{k}_1) = \{q_0\}
\]

Proof of 1:

1.1) Let \( i_k \) be such that

\[
|\mathcal{D}(i_k)| = 1 \\
\text{[Definition:]} \\
\]

1.2) Let \( \mathcal{D}(i_k) = \{q_0\} \)

[Definition:]

1.3) \( \forall (i_k \in \Delta (\text{Func}(\mathcal{G}, \mathcal{A})(i_k), S)) : \\
|\Delta (i_k [\mathcal{I}_1]), S| = 1 \)

[By the definition of \( \Delta \).]

1.4) \( \forall (i_k \in \Delta (\text{Func}(\mathcal{G}, \mathcal{A})(i_k), S)) : \\
|\Delta (i_k [\mathcal{I}_k]), S| = 1 \)

[Since \( \mathcal{D}(i_k) = \{q_0\}, i_k = q_0 \).]
(1.5) 
\[ \forall ( A \in \Delta ( \text{Func}(g, \mathfrak{A})(q), \text{Inp}_{\mathcal{A}}(i_0, \emptyset, S))) : \]

[By the definition of \text{Inp}, \text{Inp}_{\mathcal{A}}(i_0, \emptyset, S) = S.]

Q.E.D. (1)

(2) For all \( n > 0 \), if

\[ \forall ( S \in \text{Dom}(\mathcal{G})^n : \forall (i_j ; |\mathcal{L}(i_j)| < n)) : \]

\[ \left( A_0 \in \Delta ( \text{Func}(g, \mathfrak{A})(q), \text{Inp}_{\mathcal{A}}(i_j, \emptyset, S)) , \right. \]

\[ \forall (A_{n+1} \in \Delta ( \text{Func}(g, \mathfrak{A})(q), \text{Inp}_{\mathcal{A}}(i_j, (A_0, A_{n+1}, \ldots, A_{n-1}), S) )) \]

\[ |\Delta (\text{Func}(g, \mathfrak{A})(i_j [A_0, A_{n+1}, \ldots, A_l]))| = 1 \]

where \( \mathcal{L}(i_j) = \{q_0, q_{n+1}, \ldots, q_l\} \)

then

\[ \forall ( S \in \text{Dom}(\mathcal{G})^n : \forall (i_j ; |\mathcal{L}(i_j)| = n)) : \]

\[ \left( A_0 \in \Delta ( \text{Func}(g, \mathfrak{A})(q), \text{Inp}_{\mathcal{A}}(i_j, \emptyset, S)) , \right. \]

\[ \forall (A_{n+1} \in \Delta ( \text{Func}(g, \mathfrak{A})(q), \text{Inp}_{\mathcal{A}}(i_j, (A_0, A_{n+1}, \ldots, A_{n-1}), S) )) \]

\[ |\Delta (\text{Func}(g, \mathfrak{A})(i_j [A_0, A_{n+1}, \ldots, A_l]))| = 1 \]

where \( \mathcal{L}(i_j) = \{q_0, q_{n+1}, \ldots, q_l\} \)
Proof of 2:

[2.1] \( \forall (\mathcal{S} \in \text{Dom}(\mathcal{D})) \forall (i_j : |\mathcal{D}(i_j)| < \mathbb{N}) : \)

\[
\left\{ \begin{array}{l}
\mathcal{I}_0 \in \Delta (\text{Func}(\mathfrak{g}, \mathfrak{g}) (\hat{\mathcal{I}), \text{Inp}_0 (\hat{\mathcal{I}}, \emptyset), S)) \\
\mathcal{I}_{a+1} \in \Delta (\text{Func}(\mathfrak{g}, \mathfrak{g}) (\hat{\mathcal{I}}_{a+1}), \text{Inp}_{a+1} (\hat{\mathcal{I}}, (\mathcal{I}_a), S)) \\
\vdots \\
\mathcal{I}_1 \in \Delta (\text{Func}(\mathfrak{g}, \mathfrak{g}) (\hat{\mathcal{I}}_1), \text{Inp}_1 (\hat{\mathcal{I}}, (\mathcal{I}_0, \mathcal{I}_{a+1}, \ldots, \mathcal{I}_{a-1}), S)) \\
\end{array} \right.
\]

\( |\Delta (\text{Func}(\mathfrak{g}, \mathfrak{g}) (\hat{\mathcal{I}}, (\mathcal{I}_0, \mathcal{I}_{a+1}, \ldots, \mathcal{I}_1), S))| = 1 \)

where \( \mathcal{D}(\hat{\mathcal{I}}) = \{\hat{\mathcal{I}}, \hat{\mathcal{I}}_{a+1}, \ldots, \hat{\mathcal{I}}_1\} \)

[ Assumption. ]

[2.2] Let \( \hat{s}_k \) be such that

\( |\mathcal{D}(\hat{s}_k)| = \mathbb{N} \)

[ Definition. ]

[2.3] Let \( \mathcal{D}(\hat{s}_k) = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_m\} \)

[ Definition. ]

[2.4] \( \exists (f, \hat{s}_i, \hat{s}_j) : \)

\( \hat{s}_k = \mathcal{D}_f (\hat{s}_i, \hat{s}_j) \)

[ This must be the case since \(|\mathcal{D}(\hat{s}_k)| > 1\) and \( \hat{s}_k \) is a composition specification. ]

[2.5] Let \( f, \hat{s}_i, \hat{s}_j \) be such that

\( \hat{s}_k = \mathcal{D}_f (\hat{s}_i, \hat{s}_j) \)

[ Such \( f, \hat{s}_i, \hat{s}_j \) exist by [2.4]. ]

[2.6] Let \( \mathcal{D}(\hat{s}_j) = \{\hat{s}_0, \hat{s}_{a+1}, \ldots, \hat{s}_m\} \).

\( \mathcal{D}(\hat{s}_i) = \{\hat{s}_1, \hat{s}_{i+1}, \ldots, \hat{s}_i\} \)

[ Definition. ]

[2.7] \( \mathcal{D}(\hat{s}_k) = \mathcal{D}(\hat{s}_i) \cup \mathcal{D}(\hat{s}_j) \)

[ Because \( \hat{s}_k \) is \( \mathcal{D}_f (\hat{s}_i, \hat{s}_j) \). ]
Both $\mathcal{D}(\hat{i}_1)$ and $\mathcal{D}(\hat{i}_j)$ must contain at least one element of $\mathcal{D}$, because the recursive definition of connection specifications is only anchored at members of $\mathcal{D}$. The claim therefore follows from the fact that $|\mathcal{D}(\hat{i}_j)| = n$, and [2.7].

$$|\mathcal{D}(\hat{i}_1)| < n;$$

$$|\mathcal{D}(\hat{i}_j)| < n$$

$$|\mathcal{D}(\hat{i})| < n;$$

$$|\mathcal{D}(\hat{i}_j)| < n$$

$$\mathcal{D}(\hat{i}_j) \in \Delta \{\text{Func}(g, \mathcal{A})(\hat{i}_j), \text{Imp}_{\Delta}(\hat{i}_j, (), S)\}.$$ 

$$\mathcal{D}(\hat{i}_{j+1}) \in \Delta \{\text{Func}(g, \mathcal{A})(\hat{i}_{j+1}), \text{Imp}_{\Delta}(\hat{i}_{j+1}, (A), S)\}.$$ 

$$\forall m, m \in \Delta \{\text{Func}(g, \mathcal{A})(\hat{i}_m), \text{Imp}_{\Delta}(\hat{i}_m, (A, A_{a+1}, \ldots, A_{m-1}), S)\}.$$ 

$$|\Delta \{\text{Func}(g, \mathcal{A})(\hat{i}_j [A, A_{a+1}, \ldots, A_m]), S\}| = 1.$$ 

[By (2.11).]

$$U(f, \text{Func}(g, \mathcal{A})(\hat{i}_j [A, A_{a+1}, \ldots, A_m]), S)$$

is defined.

[By (2.9) and the definition of $U(\ldots)$.]

Recall that $U(f, \mathcal{A}, S)$ is only defined if $|\Delta(\mathcal{A}, S)| = 1$.]

Let $\mathcal{F}_j \equiv \text{Func}(g, \mathcal{A})(\hat{i}_j [A, A_{a+1}, \ldots, A_m])$ [Definition.]
\[
\begin{align*}
\forall \mathcal{A}_a \in \Delta \{\text{Func}(\omega, \mathcal{A})(\mathcal{A}_a), \text{Imp}_{\mathcal{A}}(i_j, \emptyset, S)\}, \\
\mathcal{A}_{a+1} \in \Delta \{\text{Func}(\omega, \mathcal{A})(\mathcal{A}_{a+1}), \text{Imp}_{\mathcal{A}+1}(i_j, (\mathcal{A}_a), S)\}, \\
\vdots \\
\mathcal{A}_m \in \Delta \{\text{Func}(\omega, \mathcal{A})(\mathcal{A}_m), \text{Imp}_m(i_j, (\mathcal{A}_a, \mathcal{A}_{a+1}, \ldots, \mathcal{A}_{m-1}), S)\}
\end{align*}
\]

\[
\begin{align*}
\forall \mathcal{A}_i \in \Delta \left( \begin{array}{c}
\text{Func}(\omega, \mathcal{A})(\mathcal{A}_i), \\
\text{Imp}_i(i_j, \emptyset, U(f, \mathcal{A}_j, S))
\end{array} \right) \\
\mathcal{A}_{i+1} \in \Delta \left( \begin{array}{c}
\text{Func}(\omega, \mathcal{A})(\mathcal{A}_{i+1}), \\
\text{Imp}_{i+1}(i_j, (\mathcal{A}_i), U(f, \mathcal{A}_j, S))
\end{array} \right) \\
\vdots \\
\mathcal{A}_l \in \Delta \left( \begin{array}{c}
\text{Func}(\omega, \mathcal{A})(\mathcal{A}_l), \\
\text{Imp}_l(i_j, (\mathcal{A}_a, \mathcal{A}_{a+1}, \ldots, \mathcal{A}_{l-1}), U(f, \mathcal{A}_j, S))
\end{array} \right)
\end{align*}
\]

\[|\Delta \{\text{Func}(\omega, \mathcal{A})(\mathcal{A}_a, \mathcal{A}_{a+1}, \ldots, \mathcal{A}_m), U(f, \mathcal{A}_j, S)\}| = 1\]

[By (2.11) and (2.1).]
\[ [2.13] \]
\[
\begin{array}{c}
\mathcal{S}_a \in \Delta \{ \text{Func}(\mathcal{F}, \mathcal{A})(\mathcal{A}_a), \text{Inp}_a(\mathcal{i}_a, (\mathcal{F}_a, S)) \}, \\
\mathcal{S}_{a+1} \in \Delta \{ \text{Func}(\mathcal{F}, \mathcal{A})(\mathcal{A}_{a+1}), \text{Inp}_{a+1}(\mathcal{i}_{a+1}, (\mathcal{F}_a, S)) \}, \\
\mathcal{S}_m \in \Delta \{ \text{Func}(\mathcal{F}, \mathcal{A})(\mathcal{A}_m), \text{Inp}_m(\mathcal{i}_m, (\mathcal{F}_a, \mathcal{F}_{a+1}, \ldots, \mathcal{F}_{m-1}, S)) \}
\end{array}
\]
\[
\forall \mathcal{S}_1 \in \Delta \left( \text{Inp}_1(\mathcal{i}_1, (\mathcal{F}_a, \mathcal{F}_{a+1}, \ldots, \mathcal{F}_m, S)) \right) \\
\Delta \{ \text{Func}(\mathcal{F}, \mathcal{A})(\mathcal{A}_1), U(\mathcal{F}_1, S) \} = 1
\]

[ By [2.12] and the definition of Inp. The expressions of the form
\[
\text{Inp}(\mathcal{i}_i, (\mathcal{F}_i, \ldots), U(\mathcal{F}_j, S))
\]
were changed to expressions of the form

\[
\text{Inp}(\mathcal{i}_i, (\mathcal{F}_a, \mathcal{F}_{a+1}, \ldots, \mathcal{F}_m, S))
\]

by appealing to part (b) of the definition of Inp. ]

\[ [2.14] \]
\[
\{ \mathcal{A}_a, \mathcal{A}_{a+1}, \ldots, \mathcal{A}_m, \mathcal{A}_{m+1}, \ldots, \mathcal{A}_n \} = \{ \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \}
\]

[ By [2.13], [2.3], and [2.6]. ]

\[ [2.15] \]
Let \( \mathcal{A}_a = \mathcal{A}_1, \mathcal{A}_{a+1} = \mathcal{A}_2, \ldots, \mathcal{A}_n = \mathcal{A}_n \)

[ Since \( \mathcal{A}_m > \mathcal{A}_i \) by the definition of >, this definition is consistent with both [2.14] and the convention that \( \mathcal{A}_1 > \mathcal{A}_2 > \ldots > \mathcal{A}_n \). ]
\[\begin{align*}
\forall \mathcal{I}_n \in \mathcal{J}_n (\mathcal{F} \cup \mathcal{F}_n \cup \mathcal{F}_{n+1}, \mathcal{F}_{n+1}) = 1 & \quad \text{[By (2.16) and the definition of } U(\cdot, \cdot) \text{.]} \\
\end{align*}\]

Lemma 3B.39: Let $\mathcal{K}_0$ and $\mathcal{K}_1$ be two function classes and let $S \in \text{Dom}(\mathcal{K}_i^l), i \in [1]$. Assume that

$$|\Delta(\mathcal{K}_i, S)| = 1.$$ 

Then

$$\Delta(\mathcal{F}(\mathcal{K}_0, \mathcal{K}_1), S) = \bigcup_{\mathcal{K} \in \Delta(\mathcal{K}_0 \cup (\mathcal{K}_1, S))} \Delta(\mathcal{F}(\mathcal{K}_0, \mathcal{K}_1), S).$$

Proof:

[1] $\forall (\mathcal{K}_1, \mathcal{K}_2 : |\Delta(\mathcal{K}_2, S)| = 1)$:

$$\Delta(\mathcal{F}(\mathcal{K}_1, \mathcal{K}_2), S) = \left\{ \mathcal{F}(H_1, H_2) : \mathcal{F}(H_1, H_2) \cap S = \emptyset, H_1 \in \mathcal{K}_1, H_2 \in \mathcal{K}_2 \right\} \cup \left\{ \mathcal{F}(H_1, H_2) \cap S : H_1 \in \mathcal{K}_1, H_2 \in \mathcal{K}_2 \right\}$$
Proof of 1:

[1.1] \[ \{ g_f(H_1, H_2) \cap S : g_f(H_1, H_2) \in g_f(\mathcal{M}_1, \mathcal{M}_2) \} = \{ g_f(H_1, H_2) \cap S : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} \]

[ By lemma 3B.18. ]

[1.2] \[ \{ g_f(H_1, H_2) \cap S : g_f(H_1, H_2) \in g_f(\mathcal{M}_1, \mathcal{M}_2) \} = \{ H_1 \cap U(f, H_2, S) : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} \]

[ By [1.1] and the definition of \( U(\cdot, \cdot, \cdot) \) ]

[1.3] \[ \{ g_f(H_1, H_2) \cap S : g_f(H_1, H_2) \in g_f(\mathcal{M}_1, \mathcal{M}_2) \} = \bigcup_{S \in \mathcal{S}(\mathcal{M}_1, \mathcal{M}_2)} \{ H_1 \cap U(f, H_2, S) : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} \]

[ By [1.2] and lemma 3B.28. ]

[1.4] \[ \{ g_f(H_1, H_2) \cap S : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} = \bigcup_{S \in \mathcal{S}(\mathcal{M}_1, \mathcal{M}_2)} \{ H_1 \cap U(f, H_2, S) : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} \]

[ By [1.3] and lemma 3B.18. ]

[1.5] \[ \left\{ \{ g_f(H_1, H_2) : g_f(H_1, H_2) \cap S = v, H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} : v \in \bigcup_{S \in \mathcal{S}(\mathcal{M}_1, \mathcal{M}_2)} \{ H_1 \cap U(f, H_2, S) : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} \right\} = \left\{ \{ g_f(H_1, H_2) : g_f(H_1, H_2) \cap S = v, H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} : v \in \bigcup_{S \in \mathcal{S}(\mathcal{M}_1, \mathcal{M}_2)} \{ H_1 \cap U(f, H_2, S) : H_1 \in \mathcal{M}_1, H_2 \in \mathcal{M}_2 \} \right\} \]

[ By [1.4]. ]

Q.E.D. (1)
164

\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \Delta (\mathcal{G}(\mathcal{K}_0, \mathcal{K}_1), S) = \]
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \left\{ \mathcal{G}(H_1, H_2) : \mathcal{G}(H_1, H_2) \cap S = v, H_1 \in \mathcal{K}_0, H_2 \in \mathcal{K}_1 \right\} \]
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \left\{ v \in \mathcal{G}(H_1, H_2) \cap S : H_1 \in \mathcal{K}_2, H_2 \in \mathcal{K}_1 \right\} \]
[ By [1]. ]

13
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \Delta (\mathcal{G}(\mathcal{K}_0, \mathcal{K}_1), S) = \]
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \left\{ \mathcal{G}(H_1, H_2) : \mathcal{G}(H_1, H_2) \cap S = v, H_1 \in \mathcal{K}_0, H_2 \in \mathcal{K}_1 \right\} \]
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \left\{ v \in \mathcal{G}(H_1, H_2) \cap S : H_1 \in \mathcal{K}_2, H_2 \in \mathcal{K}_1 \right\} \]
[ By (2) and the properties of sets. ]

4
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \Delta (\mathcal{G}(\mathcal{K}_0, \mathcal{K}_1), S) = \]
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \left\{ \mathcal{G}(H_1, H_2) : \mathcal{G}(H_1, H_2) \cap S = v, H_1 \in \mathcal{K}_0, H_2 \in \mathcal{K}_1 \right\} \]
[ By [3] and lemma 3B.28. ]

5
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \Delta (\mathcal{G}(\mathcal{K}_0, \mathcal{K}_1), S) = \]
\[ \left\{ \mathcal{G}(H_1, H_2) : \mathcal{G}(H_1, H_2) \cap S = v, H_1 \in \mathcal{K}_0, H_2 \in \mathcal{K}_1 \right\} \]
\[ v \in \mathcal{G}(H_1, H_2) \cap S : H_1 \in \mathcal{K}_0, H_2 \in \mathcal{K}_1 \}
[ By lemma 3B.28. ]

6
\[ \bigcup_{A \in \mathcal{A}(\mathcal{U}(J, n, S))} \Delta (\mathcal{G}(\mathcal{K}_0, \mathcal{K}_1), S) = \]
\[ \left\{ \mathcal{G}(H_1, H_2) : \mathcal{G}(H_1, H_2) \cap S = v, H_1 \in \mathcal{K}_0, H_2 \in \mathcal{K}_1 \right\} : v \in \Gamma (\mathcal{G}(\mathcal{K}_0, \mathcal{K}_1)) \}
[ By [5] and the definition of \( \Gamma \). ]
Lemma 3B.40: Let \( \mathcal{X} \) be a hypothesis class and let \( S \in \text{Dom}(\mathcal{X}) \), \( t \in \mathbb{N} \). Then

\[
\Delta(\mathcal{X},S) = \bigcup_{\mathcal{F} \in \Delta(\mathcal{X},S)} \Delta(\mathcal{F},S).
\]

Proof:

1. \[
\Delta(\mathcal{X},S) = \bigcup_{\mathcal{F} \in \Delta(\mathcal{X},S)} \mathcal{F}
\] [By the properties of sets.]

2. \[
\forall \mathcal{F} \in \Delta(\mathcal{X},S) : \mathcal{F} = \Delta(\mathcal{F},S)
\] [This follows from the definition of \( \Delta \).]

3. \[
\Delta(\mathcal{X},S) = \bigcup_{\mathcal{F} \in \Delta(\mathcal{X},S)} \Delta(\mathcal{F},S)
\] [By [1] and [2].]

Q.E.D. □

For the next proof it will be convenient to extend the definition of a conjunction specification somewhat:

Definition: Let \( \mathcal{D} \) be a set of function classes, let \( \mathcal{L} \) be a set, and let \( W \) be function mapping \( \mathcal{L} \) to \( \mathcal{D} \). If \( \mathcal{X} \subseteq W(\mathcal{L}) \) for some \( \mathcal{L} \in \mathcal{L} \), then we say that \( \mathcal{X} \) is a conjunction specification with respect to \( \mathcal{D} \) and \( \mathcal{L} \), and the

\[
\text{Func}(\mathcal{L},\mathcal{D})(\mathcal{X}) = \mathcal{X}.
\]
Theorem 3B.41: Let $\mathcal{G}$ be a set of function classes, and let $\mathcal{A}$ be a set $\{q_1, q_2, \ldots, q_n\}$. Assume that $\delta_i$ is a composition specification with respect to $\mathcal{G}$ and $\mathcal{A}$ and that $\mathcal{A}(\delta_i) = \{q_0, q_{i+1}, \ldots, q_m\}$. Let $S \in \text{Dom}(\mathcal{G})^I, I \in H$. Then

$$\Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_i), S) = \bigcup_{s_n \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_{i+n}), S)} \bigcup_{s_{n+1} \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_{i+n+1}), b_{p_{n+1}}(s_{n}, \delta_i), S))} \ldots \bigcup_{s_m \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_{i+m}), b_{p_m}(s_{m-1}, \delta_{i+m-1}), S))} \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_i [s_n, s_{n+1}, \ldots, s_m]), S)$$

Proof:

[The proof is by induction on the number of elements in $\mathcal{A}$.]

[1] $\forall (i_h : \mathcal{A}(\delta_h) = \{q_1\}, |\mathcal{A}(\delta_h)| = 1)$:

$$\Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_h), S) \bigcup_{s_n \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_{h+n}), b_{p_{n+1}}(s_{n}, \delta_i), S))} \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_h [s_n], S))$$

Proof of [1]:

[1.1] $\Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_h), S) = \bigcup_{s_n \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_{h+n}), b_{p_{n+1}}(s_{n}, \delta_i), S))} \Delta(\Delta(\delta_h), S))$ [By lemma 3B.40.]

[1.2] $\delta_h [s_1] = \delta_i$

[By the definition of $\delta_h [\cdot]$, since $\mathcal{A}(\delta_h) = \{q_1\}$.]

[1.3] $\Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_h), S) = \bigcup_{s_n \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A})(\delta_{h+n}), b_{p_{n+1}}(s_{n}, \delta_i), S))} \Delta(\Delta(\delta_h [s_n], S))$ [By [1.1] and [1.2].]
\[ \Delta(\text{Func}(g, a)(i_h), S) = \bigcup_{\mathcal{K} \in \Delta(\text{Func}(g, a)(i)), S} \Delta(i_h [\mathcal{K}], \text{Imp}(i_h, {\cdot}, S)) \]

[ Since \(\text{Imp}(i_h, {\cdot}, S) = S\) by the definition of \(\text{Imp}\). ]

Q.E.D. (1)

[2] If \(\forall (i_h : \mathcal{D}(i_h) = \{g_1, g_2, \cdots, g_m\}, |\mathcal{D}(i_h)| = n - 1, i > 1)\):

\[ \Delta(\text{Func}(g, a)(i_h), S) = \bigcup_{\mathcal{K} \in \Delta(\text{Func}(g, a)(i)), S} \bigcup_{\mathcal{M} \in \Delta(\text{Func}(g, a)(i), \text{Imp}(i, {\cdot}, S))} \Delta(i_h [\mathcal{K}, \mathcal{M}], S) \]

then \(\forall (i_h : \mathcal{D}(i_h) = \{g_1, g_2, \cdots, g_i\}, |\mathcal{D}(i_h)| = n, i > 1)\):

\[ \Delta(\text{Func}(g, a)(i_h), S) = \bigcup_{\mathcal{K} \in \Delta(\text{Func}(g, a)(i)), S} \bigcup_{\mathcal{M} \in \Delta(\text{Func}(g, a)(i), \text{Imp}(i, {\cdot}, S))} \Delta(i_h [\mathcal{K}, \mathcal{M}], S) \]

Proof of 2:

[2.1] \(\forall (i_h : \mathcal{D}(i_h) = \{g_1, g_2, \cdots, g_m\}, |\mathcal{D}(i_h)| = n - 1)\):

\[ \Delta(\text{Func}(g, a)(i_h), S) = \bigcup_{\mathcal{K} \in \Delta(\text{Func}(g, a)(i)), S} \bigcup_{\mathcal{M} \in \Delta(\text{Func}(g, a)(i), \text{Imp}(i, {\cdot}, S))} \Delta(i_h [\mathcal{K}, \mathcal{M}], S) \]

[ Assumption. ]
[2.2] Let \( \delta_b \) be such that

\[
\mathcal{D}(\delta_b) = \{ \hat{q}_1, \hat{q}_2, \ldots, \hat{q}_n \}, |\mathcal{D}(\delta_b)| = n
\]

[Definition.

[2.3] If \( \delta_b \) were not composite then \( |\mathcal{D}(\delta_b)| \)

would equal 1.

[2.4] Let \( \mathcal{G}_f, \delta_i, \delta_j \) be such that

\[
\delta_b = \mathcal{G}_f(\delta_i, \delta_j)
\]

[Such \( \mathcal{G}_f, \delta_i, \delta_j \) exist by [2.3].

[2.5] \[
\Delta \left( \text{Func}(\mathcal{G}, \mathcal{A}), \left( \mathcal{G}_f(\delta_i, \delta_j) \right), S \right) =
\]

\[
\bigcup_{A_b \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A}), (\delta_i, \delta_j), S)} \Delta \left( \text{Func}(\mathcal{G}, \mathcal{A}), \left( \mathcal{G}_f(\delta_i, \mathcal{A}_b) \right), S \right)
\]

[By theorem 3B.33.

[2.6] Let \( \{ \hat{q}_0, \hat{q}_{n+1}, \ldots, \hat{q}_j \} = \mathcal{D}(\delta_j) \)

[Definition.

[2.7] \[
\Delta(\text{Func}(\mathcal{G}, \mathcal{A}), (\delta_j), S) =
\]

\[
\bigcup \hspace{1cm} \bigcup_{A_b \in \Delta(\text{Func}(\mathcal{G}, \mathcal{A}), (\delta_j), S)} \bigcup \Delta \left( \text{Func}(\mathcal{G}, \mathcal{A}), \left( \delta_j, \mathcal{A}_b \right), S \right)
\]

[By [2.1] and [2.6].]
\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( \hat{G}_f(i, i) \right), S \right) = \]

\[ \bigcup_{i \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i, S \right) \right)} \]

\[ \bigcup_{i, j \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i, S \right) \right), \text{by} \left( i, \left( \lambda_i, \alpha \right) \right)} \]

\[ \ldots \]

\[ \bigcup_{i, j \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i, S \right) \right), \text{by} \left( i, \left( \lambda_i, \alpha \right) \right)} \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( \hat{G}_f(i, i), S \right) \right) \]

\[ \text{[By \([2.5]\) and \([2.7]\).]} \]

\[ \forall (\lambda_1 \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i_1 \left[ \lambda_1 \right], \alpha_1 \right), S \right)): \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( \hat{G}_f(i_1, \lambda_1), S \right) \right) = \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( \hat{G}_f(i_1, \lambda_1), \alpha_1 \right), S \right) \]

\[ \text{[By the definition of } \cup(\ldots)\].} \]

\[ \forall (\lambda_1 \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i_1 \left[ \lambda_1 \right], \alpha_1 \right), S \right)): \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( \hat{G}_f(i_1, \lambda_1), S \right) \right) = \]

\[ \bigcup_{i, j \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i, S \right) \right), \text{by} \left( i, \left( \lambda_i, \alpha \right) \right)} \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( \hat{G}_f(i, \lambda), S \right) \right) \]

\[ \text{[By \([2.9]\) and lemma \([3B.39]\).]} \]

Let \( \mathcal{D} (i) = \{ q_1, q_{i+1}, \ldots, q_m \} \) \[ \text{[Definition.]} \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i, S \right) \right) = \]

\[ \bigcup_{i \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( i, S \right) \right)} \]

\[ \bigcup_{i, j \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( j, S \right) \right), \text{by} \left( j, \left( \lambda_i, \alpha \right) \right)} \]

\[ \ldots \]

\[ \bigcup_{i, j \in \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( j, S \right) \right), \text{by} \left( j, \left( \lambda_i, \alpha \right) \right)} \]

\[ \Delta \left( \text{Func}_{\langle \varphi, \alpha \rangle} \left( j, \left[ \alpha_j \right], \alpha_{j+1}, \ldots, \alpha_m \right), S \right) \]

\[ \text{[By \([2.1]\) and \([2.11]\).]} \]
\[ \forall (\mathcal{A} \in \text{Func}(\varrho, \varepsilon)(i_j)):
\Delta \left( \text{Func}(\varrho, \varepsilon)(\hat{\varphi}_j(i_0, \mathcal{A}_0)), S \right) = \\
\bigcup_{\mathcal{A} \in \Delta \left( \text{Func}(\varrho, \varepsilon)(i_0), U(f, \mathcal{M}, S) \right)} \\
\bigcup_{\mathcal{A}_1 \in \Delta \left( \text{Func}(\varrho, \varepsilon)(i_1), U(f, \mathcal{M}, S) \right)} \\
\ldots \\
\bigcup_{\mathcal{A}_m \in \Delta \left( \text{Func}(\varrho, \varepsilon)(i_m), U(f, \mathcal{M}, S) \right)} \\
\bigcup_{\mathcal{A} \in \Delta \left( \text{Func}(\varrho, \varepsilon)(\hat{\varphi}_j(i_0, \mathcal{A}_0)), S \right)} \]

\[ \Delta \left( \text{Func}(\varrho, \varepsilon)(\hat{\varphi}_j(i_1, \mathcal{A}_1)), S \right) = \\
\bigcup_{\mathcal{A} \in \Delta \left( \text{Func}(\varrho, \varepsilon)(i_0), U(f, \mathcal{M}, S) \right)} \\
\bigcup_{\mathcal{A}_1 \in \Delta \left( \text{Func}(\varrho, \varepsilon)(i_1), U(f, \mathcal{M}, S) \right)} \\
\ldots \\
\bigcup_{\mathcal{A}_m \in \Delta \left( \text{Func}(\varrho, \varepsilon)(i_m), U(f, \mathcal{M}, S) \right)} \\
\bigcup_{\mathcal{A} \in \Delta \left( \text{Func}(\varrho, \varepsilon)(\hat{\varphi}_j(i_1, \mathcal{A}_1)), S \right)} \]

[By [2.8] and [2.13].]
\[ \forall \langle \mathcal{A} \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_0 \left[ \mathcal{K}_0, \mathcal{K}_{a+1}, \ldots, \mathcal{K}_m \right] \right), S \right) : \]

\[ \bigcup_{\mathcal{F}_0 \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_1 \left[ \mathcal{K}_0, \mathcal{K}_{a+1}, \ldots, \mathcal{K}_m \right] \right), S \right)} \]

\[ \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_1 \left[ \mathcal{K}_0 \right] \right), S \right) = \bigcup_{\mathcal{F}_1 \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_1 \left[ \mathcal{K}_0, \mathcal{K}_{a+1}, \ldots, \mathcal{K}_m \right] \right), S \right)} \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_1 \left[ \mathcal{K}_0 \right] \right), S \right) \]

[By the definition of \( U(\cdot, \cdot, \cdot) \) ]

\[ \forall \langle \mathcal{A} \in \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \right) : \]

\[ \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_0, \mathcal{K}_{a+1}, \ldots, \mathcal{K}_m \right] \right), U(f, K, S) \right)} \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_0, \mathcal{K}_{a+1}, \ldots, \mathcal{K}_m \right] \right), S \right) \]

[By 2.15 and lemma 3B.39.]

\[ \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j \right] \right), S \right) = \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j \right] \right), S \right)} \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j+1 \right] \right), S \right)} \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j+2 \right] \right), S \right)} \ldots \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_m \right] \right), S \right)} \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i+1, \mathcal{K}_j \right] \right), S \right)} \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i+1, \mathcal{K}_j \right] \right), U(f, K, S) \right)} \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j, \mathcal{K}_m \right] \right), S \right) \]

[By 2.14 and 2.16.]

\[ \forall \langle \mathcal{A} \in \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \right) : \]

\[ \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j \right] \right), S \right)} \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j \right] \right), S \right)} \ldots \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j \right] \right), S \right)} \bigcup_{\mathcal{F}_j \in \Delta \left( \text{Func}(\varrho, \varphi) \left( \mathcal{F}_j \left[ \mathcal{K}_i, \mathcal{K}_j \right] \right), S \right)} \]
\[ \Delta \left( \text{Func}(\mathcal{A}, \mathcal{B}) \left( \bar{g}(\bar{x}, \bar{y}) \right), S \right) = \bigcup_{\mathcal{A} \in \Delta(\text{Func}(\mathcal{A}, \mathcal{B}) \left( \bar{g}(\bar{x}, \bar{y}) \right))} \bigcup_{\mathcal{A} \in \Delta(\text{Func}(\mathcal{A}, \mathcal{B}) \left( \bar{g}(\bar{x}, \bar{y}) \right))} \Delta \left( \text{Func}(\mathcal{A}, \mathcal{B}) \left( \bar{g}(\bar{x}, \bar{y}) \right), S \right) \]

[By (2.17) and the definition of Inp. 1]
\[ \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right) = \bigcup_{i \in \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right)} \bigcup_{i \in \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right)} \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right) \]

[By (2.18) we have moved the union over \( \mathcal{I}_0 \) past the unions appearing to the right of it in (2.18), since those unions do not depend on \( \mathcal{I}_0 \).]

\[ \bigcup_{i \in \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right)} \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right) = \Delta \left( \text{Func}(g, g) \left( \hat{g}_f(v, v') \right), S \right) \]

[By theorem 3B.33.]
\[ \Delta \left( \text{Func}(g, g) \left( \hat{f}_j(i_i, i_j) \right), S \right) = \bigcup_{x_i \in \Delta(\text{Func}(g, g)(i_i), S)} \bigcup_{x_j \in \Delta(\text{Func}(g, g)(i_j), S)} \bigcup_{x_k \in \Delta(\text{Func}(g, g)(i_k), S)} \bigcup_{x_{k+1} \in \Delta(\text{Func}(g, g)(i_{k+1}), S)} \bigcup_{x_{m} \in \Delta(\text{Func}(g, g)(i_m), S)} \Delta \left( \text{Func}(g, g) \left( \hat{f}_j(i_i, x_k, x_{k+1}, \ldots, x_m) \right), S \right) \]

[By [2.19] and [2.20].]

\[ \mathcal{D}(i_i) = \mathcal{D}(i_i) \cup \mathcal{D}(i_j) \quad \text{[By [2.4] and the definition of } \mathcal{D}(\cdot)\text{.]} \]

[2.22]

\[ \{\hat{q}_0, \hat{q}_{a+1}, \ldots, \hat{q}_{a+1}, \hat{q}_b, \hat{q}_{b+1}, \ldots, \hat{q}_m\} = \{\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_a\} \]

[By [2.22], [2.2], [2.6], and [2.11].]

