2008

# Structural induction: towards automatic ontology elicitation

Adrian Silvescu
*Iowa State University*

**Structural induction: towards automatic ontology elicitation**

by

Adrian Silvescu

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Vasant Honavar, Major Professor
Drena Dobbs
Jack Lutz
William Robinson
Dimitris Margaritis

Iowa State University

Ames, Iowa

2008

UMI Number: 3307040

UMI Microform 3307040



ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to my advisor Dr. Vasant Honavar for guiding the research presented in this dissertation throughout my Ph.D. student years. He has been a constant source of motivation and encouragement. He helped me to develop my own views and opinions about the research I undertook. I thank him for being always accessible and for providing invaluable feedback on my work. Thanks for organizing the AI seminar which brought up thoughtful discussions and helped me to broaden my views about Artificial Intelligence. Most importantly, I thank him for his friendship and for his encouragement when I felt confused or overwhelmed. I would especially like to thank him for his support during my occasional hard times.

I would like to thank Drena Dobbs for teaching such refreshing courses in Molecular Biology and for tirelessly unraveling for me its secrets. I would like to thank William Robinson for teaching some of the best classes I took at the Iowa State University and elsewere which reopened my interests in philosophy and kept them open ever since. I would like to thank Jack Lutz for his wonderful class on Randomness and Computation. I would like to thank Dimitris Margaritis for serving on my committee and providing helpful comments. I would also like to thank David Fernandez-Baca for the very nice and exciting Algorithms classes he taught.

I would like to thank the members of the AI Lab and especially my collaborators for the wonderful times we spent doing research together. My interaction with you has made me love research and led to my bettering.

I would like to thank and gratefully acknowledge the financial support that I have received during my studies. From the Computer Science Department as a teaching assistantship. From Pioneer Hi-Bred as a fellowship for which I am very grateful as it has allowed me to bootstrap

my research program. I am also very grateful for the research assistantships funded by grants from the National Science Foundation (IIS 0219699) and the National Institutes of Health (GM066387) as they allowed me to continue my research program at a sustained pace.

# ABSTRACT

Induction is the process by which we obtain laws (and more encompassing - theories) about the world. This process can thought as aiming to derive two aspects of a theory: a Structural aspect and a Numerical aspect respectively. The Structural aspect is concerned with the entities modeled and their interrelationship, also known as ontology. The Numerical aspect is concerned with the quantities involved in the relationships among the above-mentioned entities along with uncertainties either postulated to exist in the world or inherent to the nature of the induction process.

In this thesis we will focus more on the structural aspect hence the name: Structural Induction: toward Automatic Ontology Elicitation. In order to deal with the problem of Structural Induction we need to solve two main problems: 1. We have to say what we mean by Structure (***What?***); and 2. We have to say how to get it (***How?***). In this thesis we give one very definite answer to the first question (***What?***) and we also explore how to answer the second question (***How?***) in some particular cases. A comprehensive answer to the second question (***How?***) in the most general setup will involve dealing very carefully with the interplay between the Structural and Numerical aspects of Induction and will represent a full solution to the Induction problem. This is a vast enterprise and we will only be able touch some aspects of this issue.

The main thesis presented in this work is that the fundamental structural elements based on which theories are constructed are Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into a bigger unit topologically close entities - in particular spatio-temporally close). This thesis is supported by showing that each member of the Turing equivalent class of General Generative Grammars can be decomposed in terms of

these operators and their duals (Reverse Abstraction and Reverse SuperStructuring, respectively). Thus, if we are to believe the Computationalistic Assumption (that the most general way to present a finite theory is by the means of an entity expressed in a Turing equivalent formalism) we have proved that our thesis is correct. We will call this thesis the *Abstraction + SuperStructuring thesis*. The rest of the thesis is concerned with issues in the vein opened by the second question presented above (***How?***): Given that we have established what we mean by Structure, how to get it?.

# CHAPTER 1. GENERAL INTRODUCTION

Induction is the process by which we obtain laws (and more encompassing - theories) about the world. This process can be thought as aiming to derive two aspects of a theory: a Structural / Qualitative / Algebraic aspect and a Numerical / Quantitative / Analytical (+ Probabilistic) aspect. The Structural aspect is concerned with the entities modeled and their interrelationship, in philosophical parlance known as ontology. The Numerical aspect is concerned with the quantities involved in the relationships among the above-mentioned entities along with uncertainties either postulated to exist in the world or inherent to the nature of the induction process. Given that the Structural aspect is settled then the Numerical aspect can be more easily handled by the tools of Analysis, Statistics, Machine Learning and Empirical Sciences. The problem of devising the Structural aspect automatically - Automatic Ontology Elicitation (a.k.a. Ontology Learning) is however a somewhat less explored and settled field. This problem: Structural Induction - Automatic Ontology Elicitation is the main topic of this thesis.

In order to deal with the problem of Structural Induction we need to solve two main problems: 1.We have to say what we mean by Structure (***What?***); and 2.We have to say how to get it (***How?***). In this thesis we give a very definite answer to the first question (***What?***) and we also explore how to answer the second question (***How?***) in some particular cases. A comprehensive answer to the second question (***How?***) in the most general setup will involve dealing very carefully with the interplay between the Structural and Numerical aspects of Induction and will represent a full solution to the Induction problem. This is a vast enterprise and we will only be able touch some aspects of this issue.

The main thesis presented in this work is that the fundamental structural elements based on which theories are constructed are Abstraction (grouping similar entities under one overarching

category) and Super-Structuring (grouping into a bigger unit topologically close entities - in particular spatio-temporally close). This thesis is supported by showing that each member of the Turing equivalent class of General Generative Grammars can be decomposed in terms of these operators and their duals (Reverse Abstraction and Reverse SuperStructuring, respectively). Thus, if we are to believe the Computationalistic Assumption (that the most general way to present a finite theory is by the means of an entity expressed in a Turing equivalent formalism) we have proved that our thesis is correct. We will call this thesis the *Abstraction + SuperStructuring thesis*. We prove this claim in Chapter 5 - Abstraction SuperStructuring Normal Forms. The rest of the chapters are concerned with issues in the vein opened by the second question presented above (***How?***): Given that we have established what we mean by Structure, how to get it?.

The rest of this chapter will proceed as follows: In the next section we will examine in a little bit more detail the problem of Structural Induction / Ontology Learning and also try to build some intuitions about the Abstraction and SuperStructuring operators. In a certain sense this is the starting point of an inductive argument for these structural elements, which will be augmented latter on with experimental results and also with a deductive argument in Chapter 5. Then in the last section of this chapter we give an overview of the thesis organization.

## 1.1 Structural Induction: Towards Automatic Ontology Elicitation

The overwhelming majority of problem settings rely on the existence of some primitive concepts/features (representing an initial / apriori ontology) in terms of which the problem, including its goals, is phrased and then subsequently solved. The choice of this initial ontology may have a big impact on the tractability of the problem under consideration (e.g., learning the definition of an arch is more difficult to get from raw pixel data, than from images annotated with primitive geometric objects or contours). We are interested in automatic means of deriving such an ontology. More exactly, given some data, from a particular domain under study, elicit a set of concepts and relations among them that are "relevant" to the particular domain under study. The set of concepts and their relations represent the proposed ontology.

This section is organized as follows: In the next subsection we outline our main paradigm regarding ontology elicitation. Then in the subsection following we examine a possible way to implement it. Finally, we address the problem of how to use ontologies and the possible interactions with the generation process.

### 1.1.1 The main paradigm

The main paradigm to guide our attempts to solve the ontology elicitation problem can be summarized as follows:

*"There are at least two essential operations that people make when trying to make sense of, or structure a new application domain: Super-Structuring and Abstraction. Super-Structuring is the process of grouping and subsequently naming a set of entities that occur within ""proximity"" of each other, into a more complex structural unit (e.g., four wheels and a chassis on top of them will be called a car). Abstraction, on the other hand, establishes that a set of entities belong to the same category, based on a certain notion of "similarity" among these entities, and subsequently names that category (e.g., cows, cats and dogs will be called mammals)." [13]*

Note that in order to make more precise the above stated paradigm we need to define more exactly what we mean by "proximity" *and* "similarity".

In order to define the notion of "proximity" we assume that the data from our application domain has a topology (neighborhood system) imposed over it. A topological space is a space where for each element in the space a set of neighborhoods has been defined. The neighbors in this topological space will be considered proximal entities. For example, in the case of sequence data we can define as neighbors of an entity the entities located to its left and to its right in the sequence. In the case of images we can have q neighborhood given by the four / eight adjacent pixels, or the pixels within a radius of k pixels of the pixel under question. In general, for arbitrary discrete topologies (i.e., graphs - these topologies can occur in relational learning settings or network mining for example) we can define as proximal entities, the entities that are reachable from the entity in question by traversing no more than k edges in the graph (sphere of radius k). Note that the notion of "proximity" will depend on the size of the neighborhood

system that we take into consideration. In the more traditional setup of Multivariate Statistics case all attributes of an instance are proximal to each other and we have consequently a complete graph topology within one instance.

On way to proceed to define "similarity" on the other hand, is to use a distance measure to specify how close two entities are. The distance will be calculated as a function of some features of the two entities between which the distance is computed. Next we will examine some intuitions about what we should consider when we attempt to define similarity.

**Structuralism vs. Functionalism** Based on the choice of features in terms of which we define our distance measure, we can identify at least two extreme approaches: Structuralism and Functionalism.

To recapitulate: we are trying to define a distance that reflects similarity between two entities. This naturally leads us to examine the two major philosophies for defining semantics: Structuralism and Functionalism. Assessing the similarity in meaning between two objects is dependent on how we define what we mean by those two objects. Structuralism asserts that what an object is and therefore what it means can be defined in terms of, its structural features. For example, in order to define a chair we say that it has four legs, a rectangular surface on top of them and a back support (note the analogy with class definitions in Object Oriented Programming [5]). Functionalism on the other hand asserts that the meaning of an object resides with its usage. That is, a chair is something that you can sit on, regardless of its form or shape. And if you intentionally kill somebody with that chair, then the chair is for all practical purposes a weapon. (The equivalent of functionalism in Object Oriented Programming is represented by the notion of Interface [5] which basically specifies a set of properties necessary for the object to satisfy in order for it to be able to be fit within certain contexts. For example, a Stack needs to have one way of implementing the isEmpty/Push/Pop/Clear functions in order to be used as a Stack.) Briefly stated the functionalist claim is that meaning is about ends and not about the means by which these ends are met. While Functionalism is presumably more appealing, it raises the question of how to operationalize the intuition behind it, in order to define similarity in meaning. That is, how to operationalize the intuition that two objects

**Algorithm 1** Ontology Elicitation Procedure

**until** a terminal criteria has been reached

1. Pick one of the following two operations
   (a) abstractions = *Abstract*(data);
   (b) super-structures = *SuperStructure*(data)
2. data = *Annotate*(data, [with the] abstractions / [or] super-structures)

**repeat**

are similar in meaning if they are used in the similar ways. For cases where the entities under question occur within certain contexts one way to define similarity in meaning is to say that two entities are similar if they occur in similar contexts. Note that given a topological space and hence a notion of proximity (given by neighborhood-ness) the notion of context can easily be defined in terms of proximal entities. We will return to the problem of defining similarity in meaning in the next subsection, where we will make it more precise in the case of sequences (such as for words in text and amino-acids in proteins). So far, we have established that if the data is embedded in a topological space we can define similarity between two entities by examining the similarity of proximal elements.

One final remark about Functionalism vs. Structuralism is that these two are extreme positions and that a more convenient middle ground may be found between the two by using both structural and contextual features in order to define meaning and similarity in meaning. In the next subsection we will describe a quite general method for operationalizing Ontology Elicitation based on Abstraction and SuperStructuring and illustrate it on an example.

### 1.1.2 The general method for Ontology Elicitation

A fairly generic ontology elicitation procedure base on the operations of Abstraction and SuperStructuring is summarized by the Algorithm 1.

In Algorithm 1 *Abstract* is a procedure that returns the set of the $k - ABS$ top ranked abstractions as defined by the distance that measures similarity. The more similar the entities grouped together by an abstraction, the higher this abstraction is ranked. *SuperStructure* is a

procedure that returns the set of the $k - SS$ top ranked SuperStructures formed out of entities that are close to each other according to the given notion of proximity. The structures can be ranked according to the likelihood of the composing entities occurring together to form an unit significantly above chance level. This can be easily modeled by an independence test: the more statistically dependent the components of the structure appear to be, the more likely it is that this is a true structure and not an artifact composed out of a set of entities occurring within proximity of each other by mere chance.

Note that the two operations SuperStructuring and Abstraction have contrary effects. Namely, Abstraction increases compression but decreases modeling accuracy while SuperStructuring increases modeling accuracy while increasing model size because we are considering an increasing set .

**Operationalization of the previous intuition** In order to operationalize the general method presented in the previous in 1 we need to make more precise the notions of similarity, proximity, context and true structure vs. artifact. This will enable us to specify more precisely the Abstract and SuperStructure procedures.

In the case of sequence data, given a set of sequences (e.g., text sentences or protein sequences) one way to operationalize the above paradigm is as follows:

1. proximity - occurring on adjacent positions in the sequence or being no farther than k positions;

2. context - the entities occurring one position to the left or one position to the right, or within k positions to the left or right respectively;

3. true structure vs. artifact - can be defined using an independence test that will detailed later (the more dependent a set of entities are the more likely they are to be a true structure versus an artifact).

4. similarity (in meaning between two entities (e.g., words)) - two entities (words) are similar if they are used similarly, that is, if they are used in similar contexts. This kind of idea

can be traced down to the British Empiricists [10] - but most notably Jeremy Bentham [1] and even Aristotle - and more recently to Gottlob Frege and Betrand Russell and also Zelig Harris (1954) in the philosophy of linguistics, where it resides under the name of the *distributional hypothesis* [6]. W.V.O. Quine singles it out as one of the "five milestones of empiricism:" - "the view of sentences as primary in semantics, and of names or other words as dependent on sentences for their meaning, is a fruitful idea that began perhaps with Jeremy Bentham's theory of fictions." [11]. We will make this notion of similarity between entities based on their contexts even more precise in the definition of Abstraction outlined below.

Given the previous definitions one way we can proceed to ope rationalize Abstraction and SuperStructuring is as follows:

**SuperStructure**($S \rightarrow AB$) - returns the top $k-SS$ structures made out of two components that co-occur within proximity of each other and are unlikely to occur together by chance. One way to measure this is by an independence test - the more statistically dependent two entities are, the less likely it is that the current structure is an artifact. Ranking the sequences according to this dependence likelihood allows us to order the putative structures in order to be able to pick the most promising. We can perform the independence test by measuring the Kullback-Leibler divergence [3] between the probability of two components occurring together and the probability of them occurring together as independent events (i.e., the product of the marginal probability distributions) - $KL(P(AB)||P(A)P(B))$.

Another possibility is to rank structures according to how frequently they occur or equivalently, how much they would compress the data if the structure name would be used instead of the whole structure, thus leading to an MDL type of approach. Note: Chi-squared can also be used as a test for independence, but the Chi-squared test is only an approximation of the KL divergence [8].

**Abstraction**($S \rightarrow A|B$) - returns the top $k - SS$ abstractions (clusters) of two entities, ranked according to the distance between the contexts in which the two abstracted entities appear. More exactly, we can define the distance between the left contexts of the two entities,

**Algorithm 2** Ontology Elicitation example

Step1 data:

Mary loves John.

Sue loves Curt.

Mary hates Curt.

Abstractions 1:

$A1 \rightarrow Mary|Sue$ - because they have similar right contexts: loves.

$A2 \rightarrow John|Curt$ - because they have similar left contexts: loves.

Step 2 data:

[Mary, $A1$] loves [John, $A2$].

[Sue, $A1$] loves [Curt, $A2$].

[Mary, $A1$] hates [Curt, $A2$].

Abstractions 2:

$A3 \rightarrow loves|hates$ - because of high similarity between their left and right contexts:

This illustrates how abstraction begets more Abstraction ($A3$ not possible on the raw data).

Step 3 data:

[Mary, $A1$] [loves, $A3$] [John, $A2$].

[Sue, $A1$] [loves, $A3$] [Curt, $A2$].

[Mary, $A1$] [hates, $A3$][Curt, $A2$].

Structures 3:

$S1 \rightarrow A1A3$ because it occurs three times

$S2 \rightarrow A3A2$ because it occurs three times

This illustrates how Abstraction begets more SuperStructuring ($S1$ and $S2$ are not possible on the raw data)

Structures 4:

$S3 \rightarrow S1A2$

$S4 \rightarrow A1S2$

This illustrates how SuperStructuring begets more SuperStructuring

as the Jensen-Shannon distance [9] between the probability distribution of entities that occur to the left each entity. Similarly we define a distance for the right contexts. Subsequently, in order to obtain a distance between two entities we can sum up the distances corresponding to the left and right contexts, respectively. We can even consider two types of abstractions, one for the right context and one for the left one.

**Example** In Algorithm 2 we present a run of our algorithm on a text toy data set in order to illustrate some of the intuitions and to examine the potential power of this procedure run multiple times:

Note that the abstractions and super-structures derived at one step beget more abstractions

**Algorithm 3** Generic Ontology Elicitation

repeat

1. *Propose* **New Constructs** - Using the **Data**
2. *Evaluate* the desirability of the proposed **New Constructs** according to a **Scoring Function**
3. *Select* **Posited Constructs** according to the ranking of the **New Constructs** by the **Scoring Function**
4. *Annotate* **Data** with **Posited Constructs**

until a termination criterion reached

and super-structuring at subsequent steps, which cannot be derived directly, based on the initial raw data. Note also that we can allow multiple ontologies / points of view by allowing one entity to be abstracted or super-structured in more than one way (e.g, $A3$ in $S1$ and $S2$) . The multiple ontologies / points of view are a very common thing in real life, e.g., somebody can be both a father and a researcher.

So far, we have showed one way to operationalize the ideas presented in the main paradigm. In what follows we will examine the Ontology Elicitation process from higher ground and also from the perspective of its usage in solving a task within the given domain under study. The added value obtained by using various ontological constructs in solving the task, such as the Abstraction and SuperStructuring discussed above, can also be turned into a feedback signal that can influence the the Ontology Elicitation process itself.

### 1.1.3 Ontology Elicitation and Usage overview

As we have seen before, we need to use concepts in order to define and solve tasks in different application domains. Oftentimes, however the input data is not described in terms of the most appropriate set of concepts that are needed in order to formulate and solve a problem. As a consequence there is a need for methods that can automatically elicit these concepts - i.e., Automatic Ontology Elicitation.

An overall strategy for approaching the Automatic Ontology Elicitation problem is described in Algorithm 3.

In order to operationalize this algorithm we need to solve the Proposition, Evaluation, Selection and Annotation problems. While various steps may turn out to be quite problem

dependent, one important problem that we need to address, in general, is to provide a scoring function that will allow us to execute the step 2 - *Evaluate* and step 3 - *Select*. This scoring function will attempt to assess the usefulness of the Posited Constructs for solving the task we have in hand. We will distinguish these utility assessments setups along the following two dimensions:

1. *model-free* vs. *model-based*
2. *conditional* vs. *unconditional*

We examine these distinctions in what follows.

**Model free vs. Model based scoring** The scoring functions that can be used to assess the desirability of the New Constructs can be divided on one axis into two categories: ***model-based*** and ***model-free***.

The ***model-free*** scoring functions attempts an a priori evaluation of the quality of the posited new constructs without respect to any class of models. An example of such an evaluation strategy is the information gain or Kullback-Leibler divergence outlined above. The constructed ontology is then used for modeling but its creation is total independent of its usage.

The ***model-based*** scoring functions evaluate the quality of the posited New Constructs with respect to a given model $m \in M$. Thus, given a scoring function $score : M \rightarrow \mathbb{R}$ for evaluating the quality of models in the class of models $M$, we will compute the quality of the a newly posited construct $c$ generally as: $quality(c) = score(m \cup \{c\}, Data) - score(m, Data)$. That is, the quality of a construct $c$ with respect to a model $m$ and a data set $Data$ expresses basically how much more useful the construct $c$ is aside from the already existing model $m$ in modeling the $Data$. In this way the very usefulness of a construct for the modeling task at hand is used as a driving force for the Ontology Elicitation process.

While ***model-based*** setups seem more desirable than the ***model-free*** ones, they also come at additional costs as in a naive approach for every posited construct the corresponding augmented model needs to be evaluated against the $Data$. A distinction that is similar in spirit, yet more particular, is used in the feature selection literature [14] where the ***model-free*** setup

is called the *filter*-based approach and the ***model-based*** setup is called the *wrapper*-based approach, respectively.

**Conditional vs. Unconditional** The value of ontological constructs is tightly coupled with the way the constructs are used. In humans the process of forming such constructs is triggered by certain patterns of usage. Based on the type of usage we distinguish two types of scoring functions and consequently, two types of ontology elicitation procedure:

In the ***conditional***, a.k.a. predictive case, we evaluate the quality of a posited construct with respect to how predictive it is of a certain goal. A typical model-free score function for the class conditional case is the Information Gain about the class conditioned on the appearance of the construct vs. not appearance. In the model-based case, a typical scoring function would be the difference between the class conditional likelihood / MDL of the data given the model with and without construct. Note that the real difference between the model-free and the model-based is the fact that the information gain in the model based is conditioned also on the model.