[2.23]

\[ \Delta \left( \text{Func}(g, g) \left( \hat{f}_j(i_i, i_j) \right), S \right) = \bigcup_{x_i \in \Delta(\text{Func}(g, g)(i_i), S)} \bigcup_{x_j \in \Delta(\text{Func}(g, g)(i_j), S)} \bigcup_{x_k \in \Delta(\text{Func}(g, g)(i_k), S)} \bigcup_{x_{k+1} \in \Delta(\text{Func}(g, g)(i_{k+1}), S)} \bigcup_{x_{m} \in \Delta(\text{Func}(g, g)(i_m), S)} \Delta \left( \text{Func}(g, g) \left( \hat{f}_j(i_i, x_k, x_{k+1}, \ldots, x_m) \right), S \right) \]

[By [2.21] and [2.23].]

[2.24]
\[ \Delta(\text{Func}(g, \mathcal{A})(\hat{e}_k), S) = \bigcup_{\mathcal{F}_1 \in \Delta(\text{Func}(g, \mathcal{A})(\hat{e}_1), S)} \bigcup_{\mathcal{F}_2 \in \Delta(\text{Func}(g, \mathcal{A})(\hat{e}_2), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1), S))} \bigcup_{\mathcal{F}_3 \in \Delta(\text{Func}(g, \mathcal{A})(\hat{e}_3), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1, \mathcal{F}_2), S))} \cdots \bigcup_{\mathcal{F}_n \in \Delta(\text{Func}(g, \mathcal{A})(\hat{e}_n), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{n-1}), S))} \Delta(\text{Func}(g, \mathcal{A})(\hat{e}_k), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n), S)) \]

By [2.24], since \( \mathcal{A}(\hat{e}_1, \hat{e}_2) = \hat{e}_k \) according to [2.21].

Q.E.D. (2)

Q.E.D. [By [11, [2], and induction.]}

\[ \square \]

3B.5. Proof of proposition 3.13

Theorem 3B.42: Let \( \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n \) be \( n \) hypothesis classes, and let \( l \) be a positive integer. Let \( \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \) be a constraint system for \( \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n \) and, for each \( \mathcal{A} \in \{ \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \} \), let

\[ \mathcal{X}_l = \{ \mathcal{A}(H_1, H_2, \ldots, H_n) : H_1 \in \mathcal{X}_1, H_2 \in \mathcal{X}_2, \ldots, H_n \in \mathcal{X}_n \}. \]

Let \( \oplus \) be \( n \)-ary conjunction operator and Let \( \phi \) be a conjunction specification with respect to \( \mathcal{G} = \{ \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n \} \) and \( \mathcal{S} \) for some set \( \mathcal{S} \), such that

\[ \mathcal{S}(\phi) = \{ \hat{q}_1, \hat{q}_2, \ldots, \hat{q}_n \}. \]

Assume that for any

\[ \mathcal{F}_1 \subseteq \text{Func}(g, \mathcal{A})(\hat{q}_1), \mathcal{F}_2 \subseteq \text{Func}(g, \mathcal{A})(\hat{q}_2), \ldots, \mathcal{F}_n \subseteq \text{Func}(g, \mathcal{A})(\hat{q}_n). \]

\[ |\Delta(\text{Func}(g, \mathcal{A})(\oplus), S)| \leq \sum_{\mathcal{F}_1 \in \Delta(\text{Func}(g, \mathcal{A})(\hat{q}_1), S))} \sum_{\mathcal{F}_2 \in \Delta(\text{Func}(g, \mathcal{A})(\hat{q}_2), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1), S))} \cdots \sum_{\mathcal{F}_n \in \Delta(\text{Func}(g, \mathcal{A})(\hat{q}_n), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{n-1}), S))} |\Delta(\text{Func}(g, \mathcal{A})(\oplus), \text{hypo}(\mathcal{A}, S)(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n))| \cdot \]

Let \( \mathcal{F} \) be the class

\[ \{ \oplus(F_1, F_2, \ldots, F_n) : F_1 \in \mathcal{Y}_1(F_1, F_2, \ldots, F_n), F_2 \in \mathcal{Y}_2(F_1, F_2, \ldots, F_n), \ldots, F_n \in \mathcal{Y}_n(F_1, F_2, \ldots, F_n) \}. \]
Then, for any list $S \in \text{Dom}(\mathcal{G})^*$,

$$
\Pi_{\mathcal{G}}(S) = \sum_{K_1 \in \mathcal{K}_1} \sum_{K_2 \in \mathcal{K}_2} \cdots \sum_{K_n \in \mathcal{K}_n} \sum_{x_1 \in \Delta(K_1, S)} \sum_{x_2 \in \Delta(K_2, S)} \cdots \sum_{x_n \in \Delta(K_n, S)} |\Delta(\text{Func}(\mathcal{G}), \hat{\phi}[\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n], S)|.
$$

Proof:

[1] 
$$
\forall (\mathcal{G}_i \in \mathcal{K}_i)^n : 
\forall (G_i \in \mathcal{G}_i) : 
\forall (F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2, \ldots, F_n \in \mathcal{F}_n) : 
G_i(F_1, F_2, \ldots, F_n) \in \mathcal{X}_1
$$

[By the definition of $\mathcal{X}_1$.]

[2] 
$$
\mathcal{F} \subseteq \{\hat{\phi}(F_1, F_2, \ldots, F_n) : F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2, \ldots, F_n \in \mathcal{F}_n\}
$$

[By [1] and the definition of $\mathcal{F}$ in the statement of the theorem.]

[3] 
$$
\forall (\mathcal{F}_1 \in \mathcal{X}_1, \mathcal{F}_2 \in \mathcal{X}_2, \ldots, \mathcal{F}_n \in \mathcal{X}_n) : 
|\Delta(\text{Func}(\mathcal{G}), \hat{\phi}, S)| \leq 
\sum_{x_1 \in \Delta(\mathcal{F}_1, S)} \sum_{x_2 \in \Delta(\mathcal{F}_2, S)} \cdots \sum_{x_n \in \Delta(\mathcal{F}_n, S)} |\Delta(\text{Func}(\mathcal{G}), \hat{\phi}[\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n], S)|
$$

[By assumption.]

[4] 
$$
|\{\hat{\phi}(H_1, H_2, \ldots, H_n) \cap S : H_1 \in \mathcal{F}_1, H_2 \in \mathcal{F}_2, \ldots, H_n \in \mathcal{F}_n, \mathcal{F}_1 \in \mathcal{X}_1, \mathcal{F}_2 \in \mathcal{X}_2, \ldots, \mathcal{F}_n \in \mathcal{X}_n\}|
\leq 
\sum_{\mathcal{F}_1 \in \mathcal{X}_1} \sum_{\mathcal{F}_2 \in \mathcal{X}_2} \cdots \sum_{\mathcal{F}_n \in \mathcal{X}_n} |\{\hat{\phi}(H_1, H_2, \ldots, H_n) \cap S : H_1 \in \mathcal{X}_1, H_2 \in \mathcal{X}_2, \ldots, H_n \in \mathcal{X}_n\}|
$$

[By the properties of sets.]
3B.6. **Formalized proof of proposition 3A.15**

**Theorem 3B.43:** Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be any two events defined on a set $X$. If $\mathcal{G}_1 \subseteq \mathcal{G}_2$, then $\nu(\mathcal{G}_1) \leq \nu(\mathcal{G}_2)$.

**Proof:**

$(\text{1})$

$\forall (S \subseteq X), if \mathcal{G}_1 \text{ shatters } S \text{ then } \mathcal{G}_2 \text{ shatters } S$
Proof of 1: (We assume the claim does not hold and derive a contradiction.)

[1.1] Let \( S \subseteq X \)

[1.2] \( \mathcal{A}_1 \) shatters \( S \) but \( \mathcal{A}_2 \) does not shatter \( S \).  
[Assumption.]

[1.3] Let \( S_1 \subseteq S \) be such that \( \mathcal{A}(E \in \mathcal{A}_2) : E \cap S = S_1 \)

[1.4] Let \( E_1 \in \mathcal{A}_1 \) be such that \( E_1 \cap S = S_1 \)

[Such an \( E_1 \) must exist by the definition of shattering, since \( \mathcal{A}_1 \) shatters \( S \).]

[1.5] \( E_1 \notin \mathcal{A}_2 \)

[If \( E_1 \) were in \( \mathcal{A}_2 \), [1.3] would fail to hold by [1.4].]

[1.6] \( E_1 \in \mathcal{A}_1 \land E_1 \notin \mathcal{A}_2 \)

[By [1.4] and [1.5].]

[1.7] \( \mathcal{A}_1 \notin \mathcal{A}_1 \)

[By [1.6].]

Q.E.D. (1)  
[1.7] contradicts one of the theorem's assumptions.

[2] Let \( S_0 \) be a particular set of points s.t. \( \ell \equiv |S_0| = \nu'(\mathcal{A}_1) \)

[Such an \( S_0 \) exists by the definition of \( \nu' \).]

[3] \( \mathcal{A}_1 \) shatters \( S_0 \)

[By [2] and the definition of \( \nu' \).]

[4] \( \mathcal{A}_2 \) shatters \( S_0 \)

[By [3] and [11].]

[5] \( \nu'(\mathcal{A}_2) \geq |S_0| \)

[By [4] and the definition of \( \nu' \).]

[6] \( \nu'(\mathcal{A}_2) \geq \nu'(\mathcal{A}_1) \)

[By [5] and the fact that \( \nu'(\mathcal{A}_1) = |S_0| \) according to [2].]
Theorem 3B.44: Consider a set of $k$ hypothesis classes

$$\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k$$

and their conjection

$$\mathcal{C}(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k)$$

under some operator $\circ$. Let

$$C_1, C_2, \ldots, C_k$$

be acceptable credit assignment functions for $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k$ with respect to $\circ$ and

$$C(\circ(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k))$$

for some $C$. Then

$$\forall (C(\circ(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k))) \leq \forall \left(\sum_{i=1}^{k} \forall(C_i(\mathcal{H}_i))\right).$$

Proof:

[1]

$\exists(\circ(\circ\text{ anasic}))$:

$$C(\circ(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k)) \subseteq C(\circ(C_1(\mathcal{H}_1), C_2(\mathcal{H}_2), \ldots, C_k(\mathcal{H}_k)))$$

[ By the definition of an acceptable credit assignment function. ]

[2]

$$\forall(\circ(C_1(\mathcal{H}_1), C_2(\mathcal{H}_2), \ldots, C_k(\mathcal{H}_k))) \leq \forall \left(\sum_{i=1}^{k} \forall(C_i(\mathcal{H}_i))\right)$$

[ By theorem 2A.36, since $\circ$ is ricet ]

[3]

$$\forall(C(\circ(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k))) \leq \forall \left(\sum_{i=1}^{k} \forall(C_i(\mathcal{H}_i))\right)$$

[ By [2], [11], and theorem 3B.43. ]

Q.E.D. □
4. Bounding the sample complexity of iterative learning algorithms.

It has been known for some time in the field of machine learning that simple explanations of data are more desirable than complex ones; a concise hypothesis that explains a set of training data is more likely to explain novel data as well. This has been observed empirically in the field of pattern recognition (c.f. [9]), and there, poor generalization ability was associated with classification rules that explain data in unnecessary detail, while missing regularities that might aid in classification. This is the well-known problem of overfitting data.

More recently, [6] made precise the sense in which concise hypotheses outperform complicated ones, by formally relating hypothesis size to hypothesis error. That paper showed that if a hypothesis grows more slowly than the data itself as examples are added, the error of the hypothesis can be made arbitrarily small, using a sample whose size grows only polynomially in the inverse of the desired bound on the error. It was shown that learning can be achieved if the size of the hypothesis does not exceed \( \tau \ell^\alpha \) bits, where \( \ell \) is the number of training examples, and \( \tau \) and \( \alpha \) are arbitrary constants with \( \tau \geq 0 \) and \( 0 < \alpha < 1 \). Algorithms whose hypotheses fit this criterion are referred to as Occam Algorithms in [6].

The result is interesting, because it permits the hypothesis to grow continuously (albeit only sublinearly, since \( \alpha < 1 \)) as the number of examples increases. This gives us the ability, to some extent, to overcome the restrictions imposed by having to specify a hypothesis class before learning begins. That is a useful option that does not have an analog in most other optimization problems; it is as if we were permitted to add a few processors in a scheduling problem, add colors in a graph-coloring problem, add bins in a bin-packing problem, etc. Since we expect that we will often use heuristic methods to find a hypothesis with a small empirical error, this option is especially attractive, for if we fail to find an optimal solution on the first try we may be able to make up the difference by adding a few hypotheses to our hypothesis class.

Unfortunately, it is often hard to ascertain beforehand whether the hypotheses produced by a learning algorithm
will meet the requirement of growing no faster than \( r^t \). Most learning algorithms simply update their hypotheses until all of the training examples are correctly classified, without being able to specify beforehand how many updates there will be. If the hypothesis size changes when the hypothesis is updated, the size of the final hypothesis may be difficult to predict.

This chapter concerns learning algorithms that update their hypotheses iteratively. A bound will be given on the number of iterations required to find a hypothesis that correctly explains \( t \) training examples, for any positive \( t \), and this bound will be based on the quality of the mechanism that updates the hypotheses. (In particular, we require that each update of the hypothesis explains a fixed but arbitrarily small* percentage of the previously unexplained examples.) The chapter then generalizes the result of [6] to facilitate the analysis of such learning algorithms.

In our most general result, we will also require that the change in hypothesis size be \( O(t^\alpha) \) on each update, and it is in this sense that the result of [6] is a special case of the result presented here (it is the case where the hypothesis is only updated once). However, a restriction on hypothesis growth is more benign than a restriction on overall hypothesis size, and it is easier to determine if the restriction is satisfied by an algorithm that generates successive modifications of a hypothesis.

Since better bounds on sample complexity can be obtained if stronger assumptions are made about the way hypotheses are updated, we will also derive results about update mechanisms that increase the hypothesis size by a fixed amount on each iteration, and mechanisms that update the hypothesis by conjuncting it with another hypothesis. We will also examine cases in which the final hypothesis has a nonzero error.

It should be noted that [6]'s result can sometimes also be applied to algorithms that update their hypotheses iteratively. In particular, if it is known that there will be \( u \) updates, and \( u \) is not a function of \( t \), then we can simply replace the constant \( r \) with \( ru \) in Blumer's result. However, this chapter is concerned with cases where \( u \) is a function of \( t \). We will show that the number of updates grows logarithmically in this case, at least for the kinds of learning algorithms discussed here, so it is intuitive that [6]'s result should continue to apply; however it is not immediately clear how to bound the sample complexity of iterative learning algorithms. The contribution of this chapter is provide such a bound.

4.1. Some examples of iterative learning algorithms

An approach that is often used in machine learning is to look for a hypothesis that "covers" some of the training examples

* The phrase "fixed but arbitrarily small" means that the percentage of examples explained on each iteration may not depend on the number of examples itself, e.g., 0.1% if there are 1000 examples and 0.01% if there are 10000 examples.
examples, in the sense that it classifies them correctly. After such a hypothesis has been found another hypothesis is constructed to cover some more examples, until, ultimately, all examples are covered. (Note that the hypotheses created by covering algorithms need a way to determine what inputs should be classified by which hypothesis).

However, our results apply not only to covering algorithms, but to any learning algorithm that updates its hypothesis iteratively and has the option of making the hypothesis more complex. Many learning algorithms fit this description. Consider, for example, the following paradigm for learning boolean functions:

**Paradigm 1:**

1. Let $H$ be an initial hypothesis.
2. Request an example $x$. If $H$ misclassifies $x$, then:
   1. If $x$ is positive, generalize $H$ as little as possible to include $x$.
   2. If $x$ is negative, specialize $H$ as little as possible to exclude $x$.
3. (If examples remain, go to step 2.)

This paradigm is used widely in machine learning, but it will only work if the generalization or specialization step does not "break" the hypothesis by making it err on some of the examples that had been classified correctly before the update. Therefore it is natural to expect that the hypothesis will grow in complexity (although it is also reasonable to expect that the hypothesis will not grow mindlessly, for we assume that the programmer of the learning algorithm was sophisticated enough to do more than simply construct a lookup table). We expect such learning algorithms to fall under the present model.

A more general description of the learning paradigm follows:

**Paradigm 2:**

Assumptions: We are given a sequence of examples of the function to be learned.

1. Let $H$ be some default classification rule.
2. Remove from the training list every example that is classified correctly by $H$.
3. (If no examples remain, output $H$ and halt. Otherwise, modify $H$ to correctly classify some of the remaining positive examples, and go to step 2.)
4.2. hypothesis update mechanisms

The substance of this procedure lies in the implementation of the third step, in which \( H \) must be modified. This is the step where algorithms that follow paradigm 2 differ from one another. The removal of examples in step 2 is for clarity; all that is really necessary is for the algorithm to be able to tell which examples are already explained by \( H \).

In our analysis we will describe step 3 by defining various hypothesis update mechanisms, functions that map \( \mathcal{H} \times X^t \) to \( \mathcal{H} \), where \( \mathcal{H} \) is a class of hypotheses and \( X \) is a sample space. Specifically, the first argument is a hypothesis and the second is a training sample. The hypothesis update mechanism modifies the hypothesis to make it perform better on the training sample.

4.2.1. Uniform hypothesis update mechanisms

As we define specific hypothesis update mechanisms below, we will find it useful to have a compact way of describing the repeated application of a hypothesis update mechanism to the same initial hypothesis.

Definition: Let \( Par \) be a hypothesis update mechanism, let \( H \) be a hypothesis to be updated, and let \( a_1, a_2, \ldots, a_n \) be a list of \( n \) unspecified arguments. Then:

\[
(a) \quad Par^0(H, a_1, a_2, \ldots, a_n) \equiv Par(H, a_1, a_2, \ldots, a_n).
\]

\[
(b) \quad \forall (i \in \mathbb{N}): \quad Par^i(H, a_1, a_2, \ldots, a_n) \equiv Par(Par^{i-1}(H, a_1, a_2, \ldots, a_n), a_1, a_2, \ldots, a_n).
\]

In many cases we will find that our analysis does not depend on the initial hypothesis that exists before any modifications have been performed by the hypothesis update mechanism. Because of this we will often omit the parameter that specifies this hypothesis. For example, we will say \( Par^i(S) \) instead of \( Par^i(H, S) \).
The first hypothesis update mechanism we describe is one that is required always improve the performance of the hypothesis it modifies, at least slightly.

We define a uniform hypothesis update mechanism as follows:

**Definition:** Let $\sigma > 1$. Let $\mathcal{H}$ be a set of hypotheses and $X$ be $\text{Dom}(\mathcal{H})$. Let $S \in X^i, t \in T$. A uniform hypothesis update mechanism is a hypothesis update mechanism $\text{Par}_\sigma$ with the property that:

$$\forall (i > 0):$$

$$\left| \left\{ x \in S : \text{Par}_{\sigma}^i(H, S)(x) \neq x \right\} \right| \leq \left| \left\{ x \in S : \text{Par}_{\sigma}^{i-1}(H, S)(x) \neq x \right\} \right| \left(1 - \frac{1}{\sigma}\right).$$

Recall that each training example has two parts, $x^t$ and $x^{w}$, and that if $F$ is the function to be learned, then $x^{w} = F(x^t)$. The definition of a uniform hypothesis update mechanism guarantees that each iteration will improve the performance of the hypothesis slightly, at least on the training sample. Because of this property we know that the update mechanism will eventually output a hypothesis that makes no errors on the training sample; this knowledge is the basis of our analysis.

The number of iterations needed to obtain an empirical error of zero is bounded in the next result:

**Proposition 4.1:** Let $\text{Par}_\sigma$ be a uniform hypothesis update mechanism, and define

$$\beta \equiv \frac{\sigma}{\sigma - 1}.$$

Let $\mathcal{H} = \text{Ran}(\text{Par}_\sigma)$, and let $S \in \text{Dom}(\mathcal{H})^i, t \in T$. Then

$$\forall (H \in \mathcal{H}):$$

$$\left| \left\{ x \in S : \text{Par}_{\sigma}^{\text{lim}}(H, S)(x) \neq x \right\} \right| \leq 0.$$  \hfill (4.1)
Algorithm 1:

Assumptions: \( \mathcal{X} \) is a sample from some sample space \( X \). \( \text{Par}_{r,e,v} \) satisfies the restrictions given above for some \( \sigma > 1, r > 0, 0 \leq \alpha < 1 \), and for some set \( \mathcal{G} \) of hypotheses.

1. \{Initialize \( H \)\} \hspace{1cm} \( H \leftarrow H_0 \ (H_0 \in \mathcal{G}) \);

2. \{Make repeated calls to \( \text{Par} \) until all examples are classified correctly\} \hspace{1cm} \text{Do while } \exists (x \in \mathcal{X}) : H(x) \neq H(x^*) ;

2a. \{Update \( H \) \} \hspace{1cm} \text{Let } H \leftarrow \text{Par}_{r,e,v}(H, \mathcal{X}) ;

\{ ... until it works on all examples \} \hspace{1cm} \text{end}.

3. \{Return the final hypothesis\} \hspace{1cm} \text{Return } H.

Proof: The proof is given in the appendix. \( \square \)

In other words, (4.1) gives the maximum number of incorrectly classified examples that can remain after the \( i \)th iteration of a uniform hypothesis update mechanism.

We rewrite paradigm 2 as algorithm 1 below.

4.2.2. Sublinear uniform hypothesis update mechanisms

We now introduce a hypothesis update mechanism with a further restriction.

Definition: Let \( \sigma > 1, r > 1, 0 < \alpha \leq 1 \). A sublinear hypothesis update mechanism \( \text{Par}_{\sigma,r,\alpha} \) is a hypothesis update mechanism with the properties that, for all \( S \in \text{Dom} \ (\text{Ran} \ (\text{Par}_{\sigma,r,\alpha})) \),

\( (a) \hspace{1cm} \| \text{Ran} \ (\text{Par}_{\sigma,r,\alpha})^0 (S) \| = 1 \)

\( (b) \hspace{1cm} \| \text{Ran} \ (\text{Par}_{\sigma,r,\alpha})^i (S) \| \leq r |S|^\alpha + \| \text{Ran} \ (\text{Par}_{\sigma,r,\alpha})^{i-1} (S) \| \)
When we say that

$$\|\text{Ran} (\text{Par}_{\sigma,r,o}(\cdots))\| = n$$

we mean that $n$ bits are sufficient to index all of the hypotheses in

$$\text{Ran} (\text{Par}_{\sigma,r,o}(\cdots)).$$

A sublinear uniform hypothesis update mechanism is simply a hypothesis update mechanism that is both uniform and sublinear.

If $\text{Par}_o$ is a uniform hypothesis update mechanism, the size of the hypothesis can grow by only $r|\mathcal{F}|^n$ bits during each iteration of algorithm 1. Therefore, the bound on the number of calls to $\text{Par}_o$ in proposition 4.1 means that the hypothesis finally returned can be described using

$$(\log_2|\mathcal{F}| + 1) r|\mathcal{F}|^n = \left(\frac{\ln|\mathcal{F}| + \ln(\beta)}{\ln(\beta)}\right) r|\mathcal{F}|^n$$

bits.

This allows us to determine the sample size required to ensure that, with probability $(1 - \delta)$, the error of $\mathcal{H}$ will be at most $\epsilon$, for given values $0 < \delta < 1$ and $0 < \epsilon < 1$. A bound is given in the following theorem.

**Proposition 4.2:** Let $\text{Par}_{\sigma,r,o}$ be a uniform sublinear hypothesis update mechanism, and let $\mathcal{H} = \text{Ran} (\text{Par}_{\sigma,r,o})$. Let $X$ be a sample space. Let $t \in \mathbb{N}$ and let $\epsilon$ be a constant, $(0 < \epsilon \leq 1)$, and let $\delta$ be a constant, $0 < \delta \leq 1$. Then the inequality

$$P_X \left( S \in X^t : P_X \left( \text{Par}_{\sigma,r,o}(t+1)(S)(x^*) \neq x^{(t)} \right) \geq \epsilon \right) \leq \delta$$

holds whenever

$$t \geq \max \left\{ \frac{2 \ln(\delta)}{\ln(1 - \epsilon)}, \left( \frac{A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} + 2A \right) \right) \right\}$$

where

$$A \equiv \frac{2r \ln(2)}{-\ln(1 - \epsilon)}$$

and

$$\beta \equiv \frac{\sigma}{\sigma - 1}.$$
Proof: The proof is given in the appendix.

4.2.3. Discussion.

The result of [6] is the following:

Lemma 4.3: [6] Let $L$ be a learning algorithm mapping $X^t$ to $\mathcal{H}$, for some sample space $X$ and assume that the hypotheses in $\mathcal{H}$ can be described using no more than $r^t \alpha$ bits for some $r > 1, 0 < \alpha < 1$. Then

$$P_{x^t}(s \in X^t : P_x(L(s|x^t) \neq x^t) \geq \epsilon) \leq \epsilon$$

holds whenever

$$\max \left\{ \frac{2 \ln(1/\delta)}{-\ln(1-\epsilon)}, 4^{1/(1-\alpha)} \right\}$$

where

$$A = \frac{2r \ln(2)}{-\ln(1-\epsilon)}.$$ 

When $\sigma \to 1$ in algorithm 1, so that all examples are classified correctly after a single iteration, the bound in theorem 4.3 approaches

$$\max \left\{ \frac{2 \ln(1/\delta)}{-\ln(1-\epsilon)}, 4^{1/(1-\alpha)} \right\}$$

because $1/\beta \to 0$. In [6], a bound of

$$\max \left\{ \frac{2 \ln(1/\delta)}{-\ln(1-\epsilon)}, 2A^{1/(1-\alpha)} \right\}$$

was obtained for that case. Thus lemma 4.3 is almost a special case of [6]'s result. In fact, it is possible to obtain a tighter bound that is a general case of (4.3), but the statement of that result (to say nothing of the proof) is too tedious for inclusion here.

The sample complexity of algorithm 1 is clearly polynomial in $1/\epsilon, 1/\delta, r, \sigma$, and $|\mathcal{H}|$. It is also clear that, if the complexity of $Par$ is not taken into account, the entire resource requirements of algorithm 1 are polynomial in the sample size. Thus the feasibility of running the algorithm essentially depends on the feasibility of $Par$. 
The usefulness of algorithm 1 stems from the fact that the algorithm implementing \( \text{Par} \) can return highly suboptimal hypotheses without causing the algorithm to fail, so long as enough examples are available. This suggests that many existing, heuristically based learning algorithms might be used in place of \( \text{Par} \). Even algorithms whose hypotheses are of unknown quality may turn out to be useful, so long as their performance does not deteriorate when the sample size increases.

However, proposition 4.2 is extremely general, and it is not to be expected that many learning algorithms would make full use of this generality. If stricter assumptions can be made about the learning algorithm, then we expect better bounds to be obtainable, and we will address that issue later in the chapter. First, however, we will extend lemma 4.3 to the case where the learning algorithm finds a hypothesis that correctly classifies some, but not all, of the training examples.

4.3. Inconsistent hypotheses.

Often, it is not possible for a learning algorithm to find a hypothesis that classifies all training examples correctly. A hypothesis with a nonzero empirical error is often called an inconsistent hypothesis.

Difficulties involving computational complexity may prevent a learning algorithm from finding a consistent hypothesis, in fact there are quite a few instances in which it is \( \text{NP} \)-hard to find a consistent hypothesis (Cf. [18], [7], [31]). It may also be that the training examples are noisy. Some learning algorithms (such as the CN2 algorithm of [8]) explicitly sacrifice consistency in favor of compactness. This section will discuss how the results above can be applied to such algorithms.

We can model the situation in several ways. One way is to assume that, instead of being asked to find a hypothesis with an empirical error of zero, the learning algorithm is only asked to find a hypothesis whose empirical error is less than \( \phi \), for some \( 0 < \phi < 1 \).

We must ensure that the learning algorithm actually stops when empirical error drops below \( \phi \), rather than continuing to update the hypothesis unnecessarily. We therefore make the following definition:

**Definition:** Let \( \sigma > 1 \) and \( 0 < \phi < 1 \). Let \( \mathcal{H} \) be a set of hypotheses and \( X \) be \( \text{Dom}(\mathcal{H}) \). A conservative hypothesis update mechanism with respect to \( \sigma \) and \( \phi \) is a sublinear hypothesis update mechanism with the following properties for all \( S \in \text{Dom}(\text{Ran}(\text{Par}_{\sigma,\phi})) \):
In this definition we do not actually ask for an empirical error of \( \phi \), but only for an error less than \((\phi \ell + 1)/\ell\). This is to prevent pathological behavior when \( \phi \) is smaller than \(1/\ell\). Since the empirical error cannot take on a value between \(1/\ell\) and zero (exclusive), specifying such a small \( \phi \) is equivalent to asking for a hypothesis with an empirical error of zero, and this case was already dealt with above. In the present case, we only let \( \phi \) be as small as the smallest nonzero error that can be obtained with a sample of size \( \ell \).