In the ***unconditional*** case, on the other hand, the posited construct is evaluated with respect to how well it contributes to modeling the input data. Thus, for example, in the model based case the scoring function is the difference between the likelihood / MDL score of data given the model with and without the construct. Contrast this with the conditional case, where we had the *conditional* likelihood / MDL score. In the model-free case, some examples of scores are Information Gain or the Kullback-Leibler divergence discussed above. For example, the score for SuperStructuring two adjacent entities can be the information gained by modeling the joint probability of A and B versus approximating it with the product of the their marginals - $KL(P(AB)||P(A)P(B))$. Another possible score for SuperStructuring is how much compression can be achieved by using one symbol instead of using the adjacent A and B. In the case of abstraction of two entities A and B the score can be conceived in terms of the information lost by substituting A and B with one symbol versus the compression that can be gained this way. Note that minimizing the information lost is equivalent to maximizing the negative information gained by not abstracting vs. abstracting, hence this score is also from

the information gain family.

An alternative way to name the ***conditional*** vs. ***unconditional*** distinction is *goal-driven* vs. *goal-free*. The *goal-free* / ***unconditional*** setup denotes a purely informational approach to the modeling problem where we are only interested in improving our knowledge, i.e., knowledge acquisition is treated as a goal *per se*. In the *goal-driven* / ***conditional*** setup however the knowledge and information that we acquire about the world is not valued but in as much as it allows us to better achieve our goal. Note that while the ***conditional*** setup seems more desirable because of being quite focused this may also prove a liability as the usefulness of the derived constructs may not necessarily transfer well across goals / tasks, but even to new setups of the same task.

Because of the various trade-offs among the above-mentioned dimensions it is hard to make a case for one of these extremes, and in all likelihood the most desirable results will be obtained by a nontrivial combination in the spanned space. In this thesis however we will mostly examine the conditional model-free setup as it is the fastest and easiest to evaluate.

Given this brief overview of the main ideas, intuitions and motivations that we believe to stand behind the Ontology Elicitation process we are ready to describe the results presented in the thesis.

## 1.2 Thesis Organization

The results presented in the thesis can be divided into two sets: Results that have been obtained prior to the full development of the theoretical part of the *Abstraction + SuperStructuring thesis*; and results obtained afterward.

The results obtained prior to the proof of *Abstraction + SuperStructuring thesis* are exposed in chapters 2, 3 and 4. These prior exploratory results along with earlier less comprehensive theoretical results have at least partially enabled and then further reinforced the belief that the *Abstraction + SuperStructuring thesis* in its full generality may have a chance to be proved at a point in time when this was not a given. They contain in chronological order:

1. an exploration of SuperStructuring in the temporal domain - Temporal Boolean Networks

(Chapter 2) - Learning temporal SuperStructures by the means of decision trees in the context of predicting the behavior of Genetic Regulatory Networks.

2. an exploration of SuperStructuring in the spatial domain (sequences) - Naive Bayes k - NB(k) (Chapter 3) - Various extensions of Naive Bayes, which treats input features independently, to the the case of $k$-th order dependencies in the context of function prediction for protein sequences.

3. an exploration of Abstraction learning in the Multivariate case - Learning Attribute Value Taxonomies (Chapter 4) - Learning Taxonomies (a particular way of organizing Abstractions) in the Multivariate case.

In Chapter 5 - Abstraction SuperStructuring Normal Forms - we present the theoretical argument for the of the *Abstraction + SuperStructuring thesis*. We show that Abstraction, SuperStructuring and their duals: Reverse Abstraction and Reverse SuperStructuring respectively are enough in order to produce the Turing equivalent class of General Generative Grammars. Under the Computationalistic Assumption (a.k.a. Church-Turing thesis) this means that every theory can be expressed in terms of Abstraction, SuperStructuring and their duals: Reverse Abstraction and Reverse SuperStructuring. We also show that this result can be seen as proving (under the Computationalistic Assumption) a more than 200 years claim of the philosopher David Hume regarding the "principles of connexion among ideas" [7].

In Chapter 6 - Combining Abstraction and SuperStructuring - appropriate extensions of the earlier results from chapters 3 and 4 allow us to present a baseline solution for acquiring Abstraction and SuperStructuring in combination. The experimental results show that using the two principles of Abstraction and SuperStructuring, either in isolation - Chapters 2,3 and 4, but furthermore even better in combination - Chapter 6, can produce significant improvements in the comprehensibility (as measured by size) and / or accuracy of the models produced.

In Chapter 7 - Independence and Decomposability - we present a theoretical analysis of the notion of (Variable) Independence and how such a structural analysis may impact various Numerical components, such as value functions, probabilities, relations, etc. Independence is

basically the complement of SuperStructuring - that is, if two "proximal" entities are independent then there is little need to consider their union as bigger structural unit - hence no SuperStructuring. The aforementioned analysis was triggered by the need to better understand the finer structure of SuperStructuring as simple minded solutions were revealed to be somewhat problematic in the previous explorations done in Naive Bayes k - NB(k) (Chapter 3). The naive approaches were suffering from a "double counting" problem, or more precisely "double multiplying" since they are probabilities. In Chapter 7 we explore this problem in the unified framework provided by Abelian groups which allows us to treat uniformly the additive, multiplicative and relational setups (hence "double counting" in an additive setup becomes equivalent to "double multiplying" in a multiplicative setup). Our main result shows that we can "boil down" a set of pairwise Conditional Independences (and consequently pairwise factorizations) to a "global" factorization of a function $f$ over the maximal cliques of the associated independence graph. This theorem is the natural generalization of the Hammersley-Clifford theorem [2] which holds for probability distributions, to the more general case of functions that take values into an Abelian Group. The theory developed in this chapter subsumes: factorization of probability distributions, additive decomposable functions and decomposable relations, as particular cases of functions over Abelian Groups. The main enabler for the proof is not surprisingly the Moebius Inversion lemma [12], which is the main combinatorial theorem that addresses the "double counting" problem.[1]

In Chapter 8 we conclude, present the main contributions of the thesis and also examine some possible future work.

The chapters are self contained and are organized in a paper format. Chapters 2, 3, 4 and 7 have already been published in peer-reviewed journals or conferences and Chapters 5 and 6 are to be submitted.

## Bibliography

[1] J. Bentham. *Bentham's Theory of Fictions*, (ed. C.K. Ogden), 1932.

[1] The more widely known principle of inclusion and exclusion which is also known to deal with the "double counting" problem is just a particular case of the Moebius Inversion lemma.

[2] J. Besag. Spatial Interaction and Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B (Methodological),* 36(2):192-236, 1974.

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.

[4] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed., Addison-Wesley, 2004.

[5] J. Gosling, B. Joy, G. Steele, and T. Bracha. *The Java Language Specification*. Sun Microsystems, 2000.

[6] Harris, Z. Distributional structure. *Word*, 10(23):146-162, 1954.

[7] D. Hume. *An Enquiry Concerning Human Understanding*. Hackett Publ Co. 1993.

[8] S. Kullback. *Information theory and statistics*. John Wiley and Sons, NY, 1959.

[9] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. on Information Theory*, 37(1):145-151, January 1991.

[10] J. Locke. *An Essay Concerning Human Understanding.* Nidditch, Peter H. ed., Oxford: Clarendon Press, 1975.

[11] W. V. O. Quine. Things and their Place in Theories. *Theories and Things*. Cambridge, Mass., & London: Belknap, 1981.

[12] G.-C. Rota. On the Foundations of Combinatorial Theory I: Theory of Möbius Functions, *Z. Warscheinlichkeits-theorie und Verw. Gebiete*, 2:340–368, 1964.

[13] A. Silvescu, and V. Honavar. Ontology Elicitation: Structural Abstraction = Structuring + Abstraction + Multiple Ontologies. *Learning Workshop*, Snowbird, Utah, 2003.

[14] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In H. Liu and H. Motoda, editors, *Feature extraction, construction and selection*, pages 118–135. Kluwer Academic Publisher, 1998.

# CHAPTER 2. TEMPORAL BOOLEAN NETWORK MODELS OF GENETIC NETWORKS AND THEIR INFERENCE FROM GENE EXPRESSION TIME SERIES

Modified from a paper published in *Complex Systems*[1]

Adrian Silvescu[2] and Vasant Honavar

## Abstract

Identification of genetic regulatory networks and genetic signal transduction pathways from gene expression data is one of key problems in computational molecular biology. Boolean networks offer a *discrete time Boolean* model of gene expression. In this model, each gene can be in one of two states (on or off) at any given time, and the expression of a given gene at time $t+1$ can be modeled by a *Boolean* function of the expression of at most $k$ genes at time $t$. Typically $k << n$, where $n$ is total number of genes under consideration. This paper motivates and introduces a generalization of the Boolean network model to address dependencies among activity of genes that span for more than one unit of time. The resulting model, called the *Temporal Boolean Network* or the $TBN(n,k,T)$ model, allows the expression of each gene to be controlled by a Boolean function of the expression levels of at most $k$ genes at times in $\{t \cdots t-(T-1)\}$. We apply an adaptation of a popular machine learning algorithm for decision tree induction for inference of a $TBN(n,k,T)$ network from artificially generated gene

[1]Reprinted (no permission needed as the copyrights were not transfered) from *Complex Systems*, 2001, 13(1):54-75.

[2]Primary researcher and author - designed and implemented the methods, made the theoretical arguments and wrote the paper.

expression data. Preliminary experiments with synthetic gene expression data generated from known $TBN(n, k, T)$ networks demonstrate the feasibility of this approach. We conclude with a discussion of some of the limitations of the proposed approach and some directions for further research.

## 2.1 Introduction

The central dogma of modern biology states that the functional state of an organism is determined largely by the pattern of expression of it's genes. Thus, the function of a cell, how well a cell performs its function, and even the determination of a cell type is controlled by the level at which genes are expressed in the cell. Many biological processes of interest (e.g., cellular differentiation during development, aging, disease) are controlled by complex interactions over time between hundreds of genes. Furthermore, each gene is involved in multiple functions. Therefore, understanding the nature of complex biological processes such as development, cellular differentiation, carcinogenesis, etc., requires determining the spatio-temporal expression patterns of thousands of genes, and, more importantly, seeking out the organizing principles that allow biological processes to function in a coherent manner under different environmental conditions. Given the fact that thousands of genes are involved in determining the functional state of an organism, the task of assigning functions to genes, identifying underlying genetic signalling pathways, genetic control and regulatory networks is a formidable task.

A number of emerging high-throughput technologies are revolutionizing the means by which genetic signalling pathways involving hundreds of genes can be studied. Many of these technologies exploit the power of multiplexed data acquisition from addressable solid-state arrays of biomolecules (BioArrays). Such technologies allow researchers to obtain, in a single experiment, significant amounts of biological information regarding thousands of genes. In order to determine gene expression levels, messenger RNA samples are collected from a population of cells under a given set of experimental conditions or at different times during the execution of a biological pathway or process (e.g., glycolysis [6], cell cycle [26], and development [30]). Using DNA microarrays, the levels of mRNA expression are measured. Similar methods for

determining protein concentrations [8] exist but have a lower throughput.

These advances in data aquisiton make possible, at least in principle, the inference of models of genetic (regulatory and control) networks from gene expression data. However, the large number of genes involved, complexity of the pathways, and existence of pleiotropic and multigenic interactions [24] make this a challenging task. Some issues that arise in genetic network inference from gene expression data are discussed by Thieffry *et al.* [28] and Savageau [23]. Less sophisticated methods such as clustering [7, 3, 30] can also be used in order to analyze gene expression data but the information that can be obtained by these methods are mainly restricted to either positive or negative correlations among gene expression patterns.

A variety of formal models for capturing the interactions and functional dependencies in genetic networks have been proposed in the literature. These include: *electrical circuits* [17], *Boolean networks* [12, 24, 14, 1, 10], *Fourier coefficients* [11], *Bayesian Networks* [19], *differential equations* [4], *Petri nets* [9, 16], and *Weight matrices* [29]. Each of these approaches has its own strengths and limitations in terms of the following considerations: faithfulness or accuracy of the model relative to the biological phenomenon being modeled; transparency of the model (or equivalently, its explanatory value); experimental feasibility (in terms of data requirements) of model construction, and computational tractability of automated model inference from data.

The focus of this paper is on Boolean Network models of genetic networks wherein each gene can be in one of two states – ON (expressed) or OFF (not expressed). One of their main advantages is their comprehensibility due to the transparency of the representation. It is possible to extend such network models in order to allow for a discrete set of states for each gene, instead of just two states, or even to allow states that take on real (and hence an infinite) set of values. Previous work on inference of Boolean models of genetic networks [1, 14] has focused on data that randomly sample the state transitions typically under the assumption that the state of a gene at a given time step is influenced by the states of a subset of genes in the network at the previous time step. Since many experiments involve obtaining gene expression data by monitoring the expression of genes involved in some biological process (e.g., cell or neural development) over a period of time, the resulting data is in the form of a *time series*.

In such a setting, it is of interest to understand how the expression of a gene at some stage in the process is influenced by the expression levels of other genes during the stages of the process preceding it.

In order to model the temporal dependencies that span several time steps, we introduce in this paper, a generalization of Boolean Networks called Temporal Boolean Networks. This will basically transform the Boolean Networks from a Markov(1) to Markov($T$) model where $T$ is the length of the time window during which a gene can influence another gene. We will demonstrate how Temporal Boolean Networks can be efficiently inferred using an adaptation of a greedy decision tree learning algorithm [21]. The main contribution of this paper is in terms of computational techniques for genetic network inference from gene expression time series. The insights into the nature of functionally significant interactions among genes provided by the inferred genetic networks might also suggest novel ways to exploit artificial genetic networks to solve specific computational problems e.g., in evolutionary algorithms.

The rest of the paper is organized as follows: Section 2 provides a brief overview of gene expression data analysis. Section 3 motivates and introduces the Temporal Boolean model for genetic networks. In Section 4 we present some theoretical results concerning data-driven inference of Temporal Boolean Networks from randomly sampled transitions. Section 5 describes our approach to inferring a Temporal Boolean Network from time series data. Section 6 presents experimental results that demonstrate the feasibility of the proposed approach. Section 7 concludes with a summary and discussion of some directions for further research.

## 2.2 Gene Expression Data Analysis

The widespread use of DNA microarray and related technologies have led to increased availability of gene expression data from plants and animals. Consequently, there is a growing need for sophisticated computational tools for extracting biologically significant information from gene expression data, assigning functions to genes, and identifying shared genetic signaling pathways and genetic regulatory networks. The data from a series of $m$ microarray measurements involving $n$ genes can be represented as an $m \times n$ gene expression table $\mathcal{E}$ where each of

the $n$ columns consist of $m$ entries (numbers that correspond to the measured expression levels of a single gene across $m$ measurements). Thus, the entry $e_{ti}$ in row $t$ and column $i$ of the matrix $\mathcal{E}$ denotes the expression level of gene $i$ in the $t$th measurement. The rows can represent expression values measured under different experimental conditions, or data obtained by monitoring the expression levels of genes at different times during a biological process (e.g., neural development). Given this data, a number of different types of analysis are possible [25, 27]. One of the simplest forms of analysis involves clustering of gene expression patterns (columns of the table) based on some predefined clustering criterion or distance measure, but no knowledge of the functional classes of the genes. If a subset of expression patterns form a tight cluster, it can be hypothesized that the genes in question are co-expressed, or even possibly co-regulated.

## 2.3 Temporal Boolean Networks

In order to model the dependence among the expression levels of the genes we will use a generalized version of Boolean Networks. The main advantage of these models is their high level of transparency. In this section we will examine the Boolean Networks model followed by the discussion of some issues that may arise with respect to this model. The need to address these issues provides the motivation for generalizing Boolean Networks to the Temporal Boolean Networks model, which will be defined at the end of this section.

### 2.3.1 Boolean Networks

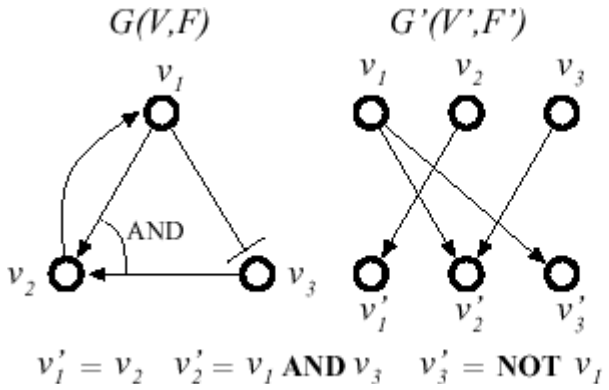
Boolean networks, introduced by Kaufmann [12] and explored in [24, 1, 14, 10] offer an attractive discrete time, boolean model for gene expression. In this model, each gene can be in one of two states either ON or OFF at any given time. The expression of a given gene at time $t+1$ is modeled by a *Boolean* function whose inputs are the expression levels of at most $k$ genes at time $t$. Typically $k << n$, where $n$ is total number of genes under consideration. We call this family of models $BN(n,k)$ networks. Some of the attractive features of this model include its conceptual simplicity and transparency as well as the availability of algorithms for automated inference of the model from expression data.

Following Akutsu *et al.* [1], we are going to give more precise definitions for the boolean networks. A Boolean Network is specified by pair $G(V, F)$, where the $V = \{v_1, ..., v_n\}$ is a set of nodes representing the genes of the network (one node for each gene) and a list $F = (f_1, ..., f_n)$ of Boolean functions. Each node $v_i$ has an associated expression value that is either 1 (for expressed) or 0 (for not expressed) that will be denoted also by $v_i$. A boolean function $f_i(v_{i_1}, ..., v_{i_k}) \in F$ with inputs from the specified nodes $v_{i_1}, ..., v_{i_k}$ is assigned to each node $v_i$. The nodes $v_{i_1}, ..., v_{i_k}$ correspond to the genes that influence the expression of the gene associated with node $v_i$, and $f_i$ represent the exact functional dependence of this influence. For example in Figure 2.1 the gene $v_2$ is influenced by genes $v_1$ and $v_3$ and the functional dependence is $v_2 = v_1\ AND\ v_3$. This functional dependence basically says that $v_2$ is expresses if and only if both $v_1$ and $v_3$ are expressed.

For a subset $U \subseteq V$, an *expression pattern* $\psi$ of $U$ is a function from $U$ to $\{0, 1\}$. An expression pattern of $V$ is called a *state* of a boolean Network. That is $\psi$ represents the states of nodes (genes), where each node is assumed to take either 0 (not-expressed) or 1 (expressed) as its state value. Typically, when the usage is clear from the context, we omit $\psi$. For example, we write $v_i = 1$ for denoting $\psi(v_i) = 1$. In a Boolean network, the expression pattern $\psi_{t+1}$ at time $t + 1$ is determined by the Boolean functions $F$ from the expression pattern $\psi_t$ at time $t$ (i.e. $\psi_{t+1}(v_i) = f_i(v_{i_1}, ..., v_{i_k})$).

For better visualization, it is convenient to consider the wiring diagram [24, 14] $G'(V', F')$ of a Boolean network $G(V, F)$ [See Figure 2.1]. For each node $v_i$ in V, let $v_{i_1}, ..., v_{i_k}$ be input nodes to $v_i$ in $G(V, F)$. For each such node $v_i$, we consider an additional node $v'_i$, and we construct an edge from $v_{i_j}$ to $v'_i$ for each $1 \leq j \leq k$. Let $G' = (V', F')$ the network with nodes $v_1, ..., v_n, v'_1, ..., v'_n$ constructed in this way. Then the expression pattern of the set $\{v'_1, ..., v'_n\}$ is determined by $v'_i = f_i(v_{i_1}, ..., v_{i_k})$. That is, the expression of pattern $\{v_1, ..., v_n\}$ corresponds to one at a time $t$ and the expression pattern of $\{v'_1, ..., v'_n\}$ corresponds to one at time $t + 1$. Moreover, it is convenient to consider the expression pattern of $\{v_1, ..., v_n\}$ as the INPUT, and the expression pattern of $\{v'_1, ..., v'_n\}$ as the OUTPUT.

The wiring diagram basically shows how the expression levels of genes at time $t$ (INPUT)




| INPUT | | | OUTPUT | | |
|---|---|---|---|---|---|
| $v_1$ | $v_2$ | $v_3$ | $v_1'$ | $v_2'$ | $v_3'$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

Figure 2.1 A Boolean Network - from left to right: the network, its wiring diagram and the functional dependency table. (Akutsu *et al.* 1999 [1])

influence the the expression levels at time $t+1$ (OUTPUT). The exact trasition diagram for every combination of INPUT values can thus be given by a table as the one in Figure 2.1.

### 2.3.2 Motivations for Generalizing the Boolean Network Model

Our proposed generalization of the Boolean network model to a Temporal Boolean Network model is motivated by the following considerations:

- Boolean networks described above are incapable of modeling the existence of latency periods (lasting more than one unit of time) between the expression of a gene and the observation of its effect. For example a gene (say $g_4$) whose inhibitory effect (say on gene $g_5$) depends on an inducer (say $g_3$) first has to bind with this inducer in order to be able to bind to the inhibition site on $g_5$. Therefore there can be a significant delay between the expression of the inhibitor gene $g_4$ and its observed effect i.e. the inhibition of the gene $g_5$.

- Not all variables that can influence the expression level of a gene are necessarily observable. For instance, assume that genes $g_1, g_2, \cdots g_{1000}$ be the genes under study. Suppose that the expression of genes $g_1$ at time $t$ might turn on a gene $g_u$ at time $t+1$. It is quite possible that $g_u$ is not among the genes that are being monitored in the experiment, or even among the genes that are currently known. Suppose gene $g_u$ being on at time $t+1$

results in a gene $g_2$ being turned on at time $t+2$. Since the expression of $g_u$ cannot be observed, the Boolean network model described above will be unable to implicate $g_1$ in the control of $g_2$.

Note that even if all the genes are monitored in an experiment the unknown factor denoted by $g_u$ may stand for a non-genetic environmental factor. Also long temporal delays observed in the our new model might indicate the presence of hidden factors, thus providing hints for their discovery and therefore contributing to the improvement of the quality of the data collected in future experiments.

In what follows, we introduce a generalization of the Boolean network model to address dependencies among the activity of genes that span for more than one unit of time. The resulting model, we will call the $TBN(n,k,T)$ model, allows the expression of each gene at time $t+1$ to be controlled by a Boolean function of the expression levels of at most $k$ genes at times in $\{t \cdots t-(T-1)\}$.

### 2.3.3 Temporal Boolean Networks

In the Temporal Boolean Networks (TBN) we will allow the state of a gene at time $t+1$ to depend on the state of other genes at times $t, t-1, ..., t-(T-1)$, instead of only $t$. The representation of such a network, by analogy with the boolean networks is given in Figure 2.2.

The only change from the Boolean Network model depicted in Figure 2.1 is that the expression level of gene $v_2$ at time $t+1$ depends on the value of the gene $v_1$ at time $t-1$ instead of time $t$. This is represented by a label of $-1$ label on the edge between $v_1$ and $v_2$. By default the edges are assumed to carry a value of 0, which means that the dependency they represent is from time $t$ to time $t+1$. In general an edge labeled as $-k$ will represent a dependency between the values at time $t-k$ and $t+1$. The wiring diagram is changed accordingly in order to be able to represent dependencies that represent the added functional $t-1$ dependencies. In general, each edge in the network can have a set of labels drawn from the set $\{0 \cdots T-1\}$. The functional dependency will be represented by a boolean table for each gene, but this time the inputs can be gene expression values at more than one time step in the past.

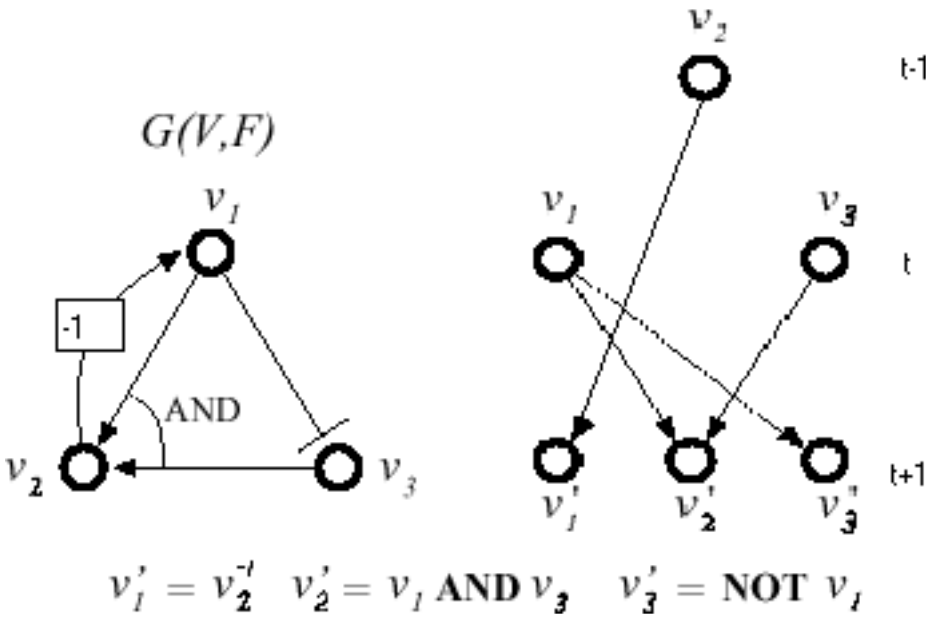



Figure 2.2 A Temporal Boolean Network

It is straightforward to extend the proposed model to allow for multiple discrete levels of expression yielding Temporal Discrete Networks $TDN(n, k, T, D)$ wherein each gene can be expressed at levels $0, 1, \cdots D-1$. Furthermore it is also possible to allow for continuous expression levels as well yielding $TCN(n, k, T, R)$ wherein the expression level of a gene $g_i$ at time $t$ is a real number in the interval $[-R, R]$.

## 2.4 Theoretical Results

Identification of a member of the $BN(n, k)$ family from noiseless as well as noisy data have been explored by Akutsu *et al.* [1, 2]. They assume that the genes in the (unknown) network to be inferred can be set to any arbitrary pattern of 0s and 1s and the resulting state at the next time step can be observed. Further, it is assumed that such "input-output" observations can be performed by uniformly randomly sampling the inputs. Under similar assumptions, we can state the corresponding results for $TBN(n, k, T)$ networks.

**Theorem 1.** *$O(2^{2k} \cdot (2k+\alpha) \cdot log\ nT)$ uniformly randomly sampled input patterns and the corresponding outputs of an (unknown) TBN $\tau \in TBN(n, k, T)$ are sufficient to guarantee exact*

*inference of $\tau$ with probability at least $1 - n^{-\alpha}$, where $\alpha > 1$ is any fixed constant.*
The proof of this theorem is a straightforward adaptation of similar results given in Akutsu *et al.* 1999 [1] in the case of Boolean networks. [See section 2.8 for additional details].

Note that although the sample and time complexities are exponential in $k$ (the degree of interaction) they are only logarithmic in $nT$ (the product of the number of genes and the maximum time span of temporal dependence $T$). Since typically $k << nT$, such exact inference is feasible for small values of $k$. However, in general, it might be necessary to sacrifice exactness of inference in exchange for computational efficiency.

It is worth noting that these theoretical guarantees (like their counterparts given by Akutsu *et al.* [1, 2]) rely on the assumption that the input patterns constitute a uniformly random sample and that the output corresponding to each such input can be observed. Since many experiments involve obtaining gene expression data by monitoring the expression of genes involved in some biological process (e.g., cell or neural development) over a period of time, the resulting data is in the form of a *time series*. Since the sequence of states in a time series are strongly temporally correlated, the assumption of uniform random sampling is no longer valid. Each such time series provides a trajectory through the state space of a genetic network. Each trajectory has a transient part which provides useful information about the underlying network and a steady state part that corresponds to an attractor of the network. Derivation of suitable bounds on the number of time series samples (as opposed to uniformly random samples) by identifying the necessary and sufficient constraints on the temporal structure of the time series data relative to the structure of the underlying network in this case is a topic of current research.

In a similar manner as in Akutsu [1] we can develop an information theoretic lower bound on the number of transitions needed to identify TBN(n,k,T) given by the following theorem:

**Theorem 2.** *At least $\Omega(2^k + k \log nT)$. INPUT/OUTPUT pairs are required in the worst case to identify a Temporal Boolean Network TBN(n,k,T).*

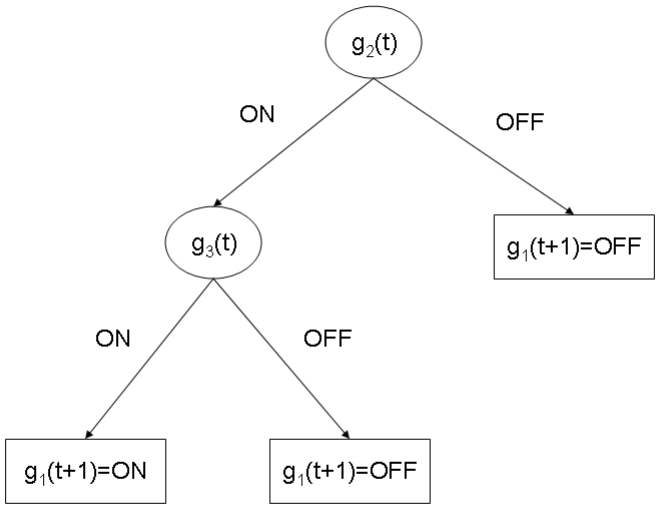



Figure 2.3 A decision tree that represents the fact that if genes $g_2$ and $g_3$ are expressed (turned ON) at time $t$ leads to the expression of gene $g_1$ (turned ON) at time $t+1$

[See section 2.8 for details]. In what follows, we describe an approach to inference of Temporal Boolean Networks from expression data in the form of a time series.

## 2.5 Inference of Temporal Boolean Networks from Time Series Data

Temporal Boolean Networks model functional dependencies among genes using Boolean functions. Boolean functions have several alternative representations. The simplest (and the most explicit) representation is a truth table shown in Fig. 1. However, functions that correspond to descriptions of natural phenomena typically lend themselves to more compact representations in the Conjunctive Normal Form (CNF), Disjunctive Normal Form (DNF), or in the form of decision trees, decision lists, etc. We have chosen to represent the Boolean functions used to describe Temporal Boolean Networks in the form of Decision Trees.

For example, consider gene $g_1$ that is induced by a complex formed from the products of

two genes $g_2$ and $g_3$ and these are the only gene that influence $g_1$. A decision tree that describes this dependence is given in Figure 2.3. Similarly, for each gene we can construct such a decision tree. The genes that control gene $g_i$'s expression appear as nodes in its corresponding decision tree.

Since every $k$-input boolean function can be represented by a decision tree of depth at most $k$, we can ensures that the resulting representation is expressive enough to describe any member of the Temporal Boolean Network family $TBN(n, k, T)$. We use $n$ decision trees, one for each gene, to collectively describe a Temporal Boolean Network model of a genetic network. The output of each decision tree represents the expression level (0 or 1) of the corresponding gene based on at most $k$ expression levels over a time window of length at most $T$.

Several algorithms for inferring decision trees from data (in the form of sample input-output pairs) are available in the machine learning literature. The ID3 [21] algorithm and its variants are based on a greedy search through the space of decision trees in order to identify a compact decision tree that adequately models the observed data and (under some reasonable assumptions) has high predictive accuracy on unobserved data. This search for a compact tree, guided by the entropy reduction (or information gain) criterion corresponds to a greedy version of the approach used in REVEAL [14]. Greedy search makes this approach computationally tractable for large genetic networks. A large body of empirical results in the machine learning literature suggest that the decision trees inferred by greedy search compare favorably with those inferred using exhaustive search in terms of predictive accuracy. Hence, we used a greedy search guided by information gain, over the space of depth $k$ decision trees.

The training data for a decision tree modeling the functional dependency of the level the expression of gene $g_i$ consists of observed input output pairs where each input is a $nT$ bit boolean vector encoding the activities of each of the $n$ genes at times $t, t-1, t-2 \cdots t-(T-1)$ and the corresponding output is the observed expression level of $g_i$ at time $t+1$. For a given gene $g_i$, $m-T$ input–output samples (or training examples) are obtained by sliding a window of length $T$ (where $T < m$) over the rows of a gene expression time series $\mathcal{E}$ (See section 2). Thus, an $m \times n$ gene expression matrix yields $m-T$ training samples for each of the $n$ decision

trees. Examples obtained from multiple time series are used for inferring a genetic network.

## 2.6 Experimental Setting and Results

The experiments described in this section were designed to explore the performance of the proposed approach to genetic network inference on randomly generated temporal boolean networks. In generating a network, we assume that the probability that the expression level of a gene at time $t+1$ depends on the expression levels of genes at time $t-\delta$ is proportional to $\zeta^{\delta}$ $\forall t$ such that $0 \leq t \leq T-1$ for some choice of $\zeta$ where $0 < \zeta \leq 1$. For each network, we generated 20 time series of length 100, by setting the expression levels over a window of length $T$ to random values and recording network's outputs over 100 time steps. Thus, each time series resulted in an $n \times (100+T)$ boolean matrix $\mathcal{E}$. The 20 time series collectively provided $100 \times 20$ training examples for each of the $n$ genes. Each time point contains the expression levels for all genes at that moment of time. A decision tree was inferred for each gene.

Then we evaluated the results in terms of the *sensitivity*, *specificity* and *accuracy* of the inferred decision tree $DT_i$ with respect to the corresponding temporal boolean network $TBN_i$ for each gene $g_i$.

We denote the fact that the decision tree $DT_i$ captures the dependence of the expression level of gene $g_i$ at time $(t+1)$ on the expression level of the gene gene $g_j$ at time $(t-\tau)$ where $\tau \in \{0, \cdots, (T-1)\}$, by writing $(\tau, j) \in DT_i$. Similarly, we denote the fact that the expression level of gene $g_i$ at time $(t+1)$ depends on the expression level of gene $g_j$ at time $(t-\tau)$ if gene $g_i$ were controlled by the temporal boolean network $TBN_i$ by writing $(\tau, j) \in TBN_i$. Let

$$DEP_{DT_i} = \{(\tau, j) | (\tau, j) \in DT_i\}$$

Let

$$DEP_{TBN_i} = \{(\tau, j) | (\tau, j) \in TBN_i\}$$

Then the *sensitivity* of the decision tree $DT_i$ for gene $g_i$ is defined as follows:

$$sensitivity(i) = \frac{|DEP_{TBN_i} \cap DEP_{DT_i}|}{|DEP_{TBN_i}|}$$

The sensitivity of the inferred decision tree measures the degree, to which this tree succeeds in capturing the dependency of gene $g_i$ on other genes, with respect to the true Temporal Boolean Network.

The *sensitivity* of the inferred set of decision trees $DT = \{DT_i | i \in \{1, \cdots, n\}\}$ relative to the corresponding TBN is given by:

$$sensitivity(DT, TBN) = \frac{1}{n} \sum_{i=1}^{n} sensitivity(i)$$

The *specificity* of the decision tree inferred for gene $g_i$ is defined as follows:

$$specificity(i) = \frac{|DEP_{TBN_i} \cap DEP_{DT_i}|}{|DEP_{DT_i}|}$$

Specificity of the inferred decision tree measures the degree to which it misleads us regarding the dependency of gene $g_i$ on the various genes in the genetic network.

The *specificity* of the inferred set of decision trees $DT = \{DT_i | i \in \{1, \cdots, n\}\}$ relative to the corresponding TBN is given by:

$$specificity(DT, TBN) = \frac{1}{n} \sum_{i=1}^{n} specificity(i)$$

The *accuracy* of the decision tree inferred for gene $g_i$ is defined to reflect the degree to which it correctly predicts the expression level of gene $g_i$ (as estimated from a set of gene expression time series data).

Let $\Lambda$ be a set of gene expression time series used to evaluate the accuracy of a decision tree inferred for gene $g_i$.

Then we will denote by $accuracy\ (i, \lambda)$ the accuracy of the tree for gene $g_i$ on a time series $\lambda \in \Lambda$. $accuracy(i, \lambda)$ represents the fraction of expression levels of gene $g_i$ from the time series $\lambda$ that are correctly predicted by the inferred decision tree $DT_i$, relative to the total size of $\lambda$. Thus, if $\lambda$ is a time series of length 100, and at 80 of the 100 time points, the expression level of gene $g_i$ predicted by $DT_i$ agrees with the corresponding values observed in $\lambda$, $accuracy(i, \lambda) = 0.8$. The *accuracy* of $DT_i$ is estimated as follows:

$$accuracy(i) = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} accuracy(i, \lambda)$$

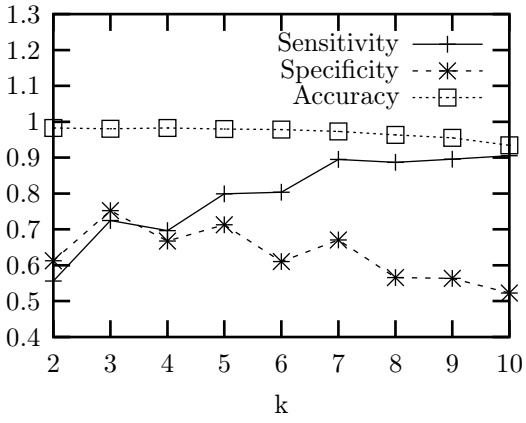



Figure 2.4 Effect of varying $k$ on inference of a boolean net with 16 genes ($n = 16$) and $T = 3$.

The accuracy of an inferred decision tree $DT_i$ was estimated using an independently generated test set of 20 time series each of length 100 generated from $TBN_i$. The estimated accuracy of the inferred set of decision trees $DT = \{DT_i | i \in \{1, \cdots, n\}\}$ relative to the corresponding TBN is given by:

$$accuracy(DT, TBN) = \frac{1}{n} \sum_{i=1}^{n} accuracy(i)$$

Each experiment consisted of 10 independent runs of this procedure (each with a new randomly generated network) and the results presented show averages over these runs.

The first experiment presented in Figure 2.4 shows the effect of varying $k$ on the sensitivity, specificity, and accuracy of inference of a boolean network with 16 genes, with $T = 3$ as $k$ is varied from 2 to 10. Similar results were obtained for networks with 32 genes. The experiment shows that the sensitivity increases as the degree of interaction $k$ increases.

The second experiment presented in Figure 2.5 explores the effectiveness of the inference algorithm when multiple levels of expression are allowed for each gene. This experiment involved inference of a 4-ary network with 16 ($n = 16$) genes, $T = 3$ as $k$ is varied from 2 to 10. This corresponds to a Temporal Discrete Network TDN ($n$=16, $k$=2-10, $T$=3, $D$=4). In this case we observe sensitivity and specificity increasing and accuracy decreasing with increase in $k$. These

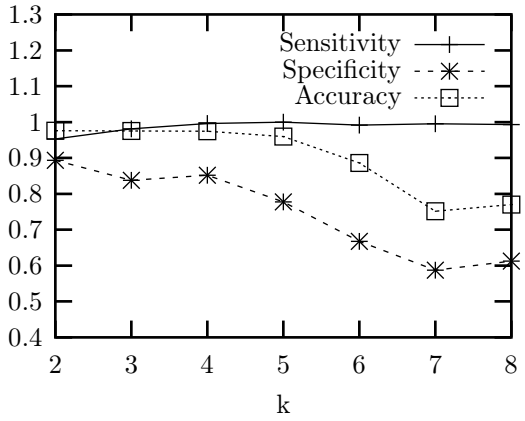



Figure 2.5 Effect of varying $k$ on inference of a 4-ary network with 16 genes ($n = 16$) and $T = 3$.

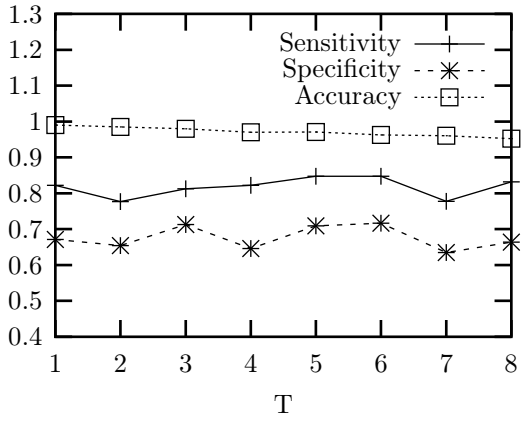



Figure 2.6 Effect of varying $T$ on inference of a boolean net with 16 genes ($n = 16$) and $T = 3$.

experiments show that as we increase the complexity of the model (in this case the number of expression levels) the sensitivity, which represents the probability that we will identify genes that represent the true dependency, increases (compare the Sensitivities in Figure 2.4 and Figure 2.5).

Further experiments revealed that the reduction in accuracy can be explained by the necessity for additional data as $k$ increases, in the conditions of a 4-ary network were for each gene we have to disambiguate among $4^{4^k}$ 4-ary functions versus $2^{2^k}$ in the 2-ary (Boolean) case (once the dependencies are known qualitatively).

The third experiment presented in Figure 2.6 explores the effect of varying $T$ on the performance of the inference algorithm. In this case, we see that for a boolean network with $n = 16$ genes, $k = 6$ the sensitivity, specificity, and accuracy vary little across different values of $T$ (from $T = 1$ to $T = 8$).