We show the following in the appendix:

\textbf{Proposition 4.4:} Let \( \text{Par}_{\sigma,\ell} \) be a conservative hypothesis update mechanism, and let \( X = \text{Ran}(\text{Par}_{\sigma,\ell}) \).

Let \( X \) be a sample space. Let \( \xi, k, \phi \) be constants, \( 0 \leq \xi, \delta, \phi < 1 \), and let \( \ell \in \mathbb{N} \). Define

\[ k = \max \left\{ \min \left\{ i : \left| \mathbb{P} \left\{ s \in X : \text{Par}_{\sigma,\ell}(S)(x^i) \neq x^o \right\} \right| \leq \phi \ell + 1 \right\} : S \in X' \right\}. \]

Then the inequality

\[ \mathbb{P}_{X'} \left( S \in X' : \mathbb{P}_{X} \left( x : \text{Par}_{\sigma,\ell}(S)(x^i) \neq x^o \right) > \phi + \ell \xi \right) \leq \delta \]

holds whenever

\[ \ell \geq \max \left\{ 1, \frac{\ln(\delta) - \ln(2)}{-\xi^2}, \frac{1}{\xi^2 (\ln(2) - 1 - 1)} - 2 \frac{\ln(1 - 1/\sigma)}{\xi^2 (1 - \alpha) / r \ln(2)} \right\}^{1/2}. \] \hspace{1cm} (4.4)
Proof: The proof is given in the appendix. □

Note that \( k \) does not appear on the right side of (4.4). To apply proposition 4.4 we therefore do have to know \( k \), we only have to know that \( k \) exists.

Unfortunately there is a potential difficulty with this result: \( \rho \) must be specified in advance in order to tell the algorithm when to stop. This is somewhat unrealistic, as a learning algorithm is more likely to stop when the empirical error of its hypothesis can no longer be decreased except at the cost of an immoderately large hypothesis.

We can solve this problem by using the upper bound we obtained in lemma 4.4 for the number of iterations of algorithm 1; clearly this upper bound still holds if the algorithm is halted prematurely. We can thus analyze the present case by using \( \log_\phi(m) + 1 \) as an upper bound on the number of iterations of the learning algorithm, as we did before.

The sample size in this case can be bounded in a manner analogous to the one used in proposition 4.4

**Proposition 4.5:** Let \( \text{Par}_{\cdot, r, a} \) be a conservative hypothesis update mechanism, and let \( \mathcal{H} = \text{Ran}(\text{Par}_{e, r, a}) \).

Let \( X \) be a sample space. Let \( \xi, \delta, \phi \) be constants, \( 0 \leq \xi, \delta, \phi < 1 \), and let \( t \in \mathbb{N} \). Define

\[
\beta \equiv \frac{\sigma}{\sigma - 1}.
\]

Let \( t \in \mathbb{N} \). Then

\[
P_X' (S \in X^t : P_X (x : \text{Par}_{\cdot, r, a}^1(S)(x') \neq x^t) > \phi + \xi) \leq \delta
\]

holds whenever

\[
t \geq \max \left\{ \frac{\ln(\delta) - \ln(2)}{\xi^2}, \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right)^{\frac{1}{2}} \right\}
\]

where

\[
A \equiv \frac{r}{\xi^2} \log_\phi(e).
\]
Proof: The proof is given in the appendix.

Example 4.6: \( t \)-element \( k \)-decision lists

One specialization of paradigm 1 produces a structure known as a decision list (Cl. [29]). A decision list maps the input space \( X \) to some set of classes \( \{c_1, c_2, \ldots, c_k\} \) and each class is associated with a \( \{0, 1\} \)-valued function, or rule, that determines whether an input should fall into the class. These rules are ordered, and if \( r_i \) is the first rule whose value is 1 for some input \( x \), then the class associated with \( r_i \) is assumed to be the class of \( x \). Thus the list has the form \( \langle r_1, c_1 \rangle, \langle r_2, c_2 \rangle, \ldots, \langle r_i, c_i \rangle \), where each \( r_j \) is a rule and \( c_i \) is the corresponding classification. The input \( x \) is said to lie in the class \( c_i \) if and only if \(-r_1(x) \land -r_2(x) \land \cdots \land r_{i-1}(x) \land r_i(x)\).

A \( k \)-decision list is defined in [29] as a decision list whose rules are boolean functions, where each function can be expressed as a conjunction of up to \( k \) variables or their negations (We will refer to such constructions as boolean conjunctions). Since there are only \( O(n^k) \) such conjunctions (if \( n \) is the number of possible variables), a learning algorithm can examine each one in turn to determine which of them can be added to the decision list to classify the greatest number of examples correctly. (We regard \( k \) as a constant in order to obtain polynomial-time performance.)

It is assumed in [29] that the target function can, in fact, be expressed as a \( k \)-decision list, so that it is always possible to eventually handle all examples in this fashion. Since the bound on the number of conjunctions bounds the size of the decision list (that is, the decision list may have no more than have \( O(n^k) \) elements), the learning algorithm is an Occam algorithm in the sense of [6], and this fact can be used to bound the required sample size. This is done in [29], where it is shown that at most

\[
O \left( \frac{n^k \ln(n) + \ln(\delta)}{\epsilon} \right)
\]

(4.5)

examples are required. However, since the complexity of the algorithm grows exponentially in \( k \), \( k \) must be a small number.

The bound can be improved if the number of elements in the list is bounded. An \( l \)-element \( k \)-decision list is simply a \( k \)-decision list with \( l \) elements. Since the \( l \) elements must, between them, classify all possible examples, it is always possible to find one decision list element that correctly classifies (100/\( l \))% of any sequence of examples correctly. By lemma 4.3, a much better bound on the sample size than (4.5) can be obtained when the target is in this class; rather than depending exponentially on \( k \), the required sample size is bounded by

\[
l \geq \max \left\{ \frac{2 \ln(\delta)}{\ln(1-\epsilon)}, \left( \frac{2A}{\ln(\delta)(1-\alpha)} \right) \ln \left( \frac{A}{\ln(\delta)(1-\alpha)} \right) + 2A \right\}
\]

(4.6)
where

\[ \beta = \frac{1}{1 - \frac{1}{r}}, \]
\[ A = \frac{2r \ln(2)}{-\ln(1 - r)}, \]

and \( r \) is the number of bits required to write down a conjunction of \( k + 1 \) variables. (The \( k \) variables in the conjunction, and an additional 1 or 0 to indicate what value should be returned if a decision rule is satisfied.) Since only \( O(k + 1) \) bits are needed to specify such a conjunction, this is an improvement of the previous bound; in particular, the bound is polynomial in \( k \). Note that \( \alpha = 0 \) for this particular algorithm.

4.4. Further restrictions on the hypothesis update mechanism.

Until now we have permitted our hypothesis update mechanisms to be fairly liberal when updating a hypothesis. All of our results have applied to sublinear hypothesis update mechanisms, which update hypotheses subject to the size restriction

\[ \| \text{Ran} \{ \text{Par}_{\sigma, \alpha} \}^i(S) \| \leq r|S|^\alpha + \| \text{Ran} \{ \text{Par}_{\sigma, \alpha} \}^{i-1}(S) \|. \]

Here, \( \text{Par} \) is permitted to choose from among all hypotheses having a certain size, and the size restriction is quite liberal.

This gives \( \text{Par} \) considerable power, but the algorithm we actually use for \( \text{Par} \) may not be able to exploit this power fully. For example, consider the following specialization of paradigm 2 for learning boolean functions:

**Paradigm 3:**

1. \{Let \( H \) be the function that classifies all instances as 0.\}

2. \{Remove from the list of examples every example that \( H \) classifies correctly.\}

3. \{If no examples remain, output \( H \) and halt. Otherwise, Find a boolean function \( r \) that can be written as a boolean conjunction, correctly classifies some examples whose classification should be 1, and misclassifies no examples whose classification should be 0. Replace \( H \) with the disjunction of \( H \) and \( r \). Go to step 2.\}

(Note that the mechanism for determining which inputs are classified by which hypotheses is implicit here. If there is any conjunction in the final hypothesis that classifies an input as 1, then that input is regarded as positive; otherwise it is regarded as negative.)

Algorithms following this paradigm include the AQ series of algorithms (c.f. [24]) and the CN2 algorithm of Clark and Niblett [8]. Here the only effect of \( \text{Par} \) is to add a boolean conjunction to the hypothesis on each iteration, and thus \( \text{Par} \) is certainly not considering all boolean hypotheses whose size is bounded by \( |H| + r|\overline{Z}|^\alpha \).
If we can tighten the specification of Par that match the limitations of our actual update algorithms, we may be able to get tighter bounds on the learning algorithm’s sample complexity.

A simple restriction consists of limiting rate at which |JF*| may increase on subsequent calls to Par. We therefore define a new kind of hypothesis update mechanism:

Definition: An additive hypothesis update mechanism Add_{σ,ρ} is a hypothesis update mechanism with the properties that, for all S ∈ Dom (Ran (Add_{σ,ρ})),

(a) \[ |\text{Ran} \{(Add_{σ,ρ})^0\}| = 0 \]

(b) \(\forall (i ≥ 0)\):

\[ (|\text{Ran} \{(Add_{σ,ρ})^i\}| = n) \rightarrow (|\text{Ran} \{(Add_{σ,ρ})^{i+1}\}| = n + \rho) . \]

If the learning algorithm iterates at most \(\log_2(m) + 1\) times (that is, if the update mechanism is also uniform), then the final hypothesis will come from a class of size no greater than

\[ r(\log_2(m) + 1) \]

We use this fact to obtain the following result:

Proposition 4.7: Let Add_{σ,ρ} be an additive hypothesis update mechanism, \(X\) be a sample space, \(t \in \mathbb{N}\). Let \(ε\) and \(δ\) be constants, \(0 < ε, δ < 1\), and let

\[ β \equiv \frac{σ}{σ - 1} . \]

Then

\[ P_x \left( S : P_x \left( z : Add_{σ,ρ}^{ln(ε)}(S)(x') \neq x'' \right) \geq ε \right) \leq δ \]

holds whenever

\[ t ≥ \max \left\{ ε, \frac{\ln(δ/2) - \ln(ε)}{\ln(1 - ε)}, \frac{2(\ln(1/2 - \ln(\ln(εβ)))}{\ln(1 - ε)}, \frac{2\ln(1/2\ln(β))}{\ln(1 - ε)} \right\} . \]
Proof: The proof is given in the appendix. \[ \square \]

Example 4.8: \(l\)-term \(k\) - DNF

In this example our learning algorithm is to learn functions from the class \(l\)-term \(k\) - DNF, that is, the class of boolean functions that can be written as a disjunction of \(l\) boolean conjunctions, where the number of variables or negated variables in each conjunction is at most \(k\). We can simply use the covering paradigm given in paradigm 3: find a boolean conjunction that correctly classifies as many as possible of the positive examples without misclassifying any negative ones, then find another boolean conjunction to correctly classify as many as possible of the remaining examples, and so on until all possible examples are covered. If the target function can be written as an \(l\)-term \(k\)-disjunctive normal formula, then there is always a conjunction that correctly classifies at least \(1/l\) of the remaining examples in step 3 of paradigm 3. This is because there is some set of \(l\) conjunctions that, between them, classify every example in \(X\) correctly. Therefore we can say that \(\sigma = 1\) in this case. The sample complexity of this algorithm is bounded by

\[
\max \left\{ \epsilon, \frac{\ln(\delta/2) - \ln(\epsilon)}{\ln(1 - \epsilon)}, \frac{2(2\ln(\delta) - \ln(\epsilon))}{-\ln(1 - \epsilon)}, \frac{2\ln(1/2\ln(\epsilon))}{\ln(1 - \epsilon)} \right\}
\]

where

\[
\sigma = \frac{l}{l-1},
\]

if one boolean conjunction is described by \(n\) bits.

4.4.1. Iteratively conjecturing hypotheses.

As the next example will show, it is also useful to require that a hypothesis update mechanism modifies \(H\) by conjecturing it with a new hypothesis, using a ricotic operator. Specifically, we define a conjunctive hypothesis update mechanism as follows: conjunctive hypothesis update mechanism

Definition: A conjunctive hypothesis update mechanism \(\text{Con}_{x,\sigma}\) is a hypothesis update mechanism with the properties that, for all \(S \in \text{Dom} (\text{Ran} (\text{Par}_{x,\sigma}))\),

\[
\text{Ran} (\text{Con}_{x,\sigma}) = \mathcal{N}
\]

(a)
(b) \[ \forall(i > 1): \]
\[ \text{Ran} \left( \text{Con}_{\mathcal{M},\varphi} \right)^i = \circ \left\{ \mathcal{N}, \text{Ran} \left( \text{Con}_{\mathcal{M},\varphi}^{i-1} \right) \right\}, \]

where \( \circ \) is a ricetic operator.

Our paradigm in this case is the given in algorithm 2.

Algorithm 2:

Assumptions: \( \mathcal{F} \) is a sample from some sample space \( \mathcal{X} \). \( \circ \) is a ricetic connection operator.

\begin{enumerate}
\item \{Initialize \( H \): \}
\[ H \leftarrow H_i \left( H_i \in \mathcal{M}^{\circ} \right). \]
\item \{Make repeated calls to Par until all examples are classified correctly: \}
\[ \text{Do while } \exists (x \in \mathcal{F}) : H(x^+) \neq H(x^-) : \]
\[ \{ \text{Update } H .... \} \quad \text{Let } H \leftarrow \text{Con}_{\mathcal{M},\varphi}(H, \mathcal{F}). \]
\[ \{ ... \text{until it works on all examples.} \} \quad \text{oD.} \]
\item \{Return the final hypothesis. \}
\[ \text{Return } H. \]
\end{enumerate}

Once again, if \( \text{Con}_{\mathcal{M},\varphi} \) is also uniform then \( \text{Con} \) is called at most \( \log_2(\ell) + 1 \) times with a sample size of \( \ell \). Therefore, by the definition of a ricetic operator, the final hypothesis of algorithm 2 (being a \( (\log_2(\ell) + 1) \)-ary connection) induces at most

\[ \left( \Pi_{\mathcal{M}}(\ell) \right)^{\left( \log_2(\ell) + 1 \right)} \]

subsets on any list of size \( \ell \). If \( \forall (\mathcal{M}) \leq \ell, \ell \geq d, \) and \( \mathcal{M}_f \) is the class of hypotheses that can be returned by algorithm 2 when it is given a sample of size \( \ell \), then, by lemma 2A.15,

\[ \Pi_{\mathcal{M}_f}(\ell) \leq \left( \Pi_{\mathcal{M}}(\ell) \right)^{\log_2 \ell + 1} \leq \left( \frac{\ell}{d} \right)^{d(\log_2(\ell) + 1)}. \]

This leads to the following result:
Proposition 4.9: Let $\text{Con}_{\mathscr{H}, \sigma}$ be a conjunctive uniform hypothesis update mechanism. Define

$$\beta = \frac{\sigma}{\sigma - 1}$$

and let

$$d \equiv \mathcal{V}^* \left( \mathcal{H}^* \right).$$

For all $t \geq 1$:

(a)

$$\mathcal{V} \left( \text{Ran} \left( \text{Con}_{\mathscr{H}, \sigma}^{\log_2(t+1)} \right) \right) \leq \max \left\{ \frac{2d}{\log_2(\beta)} \log_2 \left( \frac{\epsilon}{\log_2(\beta)} \right), 51d, \left( \frac{\epsilon}{d} \right)^{\left( \frac{\log_2(t+1)}{\log_2(\beta)} - 1 \right)^{-1}} \right\}.$$

(b)

If $\beta > 2$,

$$\mathcal{V} \left( \text{Ran} \left( \text{Con}_{\mathscr{H}, \sigma}^{\log_2(t+1)} \right) \right) \leq \max \left\{ \frac{d^*}{\gamma^*}, t_0, \frac{2d \log_2(d/\gamma)}{\gamma^*}, \left( \frac{\gamma}{d} \right)^{\left( \frac{1}{\gamma^*} - 1 \right)^{-1}} \right\},$$

where

$$t_0 \equiv \max \left\{ 80, \beta^{(\log_2(\beta) - 1)^{-1}} \right\}$$

and

$$\gamma \equiv \frac{\log_2(t_0) \log_2(\beta)}{\log_2(t_0) + \log_2(\beta)}.$$

Proof: The proof is given in the appendix. □

Example 4.10: The Vapnik-Chervonenkis Dimension of the hypothesis produced by a covering algorithm.

In the general sense, a covering algorithm is an algorithm that operates very much along the lines of paradigm 2; a rule is sought that "covers" certain inputs in the sense that it classifies them all correctly. The learning algorithm searches for a rule that covers some of the training examples, then tries to cover some of the remaining examples with another rule, and so on, until all examples are properly handled by at least one rule.

It is usually assumed that there is some way to determine which rules cover which inputs, so that we do not have to worry about an input being classified by the wrong rule. In a previous example, we used lists of the form

$$\langle r_1, c_1 \rangle, \langle r_2, c_2 \rangle, \ldots, \langle r_m, c_m \rangle$$
where $r_1, r_2, \cdots, r_m$ were rules to determine whether or not an input should be classified according to the corresponding elements $c_1, c_2, \cdots, c_m$. In a sense, $r_i$ determines which inputs are covered by the list element $(r_i, c_i)$, and all of these elements have the same class. However, $c_i$ could just as easily have been a more complex classification rule.

In general, we can write the hypothesis produced by a covering algorithm as follows:

$$\{R_1, H_1\}, \{R_2, H_2\}, \cdots, \{R_m, H_m\},$$

where $R_1, R_2, \cdots, R_m$ are boolean-valued functions, and $H_1, H_2, \cdots, H_m$ are classification functions (which may, of course, also be boolean-valued).

The value of (4.8) on an input $x$ is computed as follows: let $R_i$ be the first rule in (4.8) such that $R_i(x) = 1$. Then the value of (4.8) at $x$ is $H_i(x)$. Thus $R_i$ determines what inputs are covered by the $i$th element of the list, and $H_i$ is the classification rule to be applied to the inputs that are covered. $R_i$ makes explicit the mechanism that a covering algorithm needs to determine which inputs are covered by which hypothesis.

If algorithm 2 were used to learn an expression with the form of (4.6) (that is, if algorithm 2 were a covering algorithm), then each call $Pa\rho$ would return one of the ordered pairs in (4.8). The algorithm would halt when all of the training examples were covered, in other words, when each training example satisfied at least one of the rules $R_1, R_2, \cdots, R_m$.

Therefore we assume that the rules $R_1, R_2, \cdots, R_m$ are to be learned; we will say that they come from the hypothesis class $\mathcal{H}$. Similarly, assume that the rules $H_1, H_2, \cdots, H_m$ are also to be learned and come from the hypothesis class $\mathcal{M}$.

By theorem 2A.23, each list element has a Vapnik-Chervonenkis Dimension of at most

$$4.7(\gamma' (\mathcal{H}) + \gamma' (\mathcal{M})).$$

Thus, by proposition 4.9, the VC Dimension of class of events in which the covering hypothesis error is bounded by

$$\gamma' \left( Ran \left( Con_{\mathcal{H}, \mathcal{M}}(\epsilon^{(t)}+1) \right) \right) \leq \max \left\{ \frac{2d}{\log_2 (\delta)} \log_2 \left( \frac{\epsilon}{1 - \epsilon} \right), 0.51d, \left( \frac{1}{\delta} \right) \left( \frac{1}{\epsilon} \right) \left( \frac{1}{\epsilon (1 - \epsilon)} \right) \right\}.$$ (4.9)

where $\gamma$ is as above and

$$d = 4.7(\gamma' (\mathcal{H}) + \gamma' (\mathcal{M})).$$ (4.10)

Note that proposition 4.8 could have been applied instead if each rule-pair came from a set of size $r$. However there are hypothesis classes whose size is infinite but whose VC Dimension is finite. It is such classes that create the greatest need for sample complexity bounds based on VC Dimensions.
4.5. Discussion.

Chapters 2 and 3 were devoted to the problem of bounding sample complexities of algorithms that learn various classes of conjectured hypotheses. These bounds were obtained by comparing the empirical errors of our hypotheses to the actual errors. We did not, however, discuss the problem of actually finding hypotheses with a small empirical error.

However, the results of this chapter show that we may be able to find such hypotheses by using heuristic methods; if our first attempt to find a good hypothesis fails, we can run the heuristic algorithm again on the examples that were not handled adequately. If our heuristic consistently manages to cover a certain percentage of the examples on each iteration, then we can repeat this process until all examples are classified correctly.

In fact, heuristic learning algorithms seem to perform fairly well even on the first attempt, at least on simple problems. The literature on machine learning contains countless examples of this; for example, see [26], [8], [128]. Moreover, most machine learning research is devoted to heuristic algorithms, and this considerable body of work gives us a strong foundation for attempting to solve the problems that were left open in the analysis just presented (that is, the problems associated with finding polynomial-complexity implementations of Par).

In various places we have used the notation $|\mathcal{H}|$ to denote the number of bits needed to distinguish among the elements of $\mathcal{H}$. Of course $|\mathcal{H}|$ is just $\log_2(|\mathcal{H}|)$. We used the extra notation in order to establish a link between our results and those of [6], while at the same time adding precision to the notion of "hypothesis size." (In [6] the size of the hypothesis is the number of bits needed to represent the hypothesis, and no regard is given to the efficiency of this representation.)

4.6. Bibliographic notes

In the second chapter we presented theorem 2.4, a series of theorems that promised to make a wide range of learning algorithms amenable to analysis, but which involved the VC Dimension of the hypothesis class to be analyzed. We attempted to develop methods that could be used to bound this parameter. This chapter also starts with a result that promises to make many algorithms easy to analyze, namely lemma 4.3 from [6]. Like the results in theorem 2.4, lemma 4.3 is difficult to apply in practice, and, as in the first 2 chapters, we have presented results extend and simplify the analysis.

There are some previous results that also involve iterative learning of sorts. [22] presents an update mechanism that requests additional training examples when the hypothesis is updated. This gives the learning algorithm
considerable power but the size of the training sample cannot be bounded in advance. [10] presents a powerful hypothesis boosting mechanism, which is also a hypothesis update mechanism in the broad sense ([10]'s result also asks for additional examples). [33] discusses a potential average-case improvement in learning algorithm performance when certain incremental methods are used.
Appendix A to Chapter 4.
Proofs of results in chapter 4.

4A.1. Preliminary lemmas

Lemma 4A.11: [33]: Let A be a learning algorithm, and let X be a sample space. Let F be a function, such that
\[ X \subseteq \text{Dom}(F) \subseteq \text{Dom}(\text{Ran}(A)), \]
\[ \text{Ran}(F) \subseteq \text{Ran}(\text{Ran}(A)). \]

Let \( t \in \mathbb{N} \) and let \( c \) be a constant, \( 0 < c \leq 1 \). Then
\[ P_{X, \{S \in X : \exists (H \in \text{Ran}(A)) : (\eta(H, F, S) = 0) \land (\eta(H, F) > c)\}} \leq (1 - c)^t |\text{Ran}(A)|. \]

Proof: See [33]. \( \Box \)

Lemma 4A.12:
\[ \forall (z > 2) : \frac{z}{2} \geq \log_2(x). \]

Proof:

[1] \[ x = 0 \rightarrow \frac{z}{2} \geq \log_2(x) \quad \text{[By arithmetic.]} \]

[2] \[ \frac{d}{dx} \frac{z}{2} = 1 \quad \text{[By calculus.]} \]

[3] \[ \frac{d}{dx} \log_2(x) = \frac{1}{x \ln(2)} \quad \text{[By calculus.]} \]

[4] \[ \forall (z > 0) : \left( \frac{d}{dx} \frac{z}{2} \geq \frac{d}{dx} \log_2(x) \right) \rightarrow \left( \frac{1}{2} \geq \frac{1}{x \ln(2)} \right) \quad \text{[By [1], [2], and [3].]} \]
\[ \forall (x > 0) : \left( \frac{d x}{d^2 x} \geq \frac{d}{d x} \log_b(x) \right) \iff \left( \frac{x}{2} \geq \frac{1}{\ln(2)} \right) \] 

By [4] and algebra.

\[ \forall (x > 0) : \left( \frac{d x}{d^2 x} \geq \frac{d}{d x} \log_b(x) \right) \iff \left( x \geq \frac{2}{\ln(2)} \right) \] 

By [5] and algebra.

\[ (x = 2) \rightarrow \left( \frac{x}{2} \geq \log_b(x) \right) \] 

By arithmetic.

\[ \left( x < \frac{2}{\ln(2)} \right) \rightarrow \left( \frac{d x}{d^2 x} < \frac{d}{d x} \log_b(x) \right) \] 

By [6].

\[ \left( x < \frac{2}{\ln(2)} \right) \rightarrow \frac{x}{2} \leq \log_b(x) \] 

By [7], [8], and the properties of derivatives (since \( x < 2/\ln(2) \) implies \( x < 2 \)).

\[ \left( x \geq \frac{2}{\ln(2)} \right) \rightarrow \left( \frac{d x}{d^2 x} \geq \frac{d}{d x} \log_b(x) \right) \] 

By [6].

\[ \left( x \geq \frac{2}{\ln(2)} \right) \rightarrow \frac{x}{2} \geq \log_b(x) \] 

By [7], [10], and the properties of derivatives.

\[ \frac{x}{2} \geq \log_b(x) \] 


Q.E.D.

Lemma 4A.13: [33]: Let \( A \) be a learning algorithm, and let \( X \) be a sample space. Let \( \ell \in \mathbb{N} \) and let \( \epsilon \) be a constant, \( 0 < \epsilon \leq 1 \). Then

\[ P_{X, \ell} \left( S \in X^\ell : \exists (H \in \text{Ran}(A)) : |\eta(H, S) - \eta(H)| > \epsilon \right) \leq |\text{Ran}(A)|2e^{-2\epsilon^2} \]
Proof: See [33].

Lemma 4A.14:

\[ \forall (x \geq 1): \frac{x}{2} \geq \ln(x). \]

Proof:

[1]  
\( (x - 1) \rightarrow \left( \frac{x}{e} \geq \ln(x) \right) \)  
[ By arithmetic. ]

[2]  
\( \frac{d}{dx} \frac{x}{e} = \frac{1}{2} \)  
[ By calculus. ]

[3]  
\( \frac{d}{dx} \ln(x) = \frac{1}{x} \)  
[ By calculus. ]

[4]  
\( \left( \frac{d}{dx} \frac{x}{e} \geq \frac{d}{dx} \ln(x) \right) \leftrightarrow \left( \frac{1}{e} \geq \frac{1}{x} \right) \)  
[ By [2] and [3]. ]

[5]  
\( \left( \frac{d}{dx} \frac{x}{e} \geq \frac{d}{dx} \ln(x) \right) \leftrightarrow \left( \frac{x}{e} \geq 1 \right) \)  
[ Multiplying the right side of [4] by \( x \). ]

[6]  
\( \left( \frac{d}{dx} \frac{x}{e} \geq \frac{d}{dx} \ln(x) \right) \leftrightarrow (x \geq e) \)  
[ Multiplying the right side of [5] by 2. ]

[7]  
\( (x = e) \rightarrow \left( \frac{x}{e} = \ln(x) \right) \)  
[ By arithmetic. ]

[8]  
\( (x \geq e) \rightarrow \left( \frac{d}{dx} \frac{x}{e} \geq \frac{d}{dx} \ln(x) \right) \)  
[ By [6]. ]

[9]  
\( (x \geq e) \rightarrow \left( \frac{x}{e} \geq \ln(x) \right) \)  
[ By [7], [8], and the properties of derivatives. ]

[10]  
\( (x \leq e) \rightarrow \left( \frac{d}{dx} \frac{x}{e} \leq \frac{d}{dx} \ln(x) \right) \)  
[ By [6]. ]
4A.2. Proof of proposition 4.1

Theorem 4A.15: Let $\text{Par}_v$ be a uniform hypothesis update mechanism, and let $\mathcal{M} = \text{Ran}(\text{Par}_v)$. Let $S$ be a set of elements from $\text{Dom}(\mathcal{M})$. Then

$$\forall (H \in \mathcal{M}) : 
\left| \left\{ x \in S : \text{Par}_v(H, S)(x^i) \neq x^\sigma \right\} \right| \leq |S| \left( 1 - \frac{1}{\sigma} \right)$$

Proof:

1. Let $H$ be an arbitrary element of $\mathcal{M}$. \hspace{1cm} [Definition.]

2. $\left| \left\{ x \in S : \text{Par}_v(H, S)(x^i) \neq x^\sigma \right\} \right| \leq |S|$ \hspace{1cm} [Since $S$ can only contain $|S|$ elements.]

3. $\forall (i > 0)$:
   $$\left| \left\{ x \in S : \text{Par}_v(H, S)(x^i) \neq x^\sigma \right\} \right| \leq \left| \left\{ x \in S : \text{Par}_v^{-1}(H, S)(x^i) \neq x^\sigma \right\} \right| \left( 1 - \frac{1}{\sigma} \right)$$
   \hspace{1cm} [By the definition of a uniform hypothesis update mechanism.]

4. $\left| \left\{ x \in S : \text{Par}_v(H, S)(x^i) \neq x^\sigma \right\} \right| \leq \left[ \cdots \left[ |S| \left( 1 - \frac{1}{\sigma} \right) \right] \left( 1 - \frac{1}{\sigma} \right) \right] \cdots$
   \hspace{1cm} \underbrace{}_{a \text{ times}}$
   \hspace{1cm} [By [2], [3] and induction.]

5. $\forall (x > 0)$:
   $$\left| x \right| \left( 1 - \frac{1}{\sigma} \right) \leq \left[ x \left( 1 - \frac{1}{\sigma} \right) \right]$$
Proof of 5:

[5.1] Let \( x > 0 \)

| Definition. |

[5.2] \( x \geq [x] \)

| By the properties of the floor function. |

[5.3] \( x \left( 1 - \frac{1}{\sigma} \right) \geq [x] \left( 1 - \frac{1}{\sigma} \right) \)

| Multiplying [5.2] by \( (1 - 1/\sigma) \). |

[5.4] \( x \left( 1 - \frac{1}{\sigma} \right) \geq [x] \left( 1 - \frac{1}{\sigma} \right) \)

| By [5.3] and the properties of the floor function. |

Q.E.D. (5)

[6] \( \left\| \{ x \in S : \text{Par}_x |\sigma(H,S)(x^t) \neq x^v \} \right\| \leq \left[ |S| \left( 1 - \frac{1}{\sigma} \right) \left( 1 - \frac{1}{\sigma} \right) \right] \)

| By (4) and [5]. |

[7] \( \left\| \{ x \in S : \text{Par}_x |\sigma(H,S)(x^t) \neq x^v \} \right\| \leq \left[ |S| \left( 1 - \frac{1}{\sigma} \right)^4 \right] \)

| By combining the \( (1 - 1/\sigma) \) factors in [6]. |

Q.E.D.

Theorem 4A.16: Let \( \text{Par}_x \) be a uniform hypothesis update mechanism, and define

\( \beta \equiv \frac{\sigma}{\sigma - 1} \).