## 2.7 Summary and Discussion

This paper introduced the Temporal Boolean Networks (and Temporal Discrete Networks) which generalize the Boolean Network model in order to cope with dependencies that span over more than one unit of time. Some bounds on the size of data needed to infer Temporal Boolean Networks from time series data under uniformly random sampling assumptions are stated. We also showed how the problem of Temporal Boolean Network inference can be translated into a problem of inferring a set of decision trees. We demonstrated, through a series of experiments using artificially generated networks, the effectiveness of a simple and fast decision tree inference algorithm for inferring Temporal Boolean Networks and Temporal Discrete Networks from time series data.

The main hindrance against applying the TBN inference algorithm on real data is the lack of sufficiently large data sets. This hindrance is likely to become less serious as additional data are gathered.

The approaches to genetic network inference from gene expression data rely on the assumption that only the expression of a gene is likely to be controlled by a relatively small number

(say $k$) of genes. Biologically meaningful value of $k$ is currently unknown, but is believed to be much smaller than the total number of genes $n$ [12]. It is clear from the sample complexity results presented in this paper that purely data-driven approaches to inference of temporal boolean networks will be computationally infeasible unless $k << n$.

Work in progress is aimed at evaluating the effectiveness of the described approach for inferring genetic networks from biological gene expression time series data.

Some directions for future work include investigation of variations of the model to accommodate probabilistic Boolean and Discrete valued functional dependencies, and continuous valued expression levels as well as alternative (e.g., event-based and interval-based) representations of time. Investigation of techniques for incorporating prior knowledge (in the form of known biological constraints) into the inference algorithm represents another direction for futire research. Also of interest are active learning approaches wherein the learning algorithm helps identify promising experiments as opposed to the purely data-driven passive learning approach examined in this paper.

## 2.8 Proof of Theoretical Results

**Theorem 1.** *$O(2^{2k} \cdot (2k + \alpha) \cdot log\ nT)$ uniformly randomly sampled input patterns and the corresponding outputs of an (unknown) TBN $\tau \in TBN(n, k, T)$ are sufficient to guarantee exact inference of $\tau$ with probability at least $1 - n^{-\alpha}$, where $\alpha > 1$ is any fixed constant.*

*Proof Sketch:* The proof of this theorem is an adaptation of the corresponding theorem proved in [1] in the case of Boolean Networks. The proof in [1] is based on a brute-force algorithm for identifying a boolean function from the set of all boolean functions with less than $k$ inputs chosen from a set of $n$ possibilities (i.e., roughly $2^{2^k} \cdot n^k$ functions). A precise characterisation can be given to a minimal set of INPUT/OUTPUT pairs that will allow the algorithm to identify(in the worst case) a Boolean Network exactly. Then the bound stated by the theorem (i.e., $O(2^{2k} \cdot (2k + \alpha) \cdot log\ nT)$ ) is derived as the number of uniformly random sampled INPUT/OUTPUT pairs needed, in order to be sure that our sample will contain the minimal set with probability at least $1 - n^{-\alpha}$.

The only difference in the case of Temporal Boolean Networks is that we have to do the identification from the set of all boolean functions with less than $k$ inputs chosen from a set of $nT$ possibilities (i.e., roughly $2^{2^k} \cdot (nT)^k$ functions). Aside from this the proof follows along the same lines as the one in [1].

□

**Theorem 2.** *At least $\Omega(2^k + k \log nT)$ INPUT/OUTPUT pairs are required in the worst case to identify a Temporal Boolean Network TBN(n,k,T).*

*Proof Sketch:* The proof for this theorem follows from a standard information theoretic argument.

The general problem in the information theoretic setting is the following: "We want to identify an element $e$, from a set $S$ of finite cardinality $|S|$, by asking $D$-ary questions (i.e. that have $D$ possible answers) about where in the set $S$ the element $e$ to be found. And the problem is to find what is the minimum number of $D$-ary questions that we need in the worst case in order to identify the element $e$." The solution to this problem is as follows: "We need at least $\Omega(\log_D |S|)$ questions in order to identify the element $e$. This is because a $D$-ary question splits the set $S$ into $D$ subsets and at least one of these sets has a size of at least $|S|/D$. If we iterate this procedure for l succesive questions then we get that the size of at least one of the subsets has to be at least $|S|/D^l$. By setting this size equal to 1 (in order to obtain a 100% identification of $e$) and solving the equation we obtain a minimum of $\Omega(\log_D |S|)$ questions needed for the identification of $e$".

Returning to our problem, if we are to recast it in information theoretic terms, the element that we want to identify is a TBN from the set of all possible TBNs of the type $(n, k, T)$ (i.e., having $n$ genes, dependency of at most $k$, and having a dependency timespan of at most $T$). The questions in our case are the INPUTs (which are $n \times T$ matrices that represent the levels of expression for all $n$ genes during the previous T timesteps) and the answers to these questions are represented by the OUTPUTs (which are the levels of expression for all of the $n$ genes at the current moment of time). Therefore in our case an INPUT/OUTPUT is equivalent to an answer to a $2^n$-ary question (because there are $2^n$ possible OUTPUT patterns,

hence $2^n$ possible answers). The number of TBNs of the type $(n, k, T)$ is given by all the possible combinations of n(one for each gene) boolean functions with $k$ inputs chosen from $n \cdot T$ possibilities. Since there are $2^{2^k}$ possible Boolean functions with $k$ inputs and $\Omega((nT)^k)$ ways to chose those $k$ inputs from $n \cdot T$ possibilities, it follows that there are $\Omega((2^{2^k} \cdot (n \cdot T)^k)^n)$ possible TBNs of type (n,k,T). Now applying the information theoretic argument it follows that we need $\Omega(\log_{2^n}(2^{2^k} \cdot (n \cdot T)^k)^n)$ questions. But this is the same as $\Omega(2^k + k \log_2 nT)$ (because $\log_{a^n} b^n = \log_a b$). Which completes the proof.

□

*Note:* The previous theorem can be generalized also for the case when we have $D$-ary Temporal Networks by replacing 2 in all the places with $D$ (including the base of the logarithm).

## 2.9 PostScript

Due to to their simplicity and easiness in interpretation models of the Temporal Boolean Networks sort prove to be a useful model especially the field of Genetic Regulatory Network modeling and inference where massive amounts of data are not available in order to fit more sophisticated models. Since the publication of the original paper additional developments in this the vein of temporal modeling of Genetic Regulatory Networks and Temporal Boolean Networks have been done by other researchers [5][13][15][20].

## Bibliography

[1] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of Genetic Networks from a Small Number of Gene Expression Patterns Under the Boolean Network Model", *Pacific Symposium on Biocomputing*, 4:17-28, 1999.

[2] T. Akutsu, S. Miyano and S.Kuhara. Algorithms for Inferring Qualitative Models of Biological Networks. *Pacific Symposium on Biocomputing*, 5:290-301, 2000.

[3] A. Ben-Dor and Z. Yakhini. Clustering Gene Expression Patterns. *Journal of Computational Biology*, 6:281-297, 1999.

[4] T. Chen, H. L. He, and G.M. Church. Modeling Gene Expression with Differential Equations. *Pacific Symposium on Biocomputing*, 4:29-40, 1999.

[5] C. Cotta, and J. M. Troya. Reverse engineering of temporal Boolean networks from noisy data using evolutionary algorithms. *Neurocomputing,* 62:111-129, 2004.

[6] J. L. DeRisi, V. R. Iyer and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680-6, 1997.

[7] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science*, 95:14863-14868, 1998.

[8] S. P. Gygi, Y. Rochon, B. R. Franza, R. Aebersold. Correlation between protein and mRNA abundance in yeast. *Mollecular Biology of the Cell*, 19:1720-30, 1999.

[9] P.J.E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri Nets. *PNAS*, 95:6750-6755, 1998.

[10] T.E. Ideker, V. Thorsson, and R.M. Karp. Discovery of Regulatory Interactions Through Perturbation: Inference and Experimental Design. *Pacific Symposium on Biocomputing*, 5:302-313, 2000.

[11] H. Kargupta. A striking property of genetic code-like transformations, *Technical Report EECS-99-004*, Department of Electrical Engineering and Computer Science, Washington State University, 1999.

[12] S.A. Kauffmann. *The Origins of Order, Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[13] X. Li, S. Rao, W. Jiang, C. Li, Y. Xiao, Z. Guo, Q. Zhang, L. Wang, L. Du, J. Li, L. Li, T. Zhang, and Q. K. Wang. Discovery of time-delayed gene regulatory networks based on temporal gene expression profiling, *BMC Bioinformatics*, 7: 26, 2006.

[14] S. Liang, S. Fuhrman and R. Somogyi. REVEAL, A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures. *Pacific Symposium on Biocomputing*, 3:18-29, 1998.

[15] M.T. Matache, and J. Heidel. Random Boolean network model exhibiting deterministic chaos. *Phys. Rev. E*, 69(5):56214-56224, 2004.

[16] H. Matsuno, A. Doi, M. Nagasaki and S. Miyano. Hybrid Petri Net Representation of Gene Regulatory Network. *Pacific Symposium on Biocomputing*, 5:338-349, 2000.

[17] H.H. McAdams and L. Shapiro. Circuit Simulation of Genetic Networks. *Science*, 269:650-656, 1995.

[18] G.S. Michaels, D.B. Carr, M. Askenazi, S. Fuhrman, X. Wen and R. Somogyi. Cluster Analysis and Data Visualization of Large-Scale Gene Expression Data. *Pacific Symposium on Biocomputing*, 3:42-53, 1998.

[19] K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks, *Technical report*, Computer Science Division, University of California, Berkeley, CA, 1999.

[20] H. Oktem, , R. Pearson, and K. Egiazarian. An adjustable aperiodic model class of genomic interactions using continuous time Boolean networks (Boolean delay equations). *Chaos* 13:1167–1174, 2003.

[21] J.R. Quinlan, Induction of decision tree. *Machine Learning,* 1(1):81–106, 1986.

[22] J.R. Quinlan. Rule induction with statistical data - a comparison with multiple regression. *Journal of the Operational Research Society*, 38:347-352, 1987.

[23] M.A. Savageau. Rules for the Evolution of Gene Circuitry. *Pacific Symposium on Biocomputing*, 3:54-65, 1998.

[24] R. Somogyi and C.A. Sniegoski. Modeling the complexity of genetic Networks: Understanding Multigenic and Pleitropic Regulation. *Complexity* 1(6):45-63, 1996.

[25] R. Somogyi, H. Kitano, S. Miyano, and Q. Zheng. Molecular Network Modelling and Data Analysis. Session Introduction - *Pacific Symposium on Biocomputing*, 5:288-289, 2000.

[26] P. T. Spellman et al. Comprehensive Identification of Cell-Cycle regulated Genes of the Yeast Saccharomyces Cervisiae by MicroArray Hybridization. *Molecular Biology of the Cell*, 9:3273-97, 1998.

[27] G. Stormo. Identification of Coordinated Gene Expression and Regulatory Sequences. Session Introduction - *Pacific Symposium on Biocomputing*, 5:413-414, 2000.

[28] D. Thieffry and R. Thomas. Qualitative Analysis of Gene Networks. *Pacific Symposium on Biocomputing*, 3:77-88, 1998.

[29] D.C. Weaver, C.T. Workman, and G.D. Stormo. Modeling Regulatory Networks with Weight Matrices. *Pacific Symposium on Biocomputing*, 4:112-123, 1999.

[30] X. Wen, et al. Large-scale temporal gene expression mapping of central nervous system development. *Proceedings of the National Academy of Science*, 95:334-9, 1998.

# CHAPTER 3. LEARNING CLASSIFIERS FOR ASSIGNING PROTEIN SEQUENCES TO GENE ONTOLOGY FUNCTIONAL FAMILIES.

Modified from a paper published in the *Proceddings of the Fifth International Conference on Knowledge Based Computer Systems*[1]

Carson Andorf[2], Adrian Silvescu[3], Drena Dobbs and Vasant Honavar

## Abstract

Assigning putative functions to novel proteins and the discovery of sequence correlates of protein function are important challenges in bioinformatics. In this paper, we explore several machine learning approaches to data-driven construction of classifiers for assigning protein sequences to appropriate Gene Ontology (GO) function families using a class conditional probabilistic representation of amino acid sequences. Specifically, we represent protein sequences using class conditional probability distribution of amino acids (amino acid composition) or short (k-letter) subsequences (k-grams) of amino acids. We compare a model (NB k-grams) that ignores the statistical dependencies among overlapping k-grams with an alternative, NB(k), that uses an undirected probabilistic graphical model that captures the relevant dependencies. These two methods require only one pass through the training data during the learning phase, making them especially attractive in settings where there is a need to update the classifiers as

[1]Reprinted (no permission needed as the copyrights were not transfered) from the *Proceddings of the Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004).* India. 2004.

[2]Primary researcher and author - implemented the experimental setup, performed the experiments and wrote their description.

[3]Primary researcher and author - proposed and designed the method, experimental validation and made and wrote the theoretical arguments.

new training data become available. We also explore a support vector machine (SVM) classifier, SVM k-grams, trained on the k-gram class conditional probability distributions of sequences. We report the performance of the resulting classifiers on three data sets of functional families from the Gene Ontology (GO) database. Our results show that the NB(k) classifier outperforms NB k-grams in terms of accuracy of classification (as measured by cross-validation) by a few percentage points. SVM k-grams outperforms NB(k) in the majority of test cases. These results suggest the possibility of developing fully automated and computationally efficient approaches to construction of classifiers based on undirected graphical models of overlapping k-grams that can be easily updated as additional training data become available. Our results also show that further gains in accuracy of the classifiers are achievable (at the expense of increased computational demands and hence greater difficulty of frequent updates to the classifier as new training data become available) using SVM k-grams.

## 3.1 Introduction

Proteins are the principal catalytic agents, structural elements, signal transmitters, transporters and molecular machines in cells. Experimental determination of protein structure and function significantly lags behind the rate of growth of protein sequence databases. This situation is likely to continue for the foreseeable future. Hence, assigning proteins putative functions from sequences alone remains one of the most challenging problems in functional genomics [12]. Improvements in annotating protein sequences [19] can be expected to yield significant improvements in gene annotations. One class of sequence-based approaches relies on the comparison of the sequence in question to other sequences in a database of sequences with known function. Functional assignment is made by transference of function whenever sequences are sufficiently similar. A commonly employed notion of similarity is based on estimated sequence homology with programs such as BLAST and its derivatives [1]. Sequence searches often return multiple hits, so significant human expertise is needed for interpreting results. The reliability of homo-logs detected by multiple sequence alignment rapidly drops once the pair-wise sequence identity drops below 30 percent [19]. A second class of sequence-based approaches for assign-

ing putative functions to protein sequences rely on the detection of sequence patterns (Several automated tools for identifying conserved sequence patterns from a given set of sequences e.g., e-Motif and e-Matrix [15], [7], MEME [5] are available.) Motif databases can be queried using a protein sequence to obtain a list of conserved sequence patterns found in the sequence as well as functions associated with the respective patterns. The results can be used to assign putative functions to the protein sequence. In the case of protein families having sufficient numbers of well-characterized members, data mining approaches rooted in statistical inference and machine learning [6] offer an attractive and cost-effective approach to automated construction of classifiers for assigning putative functions to novel protein sequences. In essence, the data mining approach uses a representative training data set that encodes information about proteins with known functions to build a classifier for assigning proteins to one of the functional families represented in the training set (and if necessary, a default class indicating unknown function). The resulting classifier can then be used to assign novel protein sequences to one of the protein families represented in the training set after it has been validated using an independent test set (which was not used to build the classifier). Recent work by our group [22], [2] has explored the use of machine learning approaches to automated construction of such classifiers.

In this paper, we explore methods that use class conditional probabilities of k-grams (k-letter sub-sequences) [10] to represent amino acid sequences.

The first method uses a Naive Bayes classifier which treats each amino acid sequence as if it were simply a bag of amino acids.

The second method (NB k-grams) applies the Naive Bayes classifier to a bag of k-grams ($k > 1$). Note that NB k-grams violates the Naive Bayes assumption of independence in an obvious fashion: neighboring k-grams overlap along the sequence, adjacent k-grams have k-1 elements in common.

Our third method overcomes this problem by constructing an undirected graphical probabilistic model for k-grams, which explicitly models the dependencies among overlapping k-grams in a sequence - NB(k). This method will adequately discount the contributions of the overlaps. We train one model per functional family. During classification, just as in the case of the

Naive Bayes classifier, the sequence to be classified is assigned to the class that has the largest posterior probability given the sequence. We call the resulting classifier NB(k) to denote the fact that it models dependencies among k adjacent elements of sequences. Note that NB(1) is equivalent to NB 1-grams, which in turn is equivalent to the Naive Bayes classifier.

Our fourth method applies a support vector machine (SVM) [9], [21] to classify amino acid sequences represented using class conditional probability distributions of k-grams in the sequence. SVM-s have recently been applied successfully to many problems in computational biology including protein function classification [16] and identification of protein-protein interaction sites from sequences [23]. However, to the best of our knowledge, previous work using SVM has not utilized a class conditional k gram probability-based representation of amino acid sequences.

While SVM-s, unlike NB(k) and NB k-grams classifiers, do not have the advantage of training with only one pass through the training data, they are attractive in scenarios where higher accuracy of classification than that achievable by algorithms that make a single pass through the training data is desired. This increased accuracy comes at the expense of increased computational requirements - especially in cases where it is necessary to update the classifiers frequently as new training data become available. On a large data set a SVM classifier may take days to construct while NB(k) and NB k-grams can build a classifier in minutes. Hence, we explore an SVM that uses as input a k-gram class conditional probability distribution associated with the protein sequence to be classified. We call this fourth method SVM k-grams. This method is comparable to work using SVM-s to predict sub-cellular localization based on amino acids [14], [8]. Their research focused on using mono-peptide and di-peptide composition. We consider larger ordered polypeptide composition in our study.

We compare NB k-grams, NB(k) SVM k-grams classifiers for assigning protein sequences to the corresponding GO (the Gene Ontology [13]) taxonomy of protein functional families. The sequence data sets used in our experiments were extracted from SWISSPROT [4]. In our experiments, the NB k-gram classifier outperformed (in terms of classification accuracy), the standard Naive Bayes classifier by a large margin; the NB(k) classifier outperformed NB

1. For each class $c_j$ train the probabilistic model $M(c_j)$ of type $M$ using the sequences that belong to the class $c_j$.
2. Predict the class of the sequence $class(\bar{S})$ of a novel sequence $\overline{S}$ as:

$$class(\overline{S}) \quad = \quad \arg\max_{c_j \in C} P_{M(c_j)}(\bar{S}) P(c_j)$$

Algorithm 4 Classification using a Probabilistic Model

k-grams classifier by a few percentage points, and SVM k grams outperformed NB(k) in the majority of the test cases.

## 3.2 Method

In this section we outline the two probabilistic models we use for modeling the interactions among k consecutive elements in the sequence. First, we define a method to build a classifier associated with a probabilistic model.

### 3.2.1 Classification using a Probabilistic Model

**Notations**: Let $\Sigma$ be a finite set called alphabet (which in our case is the 20-letter amino acid alphabet). We will denote a sequence of elements from $\Sigma^*$ with $\overline{S}$, the $i$-th element of the sequence by $S_i$ and the value of the $i$-th element in the sequence by $s_i$. We will write a sequence of length $n$ as $\overline{S} = [S_1 = s_1, ..., S_n = s_n]$. Let $C = \{c_j\}_{j=1,m}$ be a finite set of (mutually exclusive) class labels for sequences.

A probabilistic model $M$ for sequences defined over the alphabet $\Sigma$ is any function that specifies a probability $P_M(\overline{S} = [S_1 = s_1, ..., S_n = s_n])$ for every sequence $\bar{S} \in \Sigma^*$. Given a model $M(c_j)$ of the type $M$ for each class $c_j \in C$ we define $P_{M(c_j)}(\bar{S}) := P_M(\bar{S}|c_j)$ (that is the probability of that the sequence $\bar{S}$ appears in class $c_j$, according to a model of type $M$ for each class: $P_M(\bar{S}|c_j)$). Given such a probabilistic model $M$ we can produce a classification scheme as follows:

Note that since $P_{M(c_j)}(\bar{S}) := P_M(\bar{S}|c_j)$ the class prediction can also be written:

$$class(\overline{S}) = \arg\max_{c_j \in C} P_M(\bar{S}|c_j)P(c_j)$$

To summarize: given that such a probabilistic model is established, we can then produce a classifier using the scheme outlined above in Algorithm 4.

### 3.2.2 Naive Bayes Classifier - NB

In the case when the probabilistic model form the previous section assumes independence among the occurrences of each symbol in the sequence $\bar{S}$ given the class label we obtain the Naive Bayes Classifier.

**Naive Bayes Classifier - NB:**

$$\begin{aligned} class(\overline{S}) &= \arg\max_{c_j \in C} P_{NB}(\bar{S} = [S_1 = s_1, ..., S_n = s_n]|c_j)P(c_j) \\ &= \arg\max_{c_j \in C} \prod_{i=1}^{n} P(S_i = s_i|c_j)P(c_j) \end{aligned}$$

Where the second line follows from the independence assumption, which in the absence of a class is:

$$P_{NB}(\bar{S} = [S_1 = s_1, ..., S_n = s_n]) = \prod_{i=1}^{n} P(S_i = s_i)$$

The Naive Bayes classifier is a particular example of the previously outlined general scheme for producing a classifier from a given probabilistic model $M$ for the input data (sequence in our case).