Let \( H = \text{Ran}(\text{Par}_x) \), and let \( S \in \text{Dom}(H)^I, I \in \mathbb{N} \). Then

\( \forall (H \in H) : \left\| \{ x \in S : \text{Par}_x |\sigma(H,S)(x^t) \neq x^v \} \right\| \leq 0. \)
Proof:

\[ \left\{ x \in S : \text{Par}_S^{\log_\beta(|S|)}(H,S)(x^+) \neq x^+ \right\} \subseteq |S| \left(1 - \frac{1}{\sigma}\right)^{\log_\beta(|S|)+1} \]

[By theorem 4A.15.]

\[ |S| \left(1 - \frac{1}{\sigma}\right)^{\log_\beta(|S|)+1} < 1 \]

Proof of 2:

[2.1]
\[ \log_\beta(|S|) < \left[ \log_\beta(|S|) + 1 \right] \]  \hspace{1cm} [By arithmetic.]

[2.2]
\[ |S| < \beta^{\log_\beta(|S|)+1} \]  \hspace{1cm} [Raising both sides of [2.1] to the power of \( \beta \).]

[2.3]
\[ |S| < \left(\frac{\sigma}{\sigma - 1}\right)^{\log_\beta(|S|)+1} \]  \hspace{1cm} [By [2.2] and the definition of \( \beta \).]

[2.4]
\[ |S| \left(\frac{\sigma}{\sigma - 1}\right)^{-\log_\beta(|S|)+1} < 1 \]  \hspace{1cm} [Dividing both sides of [2.3] by \( \left(\frac{\sigma}{\sigma - 1}\right)^{\log_\beta(|S|)+1} \).]

[2.5]
\[ |S| \left(\frac{\sigma}{\sigma - 1}\right)^{\log_\beta(|S|)+1} < 1 \]  \hspace{1cm} [By [2.4] and algebra.]

[2.6]
\[ |S| \left(\frac{\sigma - 1}{\sigma}\right)^{\log_\beta(|S|)+1} < 1 \]  \hspace{1cm} [Expanding the \( (\sigma - 1)/\sigma \) in [2.5].]

[2.7]
\[ |S| \left(\frac{1}{\sigma}\right)^{\log_\beta(|S|)+1} < 1 \]  \hspace{1cm} [Replacing \( \sigma/\sigma \) with 1 in [2.6].]
4A.3. Proof of proposition 4.2

Theorem 4A.17: Let $\text{Par}_{\sigma, \alpha}$ be a sublinear uniform hypothesis update mechanism. Let $S \in \text{Dom} (\text{Ran} (\text{Par}_{\sigma, \alpha}))$, $i \in [I]$. Then

$$\forall (k > 0):$$

(a) $\| \text{Ran} (\text{Par}_{\sigma, \alpha})^k (S < F) \| \leq kr|S|^\alpha$

(b) $\| \text{Ran} (\text{Par}_{\sigma, \alpha})^i (S) \| \leq 2^i |S|^\alpha$.

Proof:

[Proof of part (a)]

[1] $\| \text{Ran} (\text{Par}_{\sigma, \alpha})^0 (S) \| = 1$ [By the definition of a sublinear hypothesis update mechanism.]

[2] $\| \text{Ran} (\text{Par}_{\sigma, \alpha})^i (S) \| \leq r|S|^\alpha + \| \text{Ran} (\text{Par}_{\sigma, \alpha})^{i-1} (S) \|$ [By the definition of a sublinear hypothesis update mechanism.]

[3] $\forall (i > 0):$

$$\| \text{Ran} (\text{Par}_{\sigma, \alpha})^i (S) \| \leq kr|S|^\alpha$$


Q.E.D. [part (a)].
[Proof of part (b)]

[1] Let \( h \equiv \max \left\{ \left\| \text{Ran} \left( \text{Par}_{\sigma, r, a} \right)^{\ell} (S) \right\| : S \in X^\ell \right\} \)

(Definition)

[2] \[
\max \left\{ \left\| \text{Ran} \left( \text{Par}_{\sigma, r, a} \right)^{\ell} (S) \right\| : S \in X^\ell \right\} \leq \max \left\{ \kappa r |S|^a : S \in X^\ell \right\}
\]

[By theorem 4A.17.a.]

[3] \[
\max \left\{ \kappa r |S|^a : S \in X^\ell \right\} \leq \kappa r |S|^a
\]

[Because \( \forall (S \in X^\ell), |S| = \ell \) by definition.]

[4] \( h \leq \kappa r |S|^a \)

[By (1) and (3).]

[5] \[
\left\| \text{Ran} \left( \text{Par}_{\sigma, r, a} \right)^{\ell} \right\| = 2^h
\]

[By the definition of \( h \) in (1) and the definition of \( \| \cdot \|. \)]

[6] \[
\left\| \text{Ran} \left( \text{Par}_{\sigma, r, a} \right)^{\ell} (S) \right\| = 2^{\kappa r |S|^a}
\]

[By (4) and (5).]

Q.E.D. (Part (b))

\[\square\]

**Theorem 4A.18:** Let \( \text{Par}_{\sigma, r, a} \) be a uniform sublinear hypothesis update mechanism, and let \( X = \text{Ran} \left( \text{Par}_{\sigma, r, a} \right) \). Let \( X \) be a sample space. Let \( \ell \in \mathbb{N} \) and let \( \varepsilon \) be a constant, \( 0 < \varepsilon \leq 1 \). Then

\[
P_{X^\ell} \left( S \in X^\ell : P_X \left( \text{Par}_{\hat{\delta}, r, a}^{\left[\|a\|_{\ell+1}\right]} (S, F) (x) \neq F(x) \right) \geq \varepsilon \right) \leq (1 - \varepsilon)^{\ell} 2^{\left(\|a\|_{\ell+1}\right) |S|^a}.
\]
Proof:

[1] \[ \forall (S \in X^t) : \left| \{ x \in S : \text{Par}_{\delta, \sigma, \alpha}^{(t+1)}(S)(x) \neq z^\nu \} \right| \leq 0 \]


[3] \[ \delta \equiv P_X \left( S \in X^t : P_X \left( \text{Par}_{\delta, \sigma, \alpha}^{(t+1)} (S)(x) \neq z^\nu \right) \geq \epsilon \right) \]


[5] \[ \delta \leq (1 - \epsilon)^t \left| \text{Ran} \left( \text{Par}_{\delta, \sigma, \alpha}^{(t+1)} \right) \right| \]


[7] Q.E.D.

Theorem 4A.19: Let Par_{\delta, \sigma, \alpha} be a uniform sublinear hypothesis update mechanism, and let \( \mathcal{H} = \text{Ran} (\text{Par}_{\delta, \sigma, \alpha}) \). Let \( X \) be a sample space. Then the inequality

\[ P_X \left( S \in X^t : P_X \left( \text{Par}_{\delta, \sigma, \alpha}^{(t+1)} (S)(x) \neq z^\nu \right) \geq \epsilon \right) \leq \delta \]

holds whenever

\[ I \geq \max \left\{ \frac{2 \ln(\delta)}{\ln(1 - \epsilon)} \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right)^{\frac{1}{t^2}} \right\} \]

where

\[ A \equiv \frac{2\pi \ln(2)}{-\ln(1 - \epsilon)} \]

and

\[ \beta \equiv \frac{\sigma}{\sigma - 1} \]
Proof:

[1] \[ \ell \geq \max \left\{ \frac{2 \ln(\delta)}{\ln(1 - \epsilon)} \cdot \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right)^{\frac{1}{2\alpha}} \right\} \]

[ Assumption. ]

[2] \[ \ell \geq \frac{2 \ln(\delta)}{\ln(1 - \epsilon)} \]

[ By [1]. ]

[3] \[ \ell \ln(1 - \epsilon) \leq 2 \ln(\delta) \]

[ Multiplying [2] by \ln(1 - \epsilon) (recall that this quantity is negative because of the definition of epsilon). ]

[4] \[ \frac{\ell}{2} \ln(1 - \epsilon) \leq \ln(\delta) \]

[ Dividing [3] by 2. ]

[5] \[ \ln \left( (1 - \epsilon)^{\ell/2} \right) \leq \ln(\delta) \]

[ Moving the \( \ell/2 \) into the natural log in [4]. ]

[6] \[ (1 - \epsilon)^{\ell/2} \leq \delta \]

[ Raising [5] to the power of \( \epsilon \). ]

[7] \[ (1 - \epsilon)^{2 \ln(\epsilon) + 1 - \alpha} \leq (1 - \epsilon)^{\ell/2} \]

Proof of 7:

[7.1] \[ \ell \geq \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right)^{\frac{1}{2\alpha}} \]

[ By [1]. ]

[7.2] \[ \ell \geq \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right)^{\frac{1}{2\alpha}} \rightarrow \ell^{1 - \alpha} \geq \frac{A \ln(\ell)}{\ln(\beta)} + A \]

Proof of 7.2:

[7.2.1] \[ \ell^{1 - \alpha} \geq \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right) \]

[ Raising [7.1] to \( (1 - \alpha) \). ]
\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq 2 \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + 2 \ln(\beta)(1-\alpha) \] 

[ Multiplying \([7.2.1]\) by \(\ln(\beta)(1-\alpha)/A\). ]

\[ \frac{1}{2} \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + \ln(\beta)(1-\alpha) \] 

[ Dividing \([7.2.2]\) by 2. ]

\[ \frac{1}{2} \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \right) \] 

[ By lemma 4A.14. ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + \ln(\beta)(1-\alpha) + \ln \left( \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \right) \] 

[ Adding \([7.2.3]\) and \([7.2.4]\). ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} + \ln(\beta)(1-\alpha) \] 

[ Combining the first and last terms on the right side of \([7.2.5]\). ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \ell^{1-\alpha} \right) + \ln(\beta)(1-\alpha) \] 

[ Cancelling \(A/(\ln(\beta)(1-\alpha))\)'s on the right side of \([7.2.6]\). ]

\[ \ell^{1-\alpha} \geq \frac{A \ln \left( \ell^{1-\alpha} \right)}{\ln(\beta)(1-\alpha)} + A \] 

[ Multiplying \([7.2.7]\) by \(A/(\ln(\beta)(1-\alpha))\). ]

\[ \ell^{1-\alpha} \geq A(1-\alpha) \ln(t) \] 

[ Extracting \((1-\alpha)\) from the natural log on the right side of \([7.2.8]\). ]

\[ \ell^{1-\alpha} \geq A \ln(t) + A \] 

[ Cancelling \((1-\alpha)\)'s on the right side of \([7.2.9]\). ]

Q.E.D. (7.2)

\[ \ell^{1-\alpha} \geq \frac{A \ln(t)}{\ln(\beta)} + A \] 

[ By \([7.1]\) and \([7.2]\). ]
\[ T - \varepsilon \geq \frac{2 \ln(2) \ln(T) - \ln(1 - \varepsilon) \ln(\beta)}{\ln(1 - \varepsilon) \ln(\beta)} + \frac{2 r \ln(2)}{-\ln(1 - \varepsilon)} \]

\[ T \geq \frac{2 \ln(2) r \ln(\ell)^e}{-\ln(1 - \varepsilon) \ln(\beta)} + \frac{2 r \ln(2) e^e}{-\ln(1 - \varepsilon)} \]

\[ \frac{T \ln(1 - \varepsilon)}{2 \ln(2)} \geq \left( \frac{\ln(\ell)}{\ln(\beta)} + 1 \right) r e^e \]

\[ \frac{T \ln(1 - \varepsilon)}{2 \ln(2)} \geq (\log_\beta(\ell) + 1) r e^e \]

\[ \frac{T \ln(1 - \varepsilon)}{2 \ln(2)} \geq (\log_\beta(\ell) + 1) r e^e \ln(2) \]

\[ \ln \left( (1 - \varepsilon)^{\ell/2} \right) \geq (\log_\beta(\ell) + 1) r e^e \ln(2) \]

\[ \ln \left( (1 - \varepsilon)^{\ell/2} \right) \geq \ln \left( 2^{(\log_\beta(\ell) + 1) r e^e} \right) \]

\[ (1 - \varepsilon)^{\ell/2} \geq 2^{(\log_\beta(\ell) + 1) r e^e} \]

\[ (1 - \varepsilon)^{\ell/2} \geq 2^{(\log_\beta(\ell) + 1) r e^e (1 - \varepsilon)^e} \]

\[ (1 - \varepsilon)^{\ell/2} \geq 2^{(\log_\beta(\ell) + 1) r e^e (1 - \varepsilon)^e} \]

Q.E.D. (7)

\[ (1 - \varepsilon)^{\ell/2} [\log_\beta(\ell) + 1] r e^e \leq \delta \]

[ Replacing A with \( 2(r \ln(2))/(-\ln(1 - \varepsilon)) \) in (7.3), according to the definition of A. ]

[ Multiplying (7.4) by \( e^e \). ]

[ Combining terms on the right side of (7.5). ]

[ Multiplying (7.6) by \(-\ln(1 - \varepsilon)/\ln(2)\). ]

[ Replacing \( \ln(\ell)/\ln(\beta) \) with \( \log_\beta(\ell) \) in (7.7). ]

[ Multiplying (7.8) by \( \ln(2) \). ]

[ Moving \( \ell/2 \) inside the natural log on the left side of (7.9). ]

[ Moving the \( \log_\beta(\ell) + 1 \) \( r e^e \) inside the natural log on the right side of (7.10). ]

[ Raising (7.11) to the power of \( e \). ]

[ Multiplying (7.12) by \( (1 - \varepsilon)^e \). ]

[ By (7.13), since \( \log_\beta(\ell) + 1 > 1 \). ]

[ Replacing \( (1 - \varepsilon)^{\ell/2} \) with \( \delta \) in (7), as per (6). ]
\[ P_{X^t} \left( \mathcal{S} \in X^t : P_X \left( \text{Par}_{\varphi, \lambda, \alpha} \left( \mathcal{S} \right) (x^t) \neq x^w \right) \geq \epsilon \right) \leq (1 - \epsilon) \epsilon 2^t 2^{(\log t + 1) t^2} \]

[By theorem 4A.18.]

\[ P_{X^t} \left( \mathcal{S} \in X^t : P_X \left( \text{Par}_{\varphi, \lambda, \alpha} \left( \mathcal{S} \right) (x^t) \neq x^w \right) \geq \epsilon \right) \leq \delta \]

[Substituting \( \delta \) for \( (1 - \epsilon) \epsilon 2^t 2^{(\log t + 1) t^2} \) in [9], as per [8].]

\[ t \geq \max \left\{ \frac{2 \ln(\delta)}{\ln(1 - \epsilon)} \left( \frac{2A}{\ln(\beta)(1 - \alpha)} + 2A \right)^{1/2} \right\} \rightarrow \]

\[ P_{X^t} \left( \mathcal{S} \in X^t : P_X \left( \text{Par}_{\varphi, \lambda, \alpha} \left( \mathcal{S} \right) (x^t) \neq x^w \right) \geq \epsilon \right) \leq \delta \]

[By [11] and [10].]

Q.E.D.

4A.4. Proof of proposition 4.4

**Theorem 4A.20:** Let \( \text{Par}_{\alpha, \lambda, \alpha} \) be a sublinear hypothesis update mechanism, and let \( \mathcal{X} = \text{Ran} \left( \text{Par}_{\alpha, \lambda, \alpha} \right) \). Let \( X \) be a sample space. Let \( S \in X^t, t \in \mathbb{N} \), and let \( k \) be such that, for some constant \( 0 \leq \phi \leq 1 \),

\[ \left| \{x \in S : \text{Par}_{\alpha, \lambda, \alpha}^k (S) (x^t) \neq x^w \} \right| > \phi t + 1. \]

Then

\[ k < \left\lfloor \frac{\ln(\phi + 1/t)}{\ln(1 - 1/\alpha)} \right\rfloor. \]
Proof:

1. \[ \left\{ x \in S : \text{Par}_{\sigma, \alpha}^k(S)(x') \neq x'' \right\} > \phi \ell + 1 \quad \text{[Assumption.]} \]

2. \[ \left\{ x \in S : \text{Par}_{\sigma, \alpha}^k(S)(x') \neq x'' \right\} \leq \ell \left(1 - \frac{1}{\sigma} \right)^4 \quad \text{[by theorem 4A.15.]}

3. \[ \ell \left(1 - \frac{1}{\sigma} \right)^4 > \phi \ell + 1 \quad \text{[By 1 and 2.]} \]

4. \[ \ell \left(1 - \frac{1}{\sigma} \right)^k > \phi + \frac{1}{\ell} \quad \text{[By 3 and the definition of the floor function.]} \]

5. \[ \left(1 - \frac{1}{\sigma} \right)^k > \phi + \frac{1}{\ell} \quad \text{[Dividing 4 by \ell.]} \]

6. \[ \ln \left( \left(1 - \frac{1}{\sigma} \right)^k \right) > \ln \left( \phi + \frac{1}{\ell} \right) \quad \text{[By 5 and the monotonicity of the log function.]} \]

7. \[ k \ln \left(1 - \frac{1}{\sigma} \right) > \ln \left( \phi + \frac{1}{\ell} \right) \quad \text{[Extracting \( k \) from the logarithm on the left side of 6.]} \]

8. \[ k \ln \left(1 - \frac{1}{\sigma} \right) > \ln \left( \phi + \frac{1}{\ell} \right) + \ln \left(1 - \frac{1}{\sigma} \right) \quad \text{[By 7, since \( \ln(1 - 1/\sigma) \) is negative.]} \]

9. \[ k < \frac{\ln \left( \phi + 1/\ell \right)}{\ln \left(1 - 1/\sigma \right)} + 1 \quad \text{[Dividing 8 by \( \ln(1 - 1/\sigma) \) (recall that \( \ln(1 - 1/\sigma) \) is negative).]} \]

10. \[ k < \left[ \frac{\ln \left( \phi + 1/\ell \right)}{\ln \left(1 - 1/\sigma \right)} \right] \quad \text{[By 9.]} \]

Q.E.D.

**Theorem 4A.21:** Let \( \text{Par}_{\sigma, \alpha} \) be a sublinear hypothesis update mechanism, and let \( M = \text{Ran}(\text{Par}_{\sigma, \alpha}) \). Let
$X$ be a sample space. Let $S \in X^t$, $t \in \mathbb{N}$, let $\phi$ be a constant, $0 \leq \phi \leq 1$. Let

$$k \equiv \left\lfloor \frac{\ln(\phi + 1/t)}{\ln(1 - 1/\sigma)} \right\rfloor.$$

Then

$$\left| \{ z \in S : \text{Par}^t_{\sigma, r, a}(S)(z^t) \neq z^\omega \} \right| \leq \phi t + 1.$$

Proof:

[1] \hspace{1cm} \forall (i > 0) : \hspace{1cm} $\left| \{ z \in S : \text{Par}^t_{\sigma, r, a}(S)(z^t) \neq z^\omega \} \right| > \phi t + 1 \rightarrow \left( i < \frac{\ln(\phi + 1/t)}{\ln(1 - 1/\sigma)} \right)$

[ By theorem 4A.20. ]

[2] \hspace{1cm} $-\left( k < \frac{\ln(\phi + 1/t)}{\ln(1 - 1/\sigma)} \right)$ \hspace{1cm} [ By the definition of $k$. ]

[3] \hspace{1cm} $-\left( \left| \{ z \in S : \text{Par}^t_{\sigma, r, a}(S)(z^t) \neq z^\omega \} \right| > \phi t + 1 \right)$ \hspace{1cm} [ By [1] and [2]. ]

[4] \hspace{1cm} $\left| \{ z \in S : \text{Par}^t_{\sigma, r, a}(S)(z^t) \neq z^\omega \} \right| \leq \phi t + 1$ \hspace{1cm} [ By [3]. ]

Q.E.D.

Theorem 4A.22: Let $\text{Par}_{\sigma, r, a}$ be a conservative hypothesis update mechanism, and let $\mathcal{X} = \text{Ran}(\text{Par}_{\sigma, r, a})$. Let $X$ be a sample space. Let $\xi, \beta, \phi$ be constants, $0 \leq \xi, \beta, \phi < 1$, and let $t \in \mathbb{N}$. Define

$$k \equiv \max \left\{ \min \{ i : \left| \{ z \in S : \text{Par}^t_{\sigma, r, a}(S)(z^t) \neq z^\omega \} \right| \leq \phi t + 1 \} : S \in X^t \}.$$

Then the inequality

$$P_X(S \in X^t : P_X(z : \text{Par}^t_{\sigma, r, a}(S)(z^t) \neq z^\omega) > \phi + t(\xi) \leq \delta$$

holds whenever

$$t \geq \max \left\{ \frac{1}{t} \cdot \frac{1}{\ln(2)} \cdot \ln(\xi) \cdot \ln((1 - 1/\sigma)/r) \ln(2) \right\}^{1/3} \left\{ \frac{1}{\xi^2} \cdot \frac{1}{(\ln(2)^{-1} - 1)} \cdot \left( \frac{-2 \ln(-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)/r \ln(2))}{-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)/r \ln(2)} \right)^{1/3} \right\}.$$
Proof:

\[ t \geq \max \left\{ \frac{1}{\xi^2}, \frac{(2\log_2 (-\xi^2(1 - \alpha)\ln(1 - 1/\alpha)/r \ln(2)))^{1/2}}{\xi^2 (\ln(2) - 1)}, \left( \frac{-2\log_2 (-\xi^2(1 - \alpha)\ln(1 - 1/\alpha)/r \ln(2))}{} \right)^{1/2} \right\} \]

[ Assumption. ]

\[ \delta \geq 2e^{-\xi^2} \]

Proof of 2:

\[ t \geq \frac{\ln(\delta) - \ln(2)}{-\xi^2} \] [ By [1]. ]

\[ -t\xi^2 \leq \ln(\delta) - \ln(2) \] [ Multiplying [2.1] by $-\xi^2$. ]

\[ -t\xi^2 \leq \ln \left( \frac{\delta}{2} \right) \] [ Combining the two terms on the right side of [2.2]. ]

\[ e^{-t\xi^2} \leq \frac{\delta}{2} \] [ Raising both sides of [2.3] to the power of $e$. ]

\[ 2e^{-t\xi^2} \leq \delta \] [ Multiplying [2.4] by 2. ]

Q.E.D. (2)

\[ k \leq \left\lfloor \frac{\ln(\phi + 1/t)}{\ln(1 - 1/\alpha)} \right\rfloor \] [3]
Proof of 3:

[3.1] \[
    k \equiv \max \{ \min \{ i : \{|x \in S : \text{Par}_s^{\phi,\alpha}(S)(x^i) \neq x^*\}| \leq \phi + 1 : S \in X^i \} \}.
\]
[ By definition. ]

[3.2] \[
    \forall (S \in X^i) : \left\{ \left\{ z \in S : \text{Par}_s^{\phi,\alpha}(S)(S)(x^i) \neq x^* \right\} \right\} \leq \phi + 1
\]
[ By theorem 4A.21. ]

[3.3] \[
    \forall (S \in X^i) : \min \{ m : \{|x \in S : \text{Par}_s^{\phi,\alpha}(S)(x^i) \neq x^*\}| \leq \phi + 1 \} \leq \left( \frac{\ln(\phi + 1/\ell)}{\ln(1 - 1/\sigma)} \right)
\]
[ By [3.2]. ]

[3.4] \[
    k \leq \max \left\{ \left( \frac{\ln(\phi + 1/\ell)}{\ln(1 - 1/\sigma)} \right) : S \in X^i \right\}
\]
[ By [3.3] and [3.1]. ]

[3.5] \[
    k \leq \left( \frac{\ln(\phi + 1/\ell)}{\ln(1 - 1/\sigma)} \right)
\]
[ By [3.4]. ]

Q.E.D. (3)

[4] \[
    |\text{Ran} (\text{Par}_s^{\phi,\alpha})| \leq 2^{1 + \sigma
\]
[ By theorem 4A.17. ]

[5] \[
    |\text{Par}_s^{\phi,\alpha}| \leq 2^{\left( \frac{\ln(\phi + 1/\ell)}{\ln(1 - 1/\sigma)} \right) + \sigma
\]
[ By [3] and [4]. ]

[6] \[
    P_{X^i} (S \in X^i : P_X (x : \text{Par}_s^{\phi,\alpha}(S)(x^i) \neq x^*) > \phi + \ell \xi) \leq 2e^{3/4\ell} \frac{\ln(\phi + 1/\ell)}{\ln(1 - 1/\sigma)} \frac{\text{Ran} (\text{Par}_s^{\phi,\alpha})}{\sigma}
\]
Proof of 6:

[6.1] \( \forall (S \subseteq X^t) : \)
\[ |\{ x \in S : \text{Par}_{\sigma,\rho}(S)(x^t) \neq x^u \}| \leq \phi \ell + 1 \]

[By the definition of \( k \), it must satisfy this inequality.]

[6.2] \( \forall (S \subseteq X^t) : \)
\[ \eta(\text{Par}_{\sigma,\rho}(S), S) = \frac{1}{t} |\{ x \in S : \text{Par}_{\sigma,\rho}(S)(x^t) \neq x^u \}| \]

[By the definition of \( \eta \) (empirical error).]

[6.3] \( \forall (S \subseteq X^t) : \)
\[ \eta(\text{Par}_{\sigma,\rho}(S), S) \leq \frac{\phi \ell + 1}{t} \]

[By [6.1] and 16.21.]

[6.4] \( P_{X^t} \left( S \subseteq X^t : P_X (z : \text{Par}_{\sigma,\rho}(S)(z^t) \neq x^u) > \frac{\phi \ell + 1}{t} + \xi \right) \leq 2e^{-2t\xi} |\text{Ran} (\text{Par}_{\sigma,\rho})| \]

[By lemma 4A.13.]

[6.5] \( \phi + t \xi \geq \frac{\phi \ell + 1}{t} + \xi \)

Proof of 6.5:

[6.5.1] \( t \geq \frac{1}{(t - 1)\xi} \)

[By 11.1]

[6.5.2] \( (t - 1)\xi \geq \frac{1}{t} \)

[By [6.5.1] and algebra.]

[6.5.3] \( t\xi - \xi \geq \frac{1}{t} \)

[By [6.5.2] and algebra.]

[6.5.4] \( t\xi \geq \frac{1}{t} + \xi \)

[By [6.5.3] and algebra.]
\[\phi + 1 \xi \geq \frac{\phi}{\ell} + \xi \quad \text{[By [6.5.4] and algebra.]}\]

\[\phi + 1 \xi \geq \frac{\phi + 1}{\ell} + \xi \quad \text{[By [6.5.5] and algebra.]}\]

Q.E.D. (6.5)

\[P_{X_1}(S \in X': P_X(x: Par_{\tau,\omega}(S)(x') \neq x') > \phi + \xi) \leq 2e^{-2\ell^2|\text{Ran}\{Par_{\tau,\omega}(S)\}|}\]

[By [6.4] and [6.5] (since, by [6.5], fewer \(S \in X'\) satisfy the left side of [6.6] than satisfy the left side of [6.4]).]

Q.E.D. (6)

\[\ln(\phi - 1/\ell) = \ln((\phi/\ell) - 1) - \ln(\ell)\]

Proof of 8:

\[\ln(\phi - 1/\ell) = \ln\left(\frac{\phi + 1}{\ell}\right)\quad \text{[By algebra.]}\]

\[\ln(\phi - 1/\ell) = \ln\left(\frac{1}{\ell}(\phi + 1)\right)\quad \text{[By [8.1] and algebra.]}\]

\[\ln(\phi - 1/\ell) = \ln(\phi + 1) + \ln\left(\frac{1}{\ell}\right)\quad \text{[By [8.2].]}\]

\[\ln(\phi - 1/\ell) = \ln(\phi + 1) + \ln(\ell^{-1})\quad \text{[Rewriting [8.3].]}\]

\[\ln(\phi - 1/\ell) = \ln(\phi + 1) - \ln(\ell)\quad \text{[Extracting -1 from the rightmost logarithm in [8.4].]}\]
Q.E.D. (8)

[9]

\[ P_X(S \in X^t : P_{\alpha \beta}(x : \text{Par}(S))(x') \neq x^o) > \phi + \xi \leq 2e^{-2e^{t^2 \frac{\ln(\phi + \xi) - \ln(\xi)}{\ln(1 - 1/\sigma)}}} \]

[By [7] and [8].]

[10]

\[ 2e^{-2e^{t^2 \frac{\ln(\phi + \xi) - \ln(\xi)}{\ln(1 - 1/\sigma)}}} \leq 2 - t^2 \]

Proof of 10:

[10.1]

\[ \frac{\xi^2 \ell^{1-\alpha}}{\ln(2)} \geq \frac{1}{\ell^2} + \xi^2 \ell^{1-\alpha} \]

Proof of 10.1:

[10.1.1]

\[ \ell \geq \frac{1}{\xi^2 \left( \frac{1}{\ln(2)} - 1 \right)} \quad \text{[By [1].]} \]

[10.1.2]

\[ \ell \xi^2 \left( \frac{1}{\ln(2)} - 1 \right) \geq 1 \quad \text{[Multiplying [10.1.1] by } \xi^2 \left( \frac{1}{\ln(2)} - 1 \right).] \]

[10.1.3]

\[ \frac{\ell \xi^2}{\ln(2)} - \ell \xi^2 \geq 1 \quad \text{[Distributing the } \ell \xi^2 \text{ on the left side of [10.1.2].]} \]

[10.1.4]

\[ \frac{\ell \xi^2}{\ln(2)} \geq 1 + \ell \xi^2 \quad \text{[Adding } \ell \xi^2 \text{ to [10.1.3].]} \]

[10.1.5]

\[ \frac{\ell^{1-\alpha}}{\ln(2)} \geq \frac{1}{\ell^2} + \ell^{1-\alpha} \xi^2 \quad \text{[Multiplying [10.1.4] by } \ell^{1-\alpha}.] \]

Q.E.D. (10.1)

[10.2]

\[ \frac{\xi^2 \ell^{1-\alpha}}{\ln(2)} \geq \frac{\ln(\phi + \xi) - \ln(\xi)}{\ln(1 - 1/\sigma)} \]
Proof of 10.2:

[10.2.1] \( \epsilon \geq \left( \frac{-2 \ln (-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)/r \ln(2))}{-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)/r \ln(2)} \right)^{-1/n} \)

[10.2.2] \( \epsilon^{1 - \alpha} \geq \left( \frac{-2 \ln (-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)/r \ln(2))}{-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)/r \ln(2)} \right) \)

[10.2.3] \( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{2 r \ln(2)} \geq -\ln \left( \frac{-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \right) \)

[10.2.4] \( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{2 r \ln(2)} \geq -\ln \left( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \right) \)

[10.2.5] \( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \geq -\ln \left( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \right) \) - \ln \left( \frac{-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \right)

[10.2.6] \( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \geq -\ln \left( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \right) \left( \frac{-\xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \right)^{-1} \)

[10.2.7] \( \frac{\epsilon^{1 - \alpha} - \xi^2(1 - \alpha) \ln(1 - 1/\sigma)}{r \ln(2)} \geq \ln (\epsilon^{1 - \alpha}) \)
\[ \ell^{\frac{1 - \alpha}{\beta}} \geq \frac{r \ln(\ell^{1 - \alpha})}{\ln(1 - \frac{1}{\beta})} \quad \text{(Multiplying \([10.2.7]\) by \((-r \ln(2))/((1 - \alpha) \ln(1 - 1/\beta))\).)} \]

\[ \ell^{\frac{1 - \alpha}{\beta}} \geq \frac{r \ln(\ell)(1 - \alpha)}{(1 - \alpha) \ln(1 - 1/\beta)} \quad \text{(Extracting \((1 - \alpha)\) from the logarithm on the right side of \([10.2.8]\).)} \]

\[ \ell^{\frac{1 - \alpha}{\beta}} \geq \frac{r \ln(\ell)}{\ln(1 - 1/\beta)} \quad \text{(Cancelling \((1 - \alpha)\) factors on the right side of \([10.2.9]\).)} \]

\[ \ell^{\frac{1 - \alpha}{\beta}} \geq \frac{r \ln(\ell)}{\ln(1 - 1/\beta)} + \frac{r \ln(\ell^{1} + 1)}{\ln(1 - 1/\beta)} \quad \text{(By \([10.2.10]\), since \((r \ln(a + 1))/(\ln(1 - 1/\beta))\) is negative.)} \]

\[ \ell^{\frac{1 - \alpha}{\beta}} \geq \frac{r}{\ln(1 - 1/\beta)} \left(\ln(\ell^{1} + 1) - \ln(\ell)\right) \quad \text{(Combining \(r/\ln(1 - 1/\beta)\) terms in \([10.2.11]\).)} \]

Q.E.D. \((10.2)\)

\[ 2 \ell^{\frac{\ell^{1} - \alpha}{\beta}} \geq \frac{1}{\ell^{\alpha}} + \xi^{2} \ell^{1 - \alpha} + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(Adding \([10.1]\) and \([10.2]\).)} \]

\[ 2 \ell^{\frac{\ell^{1}}{\beta}} \geq 1 + \xi^{2} \ell + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(Multiplying \([10.3]\) by \(\ell^{\alpha}\).)} \]

\[ 2 \ell^{\frac{\ell^{1}}{\beta}} \geq 1 + \xi^{2} \ell + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(Multiplying \([10.4]\) by \(\ln(\ell) = 1\).)} \]

\[ 2 \log_{2}(\ell^{\alpha}) \xi^{2} \ell \geq 1 + \xi^{2} \ell + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(By \([10.5]\).)} \]

\[ 2 \log_{2}(\ell) \xi^{2} \ell - \xi^{2} \ell \geq 1 + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(Adding \(\xi^{2} \ell\) to \([10.6]\).)} \]

\[ -\xi^{2} \ell \geq 1 - 2 \log_{2}(\ell^{\alpha}) \xi^{2} \ell + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(Subtracting \(2 \log_{2}(\ell^{\alpha}) \xi^{2} \ell\) from \([10.7]\).)} \]

\[ -\xi^{2} \ell \geq 1 + \log_{2}(\ell^{-2} \xi^{2} \ell) + \frac{\ln(\ell^{1} + 1) - \ln(\ell)}{\ln(1 - 1/\beta)} r \ell^{\alpha} \quad \text{(Moving \(-2 \xi^{2} \ell\) into the logarithm in \([10.8]\).)} \]
\[ 2^{-t/2} \geq 2^{\left(\frac{R^2}{2}\right)} + 2^{\left(\frac{R^2}{2}\right)} \delta \]  

[ Raising 10.9 to the power of 2. ]

Q.E.D. (10)

\[ 2^{-t/2} \left( \frac{\ln(\frac{t}{2})}{\ln(\frac{t}{2})} \right) \delta \leq \delta \]  

[ By (2) and (10). ]

(11)

\[ P_X \{ S \in X^t : P_X \{ x : \text{Par}_{\sigma,\alpha}(S)(x^t) \neq x^t \} > \phi + \xi \} \leq \delta \]  

[ By (9) and (11). ]

Q.E.D. (11)

\[ \Box \]

4A.5. Proof of proposition 4.5

Theorem 4A.23: Let \( \text{Par}_{\sigma,\alpha} \) be a conservative hypothesis update mechanism, and let \( X' = \text{Ran}(\text{Par}_{\sigma,\alpha}) \).