Since the elements in a sequence are seldom independent next we will examine models that attempt to capture interactions among the elements in the sequence. More exactly we will try to model the direct interactions among $k$ consecutive elements.

### 3.2.3 Naive Bayes $k$-grams

The Naive Bayes $k$-grams (NB $k$-grams) method uses a sliding a window of size $k$ along each sequence to generate a bag of $k$-grams representation of the sequence. Much like in the case

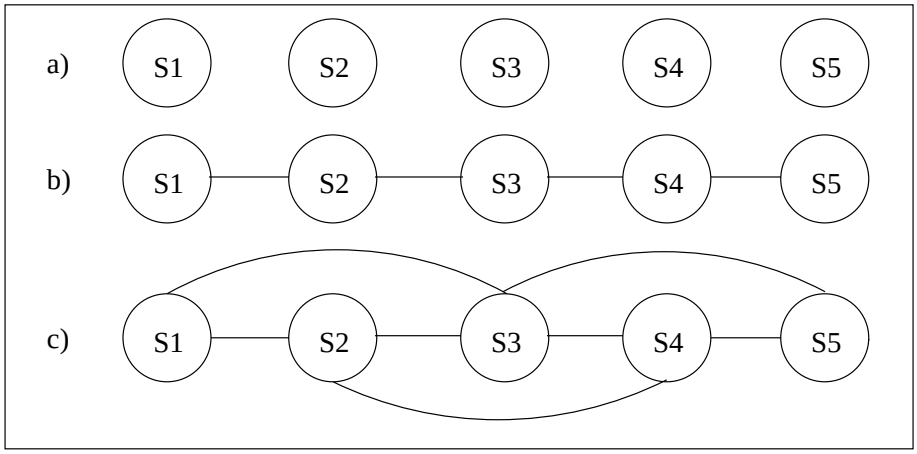



Figure 3.1 Graphical depiction of the dependence between the elements in a sequence of five elements: a) independence model (yields the Naive Bayes classifier); b) pairwise dependence; and c) 3-wise dependence.

of the Naive Bayes classifier described above treats each k-gram in the bag to be independent of the others given the class label for the sequence. The classification scheme associated with Naïve Bayes k-grams is:

$$class(\bar{S}) = \arg\max_{c_j \in C} \prod_{i=1}^{n-k+1} P(S_i = s_i, ..., S_{i+k-1} = s_{i+k-1}|c_j)P(c_j)$$

A problem with the NB $k$-grams approach is that successive $k$-grams extracted from a sequence share $k-1$ elements in common. This grossly and systematically violates the independence assumption of Naive Bayes. A systematic way to deal with this problem is presented next.

### 3.2.4 Naive Bayes (k)

We introduce the Naive Bayes (k) or the NB(k) model [20] to explicitly model the dependencies that arise as a consequence of the overlap between successive k-grams in a sequence. We represent the dependencies in a graphical form by drawing edges between the elements that are directly dependent on each other (Figure 3.1).

As a consequence of the Junction Tree Theorem for Decomposable Undirected Graphical

Models [11], the correct probability model that captures the dependencies among overlapping k-grams is given by:

$$P(\overline{S} = [S_1 = s_1, ..., S_n = s_n]) \ = \frac{\prod_{i=1}^{n-k+1} P(S_i = s_i, ..., S_{i+k-1} = s_{i+k-1})}{\prod_{i=2}^{n-k+1} P(S_i = s_i, ..., S_{i+k-2} = s_{i+k-2})} \tag{3.1}$$

Now, given this probabilistic model, we can use the standard approach to classification given a probabilistic model outlined above. It is easily seen that when k = 1, Naive Bayes 1-grams as well as Naive Bayes (1) reduce to the Naive Bayes model. The relevant probabilities required for specifying the above models can be estimated using standard techniques for estimation of probabilities using Laplace estimators [17].

### 3.2.5 On Naive Bayes k-gram vs. NB(k)

One intuitive way to try to think about the Naive Bayes k-grams case is a attempting to specify a probabilistic model by taking the product of the marginals of the directly dependent nodes as follows:

$$P\ (\overline{S} = [S_1 = s_1, ..., S_n = s_n])\ ? = \prod_{i=1}^{n-k+1} P(S_i = s_i, S_{i+k-1} = s_{i+k-1})$$

And if we are to plug this formula into our scheme for producing classifiers outlined previously we will obtain Naive Bayes $k$-grams. The only problem with this is that the right hand side is not a probability distribution (hence the question mark before the equality sign) - because the probabilities do not add up to 1 but to a smaller number as not all the possible combinations of k-grams are possible (e.g., if the first 2-gram is $ab$ then the second (overlapping) k-gram cannot be an arbitrary one but must start with a $b$). Another intuition behind it is that we are somehow "double counting" or more exactly "double multiplying" the influence of some of the nodes namely: $S_2 = s_2, ..., S_{n-1} = s_{n-1}$. For example $S_2 = s_2$ appears both in $P(S_1 = s_1, S_2 = s_2)$ and $P(S_2 = s_2, S_3 = s_3)$. Note that because the probabilities are numbers less than 1 and therefor "double multiplying" makes the numbers smaller, which in turn makes us the probabilities undershoot the 1 target unlike "double counting" which usually makes the numbers bigger and which make us overshoot the target. NB(k) is the right way to

"fix" this problem and it yields a probabilistic model (i.e., the probabilities add up to 1) and is the Decomposable Undirected Graphical Model that accurately represents the $k$-way direct dependencies depicted in Figure 3.1 as follows from the Junction Tree Theorem [11].

### 3.2.6 SVM k-grams

Note that the NB(k) algorithm was developed because NB k-grams systematically violates the independence assumption of Naïve Bayes. Against this background, it is of interest to consider other methods that can utilize class conditional k-gram frequencies without relying on the independence assumptions made by NB k-grams and without the need for explicit modeling of dependencies as in the case of NB(k). Hence, we consider a Support Vector Machine (SVM) classifier which accepts as input, a class conditional k-gram probability distribution for the protein (which is same as the one used by NB k-grams model) and outputs a class label. In this method, the SVM classifier can be seen as a second stage of a 2-stage classifier. The inputs to the SVM are supplied by the NB k-grams model.

## 3.3 Experimental setup and results

We compare the performance of the four classifiers - NB (= NB 1-gram = NB(1)), NB k-grams, NB(k) and SVM k-grams on three protein function classification data sets.

**Data Sets:** The data sets used in this study are constructed as follows: First, a set of functional classes are chosen from the GO taxonomy of protein functional families. Then, the corresponding sequences are retrieved from the SWISSPROT data-base [4]. Because the GO taxonomy has the form of a directed acyclic graph, and many proteins are multifunctional, it is possible that a given protein belongs to multiple functional families. Hence, the resulting data set is filtered to remove proteins that had multiple GO class labels to ensure that the classes are non overlapping (i.e.., mutually disjoint) - a requirement of most standard machine learning and statistical methods for classification including the methods considered in this paper. (The development of principled approaches to classification of data that are labeled with multiple class labels is largely an open problem in machine learning).

The first data set was derived from families of yeast and human kinases. These families were chosen for this study because many of them are well-characterized, with known structures and functions. The data set used in this study consisted of 396 proteins belonging to the Gene Ontology functional family GO0004672, Protein Kinase Activity. We classified them according to the highest level below GO0004672. This consists of 5 groups. In GO, their labels are GO0004672, Protein Kinase Activity (102 proteins); GO0004674, protein serine/threonine kinase activity (209 proteins); GO0004713, protein-tyrosine kinase activity (69 proteins); GO0004712, protein threonine/tyrosine kinase activity (10 proteins); and GO0004716, receptor signaling protein tyrosine kinase activity (6 proteins).

The second data set is derived from two subfamilies of GO0003824, Catalytic Activity. This division is at a higher level of GO then the previous data set and consists of 376 proteins belonging to the Gene Ontology functional family GO0004672, Protein Kinase Activity (158 proteins) and GO001684, Protein Ligase Activity (218 proteins).

The third data set is a super-set of data set two. It contains the Kinase data and Ligate data in addition to two other subfamilies of GO0003824, Catalytic Activity. These families are GO0004386, Protein Helicase Activity (110 proteins), and GO0016853, Protein Isomerase Activity (86 proteins). This data set tests the classifiers ability on a larger number of classes at a high level of GO and includes a total of 572 proteins.

**Experiments and Results:** The computational experiments were motivated by the following questions:

1. How do NB k-grams, NB(k), and SVM k-grams models compare with each other and against the baseline represented by Naïve Bayes (NB) classifier?

2. What is the effect of k (which can be viewed as a measure of the complexity of the models in question) on classification accuracy of the resulting classifiers?

NB k-grams and NB(k) models were constructed and evaluated on the three data sets for different choices of k from 1 to 4. Values of k larger than 4 were not considered because at higher values of k we run out of data to obtain reliable probability estimates. SVM k grams

| k | NB | NB k-grams | NB(k) | SVM k-grams |
|---|---|---|---|---|
| 1 | 66.1 | 66.1 | 66.1 | 84.1 |
| 2 | | 81.3 | 88.6 | 90.7 |
| 3 | | 89.9 | 92.7 | 90.3 |
| 4 | | 90.4 | 91.6 | X |

Table 3.1 Kinase data set results - classification accuracy estimated by 10-fold cross validation (Note: NB method applies only for k=1; SVM k-grams was found to be infeasible because of computational and memory requirements k > 3).

| k | NB | NB k-grams | NB(k) | SVM k-grams |
|---|---|---|---|---|
| 1 | 77.9 | 77.9 | 77.9 | 97.6 |
| 2 | | 83.5 | 84.6 | 100.0 |
| 3 | | 84.0 | 85.6 | 100.0 |
| 4 | | 85.9 | 90.7 | X |

Table 3.2 Kinase/Ligase data set results - classification accuracy estimated by 10-fold cross validation (Note: NB method applies only for k=1; SVM k-grams was found to be infeasible because of computational and memory requirements k > 3).

model, using a linear SVM kernel, was tested with values of k from 1 to 3. (Higher values of k were not explored because of computational and memory requirements). The reported accuracy estimates are based on stratified 10-fold cross validation. Within the 10-fold cross validation experiments, the majority of the individual standard deviations among classifiers were under 1% and never reached above 2%. This shows little variability among the classifiers used for these experiments.

| k | NB | NB k-grams | NB(k) | SVM k-grams |
|---|---|---|---|---|
| 1 | 56.1 | 56.1 | 56.1 | 93.9 |
| 2 | | 70.3 | 72.2 | 94.5 |
| 3 | | 79.5 | 80.8 | 94.7 |
| 4 | | 79.4 | 82.0 | X |

Table 3.3 Kinase/Ligase/Helicase/Isomerase data set results - classification accuracy estimated by 10-fold cross validation (Note: NB method applies only for k=1; SVM k-grams was found to be infeasible because of computational and memory requirements k > 3).

Because SVM is a binary classifier, and the problem calls for multi-class classifier, a separate SVM classifier was constructed for each class. The $i$-th classifier is trained using the training data from class i as positive examples and the rest of the training data as negative examples. Note that unlike SVM, NB, NB k-grams, and NB(k) can build a single multi-class classifier.

Table 1 shows the results using the Kinase data set. We obtained a classification accuracy of 66% classification using Naive Bayes alone and an accuracy of 84% using SVM 1-gram. Increasing k to 2 resulted in significant improvements in accuracy: The accuracy increased to 81.3% for NB 2-grams, 88.6% for NB(2), and 90.7% for SVM 2-grams. In the case of NB(2) this represents 22% improvement over Naive Bayes and 7% improvement in classification accuracy over NB 2- grams. SVM on 2-grams only outperformed NB(2) by about 2% in terms of classification accuracy. NB 3-grams and NB(3) had accuracies of 89.9% and 92.7% respectively, with NB(3) (92.7%) actually outperforming SVM 3-grams (90.3%). Increasing k to 4 resulted in little improvement on this data set. NB 4-grams improved by only 0.5% and NB(4), while performing better than NB 4-grams, has slightly worse accuracy relative to NB(3). This can be explained by the fact that as k increases, the probability estimates become less and less reliable (as we run out of data).

Similar results were obtained for the Kinase/Ligase and Kinase/Ligase/Isomerase/Helicase data sets. NB(4) outperforms NB(3) (by over 5% for the second data set [Table 2] and nearly 1.2% for the third data set [Table 3]) and NB 4-grams (nearly 5% [Table 2] and over 2% [Table 3]. SVM k-grams significantly outperforms NB(k), yielding 100% accuracy for the second data set and 94.7% accuracy for the third data set. This corresponds to a 14% improvement over the best accuracy of NB(k) on each of the data sets.

The experimental results demonstrate the superiority of both NB k-grams and NB(k) over Naive Bayes on all test cases using these data-sets. Furthermore, in terms of accuracy, NB(k) outperforms NB k-grams, and SVM k-grams significantly outperformed NB(k) in terms of classification accuracy in two of the three test cases. In one of the test cases, the performance of SVM k-grams was comparable to that of NB k-grams. The results collectively demonstrate the utility of representing amino acid sequences in terms of class conditional probabilities of

amino acids or k-grams of amino acids for sequence-based assignment of proteins to functional families.

## 3.4 Summary and Discussion

Development of robust methods for assigning putative functions to novel proteins and the discovery of sequence correlates of protein function are important challenges in bioinformatics.

This paper explored several methods for assigning protein sequences to functional families based on class conditional probability distributions of amino acids or short sub-sequences (k-grams) of amino acids on three data sets. The data sets were extracted from functional classes extracted from GO [13] and the corresponding sequences are extracted from SWISSPROT [4].

Our results show that the NB(k) classifier, which models the dependencies among overlapping k-grams in a sequence, consistently outperforms NB k-grams and the Naive Bayes classifier in terms of classification accuracy. SVM k-grams, which also uses the class conditional k-gram probabilities for the sequences outperforms NB(k) on two of the three data sets in terms of classification accuracy.

NB k-grams and NB(k) require only one pass through the data which makes the resulting classifiers easy to construct and update as new data become available. In contrast, at present, there are no efficient algorithms for updating SVM classifiers to incorporate new data in an incremental fashion. This makes NB(k) an attractive alternative when using large data sets or data sets that are rapidly being updated or modified.

Some directions for future work include: exploration of classifiers constructed using reduced alphabet representations of protein sequence [2]; development of principled approaches to assigning a protein sequence simultaneously to multiple classes (in the case of multifunctional proteins); incorporation of other sources of information (e.g., expression data, interaction data, structural features) to enhance the accuracy of function classification; examination of the resulting classifiers to identify testable hypotheses concerning sequence correlates of protein function and to guide the design of experiments to validate such hypotheses.

## 3.5 PostScript

Since the original publication of this paper additional results have been obtain in this vein [3].

## Bibliography

[1] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. In *Nucleic Acid Res. Sep* 1; 2(17):3389 - 3402 ,1997.

[2] C. Andorf, D. Dobbs, and V. Honavar. Discovering protein function classification rules from reduced alphabet representations of protein sequences. In *Proceedings of the Conference on Computational Biology and Genome Informatics*. Durham, North Carolina, 2002.

[3] C. Andorf, D. Dobbs, and V. Honavar. Exploring Inconsistencies in Genome Wide Protein Function Annotations: A Machine Learning Approach. *BMC Bioinformatics*, 8:284, 2007

[4] B. Boeckmann, A. Bairoch, R. Apweiler, M. Blatter, A. Estreicher, E. Gasteiger, M. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The Swiss-Prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acid Res.* 31:365 – 370, 2003.

[5] T. Bailey, M. Baker, C. Elkan, and W. Grundy. Meme, mast, and metameme: New tools for motif discovery in protein sequences. *Pattern Discovery in Biomolecular Data*. Oxford University Press, Oxford, pp. 30 – 54, 1999.

[6] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: MIT Press, 1998.

[7] A. Ben-Hur and D. Brutlag. Remote homology detection: a motif based approach. *Bioinformatics,* 19 Suppl. 1, 2003.

[8] M. Bhasin and G. Raghava. ESLpred: SVM-based method for subcellular localization of eukaryotic proteins using dipeptide composition and PSI-BLAST. *Nucleic Acids Research*, 32, 2004.

[9] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. Proceedings of the *5th Annual ACM Workshop on Computational Learning Theory*, pp. 144 – 152, Pittsburg, PA, ACM Press, 1992.

[10] E. Charniak. *Statistical Language Learning, Cambridge*. MIT Press, 1993.

[11] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[12] D. Eisenberg, E. Marcotte, and T. Xenarios, and I. Yeates. Protein function in the post-genomic era. *Nature*. 405(6788): 823-6, 2000.

[13] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genet.*, 25: 25 – 29, 2000.

[14] S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721 – 728, 2001.

[15] J. Huang and D. Brutlag. The emotif database. *Nucleic Acids Res.* Jan 1, 29(1): 202-4, 2001.

[16] G. Lanckriet, N. Cristianini, M. Jordan, and W. Noble. Kernel-based integration of genomic data using semidefinite programming. In B. Schoelkopf, K. Tsuda and J-P. Vert (Eds.), *Kernel Methods in Computational Biology*, Cambridge, MA: MIT Press, 2003.

[17] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, 1988.

[19] B. Rost. Twilight zone of protein sequence alignments. *Protein Eng.* 12(2): 85 – 94, 1999.

[20] A. Silvescu, C. Andorf, D. Dobbs, and V. Honavar. Inter-element dependency models for sequence classification, *Technical report*, Department of Computer Science, Iowa State University, 2004.

[21] V. Vapnik. *Statistical learning theory. Adaptive and learning systems for signal processing, communications, and control*. Wiley, New York, 1998.

[22] X. Wang, D. Schroeder, D. Dobbs, and V. Honavar. Automated data-driven discovery of protein function classifiers. *Information Sciences,* 155:1 – 18, 2003.

[23] C. Yan, D. Dobbs, V. Honavar. A two-stage classifier for identification of protein-protein interface residues. *Bioinformatics,* 20:371-378, 2004.

# CHAPTER 4. GENERATION OF ATTRIBUTE VALUE TAXONOMIES FROM DATA FOR ACCURATE AND COMPACT CLASSIFIER CONSTRUCTION

Modified from a paper published in the *Proceedings of the IEEE International Conference on Data Mining*[1]

Dae-Ki Kang[2], Adrian Silvescu[3] Jun Zhang[4]

and Vasant Honavar

## Abstract

Attribute Value Taxonomies (AVT) have been shown to be useful in constructing compact, robust, and comprehensible classifiers. However, in many application domains, human-designed AVTs are unavailable. We introduce AVT-Learner, an algorithm for automated construction of attribute value taxonomies from data. AVT-Learner uses Hierarchical Agglomerative Clustering (HAC) to cluster attribute values based on the distribution of classes that co-occur with the values. We describe experiments on UCI data sets that compare the performance of AVT-NBL (an AVT-guided Naive Bayes Learner) with that of the standard Naive Bayes Learner (NBL) applied to the original data set. Our results show that the AVTs generated by AVT-Learner are competitive with human-generated AVTs (in cases where such AVTs are available). AVT-NBL

[1]Reprinted (no permission needed as the copyrights were not transfered) from *Proceedings of the IEEE International Conference on Data Mining (ICDM 2004)*, Brighton, UK, November 1, 2004.

[2]Primary researcher and author - implemented the experimental setup and performed and designed the experiments.

[3]Primary researcher and author - proposed and designed the method and the validation setup.

[4]Primary researcher and author - contributed the classifier used for validation.

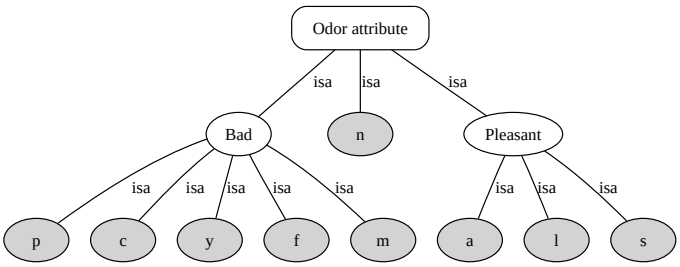



Figure 4.1 Human-made AVT from 'odor' attribute of UCI AGARICUS-LEPIOTA mushroom data set.

using AVTs generated by AVT-Learner achieves classification accuracies that are comparable to or higher than those obtained by NBL; and the resulting classifiers are significantly more compact than those generated by NBL.

## 4.1 Introduction

An important goal of inductive learning is to generate accurate and compact classifiers from data. In a typical inductive learning scenario, instances to be classified are represented as ordered tuples of attribute values. However, attribute values can be grouped together to reflect assumed or actual similarities among the values in a domain of interest or in the context of a specific application. Such a hierarchical grouping of attribute values yields an attribute value taxonomy (AVT). For example, Figure 4.1 shows a human-made taxonomy associated with the nominal attribute *'Odor'* of the UC Irvine AGARICUS-LEPIOTA mushroom data set [5].

Hierarchical groupings of attribute values (AVT) are quite common in biological sciences. For example, the Gene Ontology Consortium is developing hierarchical taxonomies for describing many aspects of macromolecular sequence, structure, and function [1]. Undercoffer et al. [34] have developed a hierarchical taxonomy which captures the features that are observable or measurable by the target of an attack or by a system of sensors acting on behalf of the target. Several ontologies being developed as part of the Semantic Web related efforts [4] also capture hierarchical groupings of attribute values. Kohavi and Provost [19] have noted the need to be able to incorporate background knowledge in the form of hierarchies over data attributes in

electronic commerce applications of data mining.

There are several reasons for exploiting AVT in learning classifiers from data, perhaps the most important being a preference for comprehensible and simple, yet accurate and robust classifiers [24] in many practical applications of data mining. The availability of AVT presents the opportunity to learn classification rules that are expressed in terms of *abstract* attribute values leading to simpler, easier-to-comprehend rules that are expressed in terms of hierarchically related values. Thus, the rule $(odor = pleasant) \rightarrow (class = edible)$ is likely to be preferred over $((odor = a) \wedge (color = brown)) \vee ((odor = l) \wedge (color = brown)) \vee ((odor = s) \wedge (color = brown)) \rightarrow (class = edible)$ by a user who is familiar with the odor taxonomy shown in Figure 4.1.

Another reason for exploiting AVTs in learning classifiers from data arises from the necessity, in many application domains, for learning from small data sets where there is a greater chance of generating classifiers that over-fit the training data. A common approach used by statisticians when estimating from small samples involves *shrinkage* [9] or grouping attribute values (or more commonly class labels) into bins, when there are too few instances that match any specific attribute value or class label, to estimate the relevant statistics with adequate confidence. Learning algorithms that exploit AVT can potentially perform *shrinkage* automatically thereby yielding robust classifiers. In other words, exploiting information provided by an AVT can be an effective approach to performing regularization to minimize over-fitting [40].

Consequently, several algorithms for learning classifiers from AVTs and data have been proposed in the literature. This work has shown that AVTs can be exploited to improve the accuracy of classification and in many instances, to reduce the complexity and increase the comprehensibility of the resulting classifiers [8, 13, 16, 32, 40, 43]. Most of these algorithms exploit AVTs to represent the information needed for classification at different levels of abstraction.

However, in many domains, AVTs specified by human experts are unavailable. Even when a human-supplied AVT is available, it is interesting to explore whether alternative groupings of attribute values into an AVT might yield more accurate or more compact classifiers. Against this background, we explore the problem of automated construction of AVTs from data. In

particular, we are interested in AVTs that are useful for generating accurate and compact classifiers.

## 4.2 Learning attribute value taxonomies from data

### 4.2.1 Learning AVT from data

We describe AVT-Learner, an algorithm for automated construction of AVT from a data set of instances wherein each instance is described by an ordered tuple of N nominal attribute values and a class label.

Let $A = \{A_1, A_2, \ldots, A_n\}$ be a set of nominal attributes. Let $V_i = \left\{v_i^1, v_i^2, \ldots, v_i^{m_i}\right\}$ be a finite domain of mutually exclusive values associated with attribute $A_i$ where $v_i^j$ is the $j^{th}$ attribute value of $A_i$ and $m_i$ is the number possible number of values of $A_i$, that is, $|V_i|$. We say that $V_i$ is the set of primitive values of attribute $A_i$. Let $C = \{C_1, C_2, \ldots, C_k\}$ be a set of mutually disjoint class labels. A data set is $D \subseteq V_1 \times V_2 \times \cdots \times V_n \times C$.

Let $T = \{T_1, T_2, \ldots, T_n\}$ denote a set of AVT such that $T_i$ is an AVT associated with the attribute $A_i$, and $Leaves(T_i)$ denote a set of all leaf nodes in $T_i$. We define a cut $\delta_i$ of an AVT $T_i$ to be a subset of nodes in $T_i$ satisfying the following two properties: (1) For any leaf $l \in Leaves(T_i)$, either $l \in \delta_i$ or $l$ is a descendant of a node $n \in \delta_i$; and (2) for any two nodes $f, g \in \delta_i$, $f$ is neither a descendant nor an ancestor of $g$ [14]. For example, $\{Bad, a, l, s, n\}$ is a cut through the AVT for *odor* shown in Figure 4.1. Note that a cut through $T_i$ corresponds to a partition of the values in $V_i$. Let $\Delta = \{\delta_1, \delta_2, \ldots \delta_n\}$ be a set of cuts associated with AVTs in $T = \{T_1, T_2, \ldots T_n\}$.

The problem of learning AVTs from data can be stated as follows: given a data set $D \subseteq V_1 \times V_2 \times \cdots \times V_n \times C$ and a measure of dissimilarity (or equivalently similarity) between any pair of values of an attribute, output a set of AVTs $T = \{T_1, T_2, \ldots, T_n\}$ such that each $T_i$ (AVT associated with the attribute $A_i$ ) corresponds to a hierarchical grouping of values in $V_i$ based on the specified similarity measure.

We use hierarchical agglomerative clustering (HAC) of the attribute values according to the distribution of classes that co-occur with them. Let $DM\left(P\left(x\right)||P\left(y\right)\right)$ denote a measure of

pairwise divergence between two probability distributions $P(x)$ and $P(y)$ where the random variables $x$ and $y$ take values from the same domain. We use the pairwise divergence between the distributions of class labels associated with the corresponding attribute values as a measure of the dissimilarity between the attribute values. The lower the divergence between the class distributions associated with two attributes, the greater is their similarity. The choice of this measure of dissimilarity between attribute values is motivated by the intended use of the AVT, namely, the construction of accurate, compact, and robust classifiers. If two values of an attribute are indistinguishable from each other with respect to their class distributions, they provide statistically similar information for classification of instances.

The algorithm for learning AVT for a nominal attribute is shown in Figure 4.2. The basic idea behind AVT-Learner is to construct an AVT $T_i$ for each attribute $A_i$ by starting with the primitive values in $V_i$ as the leaves of $T_i$ and recursively add nodes to $T_i$ one at a time by merging two existing nodes. To aid this process, the algorithm maintains a cut $\delta_i$ through the AVT $T_i$, updating the cut $\delta_i$ as new nodes are added to $T_i$. At each step, the two attribute values to be grouped together to obtain an abstract attribute value to be added to $T_i$ are selected from $\delta_i$ based on the divergence between the class distributions associated with the corresponding values. That is, a pair of attribute values in $\delta_i$ are merged if they have more similar class distributions than any other pair of attribute values in $\delta_i$. This process terminates when the cut $\delta_i$ contains a single value which corresponds to the root of $T_i$. If $|V_i| = m_i$, the resulting $T_i$ will have $(2m_i - 1)$ nodes when the algorithm terminates.

In the case of continuous-valued attributes, we define intervals based on observed values for the attribute in the data set. We then generate a hierarchical grouping of adjacent intervals, selecting at each step two adjacent intervals to merge using the pairwise divergence measure. A cut through the resulting AVT corresponds to a discretization of the continuous-valued attribute. A similar approach can be used to generate AVT from ordinal attribute values.

**AVT-Learner:**
**begin**
1. **Input** : data set D
2. For each attribute $A_i$:
3. For each attribute value $v_i^j$ :
4. For each class label $c_k$: estimate the probability $p\left(c_k|v_i^j\right)$
5. Let $P\left(C|v_i^j\right) := \left\{P\left(c_1|v_i^j\right), \ldots, P\left(c_k|v_i^j\right)\right\}$
6. Set $\delta_i \leftarrow V_i$; Initialize $T_i$ with nodes in $\delta_i$.
7. Iterate until $|\delta_i| = 1$:
8. In $\delta_i$, find $(x, y) = argmin\left\{DM\left(P\left(C|v_i^x\right)||P\left(C|v_i^y\right)\right)\right\}$
9. Merge $v_i^x$ and $v_i^y$ $(x \neq y)$ to create a new value $v_i^{xy}$.
10. Calculate probability distribution $P\left(C|v_i^{xy}\right)$.
11. $\lambda_i \leftarrow \delta_i \cup \{v_i^{xy}\} \setminus \{v_i^x, v_i^y\}$.
12. Update $T_i$ by adding nodes $v_i^{xy}$ as a parent of $v_i^x$ and $v_i^y$.
13. $\delta_i \leftarrow \lambda_i$.
14. **Output** : $T = \{T_1, T_2, \ldots, T_n\}$
**end.**

Figure 4.2 Pseudo-code of AVT-Learner

### 4.2.2 Pairwise divergence measures

There are several ways to measure similarity between two probability distributions. We have tested thirteen divergence measures for probability distributions $P$ and $Q$. In this paper, we limit the discussion to Jensen-Shannon divergence measure.

**Jensen-Shannon divergence** [29] is weighted information gain, also called Jensen difference divergence, information radius, Jensen difference divergence, and Sibson-Burbea-Rao Jensen Shannon divergence. It is given by:

$$I\left(P||Q\right) = \frac{1}{2}\left[\sum p_i log\left(\frac{2p_i}{p_i + q_i}\right) + \sum q_i log\left(\frac{2q_i}{p_i + q_i}\right)\right]$$

Jensen-Shannon divergence is reflexive, symmetric and bounded. Figure 4.3 shows an AVT of 'odor' attribute generated by AVT-Learner (with binary clustering).

## 4.3 Evaluation of AVT-Learner

The intuition behind our approach to evaluating the AVT generated by AVT-Learner is the following: an AVT that captures relevant relationships among attribute values can result in the

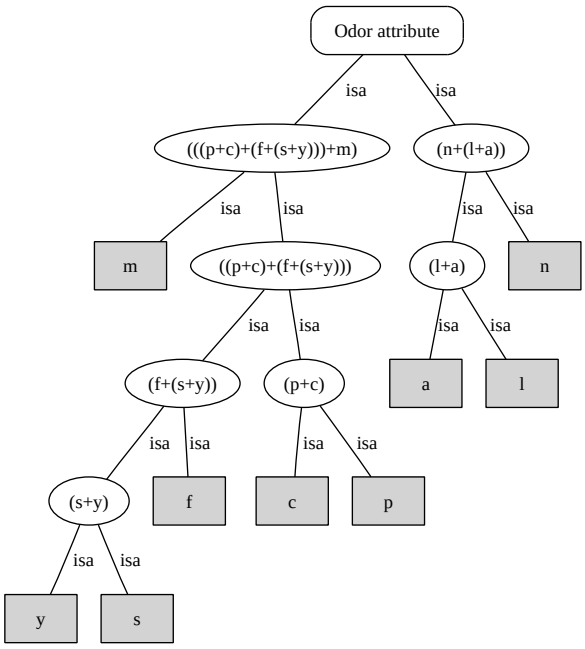



Figure 4.3 AVT of 'odor' attribute of UCI AGARICUS-LEPIOTA mushroom data set generated by AVT-Learner using Jensen-Shannon divergence (binary clustering)

generation of simple and accurate classifiers from data, just as an appropriate choice of axioms in a mathematical domain can simplify proofs of theorems. Thus, the simplicity and predictive accuracy of the learned classifiers based on alternative choices of AVT can be used to evaluate the utility of the corresponding AVT in specific contexts.

### 4.3.1 AVT guided variants of standard learning algorithms

It is possible to extend standard learning algorithms in principled ways so as to exploit the information provided by AVT. AVT-DTL [37, 43, 40] and the AVT-NBL [41] which extend the decision tree learning algorithm [26] and the Naive Bayes learning algorithm [21] respectively are examples such algorithms.

The basic idea behind AVT-NBL is to start with the Naive Bayes Classifier that is based on the most abstract attribute values in AVTs and successively refine the classifier by a scoring function - a Conditional Minimum Description Length (CMDL) score suggested by Friedman et al. [10] to capture trade-off between the accuracy of classification and the complexity of the

resulting Naive Bayes classifier.

The experiments reported by Zhang and Honavar [41] using several benchmark data sets show that AVT-NBL is able to learn, using human generated AVT, substantially more accurate classifiers than those produced by Naive Bayes Learner (NBL) applied directly to the data sets as well as NBL applied to data sets represented using a set of binary features that correspond to the nodes of the AVT (PROP-NBL). The classifiers generated by AVT-NBL are substantially more compact than those generated by NBL and PROP-NBL. These results hold across a wide range of missing attribute values in the data sets. Hence, the performance of Naive Bayes classifiers generated by AVT-NBL when supplied with AVT generated by the AVT-Learner provide useful measures of the effectiveness of AVT-Learner in discovering hierarchical groupings of attribute values that are useful in constructing compact and accurate classifiers from data.

## 4.4 Experiments

### 4.4.1 Experimental setup

Figure 4.4 shows the experimental setup. The AVT generated by the AVT-Learner are evaluated by comparing the performance of the Naive Bayes Classifiers produced by applying

- NBL to the original data set
- AVT-NBL to the original data set (See Figure 4.4).

For the benchmark data sets, we chose 37 data sets from UCI data repository [5].

Among the data sets we have chosen, AGARICUS-LEPIOTA data set and NURSERY data set have AVT supplied by human experts. AVT for AGARICUS-LEPIOTA data was prepared by a botanist, and AVT for NURSERY data was based on our understanding of the domain. We are not aware of any expert-generated AVTs for other data sets.

In each experiment, we randomly divided each data set into 3 equal parts and used 1/3 of the data for AVT construction using AVT-Learner. The remaining 2/3 of the data were used for generating and evaluating the classifier. Each set of AVTs generated by the AVT-Learner

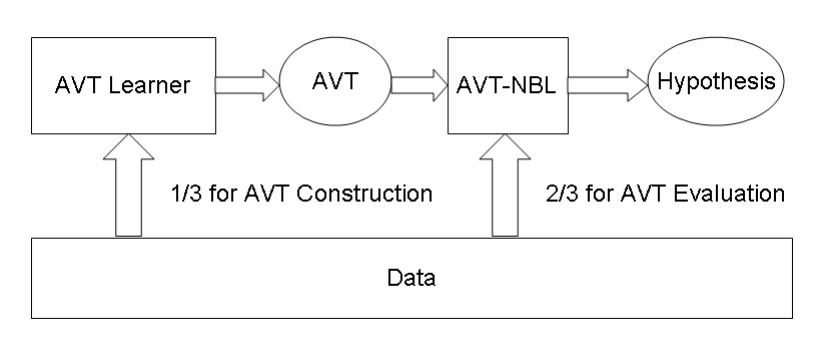



Figure 4.4 Evaluation of AVT using AVT-NBL

was evaluated in terms of the error rate and the size of the resulting classifiers (as measured by the number of entries in conditional probability tables). The error rate and size estimates were obtained using 10-fold cross-validation on the part of the data set (2/3) that was set aside for evaluating the classifier. The results reported correspond to averages of the 10-fold cross-validation estimates obtained from the three choices of the AVT-construction and AVT-evaluation. This process ensures that there is no information leakage between the data used for AVT construction, and the data used for classifier construction and evaluation.

10-fold cross-validation experiments were performed to evaluate human expert-supplied AVT on the AVT evaluation data sets used in the experiments described above for the AGARICUS-LEPIOTA data set and the NURSERY data set.

We also evaluated the robustness of the AVT generated by the AVT-Learner by using them to construct classifiers from data sets with varying percentages of missing attribute values. The data sets with different percentages of missing values were generated by uniformly sampling from instances and attributes to introduce the desired percentage of missing values.

### 4.4.2 Results

**AVT generated by AVT-Learner are competitive with human-generated AVT when used by AVT-NBL.** The results of our experiments shown in Figure 4.5 indicate that AVT-Learner is effective in constructing AVTs that are competitive with human expert-supplied AVTs for use in classification tasks with respect to the error rates and the size of the

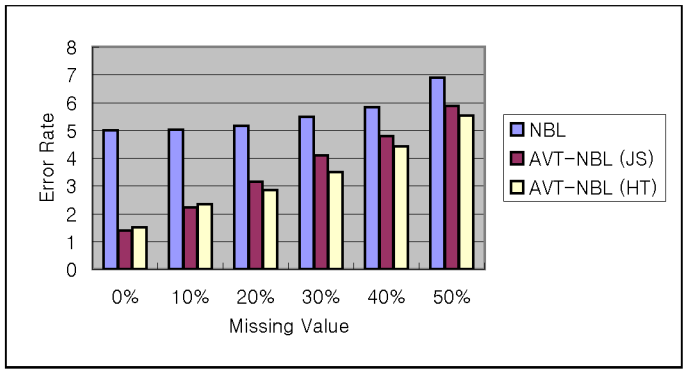



Figure 4.5 The estimated error rates of classifiers generated by NBL and AVT-NBL on AGARICUS-LEPIOTA data with different percentages of missing values. HT stands for human-supplied AVT. JS denotes AVT constructed by AVT-Learner using Jensen-Shannon divergence.

resulting classifiers.

**AVT-Learner can generate useful AVT when no human-generated AVT are available.** For most of the data sets, there are no human-supplied AVT's available. Figure 4.6 shows the error rate estimates for Naive Bayes classifiers generated by AVT-NBL using AVT generated by the AVT-Learner and the classifiers generated by NBL applied to the DERMATOLOGY data set. The results shown suggest that AVT-Learner, using Jensen-Shannon divergence, is able to generate AVTs that when used by AVT-NBL, result in classifiers that are more accurate than those generated by NBL.

Additional experiments with other data sets produced similar results. Table 4.1 shows the classifier's accuracy on original UCI data sets for NBL and AVT-NBL that uses AVTs generated by AVT-Learner. 10-fold cross-validation is used for evaluation, and Jensen-Shannon divergence is used for AVT generation. The user-specified number for discretization is 10.

Thus, AVT-Learner is able to generate AVTs that are useful for constructing compact and accurate classifiers from data.

**AVT generated by AVT-Learner, when used by AVT-NBL, yield substantially more compact Naive Bayes Classifiers than those produced by NBL** Naive Bayes

| Data | NBL | AVT-NBL |
|---|---|---|
| Anneal | 86.3029 | 98.9978 |
| Audiology | 73.4513 | 76.9912 |
| Autos | 56.0976 | 86.8293 |
| Balance-scale | 90.4 | 91.36 |
| Breast-cancer | 71.6783 | 72.3776 |
| Breast-w | 95.9943 | 97.2818 |
| Car | 85.5324 | 86.169 |
| Colic | 77.9891 | 83.4239 |
| Credit-a | 77.6812 | 86.5217 |
| Credit-g | 75.4 | 75.4 |
| Dermatology | 97.8142 | 98.0874 |
| Diabetes | 76.3021 | 77.9948 |
| Glass | 48.5981 | 80.8411 |
| Heart-c | 83.4983 | 87.1287 |
| Heart-h | 83.6735 | 86.3946 |
| Heart-statlog | 83.7037 | 86.6667 |
| Hepatitis | 84.5161 | 92.9032 |
| Hypothyroid | 95.281 | 95.7847 |
| Ionosphere | 82.6211 | 94.5869 |
| Iris | 96 | 94.6667 |
| Kr-vs-kp | 87.8911 | 87.9224 |
| Labor | 89.4737 | 89.4737 |
| Letter | 64.115 | 70.535 |
| Lymph | 83.1081 | 84.4595 |
| Mushroom | 95.8272 | 99.5938 |
| Nursery | 90.3241 | 90.3241 |
| Primary-tumor | 50.1475 | 47.7876 |
| Segment | 80.2165 | 90 |
| Sick | 92.6829 | 97.8261 |
| Sonar | 67.7885 | 99.5192 |
| Soybean | 92.9722 | 94.5827 |
| Splice | 95.3605 | 95.768 |
| Vehicle | 44.7991 | 67.8487 |
| Vote | 90.1149 | 90.1149 |
| Vowel | 63.7374 | 42.4242 |
| Waveform-5000 | 80 | 65.08 |
| Zoo | 93.0693 | 96.0396 |

Table 4.1 Accuracy of NBL and AVT-NBL on UCI data sets

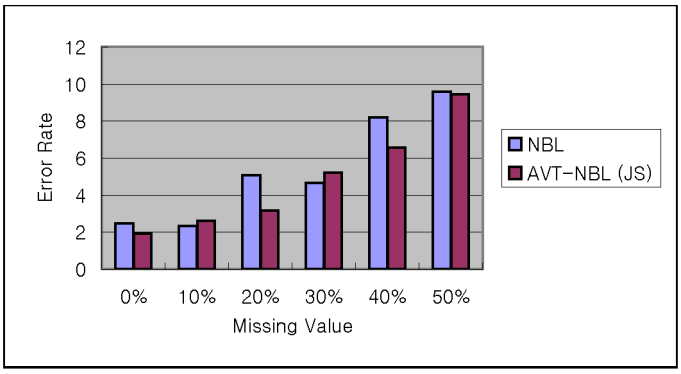



Figure 4.6 The error rate estimates of the Standard Naive Bayes Learner (NBL) compared with that of AVT-NBL on DERMATOLOGY data. JS denotes AVT constructed by AVT-Learner using Jensen-Shannon divergence.

classifiers constructed by AVT-NBL generally have smaller number of parameters than those from NBL (See Figures 4.7 for representative results). Table 4.2 shows the classifier size measured by the number of parameters on selected UCI data sets for NBL and AVT-NBL that uses AVTs generated by AVT-Learner.