Let \( X \) be a sample space. Let \( \xi, \delta, \phi \) be constants, \( 0 \leq \xi, \delta, \phi < 1 \), and let \( t \in \mathbb{N} \). Define

\[ \beta \equiv \frac{\sigma}{\sigma - 1}. \]

Let \( t \in \mathbb{N} \). Then

\[ P_X \{ S \in X^t : \left| \eta \left( \text{Par}_{\sigma,\alpha}(S) \right) - \eta \left( \text{Par}_{\sigma,\alpha}(S), S \right) \left| \geq \xi \right. \} \leq \delta \]

whenever

\[ t \geq \max \left\{ \frac{\ln(\delta) - \ln(2)}{-\xi^2}, \left( \frac{2A}{\ln(\beta)(1-\alpha)} \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + 2A \right)^{\frac{1}{\xi^2}} \right\} \]

where

\[ A \equiv \frac{r}{\xi^2} \log_2(e). \]
Proof:

\[ t \geq \max \left\{ \frac{\ln(\delta) - \ln(2)}{-\xi^2}, \left( \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \right)^{\frac{1}{1+\alpha}} \right\} \]

[ Assumption. ]

\[ 2e^{-c_1 t} \leq \delta \]

Proof of 2:

\[ \frac{\ln(\delta) - \ln(2)}{-\xi^2} \]

[ By [1]. ]

\[ -\xi^2 t \leq \ln(\delta) - \ln(2) \]

[ Multiplying [2.1] by \( -\xi^2 \). ]

\[ -\xi^2 t + \ln(2) \leq \ln(\delta) \]

[ Adding \( \ln(2) \) to [2.2]. ]

\[ 2e^{-c_1 t} \leq \delta \]

[ Exponentiating both sides of [2.3]. ]

Q.E.D. (2)

\[ 2e^{-2c_1 t} \leq e^{-\xi^2 t} \]

Proof of 3:

\[ t^\alpha \geq \frac{A \ln(t)}{\ln(\beta)} + A \]

Proof of 3.1:

\[ \frac{2A}{\ln(\beta)(1 - \alpha)} \ln \left( \frac{A}{\ln(\beta)(1 - \alpha)} \right) + 2A \]

[ By [1]. ]
\[ \ell^{1-\alpha} \geq \frac{2A}{\ln(\beta)(1-\alpha)} \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + 2A \]

[ Raising \([3.1.1]\) to the power of \(1-\alpha\). ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq 2 \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + 2 \ln(\beta)(1-\alpha) \]

[ Multiplying \([3.1.2]\) by \(\ln(\beta)(1-\alpha)/A\). ]

\[ \frac{1}{2} \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + \ln(\beta)(1-\alpha) \]

[ Dividing \([3.1.3]\) by 2. ]

\[ \frac{1}{2} \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \right) \]

[ By lemma 4A.14. ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \right) + \ln \left( \frac{A}{\ln(\beta)(1-\alpha)} \right) + \ln(\beta)(1-\alpha) \]

[ Adding \([3.1.4]\) and \([3.1.5]\). ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln \left( \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \right) + \ln(\beta)(1-\alpha) \]

[ Combining the first two natural logarithms on the right side of \([3.1.6]\). ]

\[ \frac{\ell^{1-\alpha} \ln(\beta)(1-\alpha)}{A} \geq \ln (\ell^{1-\alpha}) + \ln(\beta)(1-\alpha) \]

[ Cancelling factors inside the first natural logarithm on the right side of \([3.1.7]\). ]

\[ \ell^{1-\alpha} \geq \frac{A \ln (\ell^{1-\alpha})}{\ln(\beta)(1-\alpha)} + A \]

[ Multiplying \([3.1.8]\) by \(A/\ln(\beta)(1-\alpha)\). ]

\[ \ell^{1-\alpha} \geq \frac{A \ln (\ell)(1-\alpha)}{\ln(\beta)(1-\alpha)} + A \]

[ extracting \((1-\alpha)\) from the logarithm in the numerator of the first term in the right side of \([3.1.9]\). ]

\[ \ell^{1-\alpha} \geq \frac{A \ln (\ell)}{\ln(\beta)} + A \]

[ Cancelling \((1-\alpha)\)'s in \([3.1.10]\). ]

Q.E.D. (3.1)
By \([3.1]\) and the definition of \(A^*\).]

\[ t^{1 - \alpha} \geq \frac{\tau \ln(t)}{\xi^2 \log_2(\tau \ln(\beta))} + \frac{\tau}{\xi^2 \log_2(e)} \]

[By \([3.2]\) by \(t^\alpha\).]

\[ t \geq \frac{\tau \ln(t)}{\xi^2 \log_2(\tau \ln(\beta))} + \frac{\tau^\alpha}{\xi^2 \log_2(e)} \]

[Replacing \(\ln(t)/\ln(\beta)\) with \(\log_\beta(t)\) in \([3.3]\).]

\[ t \geq \frac{\tau \log_\beta(t) + 1}{\xi^2 \log_2(e)} \]

[Combining terms on the right side of \([3.4]\).]

\[ t \geq \frac{\tau \log_\beta(t) + 1}{\log_2(e)} \]

[By \([3.5]\) by \(\xi^2\).]

\[ \zeta^2 \geq \frac{\tau \log_\beta(t) + 1}{\log_2(e)} \]

[Subtracting \(2t\zeta^2\) from \([3.6]\).]

\[ -\zeta^2 \geq \frac{\log_2 \left( \frac{2^{\log_\beta(t) + 1} e^{\tau t}}{e} \right)}{\log_2(e)} - 2t\zeta^2 \]

[Exponentiating \(\tau \log_\beta(t) + 1\) by 2, and then taking the base 2 logarithm of this quantity, in \([3.7]\).]

\[ -\zeta^2 \geq \log_2 \left( \frac{2^{\log_\beta(t) + 1} e^{\tau t}}{e} \right) - 2t\zeta^2 \]

[Changing the base of the logarithm to \(e\) in \([3.8]\).]

\[ -\zeta^2 \geq \ln \left( \frac{2^{\log_\beta(t) + 1} e^{\tau t}}{e} \right) - 2t\zeta^2 \]

[Exponentiating \([3.9]\) by \(e\).]

\[ e^{-t\zeta} \geq \frac{2^{\log_\beta(t) + 1} e^{\tau t}}{e} e^{-2t\zeta^2} \]

[The inequality in \([3.10]\) remains valid when the left side is multiplied by 2.]

\[ 2e^{-t\zeta} \geq \frac{2^{\log_\beta(t) + 1} e^{\tau t}}{e} e^{-2t\zeta^2} \]

Q.E.D. (3)

\[ 2e^{-t\zeta} 2^{\log_\beta(t) + 1} e^{\tau t} \leq \delta \]

[By \([2]\) and \([3]\).]
Proof of proposition 4.7

**Theorem 4A.24:** Let $\text{Add}_{\alpha, \tau}$ be an additive hypothesis update mechanism, and let $S \in \text{Dom}(\text{Ran}(\text{Add}_{\alpha, \tau}))^f$, $f \in \mathbb{N}$. Let $k$ be a positive integer. Then

$$|\text{Ran}(\text{Add}_{\alpha, \tau}^k)| \leq kr.$$ 

**Proof:**

1. By the definition of an additive hypothesis update mechanism.

2. \( \forall (i \geq 0): \)

   \((|\text{Ran}(\text{Add}_{\alpha, \tau}^i)| = n) \to (|\text{Ran}(\text{Add}_{\alpha, \tau}^{i+1})| = n + r) \)

   By the definition of an additive hypothesis update mechanism.
Q.E.D. [The claim follows from [1] and [2] by induction.]

Theorem 4A.25: Let $\text{Add}_{e,r}$ be an additive hypothesis update mechanism, $X$ be a sample space, $t \in \mathbb{N}$. Let $\epsilon$ and $\delta$ be constants, $0 < \epsilon, \delta < 1$, and let $2 \geq \beta > 1$ be a constant. Then

$$P_{X^t} \left( S : P_X \left( x : \text{Add}_{e,r}^{\log_2(t)}(S)(x') \neq x^{\omega} \right) \geq \epsilon \right) \leq \delta$$

holds whenever

$$t \geq \max \left\{ \epsilon, \frac{\ln(\delta/2) - \ln(\epsilon)}{\ln(1 - \epsilon)}, \frac{2(\ln(\epsilon) - \ln(\delta/2) - \ln(\log_2(t)))}{-\ln(1 - \epsilon)}, \frac{2\ln(1/2\ln(\beta))}{\ln(1 - \epsilon)} \right\}.$$  

Proof:

[1]

$$t \geq \max \left\{ \epsilon, \frac{\ln(\delta/2) - \ln(\epsilon)}{\ln(1 - \epsilon)}, \frac{2(\ln(\epsilon) - \ln(\delta/2) - \ln(\log_2(t)))}{-\ln(1 - \epsilon)}, \frac{2\ln(1/2\ln(\beta))}{\ln(1 - \epsilon)} \right\}.$$  

[2]

$$\left| \text{Ran} \left( \text{Add}_{e,r}^{\log_2(t)} \right) \right| \leq r \left( \log_2(t) + 1 \right)$$  

[ By theorem 4A.24. ]

[3]

$$P_{X^t} \left( S : P_X \left( x : \text{Add}_{e,r}^{\log_2(t)+1}(S)(x') \neq x^{\omega} \right) \geq \epsilon \right) \leq (1 - \epsilon)^t \left| \text{Ran} \left( \text{Add}_{e,r}^{\log_2(t)+1} \right) \right|$$  

[ By lemma 4A.11. ]

[4]

$$P_{X^t} \left( S : P_X \left( x : \text{Add}_{e,r}^{\log_2(t)+1}(S)(x') \neq x^{\omega} \right) \geq \epsilon \right) \leq (1 - \epsilon)^t r \left( \log_2(t) + 1 \right)$$  

[ By [2] and [3]. ]

[5]

$$(1 - \epsilon)^t r \left( \log_2(t) + 1 \right) \leq \delta$$
Proof of 5:

[5.1] \[ r \ln(t)(1 - \varepsilon) \leq \frac{\delta}{2} \ln(\beta) \]

Proof of 5.1:

[5.1.1] \[ \delta(1 - \varepsilon)^{1/2} \leq \frac{\delta}{2} \ln(\beta) \]

Proof of 5.1.1:

[5.1.1.1] \[ t \geq \frac{2\ln(1/2\ln(\beta))}{\ln(1 - \varepsilon)} \] [By [1].]

[5.1.1.2] \[ \frac{t}{2} \geq \frac{\ln(1/2\ln(\beta))}{\ln(1 - \varepsilon)} \] [Dividing [5.1.1.1] by 2.]

[5.1.1.3] \[ \frac{t}{2} \ln(1 - \varepsilon) \leq \ln\left(\frac{1}{2\ln(\beta)}\right) \] [Multiplying [5.1.1.2] by \(\ln(1 - \varepsilon)\).]

[5.1.1.4] \[ (1 - \varepsilon)^{1/2} \leq \frac{1}{2} \ln(\beta) \] [Exponentiating [5.1.1.3] by \(e\).]

[5.1.1.5] \[\delta(1 - \varepsilon)^{1/2} \leq \frac{\delta}{2} \ln(\beta) \] [Multiplying [5.1.1.4] by \(\delta\).]

Q.E.D. (5.1.1)

[5.1.2] \[ r \ln(t)(1 - \varepsilon) \leq \delta(1 - \varepsilon)^{1/2} \]

Proof of 5.1.2:

[5.1.2.1] \[ \frac{\ln\left(\frac{\ln(t)}{\sqrt{r}}\right)}{\ln(1 - \varepsilon)} \leq \frac{t}{2} \]
Proof of 5.1.2.1:

\[ \epsilon = 1, \ell = \epsilon \to - \frac{\ln \left( \frac{\ell}{\ell} \right)}{\ln(1 - \epsilon)} \leq \frac{\ell}{2} \]

[ By arithmetic. ]

\[ \frac{d}{dc} \left( - \frac{\ln \left( \frac{\ell}{\ell} \right)}{\ln(1 - \epsilon)} \right) = \]

\[ - \frac{1}{1 - \epsilon} \ln \left( \frac{\ln(\ell)}{\delta/\rho} \right) = \]

\[ \frac{1}{1 - \epsilon} \ln \left( \frac{\ln(\ell)}{\delta/\rho} \right) \]

[5.1.2.3]

\[ \frac{d \ell}{dc} = 0 \]

[5.1.2.1.4]

\[ \frac{1}{1 - \epsilon} \ln \left( \frac{\ln(\ell)}{\delta/\rho} \right) \geq 0 \]

Proof of 5.1.2.1.4:

\[ t \geq \epsilon \]

[ By [1]. ]

\[ t \geq \epsilon^{\delta/\rho} \]

[ Since \( \delta < 1 \) and \( \rho \geq 1 \). ]

\[ \ln(t) \geq \frac{\delta}{\rho} \]

[ Taking the logarithm of both sides of 5.1.2.1.4.2. ]

\[ \frac{\ln(t)}{\delta/\rho} \geq 1 \]

[ Dividing 5.1.2.1.4.3 by \( \delta/\rho \) ]

\[ \ln \left( \frac{\ln(t)}{\delta/\rho} \right) \geq 0 \]

[ Taking the logarithm of both sides of 5.1.2.1.4.4. ]

\[ \frac{1}{1 - \epsilon} \ln \left( \frac{\ln(t)}{\delta/\rho} \right) \geq 0 \]

[ Multiplying 5.1.2.1.4.5 by \( 1/(1 - \epsilon) \). ]
Q.E.D. (5.1.2.1.4)

[5.1.2.1.5]
\[ \frac{d}{dt} \left( \frac{ln\left(\frac{w(t)}{t}\right)}{ln(1 - \epsilon)} \right) \geq \frac{d}{dt} \frac{t}{2} \]

[ By [5.1.2.1.2], [5.1.2.1.3], and [5.1.2.1.4]. ]

[5.1.2.1.6]
\( (\epsilon < 1, \epsilon = \epsilon) \rightarrow \frac{ln\left(\frac{w(t)}{t}\right)}{ln(1 - \epsilon)} \leq \frac{t}{2} \)

[ By [5.1.2.1.1], [5.1.2.1.5], and the properties of derivatives. ]

[5.1.2.1.7]
\[ \frac{d}{dt} \left( \frac{ln\left(\frac{w(t)}{t}\right)}{ln(1 - \epsilon)} \right) = \]
\[ -\frac{1}{\ln(1 - \epsilon)} \frac{\frac{d}{dt} \ln(\epsilon)}{\ln(\epsilon)} \]
\[ = \frac{1}{\ln(1 - \epsilon)} \frac{\ln(\epsilon) \frac{d}{dt}}{\ln(\epsilon)} = \]
\[ = \frac{1}{\ln(1 - \epsilon)} \frac{\ln(\epsilon) \frac{d}{dt}}{\ln(\epsilon)} \]

[5.1.2.1.8]
\[ \frac{d}{dt} \frac{t}{2} = \frac{1}{2} \]

[5.1.2.1.9]
\[ -\frac{1}{\ln(1 - \epsilon)} \frac{\ln(\epsilon)}{\ln(\epsilon)} \frac{1}{\frac{d}{dt}} \leq \frac{1}{2} \]

Proof of 5.1.2.1.9:

[5.1.2.1.9.1]
\[ t \geq \frac{2\ln\left(1/2\ln(\beta)\right)}{\ln(1 - \epsilon)} \]

[ By [1]. ]

[5.1.2.1.9.2]
\[ \beta \leq 2 \]

[ By assumption. ]

[5.1.2.1.9.3]
\[ \ln(\beta) < 1 \]

[ By [5.1.2.1.9.2], since 2 < \epsilon. ]

[5.1.2.1.9.4]
\[ \ln\left(1/2\ln(\beta)\right) < 0 \]

[ By [5.1.2.1.9.3]. ]
\[ 5.1.2.1.9.5 \]
\[ \ell \geq \frac{-2}{\ln(1 - \epsilon)} \]  
[ By \[ 5.1.2.1.9.1 \] and \[ 5.1.2.1.9.4 \]. ]

\[ 5.1.2.1.9.6 \]
\[ \frac{\ell}{2} \geq \frac{-1}{\ln(1 - \epsilon)} \]  
[ Dividing \[ 5.1.2.1.9.5 \] by 2. ]

\[ 5.1.2.1.9.7 \]
\[ \frac{1}{2} \geq \frac{-1}{\ln(1 - \epsilon)} \]  
[ Dividing \[ 5.1.2.1.9.6 \] by \( \ell \). ]

\[ 5.1.2.1.9.8 \]
\[ \frac{1}{\ln(\ell)} \leq 1 \]  
[ Since \( \ell \geq e \) by \[ 11 \]. ]

\[ 5.1.2.1.9.9 \]
\[ \frac{1}{2} \geq \frac{-1}{\ln(1 - \epsilon)} \frac{1}{\ell \ln(\ell)} \]  
[ By \[ 5.1.2.1.9.7 \] and \[ 5.1.2.1.9.8 \]. ]

Q.E.D. (5.1.2.1.9)

\[ 5.1.2.1.10 \]
\[ \frac{d}{dt} - \frac{\ln \left( \frac{1}{\ell} \right)}{\ln(1 - \epsilon)} \leq \frac{d}{dt} \frac{\ell}{2} \]  
[ By \[ 5.1.2.1.7 \], \[ 5.1.2.1.8 \], and \[ 5.1.2.1.9 \]. ]

\[ 5.1.2.1.11 \]
\[ (\epsilon < 1, \ell \geq e) \rightarrow -\frac{\ln \left( \frac{\ln(\ell)}{\ell} \right)}{\ln(1 - \epsilon)} \leq \frac{\ell}{2} \]  
[ By \[ 5.1.2.1.6 \], \[ 5.1.2.1.10 \], and the properties of derivatives. ]

Q.E.D. (5.1.2.1)

\[ 5.1.2.2 \]
\[ \ln \left( \frac{\ln(\ell)}{\ell} \right) \leq -\frac{\ell}{2} \ln(1 - \epsilon) \]  
[ Multiplying \[ 5.1.2.1 \] by \(-\ln(1 - \epsilon)\). ]

\[ 5.1.2.3 \]
\[ \frac{\ln(\ell)}{\ell} \leq (1 - \epsilon)^{-1/2} \]  
[ Exponentiating \[ 5.1.2.2 \] by \( r \). ]

\[ 5.1.2.4 \]
\[ \ln(\ell) \leq \frac{\ell(1 - \epsilon)^{-1/2}}{r} \]  
[ Multiplying \[ 5.1.2.3 \] by \( k/r \). ]

\[ 5.1.2.5 \]
\[ r \ln(\ell) \leq \ell(1 - \epsilon)^{-1/2} \]  
[ Multiplying \[ 5.1.2.4 \] by \( r \). ]
Q.E.D. (5.1.2)

\[ r \ln(r)(1 - r)^t \leq \frac{\delta}{2} \ln(\beta) \]  
| By [5.1.1] and [5.1.2], by transitivity. |

Q.E.D. (5.1)

\[ t \geq \frac{\ln(\delta/2) - \ln(r)}{\ln(1 - r)} \]  
| By [1]. |

\[ t \ln(1 - r) \leq \ln \left( \frac{\delta}{2} \right) - \ln(r) \]  
| Multiplying [5.2] by \( \ln(1 - r) \). |

\[ t \ln(1 - r) + \ln(r) \leq \ln \left( \frac{\delta}{2} \right) \]  
| Adding \( \ln(r) \) to both sides of [5.3]. |

\[ \ln \left( (1 - r)^t \right) + \ln(r) \leq \ln \left( \frac{\delta}{2} \right) \]  
| Moving the \( t \) inside the logarithm on the left side of [5.4]. |

\[ r(1 - r)^t \leq \frac{\delta}{2} \]  
| Exponentiating [5.5]. |

\[ r \ln(\beta)(1 - r)^t \leq \frac{\delta}{2} \ln(\beta) \]  
| Multiplying [5.6] by \( \ln(\beta) \). |

\[ r \ln(\beta)(1 - r)^t + r \ln(t)(1 - r)^t \leq \frac{\delta}{2} \ln(\beta) + \frac{\delta}{2} \ln(\beta) \]  
| Adding [5.7] and [5.1]. |

\[ r \ln(\beta)(1 - r)^t + r \ln(t)(1 - r)^t \leq \delta \ln(\beta) \]  
| By [5.8] and algebra. |

\[ r(1 - r)^t (\ln(\beta) + \ln(t)) \leq \delta \ln(\beta) \]  
| Combining \( r(1 - r)^t \) terms on the left side of [5.9]. |

\[ \frac{r(1 - r)^t (\ln(\beta) + \ln(t))}{\ln(\beta)} \leq \delta \]  
| Dividing [5.10] by \( \ln(\beta) \). |

\[ r(1 - r)^t \left( \frac{\ln(\beta)}{\ln(\beta)} + \frac{\ln(t)}{\ln(\beta)} \right) \leq \delta \]  
| By [5.11] and algebra. |
Proof of proposition 4.9

Theorem 4A.26: Let \( \text{Con}_{\mathcal{M}, \sigma} \) be a conjunctive uniform hypothesis update mechanism. For all \( k > 0, t > 0 \),

\[
\Pi_{\text{Ran}} (\text{Con}_{\mathcal{M}, \sigma}) (t) \leq \left( \frac{t^d}{d} \right)^k, \]

where \( d = \gamma (\mathcal{M}) \).

Proof:

[1] \( \text{Ran} (\text{Con}_{\mathcal{M}, \sigma}) = \mathcal{M} \) [By the definition of a conjunctive hypothesis update mechanism.]

[2] \( \Pi_{\text{Ran}} (\text{Con}_{\mathcal{M}, \sigma}) (t) = \Pi_{\mathcal{M}} (t) \) [By [1].]

[3] \( \forall (i > 1) : \)

\( \text{Ran} (\text{Con}_{\mathcal{M}, \sigma}) = \ast \left( \mathcal{M}, \text{Ran} (\text{Con}_{\mathcal{M}, \sigma}^{-1}) \right) \) \( \left( \ast \text{niceties} \right) \) [By the definition of a conjunctive hypothesis update mechanism.]
Theorem 4A.27: Let $\ell$ be a positive integer, $h$ be a positive integer ($h \leq \ell$), and $\beta$ be a constant ($\beta > 1$). If

$$\ell \geq \max \left\{ \frac{2h}{\log_{\beta}(\ell)} \log_{\beta} \left( \frac{e}{\log_{\beta}(\ell)} \right), 51h, \left( \frac{e}{h} \right)^{\left( \frac{\log_{\beta}(\ell+1)}{\log_{\beta}(\beta)} \right)^{-1}} \right\}$$

then

$$2^\ell \geq \left( \frac{e}{h} \right)^{\log_{\beta}(\ell+1)}$$

Proof:

11. $$\ell \geq \max \left\{ \frac{2h}{\log_{\beta}(\ell)} \log_{\beta} \left( \frac{e}{\log_{\beta}(\ell)} \right), 51h, \left( \frac{e}{h} \right)^{\left( \frac{\log_{\beta}(\ell+1)}{\log_{\beta}(\beta)} \right)^{-1}} \right\}$$

| Assumption |

[2] $$\frac{\ell}{h} \geq \log_{\beta} \left( \frac{e}{h} \right)$$
Proof of 2:

\( \left( \frac{t}{h} = 51 \right) \rightarrow \left( \frac{t}{h} \geq \log_2 \left( \frac{e^t}{h} \right) \right) \)  

[2.1] By arithmetic.

\( \frac{d}{d(\ell/h)} \frac{\ell}{h} = 1 \)  

[2.2] By calculus.

\( \frac{d}{d(\ell/h)} \log_2 \left( \frac{e^t}{h} \right) = 2 \log_2 \left( \frac{e^t}{h} \right) \frac{h}{e^t \ln(2)} \)  

[2.3] By calculus.

\( \frac{d}{d(\ell/h)} \log_2 \left( \frac{e^t}{h} \right) = 2 \log_2 \left( \frac{e^t}{h} \right) \frac{h}{e^t \ln(2)} \)  

[2.4] By [2.3] and algebra.

\( \left( \frac{t}{h} = 9.4 \right) \rightarrow \left( \frac{t}{h} \geq \frac{2}{\ln(2)} \log_2 \left( \frac{e^t}{h} \right) \right) \)  

[2.5] By calculus.

Proof of 2.5:

\( \left( \frac{t}{h} = 9.4 \right) \rightarrow \left( \frac{t}{h} \geq \frac{2}{\ln(2)} \log_2 \left( \frac{e^t}{h} \right) \right) \)  

[2.5.1] By arithmetic.

\( \frac{d}{d(\ell/h)} \frac{\ell}{h} = 1 \)  

[2.5.2] By calculus.

\( \frac{d}{d(\ell/h)} \frac{2}{\ln(2)} \log_2 \left( \frac{e^t}{h} \right) = \frac{2}{\ln(2)^2} \left( \frac{h}{e^t} \right) e^t \)  

[2.5.3] By calculus.

\( \frac{d}{d(\ell/h)} \frac{2}{\ln(2)} \log_2 \left( \frac{e^t}{h} \right) = \frac{2}{\ln(2)^2} \frac{h}{e^t} \)  

[2.5.4] By [2.5.3] and algebra.

\( \left( \frac{t}{h} \geq 9.4 \right) \rightarrow \left( \frac{2}{\ln(2)^2} \frac{h}{\ell} \leq 0.45 \right) \)  

[2.5.5]
Proof of 2.5.5:

[2.5.5.1] \[ \frac{\ell}{h} \geq 9.4 \] [ Assumption. ]

[2.5.5.2] \[ 1 \geq \frac{9.4h}{\ell} \] [ Dividing (2.5.5.1) by \( \ell/h \). ]

[2.5.5.3] \[ \frac{1}{9.4} \geq \frac{\ell}{h} \] [ Dividing [2.5.5.2] by 9.4. ]

[2.5.5.4] \[ \frac{2}{9.4\ln(2)^2} \geq \frac{2}{\ln(2)^2} \frac{\ell}{h} \] [ Multiplying [2.5.5.3] by \( 2/\ln(2)^2 \). ]

[2.5.5.5] \[ 0.45 \geq \frac{2}{\ln(2)^2} \frac{\ell}{h} \] [ By [2.5.5.4] and arithmetic. ]

Q.E.D. (2.5.5)

[2.5.6] \[ \left( \frac{\ell}{h} \geq 9.4 \right) \rightarrow \left( \frac{d}{d(\ell/h)} \frac{2}{\ln(2)} \log_2 \left( \frac{\ell}{h} \right) \leq 0.45 \right) \] [ By [2.5.4] and [2.5.5]. ]

[2.5.7] \[ \left( \frac{\ell}{h} \geq 9.4 \right) \rightarrow \left( \frac{d}{d(\ell/h)} \frac{\ell}{h} \geq \frac{d}{d(\ell/h)} \frac{2}{\ln(2)} \log_2 \left( \frac{\ell}{h} \right) \right) \] [ By [2.5.2], [2.5.3], and [2.5.6]. ]

[2.5.8] \[ \left( \frac{\ell}{h} \geq 9.4 \right) \rightarrow \left( \frac{\ell}{h} \geq \frac{2}{\ln(2)} \log_2 \left( \frac{\ell}{h} \right) \right) \] [ By [2.5.1], [2.5.7], and the properties of derivatives. ]

Q.E.D. (2.5)

[2.6] \[ \frac{\ell}{h} \geq 9.4 \] [ Because \( \ell \geq 51h \) by [1]. ]

[2.7] \[ \frac{\ell}{h} \geq \frac{2}{\ln(2)} \log_1 \left( \frac{\ell}{h} \right) \] [ By [2.5] and [2.6]. ]

[2.8] \[ 1 \geq \frac{2}{\ln(2)} \log_2 \left( \frac{\ell}{h} \right) \frac{h}{\ell} \] [ Dividing [2.7] by \( \ell/h \). ]
Proof of 3:

[3.1]

\[ t \geq \left( \frac{\epsilon}{h} \right)^{\left( \frac{\log_2(\beta)}{\log_2(\beta) - 1} \right)} \]

[ By [11]. ]

[3.2]

\[ \log_2(\epsilon) \geq \log_2 \left( \frac{\epsilon}{h} \right)^{\left( \frac{\log_2(\beta)}{\log_2(\beta) - 1} \right)} \]

[ Taking the logarithm of both sides of (3.1). ]

[3.3]

\[ \log_2(\epsilon) \geq \log_2 \left( \frac{\log_2(\beta)}{\log_2(\beta) - 1} \right)^{-1} \]

[ Extracting the exponent from the logarithm on the right side of (3.2). ]

[3.4]

\[ \frac{\log_2(\beta) - 1}{\log_2(\beta)} = \left( \frac{1/\log_2(\beta)}{1/(\log_2(\beta) - 1)} \right) \]

[ The left and right sides of the equation are equivalent by algebra. ]

[3.5]

\[ \log_2(\epsilon) \geq \log_2 \left( \frac{\log_2(\beta)}{\log_2(\beta) - 1} \right)^{-1} \]

[ By [3.3], [3.4], and substitution. ]

[3.6]

\[ \log_2(\epsilon) \left( \frac{1/\log_2(\beta)}{1/(\log_2(\beta) - 1)} - 1 \right) \geq \log_2 \left( \frac{\epsilon}{h} \right) \]
Proof of 3.6:

\[ (\beta \geq 2) \Rightarrow \frac{\log_2(\beta) - 1}{\log_2^2(\beta)} \leq 1 \]

Proof of 3.6.1:

[3.6.1.1] \[ \frac{d}{d\beta} \log_2(\beta) - 1 = \frac{1}{\ln(2) \beta} \]

[3.6.1.2] \[ \frac{d}{d\beta} \log_2^2(\beta) = 2 \log_2(\beta) \frac{1}{\ln(2) \beta} \]

[3.6.1.3] \[ \frac{d}{d\beta} \log_2^2(\beta) - \left( \frac{d}{d\beta} \log_2(\beta) - 1 \right) = 2 \log_2(\beta) \frac{1}{\ln(2) \beta} \frac{1}{\ln(2) \beta} - \frac{1}{\ln(2) \beta} \]

[By [3.6.1.1] and [3.6.1.2].]

[3.6.1.4] \[ \frac{d}{d\beta} \log_2^2(\beta) - \left( \frac{d}{d\beta} \log_2(\beta) - 1 \right) = \frac{1}{\beta \ln(2)} (2 \log_2(\beta) - 1) \]

[By [3.6.1.3], collecting \(1/\beta \ln(2)\) terms.]

[3.6.1.5] \[ \beta \geq 2 \]

[Assumption.]

[3.6.1.6] \[ 2 \log_2(\beta) > 1 \]

[By [3.6.1.5] and the properties of logarithms.]

[3.6.1.7] \[ 2 \log_2(\beta) - 1 > 0 \]

[Subtracting 1 from [3.6.1.6].]

[3.6.1.8] \[ \frac{d}{d\beta} \log_2^2(\beta) > \left( \frac{d}{d\beta} \log_2(\beta) - 1 \right) \]

[By [3.6.1.4] and [3.6.1.7], since \(\beta(\ln(2)) > 0\).]

[3.6.1.9] \[ (\beta = 2) \Rightarrow \log_2^2(\beta) > \log_2(\beta) \]

[By arithmetic.]

[3.6.1.10] \[ (\beta \geq 2) \Rightarrow \log_2^2(\beta) > \log_2(\beta) \]

[By [3.6.1.8], [3.6.1.9], and the properties of derivatives.]

[3.6.1.11] \[ (\beta \geq 2) \Rightarrow 1 > \frac{\log_2(\beta)}{\log_2^2(\beta)} \]

[Dividing the right side of [3.6.1.10] by \(\log_2^2(\beta)\).]
(\beta \geq 2) - 1 > \frac{\log_2(\beta) - 1}{\log_2^2(\beta)} \quad \text{[By 3.6.1.11.]} \]

Q.E.D. (3.6.1)

(1 \leq \beta \leq 2) - \frac{\log_2(\beta) - 1}{\log_2^2(\beta)} \leq 0

Proof of 3.6.2:

1 \leq \beta \leq 2 \quad \text{[Assumption.]} \]

\log_2(\beta) \leq 1 \quad \text{[By 3.6.2.1.]} \]

\log_2(\beta) - 1 \leq 0 \quad \text{[Subtract 1 from 3.6.2.2.]} \]

\log_2^2(\beta) \geq 0 \quad \text{Since the square of any number is positive.} \]

\frac{\log_2(\beta) - 1}{\log_2^2(\beta)} \leq 0 \quad \text{[By 3.6.2.3 and 3.6.2.4.]} \]

Q.E.D. (3.6.2)

\beta \geq 1 \quad \text{[By the definition of \beta in the statement of the theorem.]} \]

\frac{\log_2(\beta) - 1}{\log_2^2(\beta)} \leq 1 \quad \text{[By 3.6.1, 3.6.2, and 3.6.3.]} \]

\frac{\log_2(\beta) - 1}{\log_2^2(\beta)} - 1 \leq 0 \quad \text{[Subtracting 1 from 3.6.4.]} \]

Q.E.D. (3.6) \quad \text{The claim follows from 3.6.4, because 3.6.5 shows that we have multiplied 3.5 by a negative quantity to obtain 3.6.} \]

Distributing \log_2(f) on the left side of

\log_2(\varepsilon / \log_2^2(\beta)) \left( \frac{1 / \log_2^2(\beta)}{1 / (\log_2(\beta) - 1)} \right) - \log_2(f) \geq \log_2 \left( \frac{\varepsilon}{\beta} \right) \quad \text{[3.6.]} \]
\[ \log_2(\ell) \frac{\log_2(\beta) - 1}{\log_2(\beta)} - \log_2(\ell) \leq \log_2 \left( \frac{\ell}{h} \right) \]  
[By [3.7] and algebra.]

\[ \log_2(\ell) \frac{\log_2(\beta) - 1}{\log_2(\beta)} - \log_2(\ell) \leq \log_2 \left( \frac{\ell}{h} \right) \]  
[By [3.8] and algebra.]

\[ \frac{\log_2(\ell) \log_2(\beta) - 1}{\log_2(\beta)} \leq \log_2 \left( \frac{\ell}{h} \right) + \log_2(\ell) \]  
[By [3.9], since \( \log_2(\ell) \) is positive.]

\[ \log_2(\ell) \frac{\log_2(\beta) - 1}{\log_2(\beta) \log_2(\ell)} \leq \log_2 \left( \frac{\ell}{h} \right) \]  
[Combining the logarithms on the right side of [3.10].]

\[ \log_2(\ell) \frac{1}{\log_2(\beta)} \frac{1}{\log_2(\ell)} - 1 \leq \log_2 \left( \frac{\ell}{h} \right) \]  
[By [3.11] and algebra.]

Q.E.D. (3)

[Since \( \ell \) and \( h \) are both positive, by assumption.]

\[ \frac{\ell}{h} \geq 0 \]

(4)

\[ \log_2 \left( \frac{\ell}{h} \right) > 0 \]  
[Since \( \ell \geq 0 \) by assumption.]

\[ \frac{\ell}{h} \log_2 \left( \frac{\ell}{h} \right) \geq \frac{\log_2^{-1}(\beta)}{\log_2^{-1}(\beta) - 1} \frac{\log_2(\ell)}{\log_2(\beta)} \log_2 \left( \frac{\ell}{h} \right) \]
[Since \( \ell \geq 0 \) by assumption.]

\[ \frac{\ell}{h} \geq \frac{\log_2^{-1}(\beta)}{\log_2^{-1}(\beta) - 1} \frac{\log_2(\ell)}{\log_2(\beta)} \log_2 \left( \frac{\ell}{h} \right) \]
[Dividing [6] by \( \log_2(\ell/h) \).

\[ \frac{\ell}{h/\log_2(\beta)} \geq \log_2 \left( \frac{\ell}{h/\log_2(\beta)} \right) + \log_2 \left( \frac{\ell}{\log_2(\beta)} \right) \]
Proof of 8:

[8.1] 
\[ \ell \geq h \geq 0 \]

[8.2] 
\[ \beta \geq 1 \]

[8.3] 
\[ \log_2(\beta) > 0 \]

[8.4] 
\[ \frac{1}{2} \left( \frac{\ell}{h/\log_2(\beta)} \right) \geq \log_2 \left( \frac{\ell}{h/\log_2(\beta)} \right) \]

[8.5] 
\[ \frac{\ell}{\lambda} \geq \frac{2}{\log_3(\beta)} \log_2 \left( \frac{e}{\log_2(\beta)} \right) \]

[8.6] 
\[ \frac{1}{2} \left( \frac{\ell}{h/\log_2(\beta)} \right) \geq \log_2 \left( \frac{e}{\log_2(\beta)} \right) \]

[8.7] 
\[ \frac{1}{2} \left( \frac{\ell}{h/\log_2(\beta)} \right) + \frac{1}{2} \left( \frac{\ell}{h/\log_2(\beta)} \right) \geq \log_2 \left( \frac{\ell}{h/\log_2(\beta)} \right) + \log_2 \left( \frac{e}{\log_2(\beta)} \right) \]

[8.8] 
\[ \frac{\ell}{h/\log_2(\beta)} \geq \log_2 \left( \frac{\ell}{h/\log_2(\beta)} \right) + \log_2 \left( \frac{e}{\log_2(\beta)} \right) \]

Q.E.D. (8) 

[9] 
\[ \frac{\ell}{h/\log_2(\beta)} \geq \log_2 \left( \frac{\ell}{h/\log_2(\beta) \log_2(\beta)} \right) \]

[10] 
\[ \frac{\ell}{h/\log_2(\beta)} \geq \log_2 \left( \frac{e \ell}{h} \right) \]

[11] 
\[ \frac{\ell}{h} \left( \frac{1}{1/\log_2(\beta)} \right) \geq \log_2 \left( \frac{e \ell}{h} \right) \]

[12] 
\[ \frac{\ell}{h} \left( \frac{1/\log_2(\beta) - 1}{1/\log_2(\beta)} \right) \geq \log_2(\ell) \log_2 \left( \frac{e \ell}{h} \right) \]

Combining logarithms on the right side of [8].

[10] 
\[ \frac{\ell}{h/\log_2(\beta)} \geq \log_2 \left( \frac{e \ell}{h} \right) \]

Canceling the log_2(\beta) factors on the left side of [9].

[11] 
\[ \frac{\ell}{h} \left( \frac{1}{1/\log_2(\beta)} \right) \geq \log_2 \left( \frac{e \ell}{h} \right) \]

Rewriting the left side of [10].

[12] 
\[ \frac{\ell}{h} \left( \frac{1/\log_2(\beta) - 1}{1/\log_2(\beta)} \right) \geq \log_2(\ell) \log_2 \left( \frac{e \ell}{h} \right) \]

Dividing (7) by

\[ (1/\log_2(\beta))/(1/\log_2(\beta) - 1). \]
Theorem 4A.28: Let $\ell$, $a$, and $h$ be two positive integers, $1 \leq h \leq \ell$. Let $\beta > 2$ be a constant, let

$$t_0 \equiv \max \left\{ 80, \beta^{(\log_2(\beta) - 1)^{-1}} \right\}.$$
and let
\[ \gamma = \frac{\log_2(\ell_0) \log_2(\beta)}{\log_2(\ell_0) + \log_2(\beta)} . \]

If
\[ t \geq \max \left\{ \frac{e h}{\gamma^2}, \ell_0, \frac{4h \log_2(h/\gamma)}{\gamma}, \left( \frac{\gamma}{h} \right)^{(\gamma/h-1)^{-1}} \right\} , \]
then
\[ 2^t \geq \left( \frac{e h}{\beta} \right)^{h (\log_2(t)+1)} . \]

Proof:

[1]
\[ t \geq \max \left\{ \frac{e h}{\gamma^2}, \ell_0, \frac{4h \log_2(h/\gamma)}{\gamma}, \left( \frac{\gamma}{h} \right)^{(\gamma/h-1)^{-1}} \right\} \]
[ Assumption.]

[2]
\[ \frac{h \log_2(t)}{\gamma} \geq h (\log_2(t) + 1) \]

Proof of 2:

[2.1]
\[ t \geq \ell_0 \] [ By [1].]

[2.2]
\[ \gamma = \frac{\log_2(\ell_0) \log_2(\beta)}{\log_2(\ell_0) + \log_2(\beta)} \] [ By definition.]

[2.3]
\[ \frac{\log_2(\ell_0) \log_2(\beta)}{\log_2(\ell_0) + \log_2(\beta)} \leq \frac{\log_2(t) \log_2(\beta)}{\log_2(t) + \log_2(\beta)} \] [ By [2.2] and [2.1].]
Proof of 2.3:

\[ (\ell = \ell_0) \rightarrow \frac{\log_2(\ell_0) \log_2(\beta)}{\log_2(\ell_0) + \log_2(\beta)} = \frac{\log_2(\ell) \log_2(\beta)}{\log_2(\ell) + \log_2(\beta)} \quad \text{[By arithmetic.]} \]

\[ d \frac{\log_2(\ell) \log_2(\beta)}{d\ell \log_2(\ell) + \log_2(\beta)} = \frac{\log_2(\ell)}{\log_2(\ell) + \log_2(\beta)} \frac{d}{d\ell} \log_2(\ell) \log_2(\beta) + \log_2(\ell) \log_2(\beta) \frac{1}{d\ell \log_2(\ell) + \log_2(\beta)} \]

\[ \frac{\log_2(\beta)/\ell \ln(2)}{\log_2(\ell) + \log_2(\beta)} + \log_2(\ell) \log_2(\beta) \left( \frac{-1}{(\log_2(\ell) + \log_2(\beta))^2} \right) \frac{1}{\ln(2)\ell} \]

\[ d \frac{\log_2(\ell) \log_2(\beta)}{d\ell \log_2(\ell) + \log_2(\beta)} = \frac{\log_2(\beta)}{\ln(2)\ell (\log_2(\ell) + \log_2(\beta)) \left( 1 - \frac{\log_2(\ell)}{\log_2(\ell) + \log_2(\beta)} \right) \}
\]

\[ \log_2(\ell) \log_2(\beta) \leq 1 \]

Proof of 2.3.4:

\[ \beta \geq 1 \quad \text{[By the assumptions of the theorem.]} \]

\[ \log_2(\beta) \geq 0 \quad \text{[By 2.3.4.1 and the properties of logarithms.]} \]

\[ \log_2(\ell) + \log_2(\beta) \geq \log_2(\ell) \quad \text{[Adding \( \log_2(\ell) \) to both sides of 2.3.4.2.]} \]

\[ 1 \geq \frac{\log_2(\ell)}{\log_2(\ell) + \log_2(\beta)} \quad \text{[Dividing 2.3.4.3 by \( \log_2(\ell) + \log_2(\beta) \).]} \]

Q.E.D. (2.3.4)
\[ 1 - \frac{\log_2(\ell)}{\log_2(\ell) + \log_2(\beta)} \geq 0 \]  

[Subtracting \( \log_2(\ell)/(\log_2(\ell) + \log_2(\beta)) \) from both sides of [2.3.4].]

\[ \frac{d}{dt} \frac{\log_2(\ell) \log_2(\beta)}{\log_2(\ell) + \log_2(\beta)} \geq 0 \]  

[By [2.3.5] and [2.3.3] (since \( \frac{\log_2(\beta)}{\ln(2)/(\log_2(\ell) + \log_2(\beta))} > 0 \).]

\[ \frac{\log_2(\ell_0) \log_2(\beta)}{\log_2(\ell_0) + \log_2(\beta)} \leq \frac{\log_2(\ell) \log_2(\beta)}{\log_2(\ell) + \log_2(\beta)} \]  

[By [2.3.7] and [2.3.8].]

Q.E.D. (2.3)

\[ \gamma \leq \frac{\log_2(\ell) \log_2(\beta)}{\log_2(\ell) + \log_2(\beta)} \]  

[By [2.2] and [2.3].]

\[ 1 - \frac{\log_2(\ell) \log_2(\beta)}{\log_2(\ell) + \log_2(\beta)} \leq \frac{1}{\gamma} \]  

[Dividing [2.4] by \( \gamma \).]

\[ \log_2(\ell) + \log_2(\beta) \leq \log_2(\ell) \log_2(\beta) \frac{1}{\gamma} \]  

[By [2.5] and [2.4].]

\[ \frac{\log_2(\ell) + \log_2(\beta)}{\log_2(\beta)} \leq \frac{1}{\gamma} \]  

[Dividing [2.6] by \( \log_2(\beta) \).]

\[ \frac{\log_2(\ell)}{\log_2(\beta)} + 1 \leq \log_2(\ell) \frac{1}{\gamma} \]  

[By [2.7] and algebra.]

\[ \frac{\log_2(\ell)}{\log_2(\beta)} + 1 \leq \log_2(\ell) \frac{1}{\gamma} \]  

[By [2.8] and algebra.]

\[ h \left( \frac{\log_2(\ell)}{\log_2(\beta)} + 1 \right) \leq \frac{h \log_2(\ell)}{\gamma} \]  

[By [2.9] and algebra.]

\[ h (\log_2(\ell) + 1) \leq \frac{h \log_2(\ell)}{\gamma} \]  

[Changing in the base of the logarithm on the left side of [2.9].]
Q.E.D. (2)

(\frac{e^{\ell}}{h})^{h \log_b(t)/\gamma} \geq (\frac{e^{\ell}}{h})^{h (\log_b(t)+1)} \quad \text{[Exponentiating [2].]}

2^t \geq (\frac{e^{\ell}}{h})^{h \log_b(t)/\gamma}

Proof of 4:

[4.1] \frac{e^{\ell}}{h} \geq \log_b(t)

Proof of 4.1:

[4.1.1] t \geq \frac{2h \log_b \left( \frac{h}{\gamma} \right)}{\gamma} \quad \text{[By [1].]}

[4.1.2] \frac{te}{2h} \geq \log_b \left( \frac{h}{\gamma} \right) \quad \text{[Dividing [4.1.1] by } 2h/\gamma.]

[4.1.3] \frac{te}{2h} \geq 2

Proof of 4.1.3:

[4.1.3.1] t \geq \frac{4h}{\gamma} \quad \text{[By [1].]}

[4.1.3.2] \frac{te}{h} \geq 4 \quad \text{[By [4.1.3.1] and algebra.]}

Q.E.D. (4.1.3) \quad \text{[The claim follows from [4.1.3.2].]}

[4.1.4] \frac{te}{2h} \geq \log_b \left( \frac{h}{\gamma} \right) \quad \text{[By [4.1.3] and lemma 4A.12.]}

[4.1.5] \frac{te}{2h} + \frac{te}{2h} \geq \log_b \left( \frac{h}{\gamma} \right) + \log_b \left( \frac{h}{\gamma} \right) \quad \text{[Adding [4.1.3] and [4.1.4].]}
Adding the two terms on the left side of \[4.1.5\].

Combining the logarithms on the right side of \[4.1.6\].

Cancelling the \( d/\gamma \) factors on the right side of \[4.1.7\].

Q.E.D. (4.1)

\[
\frac{\ell}{h} \geq 2 \log_2(\ell) \log_2\left(\frac{\ell}{h}\right)
\]

Proof of 4.2:

\[4.2.1\]
\[
\ell \geq \left(\frac{\gamma}{h}\right)^{(\gamma/h-1)^{-1}}
\]

[ By III. ]

\[4.2.2\]
\[
\log_2(\ell) \geq \log_2\left(\left(\frac{\gamma}{h}\right)^{(\gamma/h-1)^{-1}}\right)
\]

[ Taking the log base 2 of \[4.2.1\]. ]

\[4.2.3\]
\[
\log_2(\ell) \geq \frac{\log_2(\gamma/h)}{\gamma/h - 1}
\]

[ Extracting the exponent from the logarithm in \[4.2.2\]. ]

\[4.2.4\]
\[
1 \geq \frac{\log_2(\gamma/h)}{(\gamma/h - 1) \log_2(\ell)}
\]

[ Dividing \[4.2.3\] by \( \log_2(\ell) \). ]

\[4.2.5\]
\[
\left(\frac{\gamma}{h} - 1\right) \log_2(\ell) \geq \log_2\left(\frac{\gamma}{h}\right)
\]

[ Multiplying \[4.2.4\] by \( (\gamma/h - 1) \log_2(\ell) \). ]

\[4.2.6\]
\[
\log_2(\ell) \frac{\gamma}{h} - \log_2(\ell) \geq \log_2\left(\frac{\gamma}{h}\right)
\]

[ Distributing the \( \log_2(\ell) \) on the left side of \[4.2.5\]. ]

\[4.2.7\]
\[
\log_2(\ell) \frac{\gamma}{h} \geq \log_2\left(\frac{\gamma}{h}\right) + \log_2(\ell)
\]

[ Adding \( \log_2(\ell) \) to \[4.2.6\]. ]

\[4.2.8\]
\[
\log_2(\ell) \frac{\gamma}{h} \geq \log_2\left(\frac{\ell}{h}\right)
\]

[ Combining the logarithms on the left side of \[4.2.7\]. ]
\[ \ell \geq 2 \log_2^2(\ell) \]

Proof of 4.2.9:

\[ (\ell = 80) \to \{ \ell \geq 2 \log_2^2(\ell) \} \quad \text{By arithmetic.} \]

\[ \frac{d}{dt} \ell = 1 \]

\[ \frac{d}{dt} \log_2^2(\ell) = \frac{4 \log_2(\ell)}{\ln(2)} \frac{1}{\ell} \]

\[ (\ell = 80) \to \]

\[ 4 \log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} < 1 \quad \text{By arithmetic.} \]

\[ \frac{d}{dt} 4 \log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} = \]

\[ 4 \frac{1}{\ln(2)} \left( \frac{1}{\ln(2)} - \frac{\log_2(\ell)}{\ell} \right) \]

\[ \frac{d}{dt} 4 \log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} = \quad \text{Collecting terms in the logarithm on the right side of [4.2.9.5].} \]

\[ 4 \frac{1}{\ln(2)} \left( \frac{1}{\ln(2)} - \frac{\log_2(\ell)}{\ell} \right) \]

\[ (\ell \geq 80) \to \left( \log_2(\ell) > \frac{1}{\ln(2)} \right) \quad \text{By arithmetic and the properties of logarithms.} \]

\[ (\ell \geq 80) \to \left( 0 > \frac{1}{\ln(2)} - \log_2(\ell) \right) \quad \text{Adding } -\log_2(\ell) \text{ to the right side of [4.2.9.7].} \]

\[ (\ell \geq 80) \to \]

\[ 0 > 4 \frac{1}{\ln(2)} \left( \frac{1}{\ln(2)} - \frac{\log_2(\ell)}{\ell^2} \right) \]

\[ \text{Multiplying the right side of [4.2.9.8] by } 4/(\ln(2))^2. \]
(4.2.9.10) \[ (\ell \geq 80) \to \frac{d}{d\ell} 4\log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} < 0 \] \[ \text{By [4.2.9.6] and [4.2.9.9].} \]

(4.2.9.11) \[ \frac{d}{d\ell} 1 = 0 \]

(4.2.9.12) \[ (\ell \geq 80) \to \frac{d}{d\ell} 4\log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} \leq 1 \] \[ \text{By [4.2.9.10] and [4.2.9.11].} \]

(4.2.9.13) \[ (\ell \geq 80) \to 4\log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} \leq 1 \] \[ \text{By [4.2.9.4] and [4.2.9.12].} \]

(4.2.9.14) \[ (\ell \geq 80) \to \frac{d}{d\ell} 4\log_2(\ell) \frac{1}{\ln(2)} \frac{1}{\ell} \geq \frac{d}{d\ell} 2\log_2(\ell) \] \[ \text{By [4.2.9.2], [4.2.9.3], and [4.2.9.13].} \]

(4.2.9.15) \[ (\ell \geq 80) \to \ell \geq 2\log_2(\ell) \] \[ \text{By [4.2.9.11] and [4.2.9.14].} \]

(4.2.9.16) \[ \ell \geq 80 \] \[ \text{By [1].} \]

(4.2.9.17) \[ \ell \geq 2\log_2(\ell) \] \[ \text{By [4.2.9.15] and [4.2.9.16].} \]

Q.E.D. (4.2.9)

(4.2.10) \[ \frac{\ell}{\log_2(\ell)} \geq 2\log_2(\ell) \] \[ \text{Dividing [4.2.9] by \log_2(\ell).} \]

(4.2.11) \[ \frac{\ell}{\log_2(\ell)} \geq 0 \] \[ \text{Since } \ell \geq 1. \]

(4.2.12) \[ \frac{\log_2(\ell)}{h} \geq 0 \] \[ \text{Since } \ell \geq 1, \gamma \geq 0, h \geq 1. \]

(4.2.13) \[ \frac{\ell}{\log_2(\ell)} \frac{\gamma}{h} \geq 2\log_2(\ell) \log_2 \left( \frac{\ell \gamma}{h} \right) \] \[ \text{Multiplying [4.2.8] and [4.2.10]; [4.2.11] and [4.2.12] establish that the sense of the inequality is correct.} \]
\[ \ell \gamma \geq 2 \log_2(\ell) \log_2 \left( \frac{\ell \gamma}{h} \right) \quad \text{[Cancelling the } \log_2(\ell) \text{ factors on the left side of (4.2.13).]} \]

Q.E.D. (4.2)

[4.3] Define

\[ \lambda \equiv 2 \log_2(\ell) \quad \text{[Definition.]} \]

[4.4] \[
\log_2(\ell) \left( \log_2 \left( \frac{e}{\gamma} \right) + \log_2 \left( \frac{\ell \gamma}{h} \right) \right) \leq \lambda \log_2 \left( \frac{\ell \gamma}{h} \right)
\]

Proof of 4.3:

[4.4.1] \[ \lambda \geq 2 \log_2(\ell) \quad \text{[By the definition of } \lambda \text{.]} \]

[4.4.2] \[ \lambda - \log_2(\ell) \geq \log_2(\ell) \quad \text{[Subtracting } \log_2(\ell) \text{ from (4.4.1).]} \]

[4.4.3] \[ \ell \geq \frac{de}{\gamma^2} \quad \text{[By (11).]} \]

[4.4.4] \[ \ell \gamma \geq \frac{e}{\gamma} \quad \text{[Multiplying (4.4.3) by } \gamma/h \text{.]} \]

[4.4.5] \[
\log_2 \left( \frac{\ell \gamma}{h} \right) \geq \log_2 \left( \frac{e}{\gamma} \right) \quad \text{[Taking logarithms of (4.4.4).]} \]

[4.4.6] \[
\log_2 \left( \frac{\ell \gamma}{h} \right) (\lambda - \log_2(\ell)) \geq \log_2 \left( \frac{e}{\gamma} \right) \log_2(\ell)
\]

[Multiplying (4.4.2) and (4.4.5).]

Proof of 4.4.6:

[4.4.6.1] \[ \gamma > 1 \]
Proof of 4.4.6.1:

[4.4.6.1.1]
\[ t_0 \geq \beta^{(\log_2(\beta)-1)^{-1}} \quad \text{[By the definition of } t_0\text{.]} \]

[4.4.6.1.2]
\[ \log_2(t_0) \geq \log_2 \left( \beta^{(\log_2(\beta)-1)^{-1}} \right) \]
\[ \text{[Taking logarithms on both sides of [4.4.6.1.1].]} \]

[4.4.6.1.3]
\[ \log_2(t_0) \geq \frac{\log_2(\beta)}{\log_2(\beta) - 1} \quad \text{[Extracting the exponent from the logarithm on the right side of [4.4.6.1.2].]} \]

[4.4.6.1.4]
\[ \log_2(t_0) \left( \log_2(\beta) - 1 \right) \geq \log_2(\beta) \]
\[ \text{[Multiplying [4.4.6.1.3] by } \log_2(\beta) - 1 \text{ (recall that } \beta \geq 2, \text{ hence } \log_2(\beta) - 1 \geq 0).]} \]

[4.4.6.1.5]
\[ \log_2(t_0) \log_2(\beta) - \log_2(t_0) \geq \log_2(\beta) \]
\[ \text{[Distributing } \log_2(t_0) \text{ on the left side of [4.4.6.1.4].]} \]

[4.4.6.1.6]
\[ \log_2(t_0) \log_2(\beta) \geq \log_2(\beta) + \log_2(t_0) \]
\[ \text{[Adding } \log_2(t_0) \text{ to [4.4.6.1.5].]} \]

[4.4.6.1.7]
\[ \frac{\log_2(t_0) \log_2(\beta)}{\log_2(\beta) + \log_2(t_0)} \geq 1 \]
\[ \text{[Dividing [4.4.6.1.6] by } \log_2(\beta) + \log_2(t_0).]} \]

Q.E.D. (4.4.6.1) \quad \text{[The claim follows from [4.4.6.1.7] and the definition of } \gamma\text{.]} \]

[4.4.6.2]
\[ t \geq h \quad \text{[By the definition of } h \text{ and } t\text{.]} \]

[4.4.6.3]
\[ \frac{t}{h} \geq 1 \quad \text{[Dividing [4.4.6.2] by } h\text{.]} \]

[4.4.6.4]
\[ \frac{t}{h} \gamma \geq 1 \quad \text{[Multiplying [4.4.6.1] and [4.4.6.3].]} \]

[4.4.6.5]
\[ \log_3 \left( \frac{t}{h} \right) \geq 0 \quad \text{[Taking logarithms of [4.4.6.4].]} \]
\[4.4.6.6\] 
\[\lambda - \log_2(\ell) \geq \log_2(\ell) \quad \text{By } [4.4.2].\]

\[4.4.6.7\] 
\[\log_2(\ell) \geq 0 \quad \text{Since } \ell \geq 1.\]

\[4.4.6.8\] 
\[\lambda - \log_2(\ell) \geq 0 \quad \text{By } [4.4.6.6] \text{ and } [4.4.6.7].\]

\[4.4.6.9\] 
\[\log_2\left(\frac{\ell \gamma}{h}\right) (\lambda - \log_2(\ell)) \geq \log_2\left(\frac{\gamma}{h}\right) \log_2(\ell) \]

\[\text{Multiplying } [4.4.6.5] \text{ and } [4.4.6.8]. \text{ The sense of the inequality is preserved because of } [4.4.6.5], [4.4.6.8] \text{ and } [4.4.6.7].\]

Q.E.D. (4.4.6)

\[4.4.7\] 
\[\lambda \log_2\left(\frac{\ell \gamma}{h}\right) - \log_2\left(\frac{\ell \gamma}{h}\right) \log_2(\ell) \geq \log_2\left(\frac{\gamma}{h}\right) \log_2(\ell) \]

\[\text{Distributing } \log_2(\ell \gamma/h) \text{ on the left side of } [4.4.6].\]

\[4.4.8\] 
\[\lambda \log_2\left(\frac{\ell \gamma}{h}\right) \geq \log_2\left(\frac{\gamma}{h}\right) \log_2(\ell) + \log_2\left(\frac{\ell \gamma}{h}\right) \log_2(\ell) \]

\[\text{Adding } (\ell \gamma/h) \log_2(\ell) \text{ to } [4.4.7].\]

\[4.4.9\] 
\[\lambda \log_2\left(\frac{\ell \gamma}{h}\right) \geq \log_2(\ell) \left(\log_2\left(\frac{\gamma}{h}\right) + \log_2\left(\frac{\ell \gamma}{h}\right)\right) \]

\[\text{Collecting } \log_2(\ell) \text{ terms on the right side of } [4.4.8].\]

Q.E.D. (4.4)

\[4.5\] 
\[\frac{\ell \gamma}{h} \geq \lambda \log_2\left(\frac{\ell \gamma}{h}\right)\]

Proof of 4.5:

\[4.5.1\] 
\[\frac{\ell \gamma}{h} \geq 2 \log_2(\ell) \log_2\left(\frac{\ell \gamma}{h}\right) \quad \text{By } [4.2].\]

\[4.5.2\] 
\[\lambda = 2 \log_2(\ell) \quad \text{By the definition of } \lambda.\]
[4.5.3] \[ \frac{t \gamma}{h} \geq \lambda \log_2 \left( \frac{t \gamma}{h} \right) \] [ Substituting [4.5.2] into [4.5.1]. ]

Q.E.D. (4.5)

[4.6] \[ \frac{t \gamma}{h} \geq \log_2(t) \left( \log_2 \left( \frac{e}{\gamma} \right) + \log_2 \left( \frac{t \gamma}{h} \right) \right) \] [ By [4.4] and [4.5]. ]

[4.7] \[ \frac{t \gamma}{h} \geq \log_2(t) \left( \log_2 \left( \frac{e}{\gamma} \frac{t \gamma}{h} \right) \right) \] [ Combining the inner logarithms on the right side of [4.6]. ]

[4.8] \[ \frac{t \gamma}{h} \geq \log_2(t) \left( \log_2 \left( \frac{e t \gamma}{h} \right) \right) \] [ Cancelling \( \gamma \) on the right side of [4.7]. ]

[4.9] \[ t \geq \frac{h}{\gamma} \log_2(t) \left( \log_2 \left( \frac{e t \gamma}{h} \right) \right) \] [ Multiplying [4.8] by \( h/\gamma \). ]

[4.10] \[ t \geq \log_2 \left( \left( \frac{e t \gamma}{h} \right)^{\frac{1}{h \log_2(t)}} \right) \] [ Moving \( (d/\gamma) \log_2(t) \) into the logarithm on the right side of [4.9]. ]

[4.11] \[ 2^t \geq \left( \frac{e t \gamma}{h} \right)^{\frac{1}{h \log_2(t)}} \] [ Exponentiating [4.10]. ]

Q.E.D. (4)

[5] \[ 2^t \geq \left( \frac{e t \gamma}{h} \right)^{\left( \log_2(t) + 1 \right)} \] [ By [3] and [4]. ]

Q.E.D.