These results suggest that AVT-Learner is able to group attribute values into AVT in such a way that the resulting AVT, when used by AVT-NBL, result in compact yet accurate classifiers.

## 4.5 Summary and discussion

### 4.5.1 Summary

In many applications of data mining, there is a strong preference for classifiers that are both accurate and compact [19, 24]. Previous work has shown that attribute value taxonomies can be exploited to generate such classifiers from data [40, 41]. However, human-generated AVTs are unavailable in many application domains. Manual construction of AVTs requires a great deal of domain expertise, and in case of large data sets with many attributes and many values for each attribute, manual generation of AVTs is extremely tedious and hence not feasible in practice. Against this background, we have described in this paper, AVT-Learner, a simple algorithm for automated construction of AVT from data. AVT-Learner recursively groups values of attributes

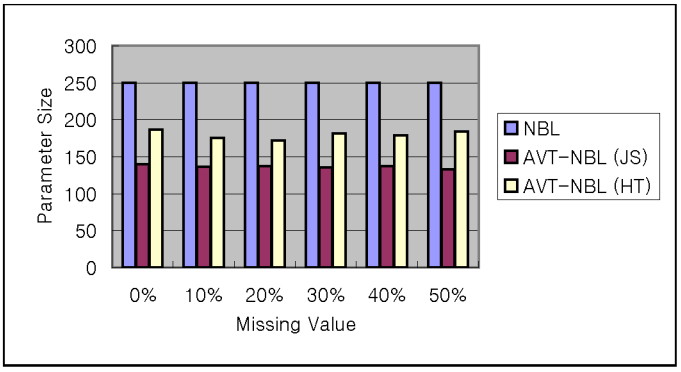



Figure 4.7 The size (as measured by the number of parameters) of the Standard Naive Bayes Learner (NBL) compared with that of AVT-NBL on AGARICUS-LEPIOTA data. HT stands for human-supplied AVT. JS denotes AVT constructed by AVT-Learner using Jensen-Shannon divergence.

| Data | NBL | AVT-NBL |
|---|---|---|
| Audiology | 3720 | 3600 |
| Breast-cancer | 104 | 62 |
| Car | 88 | 80 |
| Dermatology | 906 | 540 |
| Kr-vs-kp | 150 | 146 |
| Mushroom | 252 | 124 |
| Nursery | 140 | 125 |
| Primary-tumor | 836 | 814 |
| Soybean | 1919 | 1653 |
| Splice | 864 | 723 |
| Vote | 66 | 66 |
| Zoo | 259 | 238 |

Table 4.2 Parameter size of NBL and AVT-NBL on selected UCI data sets

| Data | 2-ary | 3-ary | 4-ary |
|---|---|---|---|
| Nursery | 90.3241 | 90.3241 | 90.3241 |
| Audiology | 76.9912 | 76.5487 | 76.9912 |
| Car | 86.169 | 86.169 | 86.169 |
| Dermatology | 98.0874 | 97.541 | 97.541 |
| Mushroom | 99.5938 | 99.7292 | 99.7538 |
| Soybean | 94.5827 | 94.4363 | 94.4363 |

Table 4.3 Accuracy of NBL and AVT-NBL for k-ary AVT-Learner

based on a suitable measure of divergence between the class distributions associated with the attribute values to construct an AVT. AVT-Learner is able to generate hierarchical taxonomies of nominal, ordinal, and continuous valued attributes. The experiments reported in this paper show that:

- AVT-Learner is effective in generating AVTs that when used by AVT-NBL, a principled extension of the standard algorithm for learning Naive Bayes classifiers, result in classifiers that are substantially more compact (and often more accurate) than those obtained by the standard Naive Bayes Learner (that does not use AVTs).
- The AVTs generated by AVT-Learner are competitive with human supplied AVTs (in the case of benchmark data sets where human-generated AVTs were available) in terms of both the error rate and size of the resulting classifiers.

### 4.5.2 Discussion

The AVTs generated by AVT-Learner are binary trees. Hence, one might wonder if k-ary AVTs yield better results when used with AVT-NBL. Figure 4.8 shows an AVT of 'odor' attribute generated by AVT-Learner (with quaternary clustering). Table 4.3 shows the accuracy of AVT-NBL when k-ary clustering is used by AVT-Learner. It can be seen that AVT-NBL generally works best when binary AVTs are used. It is because reducing internal nodes in AVT-Learner will eventually reduce the search space for possible cuts in AVT-NBL, which leads to generating a less compact classifier.

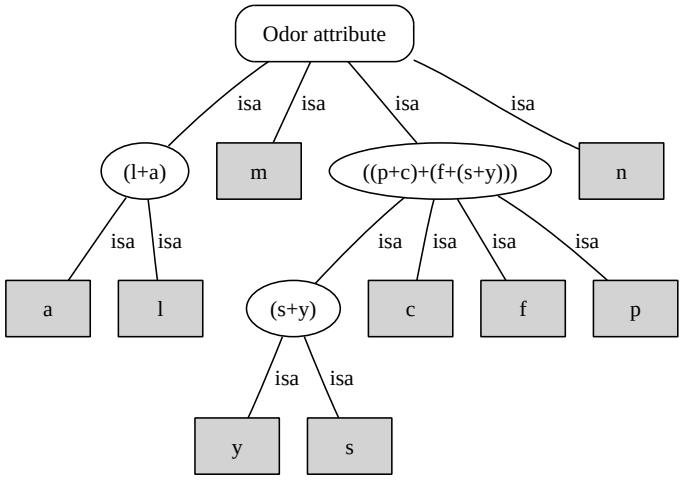



Figure 4.8 AVT of 'odor' attribute of UCI AGARICUS-LEPIOTA mushroom data set generated by AVT-Learner using Jensen-Shannon divergence (with quaternary clustering)

### 4.5.3 Related work

Gibson and Kleinberg [12] introduced STIRR, an iterative algorithm based on non-linear dynamic systems for clustering categorical attributes. Ganti et. al. [11] designed CACTUS, an algorithm that uses intra-attribute summaries to cluster attribute values. However, both of them did not make taxonomies and use the generated for improving classification tasks. Pereira et. al. [25] described distributional clustering for grouping words based on class distributions associated with the words in text classification. Yamazaki et al., [37] described an algorithm for extracting hierarchical groupings from rules learned by FOCL (an inductive learning algorithm) [23] and reported improved performance on learning translation rules from examples in a natural language processing task. Slonim and Tishby [29, 30] described a technique (called the agglomerative information bottleneck method) which extended the distributional clustering approach described by Pereira et al. [25], using Jensen-Shannon divergence for measuring distance between document class distributions associated with words and applied it to a text classification task. Baker and McCallum [3] reported improved performance on text classification using a technique similar to distributional clustering and a distance measure, which upon closer examination, can be shown to be equivalent to Jensen-Shannon divergence [29].

To the best of our knowledge, there has been little work on the evaluation of techniques for generating hierarchical groupings of attribute values (AVTs) on classification tasks using a broad range of benchmark data sets using algorithms such as AVT-DTL or AVT-NBL that are capable of exploiting AVTs in learning classifiers from data.

### 4.5.4 Future work

Some directions for future work include:

- Extending AVT-Learner described in this paper to learn AVTs that correspond to tangled hierarchies (which can be represented by directed acyclic graphs (DAG) instead of trees).
- Learning AVT from data for a broad range of real world applications such as census data analysis, text classification, intrusion detection from system log data [15], learning classifiers from relational data [2], and protein function classification [36] and identification of protein-protein interfaces [38].
- Developing algorithms for learning hierarchical ontologies based on part-whole and other relations as opposed to ISA relations captured by an AVT.
- Developing algorithms for learning hierarchical groupings of values associated with more than one attribute.

## 4.6 Postscript

Since the original publication of this paper additional work has been done in terms of using Taxonomies generated by the algorithm described here in other learning setups [39] [35] [18].

## Bibliography

[1] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock.

Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25(1):25–29, 2000.

[2] A. Atramentov, H. Leiva, and V. Honavar. A multi-relational decision tree learning algorithm - implementation and experiments. In T. Horváth and A. Yamamoto, editors, *Proceedings of the 13th International Conference on Inductive Logic Programming (ILP 2003). Vol. 2835 of Lecture Notes in Artificial Intelligence : Springer-Verlag*, pages 38–56, 2003.

[3] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM Press, 1998.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 0:0, May 2001.

[5] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[6] J. Burbea and C.R. Rao. Entropy differential metric, distance and divergence measures in probability spaces: A unified approach. *J. Multi. Analysis*, 12:575–596, 1982.

[7] J. Burbea and C.R. Rao. On the convexity of some divergence measures based on entropy functions. *IEEE Trans. on Inform. Theory*, IT-28:489–495, 1982.

[8] V. Dhar and A. Tuzhilin. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):926–938, 1993.

[9] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[10] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, 1997.

[11] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus - clustering categorical data using summaries. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 73–83. ACM Press, 1999.

[12] D. Gibson, J.M. Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. *VLDB Journal: Very Large Data Bases*, 8(3–4):222–236, 2000.

[13] J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhr Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AIII Press/MIT Press, 1996.

[14] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial intelligence*, 36:177 – 221, 1988.

[15] G. Helmer, J.S.K. Wong, V.G. Honavar, and L. Miller. Automated discovery of concise predictive rules for intrusion detection. *J. Syst. Softw.*, 60(3):165–175, 2002.

[16] J. Hendler, K. Stoffel, and M. Taylor. Advances in high performance knowledge representation. *Technical Report CS-TR-3672*, University of Maryland Institute for Advanced Computer Studies Dept. of Computer Science, 1996.

[17] H. Jeffreys. An invariant form for the prior probability in estimation procedures. In *Proceedings of the Royal Society, London, Ser. A, 186*, pages 453–461, London, UK, 1946.

[18] D-K. Kang, J. Zhang, A. Silvescu, and V. Honavar. Multinomial event model based abstraction for sequence and text classification. In *In: Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA 2005)*, 2005.

[19] R. Kohavi and F. Provost. Applications of data mining to electronic commerce. *Data Min. Knowl. Discov.*, 5(1-2):5–10, 2001.

[20] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22:79–86, 1951.

[21] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.

[22] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[23] M. Pazzani and D. Kibler. The role of prior knowledge in inductive learning. *Machine Learning*, 9:54–97, 1992.

[24] M.J. Pazzani, S. Mani, and W.R. Shankle. Beyond concise and colorful: Learning intelligible rules. In *Knowledge Discovery and Data Mining*, pages 235–238, 1997.

[25] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *31st Annual Meeting of the ACL*, pages 183–190, 1993.

[26] J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[27] J.S. Schlimmer. Concept acquisition through representational adjustment. *Technical Report 87-19*, Department of Information and Computer Science, University of California, 1987. Doctoral dissertation.

[28] R. Sibson. Information radius. *Z. Wahrs. und verw Geb.*, 14:149–160, 1969.

[29] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Proceedings of the 13th Neural Information Processing Systems (NIPS 1999 )*, 1999.

[30] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–215. ACM Press, 2000.

[31] I.J. Taneja. New developments in generalized information measures. *Advances in Imaging and Electron Physics*, 91:37–135, 1995.

[32] M. Taylor, K. Stoffel, , and J. Hendler. Ontology based induction of high level classification rules. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[33] F. Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46:1602–1609, 2000.

[34] J.L. Undercoffer, A. Joshi, T. Finin, and J. Pinkston. A Target Centric Ontology for Intrusion Detection: Using DAML+OIL to Classify Intrusive Behaviors. *Knowledge Engineering Review*, 0:0, January 2004.

[35] F. Vasile, A. Silvescu, D.-K. Kang, and V. Honavar. Tripper: Rule learning using taxonomies. In *Proceedings of the Tenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006)*, 2006.

[36] X. Wang, D. Schroeder, D. Dobbs, and V.G. Honavar. Automated data-driven discovery of motif-based protein function classifiers. *Inf. Sci.*, 155(1-2):1–18, 2003.

[37] T. Yamazaki, M.J. Pazzani, and C.J. Merz. Learning hierarchies from ambiguous natural language data. In *International Conference on Machine Learning*, pages 575–583, 1995.

[38] C. Yan, D. Dobbs, and V. Honavar. Identification of surface residues involved in protein-protein interaction – a support vector machine approach. In A. Abraham, K. Franke, and M. Koppen, editors, *Intelligent Systems Design and Applications (ISDA-03)*, pages 53–62, 2003.

[39] J. Zhang, D.-K. Kang, A. Silvescu, and V. Honavar. Learning accurate and concise naive bayes classifiers from attribute value taxonomies and data. *Knowledge and Information Systems*, 9(2):157–179, 2006.

[40] J. Zhang and V. Honavar. Learning decision tree classifiers from attribute value taxonomies and partially specified data. In *the Twentieth International Conference on Machine Learning (ICML 2003)*, Washington, DC, 2003.

[41] J. Zhang and V. Honavar. AVT-NBL: An algorithm for learning compact and accurate naive bayes classifiers from attribute value taxonomies and data. In *International Conference on Data Mining (ICDM 2004)*, 2004. To appear.

[42] J. Zhang and V. Honavar. Learning naive bayes classifiers from attribute value taxonomies and partially specified data. In *International Conference on Intelligent System Design and Applications (ISDA 2004)*, 2004.

[43] J. Zhang, A. Silvescu and V. Honavar. Ontology-driven induction of decision trees at multiple levels of abstraction. In *Proceedings of Symposium on Abstraction, Reformulation, and Approximation 2002. Vol. 2371 of Lecture Notes in Artificial Intelligence: Springer-Verlag*, 2002.

# CHAPTER 5. ABSTRACTION SUPERSTRUCTURING NORMAL FORMS

Modified from a paper to be submitted to *Information and Computation*

Adrian Silvescu

## Abstract

Induction is the process by which we obtain laws (or more encompassing - theories) about the world. A theory can be viewed as consisting of essentially two aspects: a structural and a numeric one. In this paper we are concerned with the structural aspect of theories, which characterizes the variables involved in the theory and their relationships. A priori there are an infinite number of types of possible structurral laws that can be postulated as holding in a theory . We ask the question whether we can find a finite and minmalistic set of structural elements in terms of which any theory can be decomposed. We identify Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into a unit topologically close entities - in particular spatio-temporally close) as the essential structural elements in the induction process. We shown that only two more structural elements are needed in order to obtain the full blown capacity of Turing Machines - the dual versions: Reverse Abstraction and Reverse SuperStructuring. We explore the implications of this theorem about the nature of hidden variables and radical positivism. We also show that our main theorem can be seen as a proof, under the Computationalistic Assumption (a.k.a. Church-Turing thesis), of a more than 200 years old claim of the philosopher David Hume about the principles of connexion among ideas.

## 5.1 Introduction

Induction is the process by which we obtain laws (or more encompassing - theories) about the world. The development of a "logic" of induction has been a long standing desideratum of philosophers, scientists, statisticians and AI researchers ever since the very inception of their respective fields of inquiry. Such a logic is supposed to provide a "consistent and complete" methodology on how to derive laws / theories pertaining to a certain world, based on a finite (but potentially unbounded) set of interactions with it. While the logic of induction itself is desirable to be "consistent and complete", the laws and theories produced as a result of the induction process at any point in time, based on a finite set of interactions, need not necessarily be a "consistent and / or complete" theory of the world to which they pertain. This would be too ambitious. Rather, a more realistic requirement is that the theories are to be consistent and complete in the limit and also in a probabilistic sense. More exactly, given increasingly more experimental evidence and not terribly unlikely samples, the induction process is increasingly more likely to rule out inconsistent laws if they have been postulated and increasingly more likely to postulate the existence of a consistent law that has not been postulated already. Furthermore, the laws need not be absolute statements but can rather be statements about the probability that a certain statement is going to hold.

The nature of the laws involved in a theory can be viewed as consisting of essentially two aspects: the qualitative and the quantitative aspect, respectively. The qualitative aspect is mainly concerned with the entities among which such a law is postulated and the nature of their interrelationship. In philosophical parlance, the qualitative aspect of laws and theories is concerned with ontology. The quantitative aspect, on the other hand, is usually the combination of two sub-aspects, the numeric and the probabilistic one. The numeric part is concerned with the quantities involved in the law, while the probabilistic one is concerned with the uncertainties associated with the law. Very often these two sub-aspects are entangled, for example in the simple case of the normal distribution the mean stands for the numeric sub-aspect of a certain quantity, while the variance represents the uncertainty sub-aspect.

Once the qualitative aspect of a certain law is established then the quantitative (numer-

ical + uncertainty) aspect becomes the subject of experimental science and statistics. The process which produces the qualitative aspect of laws (i.e., ontology learning), is a somewhat less developed field. A solution to the inductive problem however, requires addressing both the qualitative and quantitative aspect. In other terms, all aspects of hypothesis generation and hypothesis testing should be coped with. While hypothesis testing is nicely handled by statistics, the hypothesis generation process and especially its qualitative aspect are in need of more investigation.

In this paper we focus on the qualitative part of a theory, namely ontology. We attempt to identify a set of fundamental structural elements in terms of which the structure of a theory can be decomposed and whose invention should be targeted by the structural induction process. In the general case there is an infinity of possible types of laws that we can postulate as holding within a theory of the world. We seek a finite and minimalistic set of fundamental structural elements that will allow us to express any set of laws in terms of them. Such a set will render induction more tractable as at any step we will have to pick from a finite set of possible constructs rather than an infinite one.

Solutions to the full blown induction problem have already been proposed, under the Computationalistic Assumption, in [16], [17], [15], [7]. Computationalism (a.k.a. Church-Turing thesis) assumes that any theory can be described by a Turing machine, and quite convincing arguments have been made in this direction [18], [2] and references therein. In this paper we also embrace the Computationalistic Assumption as the above-mentioned authors. However, rather than using the Turing machine in order to investigate the problem of induction we will use the alternative, yet equivalent, mechanism of generative grammars. Such a change is motivated by the relative opacity of Turing machines from the structural point of view. We desire a model that decomposes into parts and where the final outcome can be viewed as a composition of these parts. For such a purpose generative grammars are a good candidate, but not the only one. For example lambda calculus is a possible alternative which is Turing equivalent while being compositional at the same time (see [10] for some work in this direction). While different on the surface, various compositional models are in essence very similar and the obtained

results in one framework can easily be ported among them, therefore we will limit ourselves to the case of generative grammars.

The rest of the paper is organized as follows: First we examine some of the intuitions behind the structural induction process. Then we identify the fundamental structural elements for various classes of grammars yielding a set of normal form theorems whose persistent elements are Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into a unit topologically close entities - in particular spatio-temporally close). We then proceed with a discussion and interpretation of the theoretical results in a larger context along with a few more interesting theoretical results. Finally, we conclude.

## 5.2 Compositionality and Structural Induction

Induction aims at finding patterns (laws) from a stream of observations, sometimes lumped together into more encompassing theories. For the sake of simplicity we will only address the problem of observational studies in this paper, and set aside for future the more general problem where actions are also available to the agent. One way to encode the knowledge that is extracted from the stream of observations is to use a Turing machine. Therefore a way to solve the induction problem is to enumerate all the Turing machines (dovetailing in order to cope with the countable infinity) and pick one that strikes a good balance between the predictability (of the finite experience stream) and size [16], [7]. An alternative to picking a certain model as our current model of the world is the Bayesian paradigm which attempts to use all models along with a weight that measures, again, the trade-off between the predictability (of the finite experience stream) and size of the model [7]. In the Bayesian paradigm, prediction is done by take a weighted vote among the predictions of all the models with their respective weights.

However, by examining the way people try to address the problem of induction it seems that they tend to invent an increasing set of concepts which serve as stepping stones in the building of increasingly more complicated theories and concepts. This buildup of increasing complexity by gradually aggregating a few "less complex" entities into more complicated ones is called compositionality. In contrast to the Turing Machines enumeration approach to in-

duction the compositional approach exhibits at least two advantages: better transparency and computational speedup. Transparency is due to the decomposability aspect while computational speedup is obtained by the possibility to reuse already made "less complex" entities when building the bigger ones as opposed to starting afresh every time as it is the case with the Turing Machines enumeration. In quite a general sense compositionality is the key to turning an Exhaustive Enumeration solution into a Dynamic Programming one. Another important advantage of compositionality, from the point of view of induction is that it allows for a more data driven approach. Exhaustive Enumeration in the inductive setup is basically a Generate and Test kind of approach where we take the effort to generate the a full blown hypothesis, test it on the data and then repeat the process for zero. Having a compositional model allows us to assess the usefulness of various parts only once and and the reuse them in bigger components much like in Dynamic Programming or its heuristics based generalization - $A^*$.