---

**Theorem 4A.29:** Let Con_{\psi, \sigma} be a conjunctive uniform hypothesis update mechanism. For all \( k > 0, t \geq 1 \),

\[ \Pi_{Rem} \left( Con_{\psi, \sigma} \right) (t) \leq (1.1^\sigma(t))^k. \]
Proof:

[1] \[ \operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{t} \right) = \mathcal{H} \]  
[ By the definition of a uniform hypothesis update mechanism. ]

[2] \[ \forall (t \geq 1) : \]  
\[ \Pi_{\operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{t} \right)}(t) = \Pi_{\mathcal{H}}(t) \]  
[ This follows from [1]. ]

[3] \[ \forall (i \geq 1) : \]  
\[ \exists (\oplus : \Phi : \text{ricetic}) : \]  
\[ \operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{i+1} \right) = \oplus \left( \operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{i} \right), \mathcal{H} \right) \]  
[ By the definition of a uniform hypothesis update mechanism. ]

[4] \[ \forall (i \geq 1) : \]  
\[ \Pi_{\operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{i+1} \right)}(t) \leq \Pi_{\mathcal{H}}(t) \Pi_{\operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{i} \right)}(t) \]  
[ By [3] and the definition of a ricetic operator. ]

[5] \[ \forall (i \geq 1) : \]  
\[ \Pi_{\operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{i+1} \right)}(t) \leq \left( \Pi_{\mathcal{H}}(t) \right)^{i} \]  
[ By [1], [4], and induction. ]

Q.E.D.

Theorem 4A.30: Let \( \text{Con}_{\mathcal{H},\sigma} \) be a conjective uniform hypothesis update mechanism. For all \( k \in \mathbb{N} \),

\[ V \left( \operatorname{Ran} \left( \text{Con}_{\mathcal{H},\sigma}^{k} \right) \right) \leq 2k V \left( \mathcal{H} \right) \log_{2}(ek). \]

Proof:
\[ \forall (t > 0) : \\
\Pi_{R_m(\text{Con}_{\mathcal{M}, e})}^{t} \leq (\Pi_{\mathcal{M}}(t))^{k} \] 
[ By theorem 4A.29. ]

\[ \forall (t > 0) : \\
\Pi_{R_m(\text{Con}_{\mathcal{M}, e})}^{t} \left( \frac{e^{t}}{V(\mathcal{M})} \right)^{k V(\mathcal{M})} \] 
[ By (1) and lemma 2A.15. ]

\[ (2^{t} \geq \Pi_{R_m(\text{Con}_{\mathcal{M}, e})}^{t}) \rightarrow (t \geq V(\text{Ran}(\text{Con}_{\mathcal{M}, e}))) \] 
[ By the definition of \( V \). ]

\[ (2^{t} \geq \left( \frac{e^{t}}{V(\mathcal{M})} \right)^{k V(\mathcal{M})}) \rightarrow (t \geq V(\text{Ran}(\text{Con}_{\mathcal{M}, e}))) \] 
[ By (2) and (3). ]

\[ (t \geq 2k V(\mathcal{M}) \log_{2}(e)) \rightarrow (2^{t} \geq \left( \frac{e^{t}}{V(\mathcal{M})} \right)^{k V(\mathcal{M})}) \]

Proof of 5:

\[ 5.1 \]
Let \( d \equiv V(\mathcal{M}) \)  
[ Definition. ]

\[ 5.2 \]
\( t \geq 2kd \log_{2}(e) \)  
[ Assumption. ]

\[ 5.3 \]
\( \frac{t}{2kd} \geq \log_{2}(e) \)  
[ Dividing (5.2) by \( 2kd \). ]

\[ 5.4 \]
\( \frac{t}{kd} \geq 2 \)

Proof of 5.4:

\[ 5.4.1 \]
\( t \geq 2kd \log_{2}(e) \)  
[ By (5.2). ]

\[ 5.4.2 \]
\( \frac{t}{kd} \geq 2 \log_{2}(e) \)  
[ By (5.4.1). ]

\[ 5.4.3 \]
\( \frac{t}{kd} \geq 2 \log_{2}(e) \)  
[ Since \( k \in \mathbb{N} \) by assumption. ]
\[ \frac{t}{kd} \geq 2 \] 

Q.E.D. (5,4)

\[ \frac{t}{kd} \geq \log_2 \left( \frac{t}{kd} \right) \] 

[ By (5.4) and lemma 4A.12. ]

\[ \frac{t}{kd} \geq \log_2 \left( \frac{t}{kd} \right) + \log_2(ek) \] 

[ Adding (5.4) and (5.5). ]

\[ \frac{t}{kd} \geq \log_2 \left( \frac{t}{kd} \cdot ek \right) \] 

[ Combining the logarithms on the right side of (5.6). ]

\[ \frac{t}{kd} \geq \log_2 \left( \frac{et}{d} \right) \] 

[ Cancelling \( k \)'s on the right side of (5.7). ]

\[ \ell \geq \log_2 \left( \frac{et}{d} \right) kd \] 

[ Multiplying (5.8) by \( kd \). ]

\[ \ell \geq \log_2 \left( \left( \frac{et}{d} \right)^{kd} \right) \] 

[ Placing the \( kd \) inside the logarithm on the right side of (5.9). ]

\[ 2^\ell \geq \left( \frac{et}{d} \right)^{kd} \] 

[ Exponentiating (5.10). ]

Q.E.D. (5) [ The claim follows from (5.1) and (5.11). ]

(6)

\[ (t \geq 2k \kappa'(\mathcal{A}) \log_2(ek)) \rightarrow (t \geq \kappa'(\text{Ran} \ (\text{Con}_\kappa))) \] 

[ By (4) and (5). ]

(17)

\[ 2k \kappa'(\mathcal{A}) \log_2(ek) \geq \kappa'(\text{Ran} \ (\text{Con}_\kappa)) \] 

[ This follows from (6). ]

Q.E.D. 

Theorem 4A.31: Let $\text{Con}_{\mathcal{W}, \sigma}$ be a conjunctive uniform hypothesis update mechanism. Define

$$\beta = \frac{\sigma}{\sigma - 1}$$

and let

$$d \equiv \gamma(\mathcal{W}).$$

For all $t \geq 1$:

(a)

$$\gamma \left( \text{Ran} \left( \text{Con}_{\mathcal{W}, \sigma}^{\log_d (t+1)} \right) \right) \leq \max \left\{ \frac{2d}{\log_2(\beta)}, \log_2 \left( \frac{e}{\log_2(\beta)} \right), 51d, \left( \frac{e}{d} \right)^{(n^* - 1)} \right\}.$$

(b)

If $\beta > 2$,

$$\gamma \left( \text{Ran} \left( \text{Con}_{\mathcal{W}, \sigma}^{\log_d (t+1)} \right) \right) \leq \max \left\{ \frac{de}{\gamma}, \ell_0, \frac{2d \log_2(d/\gamma)}{\gamma}, \left( \frac{e}{d} \right)^{(n^* - 1)} \right\},$$

where

$$\ell_0 \equiv \max \left\{ 80, \beta(n^* - 1) \right\}$$

and

$$\gamma \equiv \frac{\log_2(\ell_0) \log_2(\beta)}{\log_2(\ell_0) + \log_2(\beta)}.$$

Proof:

1. $\forall (t > 0)$:

$$\Pi_{\text{Ran} \left( \text{Con}_{\mathcal{W}, \sigma} \right)}(t) \leq \left( \frac{td}{d} \right)^{kd}$$

[ By theorem 4A.26. ]

2. $t^{\ell} \geq \Pi_{\text{Ran} \left( \text{Con}_{\mathcal{W}, \sigma}^{\log_d (t+1)} \right)}(t) \rightarrow \left( t^{\ell} \geq \gamma \left( \text{Ran} \left( \text{Con}_{\mathcal{W}, \sigma}^{\log_d (t+1)} \right) \right) \right)$

[ By [1] and the definition of $\gamma$. ]
\[\ell \geq \max \left\{ \frac{2h}{\log_2(\beta)} \log_2 \left( \frac{e}{\log_2(\beta)} \right), 51h, \left( \frac{\left( \frac{h}{\gamma} \right)^{\gamma^{1/\gamma-1}}}{\gamma} \right)^2 \right\} \rightarrow 2\ell' \geq \left( \frac{e\ell}{h} \right)^{h(\log_2(\ell)+1)}\] 

[By theorem 4A.27.]

\[\ell \geq \max \left\{ \frac{2h}{\log_2(\beta)} \log_2 \left( \frac{e}{\log_2(\beta)} \right), 51h, \left( \frac{\left( \frac{h}{\gamma} \right)^{\gamma^{1/\gamma-1}}}{\gamma} \right)^2 \right\} \rightarrow \ell \geq \gamma \left( \text{Ran} \left( \text{Con}_{\mathcal{M}_{\mu,\rho}}^{(\log_2(\ell)+1)} \right) \right)\]

[By (3) and (2).]

\[\left( \beta > 2 \right) \land \ell \geq \max \left\{ \frac{eh}{\gamma^2}, 4h, 2h \log_2 \left( \frac{h}{\gamma} \right), \left( \frac{1}{\gamma} \right)^{\gamma^{1/\gamma-1}} \right\} \rightarrow 2\ell' \geq \left( \frac{e\ell}{h} \right)^{h(\log_2(\ell)+1)}\] 

[By theorem 4A.28.]

\[\left( \beta > 2 \right) \land \ell \geq \max \left\{ \frac{eh}{\gamma^2}, 4h, 2h \log_2 \left( \frac{h}{\gamma} \right), \left( \frac{1}{\gamma} \right)^{\gamma^{1/\gamma-1}} \right\} \rightarrow \ell \geq \gamma \left( \text{Ran} \left( \text{Con}_{\mathcal{M}_{\mu,\rho}}^{(\log_2(\ell)+1)} \right) \right)\]

[By (5) and (3).]

Q.E.D. [4] and [6] are what was to be proved.
Appendix B. Computational Learning Theory

This appendix is intended as an introduction to the model of machine learning used in this dissertation. We use what is sometimes referred to as the pac model of learning; the acronym pac means probably approximately correct, and the reason for it will become clear below. This model is attributed to [30], who suggested the use of statistical methods in the analysis of machine learning algorithms, and also suggested that the tractability of machine learning algorithms should be an issue. The term “learnability” was coined by [30] in reference to the overall feasibility of a learning problem. However, much of the statistical theory used in analyzing machine learning algorithms, and much of the content of this appendix, was developed in the field of nonparametric pattern recognition (Cf. [33]).

To express a learning problem formally, we ask several questions:

- What are we trying to learn?
- What information is available to a learning algorithm?
- What constitutes success for a learning algorithm?
- What are the algorithm’s resource requirements?

This appendix will consider each of these questions in turn.

B.1. What is learning?

Psychology, computer science, and several other fields are concerned to various extents about the nature of learning, but in general the word is used quite broadly, and it is rare for two disciplines to agree on what it means. Even within machine learning there is no accepted definition, but in this field there are some points that are generally
agreed upon:

First, it is usually assumed that a learned algorithm is supposed interact with its environment in a certain way; it performs a mapping from some space of inputs to some space of outputs. Since machine learning is more or less a branch of artificial intelligence, one might say that the algorithm senses something about its surroundings (e.g., it receives a stimulus) and reacts with some appropriate response, which may be an action, or simply a display of information about the thing it has sensed.

Second, although "learning by being told," or, analogously, "learning by being programmed," is important in many areas, it is not generally considered to be a part of machine learning. Likewise, learning by memorization is not usually regarded as machine learning, because there, the primary issue is converting information into a form that a machine can understand. This is a complex and important problem in itself, but we will not deal with it here.

The algorithms discussed in this dissertation learn from examples; that is, they learn how events depend upon one another by observing those events. Their intent is to find a rule that will allow them to determine whether or not an event occurred, even if it cannot be observed directly, by looking at other events instead. However, this characterization should be understood in its broadest sense; it may be, for example, that the event we are interested in is "those times when it is appropriate to move the second joint of the arm 19 degrees to the left."

To distinguish learning from mere memorization, we require that the rules we learn should work correctly on inputs that were not included in the training examples. However, this requirement should not be taken to mean that an algorithm may not memorize its training examples. If a learning domain contains only one possible input, then learning in that domain is considered trivial, and not regarded as being impossible simply because there are no novel inputs left after the first example has been seen.

We have said that we will not regard "learning by being programmed" as being a part of machine learning, but it is difficult to say what does or does not constitute programming. If to program is simply to convey information that allows a computer to perform some function, then all learning from examples is the result of programming, as long as the learning itself is done by a computer. The criterion we use instead will be based on the statistical properties of the set of training examples; we ask that the choice of training examples be independent of the choice that determines what rule is to be learned, except insofar as the latter choice may be reflected in a supervisory input (during its learning phase, an algorithm is often told what response is "correct" for a given stimulus so that it can make corrections to its proposed input-to-output mapping function. This "correct response" is known as a supervisory input).
B.1.1. Learnability.

Below, we will outline a set of criteria for successful learning. It is often said that if there is an algorithm that meets these criteria for a given set of learning problems, then that set (or class) of problems is learnable.

In general, the use of the word "learnability" in connection with the specific framework that will be used here has led to some objections (Cf. L. Bimbbaum in [1]) because it implies a definition of "learning" that is not generally accepted (again, because so many natural and artificial phenomena have been characterized as learning). For this reason, it is common to rephrase the definition and say that a class of concepts is "probably approximately correctly," or \textit{pac-learnable} if it meets the criteria given below.

Nonetheless the framework given here has broad applicability, and, more importantly, it gives us a criterion by which to decide if our learning algorithms are successful. However, it should be kept in mind that \textit{learnability}, as it is defined here and elsewhere, does not refer to learning in general, but rather to a particular kind of learning.

B.1.2. Inductive Bias (what are we trying to learn?)

In formalizing a learning problem, we must first recognize that we cannot construct a learning algorithm that can simply "learn anything." This is obvious in one sense because of the set of all functions is uncountably infinite, which implies that some functions cannot be specified with a finite representation.

However, even if we attempt to write a program that can learn any possible \textit{algorithm}, we cannot succeed. We can demonstrate this with a thought-experiment: consider a domain in which there are an infinite number of inputs, so that learning cannot be achieved by simply memorizing an output for each possible input (generally the criterion that distinguishes learning from mere memorization is whether or not the learned mapping can works correctly on inputs that were not previously seen). In this domain, we will present a series of examples that can be classified as being either "positive" or "negative," the role of the learning algorithm is to find a rule that can classify the inputs correctly.

Consider some set of \(m\) unique examples. There are \(2^m\) ways of dividing the examples into two subsets, and hence \(2^m\) unique rules for classifying them. Assume that there is a "teacher" who not only presents the examples, but also decides which classification rule is the right one. During training, the teacher can change its mind to some extent about what the correct rule is, as long as the new rule is not inconsistent with any of the examples that
the algorithm has already seen. There will be no way for the algorithm to know that the desired learning rule has changed.

If the learning algorithm is supposed to be able to learn any possible classification rule, then the teacher may decide on the correct classification for any novel input just before that input is presented to the algorithm. But suppose the teacher has access to the algorithm's state of computation. This is certainly possible when the learning algorithm is implemented by a computer program. The teacher can observe the program's current hypothesis, and simply say that the "correct" classification is the opposite of what this hypothesis will guess for any novel input. In this way the learning algorithm can be forced to make only mistakes.

Examples of things that humans cannot learn often come from the area of language understanding. We can illustrate this problem by starting with the simple noun phrase

"The cat ran."

It is grammatically correct to replace "the cat" with "the cat the dog bit." We can go still further, replacing "the dog" with "the dog the bird pecked" and "the bird" with "the bird flying west" to get:

"The cat the dog the bird flying west pecked bit ran."

Although this is a correct English sentence, it is not really one that can be parsed by humans, and it is not clear that the problem of learning to understand such sentences is within the capabilities of the brain.

In order to avoid problems like these, it is common to place some restriction on the mappings that an algorithm may have to learn. The usual approach is to specify some class of possible concepts, and then to ask whether learning is possible for that particular class. Some possible classes are:

axis-parallel hyperrectangles: regions in n-dimensional Euclidean space having the form of a hyperrectangle, each of whose edges is parallel to an axis.

decision trees: trees used for classifying sets of attribute-value pairs. Each edge in the tree corresponds to a possible attribute-value combination, and each leaf is associated with a classification for the sets of attribute-value pairs that correspond to a path from the
leaves to the root of the tree.

disjunctive normal formulae: disjunctions of boolean conjunctions.

n-ary boolean formulae: arbitrary functions mapping \( \{0,1\}^n \) to \( \{0,1\} \), for some \( n \).

A part of the problem at hand is the apparent difficulty associated with learning concepts of even moderate simplicity. Of the concept classes listed above, only the first can be “learned” in the sense that we will describe below.

If \( A \) is some particular learning algorithm, it is often interesting to look at the set of rules that \( A \) might find given appropriate training data. If we see \( A \) as a function that maps the set of training samples to the set of classification rules, this set of rules is just the range of \( A \). This range is sometimes called the hypothesis class of \( A \).

Restricting an algorithm’s hypothesis class is not the only way to constrain a learning problem; we can also make assumptions about the domain in which the learned rule is expected to operate (this is almost always the domain from which the training data comes as well). For example, one might assume that the probability density \( P(X) \) is known, along with the conditional density \( P(X|e) \) for each class \( e \) into which a point in \( X \) might fall. In that case simple Bayesian estimation can be applied, and in fact if the densities are known only up to certain parameters, it may be more efficient to estimate these parameters than to use the nonparametric methods that must be applied when nothing is known about the input distributions at all.

Such prior assumptions about the learning domain are sometimes referred to as inductive bias (this includes assumptions about the class of the target concept).

B.1.3. What information is available to a learning algorithm?

It has already been mentioned that the primary concerns of this paper are algorithms that learn from examples. To construct them, we first specify a space \( \mathcal{X} \) of possible inputs. Points in this space are called instances. In our formalism, we assume that there is a probability measure \( P_\mathcal{X} \) defined on \( \mathcal{X} \), and, when a training example is desired, an input is drawn at random according to this distribution.
We assume that there is some function $F$ which is able to map the inputs we draw to their “correct” mapping in the space of possible outputs. In general $F$ is what is supposed to be learned. This function is often referred to as the target function in the learning problem.

The goal is for the learning algorithm to learn by observing the ways in which the target function behaves at the points that are used as training examples. Therefore we do not allow information about the target function to be transmitted by the choice of training examples; we require that $P_x$ does not depend on $F$. It should be pointed out that we are not assuming the independence of an instance $x$ and its mapping $F(x)$, the problem would be quite hopeless if no such dependency existed.

Usually in literature on supervised learning, an example is an ordered pair

\[ (x, F(x)), \]

where $x \in \mathcal{S}$ and where $F$ is the target function. In this thesis we will often find it unnecessary to treat the elements of this pair as if they came from different domains, however. If $\mathcal{S}$ is the range of $F$, then we will define $X$ as $\mathcal{S} \times \mathcal{S}$. We will adopt the convention that if $x \in X = \{v, F(v)\}$, then $x''$ will refer to $v$, the input that the learning algorithm receives, while $z'''$ will refer to $F(v)$, the desired output. We will assume that there is a probability measure $P_X$ associated with points in $X$, with the property that

\[ P_X(v) = P_X((v, F(v))). \]

A sequence of examples which are drawn at random (with replacement) from $X$ according to $P_X$, and classified according to $F$, is a training sample, or simply a sample, of $F$. Such samples will constitute the inputs of our learning algorithms.

### B.1.4. What constitutes success in learning?

The quality of a learned hypothesis or function is often judged by the expected loss due to misclassification, according to some loss function $Q$, that is incurred when that hypothesis is used for making classifications. If $H$ is the learned function and $F$ is the target function, then the expected loss is defined as

\[ \eta = \int_X Q(F, H, x) dP_X. \]
Some possible choices for $Q(F, H, x)$ are $|H(x) - F(x)|$, $(H(x) - F(x))^2$, or simply $H(x) \triangleright F(x)$, where

$$\delta \triangleright \delta = \begin{cases} 1, & \text{if } a \neq b, \\ 0, & \text{otherwise.} \end{cases}$$

It is assumed that $Q(\cdot, \cdot, x)$ quantifies the loss that will be incurred by misclassifying $x$; we can also view it as a measure of the deviation between two functions at $x$. This paper will use the last of the three functions mentioned above; this makes $\eta$ the probability of obtaining an incorrect classification.

We can reformulate (B.11) in terms of $X$:

$$\eta = \int_X Q(H, x) \, dP_X; \quad (B.12)$$

where the target function $F$ is implicitly defined when we decide which part of an input $x$ will be $x^*$. Thus the three choices for $Q$ can be rewritten $|H(x^*) - x^*|$, $(H(x^*) - x^*)^2$, and $H(x^*) \triangleright x^*$.

In general, the quality of the hypothesis depends on the training sample, and therefore we will not be able to predict $\eta$ deterministically in advance. Instead, we will simply ask for the probability (in the space of possible samples) that $\eta$ will exceed some arbitrarily chosen bound $\epsilon$ ($\epsilon$ is sometimes referred to as the accuracy parameter of the learning algorithm). Our goal is to be able to specify an arbitrary $0 < \epsilon < 1$, and expect the learning algorithm to have an arbitrarily high probability of finding a hypothesis $H$ for which $\eta$ is less than $\epsilon$. (What is meant by an "arbitrarily high probability" is that we also wish to be able to specify what probability the algorithm should have of finding a good hypothesis. However, increasing this probability may cause the algorithm to take more time and require more examples. The same is true if $\epsilon$ is decreased). The probability that the algorithm will not find a good hypothesis is usually denoted by $\delta$, and sometimes referred to as the confidence parameter of the algorithm.

It should be noted that (B.12) is not useful in practice except when $H$ is used to classify inputs drawn according to $P_X$. This is, in some sense, a weakness of this framework, but it is also a limitation of learning in general, for it is always possible to make a good hypothesis bad by changing the input distribution. As an extreme example, one could make a particular point $v_0 \in \mathcal{X}$ an exception to whatever classification rule applied to the rest of the domain. One could choose $P_{X'}$ so that the probability of drawing $v_0$ was zero during training, but then test the hypothesis in a domain where $v_0$ was drawn with probability 1. It is clear that no learning algorithm can consistently do better than chance in such a situation, and an adversary with knowledge of the algorithm's structure may well be able to force worse-than-chance outcomes ($\eta > 0.5$) by making judicious choices of $F$ and $v_0$. 
B.1.5. Learnability.

In many cases of interest, it is trivial to obtain a hypothesis that perfectly classifies all inputs having a nonzero probability; if the set of possible hypotheses is enumerable, one can simply try each one of them in turn until a suitable one is found.

Of course, this approach might take considerable time, and in many cases the learning algorithm might require so many examples that learning is impractical, although not impossible. It is reasonable to require that, for all concepts in the target class, a learning algorithm be able to find a hypothesis in polynomial time. The idea was formalized in [30]: A class of concepts $\mathcal{F}$ is learnable if and only if, for any arbitrary $0 < \epsilon \leq 1$ and any arbitrary $0 < \delta \leq 1$, there exists an algorithm which:

(a) Has probability greater than $(1 - \delta)$ of finding a hypothesis $H$ whose error,

$$\int_X H(x^*) \oplus z^* \, dP_X,$$

is no greater than $\epsilon$, so long as $P$, the function that takes $z^*$ to $z^*$ for all $x \in X$, is in the target class $\mathcal{F}$.

(b) Requires a sample whose size grows at most polynomially in $1/\delta$, $1/\epsilon$, and the number of bits needed to specify a point in $X$.

(c) Has resource requirements which grow no faster than polynomially in the size of the training sample.

We require the algorithm to work for arbitrarily small values of $\epsilon$, and it was once hoped that learning algorithms might be easier to write if $\epsilon$ were held constant instead. However, this is not the case, at least for algorithms that do not depend on the form of $P_X$ (most of the algorithms presented here have that property). It can be shown ([110]) that if such an algorithm consistently produces hypotheses that perform better than chance for target functions in some class $\mathcal{F}$, then it is possible to write a learning algorithm for $\mathcal{F}$ that achieves arbitrarily small values of $\epsilon$ in polynomial time.

Moreover, if we raise $\epsilon$, we can almost always lower $\delta$ without increasing an algorithm's resource requirements. For this reason we expect that a class of concepts that is unlearnable in the sense described above will still be unlearnable if $\epsilon$ and $\delta$ are held fixed.

We have already said that we are not primarily concerned with the specific methods by which learning algorithms find a hypothesis. This increases the generality of our results, but it is useful for another reason as well: many existing learning algorithms can be seen empirically to execute with reasonable resource demands, but it is often
quite difficult to prove that they will always do so, or even to say under what circumstances they will do so.

Our results do not depend on formal proofs about the time our memory requirements of learning algorithms, they merely require that an algorithm obtain a hypothesis by one means or another. Once the hypothesis has been obtained, we analyze its quality in terms of its performance on the training data.

Our analyses involve the size of the training sample, and the training sample is, of course, one of the algorithm's resources. However, in our analyses, the number of examples needed to obtain a hypothesis of a certain quality is completely independent of the how the learning algorithm is implemented; it depends, rather, on the set of functions that the learning algorithm chooses from when it chooses a hypothesis. For this reason, the research presented here complements the bulk of the research that has been done in machine learning during the past several decades, as this research was centered on the problem of finding learning algorithms that used (more or less) reasonable amounts of time and memory, and produced hypotheses with good empirical behaviour on their training data.

Note that for each hypothesis $H$, the set of points
\[\{x \in X : H(x^*) \neq x^*\}\]
is an event in $X$. In this thesis our primary concern will be bounding the probability of this event for various hypotheses $H$.

The next section gives an example of how an algorithm can be demonstrated to learn concepts in a particular class (and hence, how a class of concepts can be shown to be learnable.)

### B.2. An example: learning boolean monomials.

To illustrate the ideas presented in the previous section, this section will present a simple learning algorithm, and show one way in which the performance of the algorithm can be analyzed. The set of functions to be learned in this case consists of all boolean functions over $n$ variables (for an arbitrary $n$) that can be written as a conjunction of attribute-value pairs.

In a boolean domain, any instance can be represented as an ordered list of 1's and 0's. Here, as in many other domains, we can say that the elements of the list correspond to the attributes of the thing being described by the list. If the $j$th element in the list is 1, we say that the thing being described possesses the $j$th attribute, whereas a 0 would indicate that the attribute was absent. For example, a 1 in the first position of the list might indicate that the object being classified has hair, while a 1 in the second position could mean that it has wheels.

Attribute-value pairs can be viewed as events in $X^n$, and this view is especially useful in boolean domains,
because the unions and intersections of such events can be used to specify boolean functions on $\mathcal{X}$. Thus, if $a$ and $b$ are boolean variables, then $a \land b$ corresponds to the boolean function $a \land b$ and so on. We will refer to a boolean attribute-value pair as a literal. If $a$ is a literal or any other event we will use $\overline{a}$ to refer to the nonoccurrence of $a$. If $a$ and $b$ are two events, and the occurrence of $a$ implies that $b$ has also occurred, we will simply say that $a$ implies $b$, or simply that $a \Rightarrow b$. Note that $a \Rightarrow b$ is equivalent to $a \subseteq b$, and not to $a \subseteq b$.

For the purpose of defining inductive bias, boolean functions are often classified by the properties of the expressions that represent them. For example, the following classifications are common in the literature on machine learning:

- **monomials**: conjunctions of literals.
- **clauses**: disjunctions of literals.
- **$DNF$ formulae**: disjunctions of monomials.
- **$CNF$ formulae**: conjunctions of clauses.
- **$k-DNF$ formulae**: disjunctions of monomials having at most $k$ literals each.
- **$k-CNFS$ formulae**: conjunctions of clauses having at most $k$ literals each.

This subsection presents a simple algorithm, $Monomial$, that can be used to learn one of the simplest of these classes, namely that of boolean monomials. Various forms of this algorithm appear in many papers (such as [30], [16], [23]). The algorithm maintains a list of literals, and uses positive examples to eliminate those that cannot appear in the target function.

An informal explanation of $Monomial$ might go as follows: if the target function $F$ can be represented as a conjunction of literals, then every literal appearing in this representation must be implied by every positive instance. $Monomial$ tries to approximate $F$ by building a conjunction whose literals all fit this criterion, and it does so by eliminating all literals that do not.

However, if one of these "inappropriate" literals is not found and removed from $L$, it can cause the resulting hypothesis to err. Any instance that implies such a literal will be classified as 0 by $H$, although it may, in fact, be a positive instance. Our goal will be to bound the probability of encountering such a literal in an instance that is supposed to be classified by $H$. 
Algorithm 3: Monomial:

Assumptions:

- $X$ is a sample space consisting of points in $\{0, 1\}^n$, for some positive $n$.
- $F$ is a function mapping $X$ to $\{0, 1\}$.
- $Ex()$ is an oracle that returns an example $(x^e, x^w)$, where $x^e \in X$ is drawn at random according to $P_x$, and $x^w = F(x^e)$.
- $m$ is a positive integer.
- $L$ is a set of literals, initially consisting of every literal that is possible in $X$.

1 {The main loop is executed $m$ times, resulting in a sample of size $m$. On each iteration, we:
} Do $m$ times:
2 {Draw a random example.} $x \leftarrow Ex()$.
3 {If the example is positive:} If $x^w = 1$:
4 {Remove from $L$, all literals that did not occur in $x$.} For each literal $l$ such that $x^e \neq l$, let $L \leftarrow L - \{l\}$.
5 {Return a hypothesis.} Let $H$ be the boolean function represented by the conjunction of the literals remaining in $L$. Return $H$ and halt.

We now restate our informal explanation in the following lemma:

Lemma B.32: The hypothesis $H$ returned by Monomial$(\cdot)$ errs on an instance $x$ only if:

(a) $x$ is a positive instance, and
(b) there is some literal $l$ in $L$ such that $\overline{l}$ appears in $x$. 
Proof: Since \( L \) initially contains all possible literals, the monomial represented by the conjunction of the literals in \( L \) initially maps all possible instances to 0 (specifically, \( L \) contains both \( \overline{t} \) and \( t \) for each literal \( l \) in the domain. Hence the intersection of the events in \( L \) is an event that never occurs). Furthermore, if a literal is removed from \( L \) its negation must have appeared in a positive example, so the literal itself cannot appear in the set of literals in the conjunction that represents \( F \) (which we will denote by \( M \)) if \( F \) is a monomial concept. Thus the literals in \( L \) always form a superset of the literals appearing in \( M \), but this means that there is no assignment of truth values to literals that will make the conjunction of the literals in \( L \) true, and still make the monomial \( F \) false.

Part (b) is obvious: if a truth assignment makes the conjunction of literals in \( L \) false then that truth assignment must make at least one of the literals in \( L \) false, and hence the truth assignment contains the negation of that literal.

\( \square \)

Of course, the hypothesis finally returned by this algorithm depends on the sequence of examples that was used to train it. If \( \mathcal{F} \) is such a sequence, then we will call the resulting hypothesis \( \mathcal{H}_\mathcal{F} \), and we will let \( E_\mathcal{F} \) denote the set of literals that appear in \( \mathcal{H}_\mathcal{F} \) but should not (these are just the variables that are in \( \mathcal{H}_\mathcal{F} \) but are not in the target concept \( F \)). Lemma B.31 states that the events that cause \( \mathcal{H}_\mathcal{F} \) to fail are the negations of the events in \( E_\mathcal{F} \). In other words, if \( l \in E_\mathcal{F} \) then \( \mathcal{H}_\mathcal{F} \) will misclassify an instance that implies \( \overline{t} \).

If \( P_X(l) \) is the probability of the literal \( l \) in \( X^* \) (that is, the probability of drawing an instance that implies \( l \)), then the expected loss

\[
\eta = \sum_{x' \in X} H(x') \cap \overline{F}(x') \ P_X(x')
\]

is at most

\[
\sum_{l_j \in E_\mathcal{F}} P_X(l_j).
\]  

Unfortunately we cannot find a numerical value for (B.13) until we know \( E_\mathcal{F} \), and we do not have an a priori knowledge of \( E_\mathcal{F} \) since it depends on the training sample \( \mathcal{F} \), from which the algorithm learns. Since \( \mathcal{F} \) is a random variable (from the space \( \mathcal{X}^t \) for some positive integer \( t \)), we cannot deterministically predict \( E_\mathcal{F} \).

Instead, our approach will be to choose some \( 0 < \epsilon \leq 1 \) and try to determine the probability (in \( X^* \)) with which \( \eta \) will exceed \( \epsilon \). To this end, suppose that \( m \) is an upper bound on the number of literals in \( E_\mathcal{F} \). In this case, the error of the hypothesis will certainly be less than \( \epsilon \) if the negations of the literals in \( E_\mathcal{F} \) all have probabilities less than \( \epsilon/m \).

Therefore, we define \( C \) as the set of literals in \( X^* \) whose probability is greater than \( \epsilon/m \). The probability of generating a hypothesis with an error greater than \( \epsilon \) is no greater than the probability of drawing a sample \( \mathcal{F} \) such
that

\[ \exists(i \in E_F) \text{ s.t. } \overline{t} \in C. \]

This probability is no greater than

\[ \sum_{t \in C} P_{x^t}(\overline{t} : t \in E_F) \quad (B.14) \]

for a given \( t \), where \( P_{x^t} \) denotes a probability in the space \( x^t \).

Any literal appearing in \( E_F \) must also be in \( H_F \), according to lemma B.32. But inspection of the algorithm \textit{Monomial} will show that, if \( t \in H_F \), then \( \overline{t} \) did not occur in any of the training examples seen by the algorithm. If \( \overline{t} \in C \) then

\[ \left(1 - \frac{\epsilon}{m}\right)^t, \]

so (B.14) is no greater than

\[ \sum_{t \in C} \left(1 - \frac{\epsilon}{m}\right)^t. \quad (B.15) \]

But the summation is over 2\( n \) or fewer literals, since there are only 2\( n \) literals in the domain. Thus the probability of finding a hypothesis whose error is greater than \( \epsilon \) is at most

\[ 2n \left(1 - \frac{\epsilon}{m}\right)^t. \quad (B.16) \]

Of course \( m \) can also be no greater than 2\( n \); in fact, it can be no greater than \( n \) because \( n \) literals are necessarily eliminated in the first iteration of \textit{Monomial}. But perhaps we have reason to believe that the target concept contains fewer literals still; then \( m \) is less than \( n \), and the probability of finding a bad hypothesis decreases accordingly.

Now we ask how many examples \textit{Monomial} should get. To answer this question, we first recall the definition of \( \delta \): it is a desired bound on the probability that the final hypothesis will have an error greater than \( \epsilon \). We have already argued that this probability is bounded by (B.15). Therefore we want to arrange things so that

\[ 2n \left(1 - \frac{\epsilon}{m}\right)^t \leq \delta. \]

We solve this inequality for \( t \) to find that we should draw no fewer than

\[ \frac{\ln \delta - \ln(2n)}{\ln(1 - \epsilon/m)} \quad (B.17) \]

training examples.

Note that the literals in \( L \) can replaced by arbitrary events in \( \mathcal{F} \) as long as the set \( L \) does not grow to large to enumerate during step 2 of \textit{Monomial}. In [30]'s presentation of the algorithm, \( L \) consisted of all boolean
disjunctions containing \( k \) or fewer literals for some constant \( k \) (note that \(|I|\) grows exponentially in \( k \)). Thus \textit{Monomial} can be used to learn \( k - CNF \). If we restrict ourselves to \( k - CNF \) functions that have no more than \( m \) clauses, there will be at most \( m \) events in \( E_S \) for any \( S \), and the bound (B.17) improves accordingly.

\( I \) can also be a list of arbitrarily complex predicates; the requirement is simply that the predicates be known beforehand. On the other hand we cannot simply try the algorithm with one set of predicates, see if \textit{Monomial} succeeds, and try another set if it fails. If we did this, the probability of finding a bad hypothesis might be as large as (B.16) for each attempt (as least as far as we know from the analysis above). Thus \( k \) could be as large as

\[
2n \left(1 - \frac{1}{e}\right)^m
\]

if \( h \) attempts were to be made.

\textbf{B.3. Sample complexity in classification algorithms.}

The paradigm we used to analyze \textit{Monomial}, in it's most general form, is the following: Given a learning algorithm \( A \), and a sample \( \mathcal{F} \), bound the error of \( H = A(\mathcal{F}) \) in terms of \( H \)'s empirical error on the training sample:

\[
\hat{\eta}(H, \mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{x \in \mathcal{F}} H(x') \oplus z^e.
\]  

(B.18)

This section surveys existing results that relate the empirical error (B.18) of a hypothesis to its error in the space \( X \), the latter given by

\[
\eta = \int_X H(z') \oplus z^e \, dp_x.
\]  

(B.19)

We will sometimes refer to \( \eta(H) \) as the \textit{actual error} of \( H \), to distinguish it from the empirical error.

We can relate the actual and empirical errors of a hypothesis using Hoeffding's inequality [17], which, in the current context, states that

\[
P_{X^l}(\mathcal{F} : |\eta(H) - \hat{\eta}(H, \mathcal{F})| > \alpha) < 2e^{-2\alpha^2l},
\]  

(B.20)

where \( P_{X^l} \) represents a probability in the space of samples having size \( l \). Informally, (B.20) states that the probability is less than \( 2e^{-2\alpha^2l} \) of drawing a sample \( \mathcal{F} \) for which the empirical error of the hypothesis \( H \) differs from its true error by more than \( \alpha \), if \( \mathcal{F} \) contains at least \( l \) examples.

Let \( \mathcal{M} \) be the set of hypotheses from which a learning algorithm may choose (\( \mathcal{M} \) is the algorithm's hypothesis class, as in chapter 2). The probability of choosing a hypothesis for which the empirical error exceeds the actual error by more than \( \alpha \) is certainly no greater than the sum of the probabilities (B.20) over the individual hypotheses;
if there are \(|\mathcal{H}|\) hypotheses the probability is no greater than

\[ |\mathcal{H}|2e^{-2n\delta}. \]

Equating this quantity with \(\delta\) and solving for \(\ell\) we find that, in all cases, a sample of size at least

\[ \frac{\ln(\delta) - \ln(|\mathcal{H}|)}{-4\alpha^2} \quad (B.21) \]

will ensure with probability \(1 - \delta\) that the empirical error and the actual error of a hypothesis differ by no more than \(\alpha\).

The bound can be improved if we only permit the algorithm to consider hypotheses that are consistent with the training sample, that is, hypotheses that do not misclassify any of the training examples (those \(H \in \mathcal{H}\) with the property that \(\eta(H, F) = 0\) for the particular \(F\) that is chosen as a training sample). If a hypothesis \(H\) has an error greater than \(\epsilon\), the probability of drawing a sample \(F\) such that \(\eta(H, F) = 0\) is at most

\[ (1 - \epsilon)^{\ell} \]

if \(F\) contains \(\ell\) examples. The probability that the learning algorithm will choose a hypothesis that has an error greater than \(\epsilon\) is thus no more than

\[ |\mathcal{H}|(1 - \epsilon)^{\ell}, \]

so that no more than

\[ \frac{\ln(\delta) - \ln(|\mathcal{H}|)}{\ln(1 - \epsilon)} \quad (B.22) \]

examples are needed to ensure that the probability of choosing such a hypothesis is less than \(\delta\).

More refined bounds than those given above can often be obtained by using a parameter known as the Vapnik-Chervonenkis Dimension of a hypothesis class.

To introduce this concept we begin with a more basic one; that of the number of partitions that a hypothesis class can induce on a sequence of examples.

If \(\mathcal{H}\) is a set of \(\{0, 1\}\)-valued hypotheses and \(F\) is a sequence of examples drawn from \(\text{Ran}(\mathcal{H})\), then we can say that each \(H \in \mathcal{H}\) partitions \(F\) into those elements in

\[ \{x \in F : H(x) = 0\} \]

and those that lie in

\[ \{x \in F : H(x) = 1\}. \]
What concerns is here is how many unique partitions can be induced on \( \mathcal{F} \) in this way by members of \( \mathcal{N} \). For although each \( H \in \mathcal{N} \) can produce a unique partition under the right circumstances, this does not always happen. In particular, there may not be up to \( |\mathcal{N}| \) ways different ways to partition \( \mathcal{F} \), especially when we take into account that if \( x_1 = x_2 \), then \( x_1 \) must lie in the same partition as \( x_2 \) regardless of which hypothesis induces the partitioning. Thus, if there are only ten unique points in a training sequence, there are only \( 2^{10} \) ways to partition the sequence in the manner we have described. If \( \mathcal{N} \) contains more than \( 2^{10} \) hypotheses there must be two or more unique hypotheses in \( \mathcal{N} \) that nonetheless induce the same partitioning on \( \mathcal{F} \).

We denote by \( \Pi_{\mathcal{N}}(\mathcal{F}) \) the number of unique partitions that \( \mathcal{N} \) can induce on \( \mathcal{F} \).

To see how \( \Pi_{\mathcal{N}}(\mathcal{F}) \) relates to machine learning, notice that if two hypotheses \( H_1 \) and \( H_2 \) induce the same partition on \( \mathcal{F} \), then they are equivalent with respect to \( \mathcal{F} \) in that, if \( H_1 \) misclassifies an example in \( \mathcal{F} \) then \( H_2 \) misclassifies the example too, while \( H_1 \) correctly classifies any example that \( H_2 \) classifies correctly. Therefore, from the point of view of the algorithm that bases its choice of hypotheses solely on training examples, there is no reason to prefer \( H_1 \) over \( H_2 \), or vice versa.

Clearly such an algorithm can do something productive if it is given more training examples: if the right example turns up it may be able to determine if \( H_1 \) or \( H_2 \) is the better hypothesis.

Let us now consider the other extreme. Suppose that each hypothesis in \( \mathcal{N} \) induces a unique partitioning on \( \mathcal{F} \). Then we can proceed as follows: first, we partition \( \mathcal{F} \) into

\[
\{ x \in \mathcal{F} : x^\mu = 1 \} \quad \text{(B.23)}
\]

and

\[
\{ x \in \mathcal{F} : x^\mu = 0 \} \quad \text{(B.24)}
\]

If any hypotheses in \( \mathcal{N} \) induce this partitioning, then they are distinguished by virtue of classifying each example correctly. If every hypothesis in \( \mathcal{N} \) induces a unique partition, then the "correct" partitioning ((B.22) and (B.23)) can be induced by at most one of the hypotheses, so we have isolated a single hypothesis that is "best" for this particular training sample. Further examples may reduce our confidence in this hypothesis but they will not give us another hypothesis that classifies every example correctly. In fact, if we know that the function \( F \) we are trying to learn is identical with one of the hypotheses in \( \mathcal{N} \), then we have discovered exactly which hypothesis \( F \) is identical to (we have, in the terminology of machine learning, identified the target function).

However, it may be overambitious to look for a sample that can actually be partitioned in a unique way by every hypothesis in \( \mathcal{N} \). \( \mathcal{N} \) may contain an infinite number of hypotheses, and even if it does not, it may be that we cannot get \( |\mathcal{N}| \) partitionings of a sample unless it contains some examples that have a very low probability.
Another way to estimate the usefulness of a sample is by asking whether or not

$$\Pi_{\mathcal{F}}(\mathcal{X}) < 2^{|\mathcal{F}|}.$$ 

In a sample where this inequality does not hold, every example is useful if we are looking for a hypothesis that classifies all examples correctly, because in this case every example can eliminate at least one hypothesis from consideration.

To illustrate this, consider a sample $\mathcal{X}$ and an example $x_0$ such that

$$\Pi_{\mathcal{F}}(\mathcal{X} + x_0) = 2^{|\mathcal{F}|+1}. \tag{B.25}$$

Assume that

$$\{x \in \mathcal{X} : x^* = 1\}, \{x \in \mathcal{X} : x^* = 0\} \tag{B.26}$$

is one of the several partitions that $\mathcal{F}$ can induce on $\mathcal{X}$. Any hypothesis that classifies all examples correctly will induce the partition (B.26) on $\mathcal{X}$, but all hypotheses that induce (B.26) may not be equally good. We would like to be able to differentiate among these hypotheses.

Now consider the sample $\mathcal{X} + x_0$. Since there cannot be more than $2^{|\mathcal{F}|}$ partitions of $\mathcal{X}$ in our scheme, it follows from (B.25) that there are twice as many partitions of $\mathcal{X} + x_0$ as there are of $\mathcal{X}$. Now consider any partitioning

$$\{x \in \mathcal{X} : H(x') = 1\}, \{x \in \mathcal{X} : H(x') = 0\} \tag{B.27}$$

of $\mathcal{X}$. For this partitioning there are at most two partitionings of $\mathcal{X} + x_0$ which are identical to (B.26) except that $x_0$ is added to one of the two partitions:

$$\{x \in \mathcal{X} : H(x') = 1\} + x_0, \{x \in \mathcal{X} : H(x') = 0\} \tag{B.28}$$

and

$$\{x \in \mathcal{X} : H(x') = 1\}, \{x \in \mathcal{X} : H(x') = 0\} + x_0. \tag{B.29}$$

In order for there to be twice as many partitionings of $\mathcal{X} + x_0$ as there are of $\mathcal{X}$ (as we stipulated in (B.25)), every partitioning of $\mathcal{X}$ must correspond to two partitionings of $\mathcal{X} + x_0$ as in (B.28) and (B.29). But this means that, for every partitioning $p$ of $\mathcal{X}$, there is at least one hypothesis in $\mathcal{F}$ that induces $p$ classifies $x_0$ as 1, and at least one that classifies $x_0$ as 0.

Since this holds of every partitioning, it holds, in particular, of (B.27). But if $x^* \neq 0$, then the hypothesis that classifies $x_0$ as 1 will be eliminated from consideration after we see $x_0$. Likewise, if $x^* = 1$, the hypothesis that classifies $x_0$ as 0 will be eliminated.
In short, we are guaranteed that we will be able to eliminate at least one hypothesis from consideration after seeing $x_0$.

But if $\Pi_{\mathcal{H}}(\mathcal{F} + x_0)$ were less than $2^{2\ell+1}$ we would not have the same guarantee, because we would no longer be able to argue that each partitioning of $\mathcal{F}$ corresponded to two partitionings of $\mathcal{F} + x_0$, as in (B.28) and (B.29). Hence it may be that there are no hypotheses that can be eliminated as the result of seeing $x_0$.

In some sense, we have reached a point of diminishing returns. We may still benefit from training examples, but we are not guaranteed to benefit from an arbitrarily chosen unique example.

This point of diminishing returns is characterized by the Vapnik-Chervonenkis Dimension of $\mathcal{H}$, which we denote by $V(\mathcal{H})$. Specifically,

$$V(\mathcal{H}) = \max \left\{ |\{ z : z \in \mathcal{F} \} : \mathcal{F} \text{ is drawn from } \text{Ran}(\mathcal{H}) \text{ and } \Pi_\mathcal{F}(\mathcal{F}) = 2^{1|\{ z \in \mathcal{F} \}|} \right\}.$$ 

In other words, it is the largest $\ell$ for which some subset $S$ of $\text{Ran}(\mathcal{H})$ can be partitioned in $2^{1\ell}$ ways. (Note that in our formal definition we replaced the sequence $\mathcal{F}$ with the set $\{ z : z \in \mathcal{F} \}$. This was done because we only wished to consider the number of unique examples in $\mathcal{F}$ and not the total number of examples in the sequence.

Several authors have presented results that use the Vapnik-Chervonenkis dimension of a hypothesis class used to bound (in probability) the inaccuracy of a hypothesis in terms of its empirical error.

**Theorem B.33:**

(a) [32]: For any class $\mathcal{H}$ of hypotheses, if $X = \text{Dom}(\mathcal{H}) \times \text{Ran}(\mathcal{H})$, then

$$P_{\mathcal{X}} \left( \exists H \in H \text{ s.t. } |\phi_H(H) - \eta(H)| \geq \xi \right)$$

is less than $\delta$ if

$$m \geq \frac{16}{\xi^2} \left( d \ln \frac{16d}{\delta} - \ln \frac{1}{4} \right).$$

(b) [7]: For any class $\mathcal{H}$ of hypotheses, if $X = \text{Dom}(\mathcal{H}) \times \text{Ran}(\mathcal{H})$, then

$$P_{\mathcal{X}} \left( \exists H \in H \text{ s.t. } \frac{\phi_H(H)}{\eta(H)} \geq \xi \right)$$

is less than $\delta$ if

$$m \geq \max \left( \frac{8}{(1 - \xi)^2} \ln \frac{8}{\delta}, \frac{16d}{(1 - \xi)^2} \ln \frac{16}{(1 - \xi)^2} \right).$$
(c) [17]: For any class $\mathcal{H}$ of hypotheses, if $X = \text{Dom}(\mathcal{H}) \times \text{Ran}(\mathcal{H})$ and $d = \mathcal{V}(\mathcal{H})$ then the probability that a hypothesis whose empirical error is zero will have an actual error greater than $\epsilon$ is bounded by $\delta$, if the training sample contains at least
\[
\max \left( \frac{4}{\epsilon} \log_2 \frac{2}{\delta}, \frac{8d}{\epsilon} \log_2 \frac{13}{\delta} \right)
\]
examples.

(d) [19]: For any class $\mathcal{H}$ of hypotheses, let $X = \text{Dom}(\mathcal{H}) \times \text{Ran}(\mathcal{H})$ and $d = \mathcal{V}(\mathcal{H})$. If $X$ is countable then the probability that a hypothesis whose empirical error is zero will have an actual error greater than $\epsilon$ is bounded by $\delta$, if the training sample contains at least
\[
\frac{1}{\epsilon} \left( \log_2(|X|) d \ln(2) + \ln \left( \frac{1}{\delta} \right) \right)
\]
examples.

e [19]: For any class $\mathcal{H}$ of hypotheses, let $X = \text{Dom}(\mathcal{H}) \times \text{Ran}(\mathcal{H})$ and $d = \mathcal{V}(\mathcal{H})$. The probability that a hypothesis whose empirical error is zero will have an actual error greater than $\epsilon$ is bounded by $\delta$, if the training sample contains at least
\[
\frac{8}{\epsilon} \left( 4d \ln \left( \frac{32}{\epsilon} \right) + \ln \left( \frac{2}{\delta} \right) \right)
\]
examples.


In spite of the fact that computational learning is usually concerned with the overall tractability of a machine learning problem, this thesis deals mostly with the requirement that the sample size grow only polynomially in the various parameters that specify the problem (see above). The other tractability requirement, that the resource requirements of the learning algorithm be polynomial in the sample size, is less important because heuristics can be used to find a hypothesis once we have a sample.

The bulk of this thesis deals with ways of relating the empirical error of hypotheses, measured on a training sample, to the hypotheses' actual error in $X$. Instead of assuming that a problem will be abandoned if an optimal
solution cannot be found, we will assume that a suboptimal hypothesis is better than none at all, so that heuristic learning algorithms will not, by any means, be ruled out for finding hypotheses.

In fact, the results of this thesis are completely complementary to research that has been done with the purpose of finding algorithms with good empirical performance on specific problems (a large portion of the machine learning research to date has had this goal). If we have, by one means or another, discovered an algorithm that has good empirical performance on a given type of problem, we are often not concerned very deeply with a formal analysis of the time and memory requirements of the algorithm, because this can be difficult to do, and we already have observed that the algorithm usually does not require unreasonable resources of this type. Moreover, if the algorithm does not execute with reasonable resource demands in a particular instance, we will be able to see this.

What we cannot easily determine by observation is whether or not the algorithm has had enough training examples to produce a good hypothesis. We could test the hypothesis with data that was not used for training, but what is to be done if we discover that the hypothesis is not as good as we had hoped? If add the test data to the training data and obtain a new hypothesis, we will have no way to test the new hypothesis; if, instead, we rerun the algorithm on completely new data, then we will increase our likelihood of drawing a training sample that is uncharacteristic of the real learning domain and thus makes a bad hypothesis look good (for we must take into account the probability that this will happen on the first attempt or on the second attempt or the third attempt, etc.).

The results in the last subsection have the advantage of being valid when all available data has been used for training. They therefore give us the information that we could not easily obtain empirically when we executed the learning algorithm: information, specifically, about the quality of the learned hypothesis. In addition, as we have already noted, they make no specific reference to the algorithm that was used to find this hypothesis. Thus they can, in principle, be used any learning algorithm, regardless of whether formal methods were used to construct the algorithm.

These results do, however, require some parameter that describes the complexity of the hypotheses that the algorithm can produce; some use the number of possible hypotheses, while others use the Vapnik-Chervonenkis Dimension of the class of hypotheses. The first two chapters of this thesis describe ways in which these parameters can be obtained.

B.4. Bibliography

There are a number of formulations of the concepts given in section one of this appendix; the one that was used in this appendix comes from [33], except for the requirement that algorithms have polynomial resource requirements,
which is from [30]. The present formulation is also widely used elsewhere (Cf. [21]). [30]'s formulation was similar except in that the accuracy and confidence parameters were identical there.

There are a number of different paradigms that can be used for training: in this appendix we stated that the supervisory input was provided to the learning algorithm along with a randomly drawn input from X, but it would also have been possible (for example) to have the algorithm output a guess for each instance, and then tell the algorithm whether the guess was correct or incorrect. A number of these variants are discussed in [15], and some are shown to be equivalent to one another.

The learning algorithm *Monomial* was given in [30], with the predicates in I consisting of \( k \)-ary disjunctions of boolean variables. The analysis of the algorithm in this appendix original, but it was chosen because of its simplicity, and it does not give tight results. Better results are obtained in [16] and [23]; the former paper also shows that negative examples can be used to prune a monomial hypothesis and reduce the number of examples needed by the algorithm.
References


<table>
<thead>
<tr>
<th>Lemma/Proposition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lemma 3B.24</td>
<td>130</td>
</tr>
<tr>
<td>Lemma 3B.25</td>
<td>131</td>
</tr>
<tr>
<td>Lemma 3B.27</td>
<td>134</td>
</tr>
<tr>
<td>Lemma 3B.28</td>
<td>135</td>
</tr>
<tr>
<td>Lemma 3B.30</td>
<td>137</td>
</tr>
<tr>
<td>Lemma 3B.39</td>
<td>162</td>
</tr>
<tr>
<td>Lemma 3B.41</td>
<td>165</td>
</tr>
<tr>
<td>Lemma 4.3</td>
<td>187</td>
</tr>
<tr>
<td>Lemma 4A.12</td>
<td>200</td>
</tr>
<tr>
<td>Lemma 4A.13</td>
<td>201</td>
</tr>
<tr>
<td>Lemma 4A.14</td>
<td>202</td>
</tr>
<tr>
<td>Lemma B.32</td>
<td>269</td>
</tr>
<tr>
<td>Linear discriminant functions</td>
<td>22</td>
</tr>
<tr>
<td>(S_1 \cap S_2) ((S_1) and (S_2) lists)</td>
<td>8</td>
</tr>
<tr>
<td>(S \cap E) ((S) a list, (E) a set)</td>
<td>8</td>
</tr>
<tr>
<td>(\log_e)</td>
<td>5</td>
</tr>
<tr>
<td>Misclassification of an example</td>
<td>3</td>
</tr>
<tr>
<td>(\Pi)</td>
<td>6</td>
</tr>
<tr>
<td>Occam algorithm</td>
<td>180</td>
</tr>
<tr>
<td>(\omega) superscript</td>
<td>2</td>
</tr>
<tr>
<td>(P) notation</td>
<td>5</td>
</tr>
<tr>
<td>(\text{pac-learnability})</td>
<td>261</td>
</tr>
<tr>
<td>(\Pi\left(\left{S\right}\right))</td>
<td>9</td>
</tr>
<tr>
<td>(\Pi\left(\left{f\right}\right))</td>
<td>9</td>
</tr>
<tr>
<td>(\Pi\left(\left{\cdot\right}\right)) notation</td>
<td>9</td>
</tr>
<tr>
<td>Probably approximately correct</td>
<td>261</td>
</tr>
<tr>
<td>Proposition 2.11</td>
<td>20</td>
</tr>
<tr>
<td>Proposition 2.12</td>
<td>21</td>
</tr>
<tr>
<td>Proposition 2.14</td>
<td>24</td>
</tr>
<tr>
<td>Proposition 2.7</td>
<td>14</td>
</tr>
<tr>
<td>Proposition 2.7</td>
<td>14</td>
</tr>
<tr>
<td>Proposition 2.8</td>
<td>15</td>
</tr>
<tr>
<td>Proposition 3.11</td>
<td>102</td>
</tr>
<tr>
<td>Proposition 3.11</td>
<td>102</td>
</tr>
<tr>
<td>Proposition 3.13</td>
<td>106</td>
</tr>
<tr>
<td>Proposition 3.3</td>
<td>89</td>
</tr>
<tr>
<td>Proposition 3.5</td>
<td>95</td>
</tr>
<tr>
<td>Proposition 3.9</td>
<td>100</td>
</tr>
<tr>
<td>Proposition 3.9</td>
<td>100</td>
</tr>
<tr>
<td>Proposition 3A.15</td>
<td>110</td>
</tr>
<tr>
<td>Proposition 4.1</td>
<td>184</td>
</tr>
<tr>
<td>Proposition 4.10</td>
<td>196</td>
</tr>
<tr>
<td>Proposition 4.2</td>
<td>186</td>
</tr>
<tr>
<td>Proposition 4.4</td>
<td>189</td>
</tr>
<tr>
<td>Proposition 4.6</td>
<td>190</td>
</tr>
<tr>
<td>Proposition 4.7</td>
<td>193</td>
</tr>
</tbody>
</table>

\[\mathcal{B}(\delta_i)\] | 149 |
\[\mathbb{R}\] | 6 |
\(\text{Run notation}\) | 6 |
\(\text{ricetic combination}\) | 19 |
\(\text{rule based systems}\) | 22 |
\(\text{rules}\) | 22 |
\(\text{sample complexity}\) | 7 |
\(\text{shattering a set}\) | 9 |
\(S^m\) notation | 5 |
\(\square\) | 6 |
\(\text{sublinear hypothesis update mechanism}\) | 185 |
\(\text{supervisory input}\) | 260 |
\(\text{Target function}\) | 2 |
\(\text{theorem 2A.23}\) | 46 |
\(\text{theorem 2.4}\) | 11 |
\(\text{theorem 2.4}\) | 12 |
\(\text{theorem 2A.18}\) | 34 |
\(\text{theorem 2A.23}\) | 46 |
\(\text{theorem 2A.25}\) | 51 |
\(\text{theorem 2A.30}\) | 60 |
\(\text{theorem 2A.31}\) | 61 |
\(\text{theorem 2A.32}\) | 62 |
\(\text{theorem 2A.33}\) | 67 |
\(\text{theorem 2A.34}\) | 70 |
\(\text{theorem 2A.35}\) | 76 |
\(\text{theorem 2A.37}\) | 78 |
\(\text{theorem 2A.38}\) | 79 |
\(\text{theorem 3B.16}\) | 111 |
\(\text{theorem 3B.20}\) | 119 |
\(\text{theorem 3B.22}\) | 123 |
\(\text{theorem 3B.26}\) | 132 |
\(\text{theorem 3B.30}\) | 137 |
\(\text{theorem 3B.32}\) | 140 |
\(\text{theorem 3B.33}\) | 142 |
\(\text{theorem 3B.38}\) | 155 |
\(\text{theorem 3B.41}\) | 166 |
\(\text{theorem 3B.42}\) | 176 |
\(\text{theorem 3B.43}\) | 177 |
\(\text{theorem 3B.44}\) | 179 |
\(\text{theorem 4A.15}\) | 203 |
\(\text{theorem 4A.16}\) | 204 |
\(\text{theorem 4A.17}\) | 206 |
\(\text{theorem 4A.18}\) | 207 |
\(\text{theorem 4A.19}\) | 208 |
\(\text{theorem 4A.20}\) | 212 |
\(\text{theorem 4A.22}\) | 214 |
\(\text{theorem 4A.22}\) | 215 |
\(\text{theorem 4A.23}\) | 222 |
theorem 4A.24 ...................................................... 226
theorem 4A.25 ...................................................... 227
theorem 4A.26 ...................................................... 233
theorem 4A.27 ...................................................... 234
theorem 4A.28 ...................................................... 243
theorem 4A.29 ...................................................... 253
theorem 4A.30 ...................................................... 254
theorem 4A.31 ...................................................... 257
theorem B.33 ...................................................... 277
θ notation ............................................................. 108
Training example ................................................... 2
Training sample .................................................... 2
\(\mathcal{U}(\cdot, \cdot)\) notation ........................................ 99
uniform hypothesis update mechanism ...................... 183
uniform sublinear hypothesis update mechanism ........ 186
\(\mathcal{V}\) notation .................................................... 78
\(V\) ..................................................................... 8
Vapnik-Chervonenkis Dimension .................................. 8
VC Dimension ....................................................... 8
The space \(\mathcal{E}\) ...................................................... 1
The space \(\mathcal{X}\) ...................................................... 2
The space \(\mathcal{Y}\) ...................................................... 2
Add\(^{(\cdot, \cdot)}\) .......................................................... 183
Con\(^{(\cdot, \cdot)}\) .......................................................... 183
\(\text{Par}^{(\cdot, \cdot)}\) \((\text{Par} a \text{ hypothesis update mechanism})\) ...... 183
\(|n|\) \((n \text{ numeric})\) .................................................. 6
\(|\cdot|\) notation ........................................................ 5
\(|S|\) \((S \text{ a list})\) ..................................................... 9
VITA

Christoph Cornelius Michael