Structural Induction can be characterized mainly along the following lines: We start with a stream of observations and aim at finding patterns (laws) which "explain" it, which we potentially collated into more encompassing theories. The patterns found are not necessarily described solely in terms of the input observations, but may also use additional "internal variables". The role of these internal variables is to simplify explanation. The observables and the internal variables in terms of which the explanation is offered constitute the ontology[1] - i.e., the set of concepts and their interrelationships deemed relevant for the agent's theorizing about its experiences. Structural induction aims at finding appropriate internal variables and patterns among them - or equivalently learning (inducing) the ontology. Each found pattern can be in turn labeled as an "internal variable" thus bootstrapping the process. New patterns can then in turn be formed out of these internal variables which stand for other patterns, thus allowing to build increasingly more complicated patterns while maintaining local simplicity. Each pattern

[1]The ontology in this case does not have any claims of universality as it is often the case in the philosophical literature. Rather is just a set of concepts and interrelations among them that afford the expression of theories. These concepts may change during the life of the agent and they are void of any metaphysical claims. Thus our ontology is situated at the epistemological rather than the metaphysical level. A more appropriate name for it would probably be epiology in order to distinguish it from the philosophical usage which implies a metaphysical level. However in the Artificial Intelligence literature the term is mostly used in the epistemological sense anyways, even though seldom explicitly stated, and this is the way we will use it too.

is formed out of a small [2] set of variables which can stand in turn for more complicated patterns and so on and so forth. Such bootstrapping allows for the buildup of indeed very complicate patterns based on very simple rules. (e.g., - fractals generated by rewriting systems).

The invention of internal variables to aid the explanation process is not without perils. Arbitrary invention may lead to internal variables such as the one which stands for the truth value of the sentence: "In heaven, if it rains, do the angels get wet or not?". Or the perhaps the even more famous, yet quantitative: "How many angels can fit on the head of a pin?". Sentences of these type, while perfectly valid from the syntactic point of view are problematic when we attempt to determine their truth or quantitative values respectively, based on the experience data. The need to expose these types of problematic contraptions led to the demarcation problem in philosophy at the beginning of the twentieth century (e.g., [12]). The main intuition behind a demarcation criterion between "non-sensical" internal variables such as the above-mentioned and "reasonable" ones is that: If the world will look the same from the point of view of the set of potential experiences regardless of the value of the internal variable then the variable is non-sensical (or in other worlds independent of the data - "senses").

One solution for the demarcation criteria calls for tagging as non-scientific all sentences / "internal variables" , that do not have any direct connection to experience. This is a radical interpretation of an idea that runs back in the history of Philosophy from Positivism through the Empiricists and Scholastics down to at least to Aristotle's *"Nihil est in intellectu quod non prius fuerit in sensu"*[3]. The direct connection requirement restricted the no-nonsense theories to those formed out empirical laws [1] (i.e, laws that relate only measurable quantities) . However a good deal of scientists including Albert Einstein, while sympathetic to the positivists ideas, were successfully using hidden variables in their theories, which have only an indirect connection to observations. Such odds led to many revisions of the positivists doctrine culminating with Carnap's full blown re-introduction of hidden variables - in An Introduction to the Philosophy of Science [3]. Hidden variables allow for more compact explanations being able sometimes

[2] or at the very least finite

[3] There is nothing in the mind that was not previously in the senses - this is also known as the *tabula rasa* or *blank slate* hypothesis

even to turn an infinite representation into a finite one as it is the case with Hidden Markov Models (HMMs) versus Markov Models of depth k (MM(k)), or in the deterministic case Finite State Machines versus Finite History Lookup Tables.

In the remainder of this paper we will use the Turing equivalent, yet compositional, model of Generative Grammars. We show that in order for our models of the world to be Turing equivalent we need to allow for hidden variables. Furthermore based on our characterization theorems we identify the two fundamental structural elements that postulate hidden variables: Reverse Abstraction and Reverse SuperStructuring. These two along with their direct duals Abstraction and SuperStructuring, will be enough to characterize the full-blown Turing equivalent formalism of Generative Grammars.

## 5.3 Abstraction SuperStructuring Normal Forms

In this section we prove the main results of the paper: a series of characterization theorems of two main classes of Generative Grammars: Context-Free and General Grammars, in terms of a small set of fundamental structural elements. We start by recapitulating the definitions and notations for generative grammars and the theorem that claims the equivalence between Generative Grammars and Turing Machines. We then draw the connections between the induction process and the formalism of generative grammars and examine the main motivation for trying to find a minimalistic set of fundamental structural elements. Finally, we prove the main results and make some remarks along the way.

### 5.3.1 Generative Grammars and Turing Machines

**Definitions (Grammar)** A (generative) grammar is a quadruple $(N, T, S, R)$ where $N$ and $T$ are disjoint finite sets called NonTerminals and Terminals, respectively, $S$ is a distinguished element from $N$ called the start symbol and $R$ is a set of rewrite rules (a.k.a. production rules) of the form $(l \rightarrow r)$ where $l \in (N \cup T)^* N (N \cup T)^*$ and $r \in (N \cup T)^*$. Additionally, we call $l$ the left hand side (lhs) and $r$ the right hand side (rhs) of the rule $(l \rightarrow r)$. The language generated by a grammar is defined by $L(G) = \{w \in T^* | S \stackrel{*}{\rightarrow} w\}$ where $\stackrel{*}{\rightarrow}$ stands for the

reflexive transitive closure of the rules from $R$. Furthermore $\stackrel{+}{\rightarrow}$ stands for the transitive (but not reflexive) closure of the rules from $R$. We say that two grammars $G$,$G'$ are equivalent if $L(G) = L(G')$. The steps contained in a set of transitions $\alpha \stackrel{*}{\rightarrow} \beta$ is called a derivation. If we want to distinguish between derivations in different grammars we will write $\alpha \stackrel{*}{\rightarrow}_G \beta$ or mention it explicitly. We denote by $\epsilon$ the empty string in the language. We will sometimes use the shorthand notation $l \rightarrow r_1|r_2|...|r_n$ to stand for the set of rules $\{l \rightarrow r_i\}_{i=1,n}$. See e.g., [13] for more details and examples.

**Definition (Grammar Types)** Let $G = (N, T, S, R)$ be a grammar. Then

1. G is a **regular grammar (REG)** if all the rules $(l \rightarrow r) \in R$ have the property that $l \in N$ and $r \in (T^* \cup T^*N)$.

2. G is **context-free grammar (CFG)** if all the rules $(l \rightarrow r) \in R$ have the property that $l \in N$.

3. G is **context-sensitive grammar (CSG)** if all the rules $(l \rightarrow r) \in R$ have the property that they are of the form $\alpha A\beta \rightarrow \alpha\gamma\beta$ where $A \in N$ and $\alpha, \beta, \gamma \in (N \cup T)^*$ and $\gamma \neq \epsilon$. Furthermore if $\epsilon$ is an element of the language one rule of the form $S \rightarrow \epsilon$ is allowed and furthermore the restriction that $S$ does not appear in the right hand side of any rule is imposed. We will call this last sentence the $\epsilon - Ammendment$.

4. G is **general grammar (GG)** if all the rules $(l \rightarrow r) \in R$ have no additional restrictions.

**Theorem 0.** *The set of General Grammars are equivalent in power with the set of Turing Machines. That is, for every Turing Machine T there exists a General Grammar G such that* $L(G) = L(T)$ *and viceversa.*

*Proof.* The proof of this theorem is a well known result. See for example [13]. □

Because of this theorem we can concentrate on the Generative Grammar formalism which is compositional as opposed to using the rather opaque framework of Turing Machines. Similar results of equivalence exists for transductive[4] versions of Turing machines and grammars as

[4]where an output string $y$ is computed as a function of the input string $w$ rather than just attempting to recognize whether $w$ belongs to the language

opposed to the recognition versions outlined in this subsection, see e.g., [2] and references therein. For the sake of simplicity, but without loss of generality - as the results are easily portable, we will concentrate on the recognition setup rather than the transductive one.

### 5.3.2 Structural Induction, Generative Grammars and Motivation

Before proceeding with the main results of the paper we examine the connections between the generative grammars setup and our structural induction problem. The terminals in the grammar formalism denote the set of observables in our induction problem. The NonTerminals stand for internal variables in terms of which the observations (terminals) are explained. The "explanation" is given by a derivation of the stream of observations from the initial symbol $S \stackrel{*}{\rightarrow} w$. The NonTerminals that appear in the derivation are the internal variables in terms of which the surface structure given by the stream of observations $w$ is explained. Given such a correspondence, the task of Structural Induction is to find an appropriate set of NonTerminals $N$ and a set of rewrite rules $R$ that will allow us to derive (explain) the input stream of observations $w$ from the initial symbol $S$. The process of Structural Induction may invent a new rewrite rule $l \rightarrow r$ under certain conditions and this new rule may contain in turn new NonTerminals (internal variables) which are added to the already existing ones. The common intuition is that $l$ is a simpler version of $r$, as the final goal is to reduce $w$ to $S$. The terminals constitute the input symbols (standing for observables), the NonTerminals constitute whatever additional "internal" variables we may feel are needed, the rewrite rules describe their interrelationship and altogether they constitute the ontology.

The correspondence between the terms used in Structural Induction and Generative Grammars is outlined in Table 5.1.

While Structural Induction may invent any kind of rewrite rule of the form $l \rightarrow r$, potentially introducing new NonTerminals, the problem is that there are infinitely many such rules that we could invent at any point in time. In order to make the process more well defined we ask whether it is possible to find a set of fundamental structural elements which is finite and minimalistic, and if possible quite intuitive such that all the rules (or more precisely sets of

| Structural Induction | Generative Grammar |
|---|---|
| Observables | Terminals $T$ |
| Internal Variables | NonTerminals $N$ |
| Law / Theory | production rule(s) $l \rightarrow r$ |
| Ontology | Grammar $G$ |
| Observations Stream | word $w$ |
| Explanation | Derivation $S \overset{*}{\rightarrow} w$ |
| Partial Explanation | Derivation $\alpha \overset{*}{\rightarrow} w$ |

Table 5.1 Correspondence between Structural Induction and Generative Grammars

rules) can be re-expressed in terms of these elements. This would establish a normal form in terms of a finite set of elements and then the problem of generating laws will be reduced to making appropriate choices from this set without the danger of loosing completeness.

In the next subsection we will attempt to decompose the rules $l \rightarrow r$ into a small finite set of fundamental structural elements which will allow us to design better structure search mechanisms. Two structural elements stand out across our various results: Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into one unit topologically close entities).

### 5.3.3 ASNF Theorems

**Issue** ($\epsilon - Construction$). In the rest of the paper we will prove some theorems that impose various sets of conditions on a grammar $G$ in order for the grammar to be considered in a certain Normal Form. If $\epsilon \in L(G)$ however, we will allow two specific rules of the grammar $G$ to be exempted from these constraints and still consider the grammar in the Normal Form. More exactly if $\epsilon \in L(G)$ and given a grammar $G'$ such that $L(G') = L(G \backslash \{\epsilon\})$ and $G' = (N', T, S', R')$ is in a certain Normal Form then the grammar $G = (N \cup \{S\}, T, S, R = R' \cup \{S \rightarrow \epsilon, S \rightarrow S'\})$ where $S \notin N'$ will also be considered in that certain Normal Form despite the fact that the two productions $\{S \rightarrow \epsilon, S \rightarrow S'\}$ may violate the conditions of the Normal Form. These are the only productions that will be allowed to violate the Normal Form conditions. Note that $S$ is a brand new NonTerminal and does not appear in any other productions aside

from these two. Without loss of generality we will assume in the rest of the paper that $\epsilon \notin L(G)$. This is because if $\epsilon \in L(G)$ we can always produce using the above-mentioned construction a grammar $G''$ that is in a certain Normal Form and $L(G'') = L(G')$ from a grammar $G'$ that is in that Normal Form and satisfies $L(G') = L(G \backslash \{\epsilon\})$. We will call the procedure just outlined the $\epsilon - Construction$.

Furthermore we will call the following statement the $\epsilon - Ammendment$: Let $G = (N, T, S, R)$ be a grammar, if $\epsilon$ is an element of the language $L(G)$ one rule of the form $S \rightarrow \epsilon$ is allowed and furthermore the restriction that $S$ does not appear in the right hand side of any rule is imposed.

First we prove a weak form of the Abstraction SuperStructuring Normal Form for Context Free Grammars.

**Theorem 1 (Weak-CFG-ASNF).** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a Context Free Grammar. Then there exists a Context Free Grammar $G'$ such that $L(G) = L(G')$ and $G'$ contains only rules of the following type:*

1. $A \rightarrow B$
2. $A \rightarrow BC$
3. $A \rightarrow a$

*Proof .* Since $G$ is a CFG it can be written in the Chomsky Normal Form [13]. That is, such that it contains only productions of the forms 2 and 3. If $\epsilon \in L(G)$ a rule of the form $S \rightarrow \epsilon$ is allowed and $S$ does not appear in the rhs of any other rule ($\epsilon - Ammendment$). Since we have assumed that $\epsilon \notin L(G)$ we do not need to deal with the $\epsilon - Ammendment$ and therefore we proved the theorem.

□

**Remarks.**

1. We will call the rules of type 1 Renamings (REN).
2. We will call the rules of type 2 SuperStructures (SS) or compositions.

3. The rules of the type 3 are just convenience renamings of observables into internal variables in order to uniformize the notation and we will call them Terminal (TERMINAL).

We are now ready to prove the the Weak ASNF theorem for the general case.

**Theorem 2 (Weak-GEN-ASNF).** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a General (unrestricted) Grammar. Then there exists a grammar $G'$ such that $L(G) = L(G')$ and $G'$ contains only rules of the following type:*

1. $A \to B$
2. $A \to BC$
3. $A \to a$
4. $AB \to C$

*Proof .* Every general grammar can be written in the Generalized Kuroda Normal Form(GKNF) [13]. That is, it contains only productions of the form:

1. $A \to \epsilon$
2. $AB \to CD$
3. $A \to BC$
4. $A \to a$

Assume that we have already converted our grammar in the GKNF. We will prove our claim in 3 steps.

**Step 1**. For each $\{AB \to CD\}$ we introduce a new NonTerminal $X_{AB,CD}$ and the following production rules $\{AB \to X_{AB,CD}\}$ and $\{X_{AB,CD} \to CD\}$ and eliminate the old $\{AB \to CD\}$ production rules. In this way we ensure that all the rules of type 2 in GKNF have been rewritten into rules of types 2 and 4 in the GEN-ASNF.

The new grammar generates the same language. To see this let $oldG = (N_{oldG}, T, S, R_{oldG})$ denote the old grammar and $newG = (N_{newG}, T, S, R_{newG})$ denote the new grammar. Let

$\gamma \in N^+_{oldG}$ and $\gamma \stackrel{*}{\rightarrow} w$ be a derivation in *oldG*. In this derivation we can replace all the uses of the production $AB \rightarrow CD$ with the sequence $AB \rightarrow X_{AB,CD} \rightarrow CD$ and get a derivation that is valid in *newG*, since all the other productions are common between the two grammars. Thus we have proved that for all $\gamma \in N^+_{oldG}$ we can convert a valid derivation from *oldG*, $\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in *newG* and in particular this is true also for $S \stackrel{*}{\rightarrow} w$. Therefore $L(oldG) \subseteq L(newG)$. Conversely, let $\gamma \in N^+_{oldG}$ and $\gamma \stackrel{*}{\rightarrow} w$ be a valid derivation in *newG* then whenever we use the rule $AB \rightarrow X_{AB,CD}$ in this derivation $\gamma \stackrel{*}{\rightarrow} \alpha AB\beta \rightarrow \alpha X_{AB,CD}\beta \stackrel{*}{\rightarrow} w$ let $\gamma \stackrel{*}{\rightarrow} \alpha AB\beta \rightarrow \alpha X_{AB,CD}\beta \stackrel{*}{\rightarrow} \delta X_{AB,CD}\eta \rightarrow \delta CD\eta \stackrel{*}{\rightarrow} w$ be the place where the $X_{AB,CD}$ that occurs between $\alpha$ and $\beta$ is rewritten (used in the lhs of a production, even if it rewrites to the same symbol) for the first time. Then necessarily the $X_{AB,CD} \rightarrow CD$ rule is the one that applies since it is the only one that has $X_{AB,CD}$ in the lhs. Furthermore, as a consequence of Lemma 1 we have that $\alpha \stackrel{*}{\rightarrow} \delta$ and $\beta \stackrel{*}{\rightarrow} \eta$ are valid derivations in *newG*. Therefore we can bring the production $X_{AB,CD} \rightarrow CD$ right before the use of $AB \rightarrow X_{AB,CD}$ as follows $\gamma \stackrel{*}{\rightarrow} \alpha AB\beta \rightarrow \alpha X_{AB,CD}\beta \rightarrow \alpha CD\beta \stackrel{*}{\rightarrow} \delta CD\beta \stackrel{*}{\rightarrow} \delta CD\eta \stackrel{*}{\rightarrow} w$ and still have a valid derivation in *newG*. We can repeat this procedure for all places where rules of the form $AB \rightarrow X_{AB,CD}$ appear. In this modified derivation we can replace all the uses of the sequence $AB \rightarrow X_{AB,CD} \rightarrow CD$ with the production $AB \rightarrow CD$ and obtain a derivation that is valid in *oldG* since all the other productions are common between the two grammars. Thus we have proved that for all $\gamma \in N^+_{oldG}$ we can convert a valid derivation from *newG* ,$\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in *oldG* and in particular this is true also for $S \stackrel{*}{\rightarrow} w$ since $S \in N^+_{oldG}$, and therefore $L(newG) \subseteq L(oldG)$. Therefore we have proved that the grammars are equivalent, i.e., $L(oldG) = L(newG)$.

**Step 2**. Returning to the main argument, so far our grammar has only rules from Weak-GEN-ASNF safe for the $\epsilon$-productions $A \rightarrow \epsilon$. Next we will eliminate $\epsilon$-productions $A \rightarrow \epsilon$ in two steps. First for each $A \rightarrow \epsilon$ we introduce a new NonTerminal $X_{A,E}$ and the productions $X_{A,E} \rightarrow E$ and $E \rightarrow \epsilon$ and eliminate $A \rightarrow \epsilon$, where $E$ is a distinguished new NonTerminal (that will basically stand for $\epsilon$ internally). This insures that we have only one $\epsilon$-production, namely $E \rightarrow \epsilon$ and $E$ does not appear on the lhs of any other production and also that all the

rules that rewrite to $E$ are of the form $A \rightarrow E$.

The new grammar generates the same language. We will use a similar proof technique as in the previous step. Let $oldG = (N_{oldG}, T, S, R_{oldG})$ denote the old grammar and $newG = (N_{newG}, T, S, R_{newG})$ denote the new grammar. Let $\gamma \in N^+_{oldG}$ and $\gamma \stackrel{*}{\rightarrow} w$ be a derivation in $oldG$. In this derivation we can replace all the uses of the production $A \rightarrow \epsilon$ with the sequence $X_{A,E} \rightarrow E \rightarrow \epsilon$ and get a derivation that is valid in $newG$, since all the other productions are common between the two grammars. Thus we have proved that for all $\gamma \in N^+_{oldG}$ we can convert a valid derivation from $oldG$ ,$\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in $newG$ and in particular this is true also for $S \stackrel{*}{\rightarrow} w$, therefore $L(oldG) \subseteq L(newG)$. Conversely, let $\gamma \in N^+_{oldG}$ and $\gamma \stackrel{*}{\rightarrow} w$ be a valid derivation in $newG$ then whenever we use the rule $X_{A,E} \rightarrow E$ in this derivation $\gamma \stackrel{*}{\rightarrow} \alpha X_{A,E}\beta \rightarrow \alpha E\beta \stackrel{*}{\rightarrow} w$ let $\gamma \stackrel{*}{\rightarrow} \alpha X_{A,E}\beta \rightarrow \alpha E\beta \stackrel{*}{\rightarrow} \delta E\eta \rightarrow \delta\eta \stackrel{*}{\rightarrow} w$ be the place where the $E$ that occurs between $\alpha$ and $\beta$ is rewritten(used in the lhs of a production, even if it rewrites to the same symbol) for the first time. Then necessarily the $E \rightarrow \epsilon$ rule is the one that applies since it is the only one that has $E$ in the lhs. Furthermore, as consequence of Lemma 1 we have that $\alpha \stackrel{*}{\rightarrow} \delta$ and $\beta \stackrel{*}{\rightarrow} \eta$ are valid derivations in $newG$. Therefore we can bring the production $E \rightarrow \epsilon$ right before the use of $X_{A,E} \rightarrow E$ as follows $\gamma \stackrel{*}{\rightarrow} \alpha X_{A,E}\beta \rightarrow \alpha E\beta \rightarrow \alpha\beta \stackrel{*}{\rightarrow} \delta\beta \stackrel{*}{\rightarrow} \delta\eta \stackrel{*}{\rightarrow} w$ and still have a valid derivations in $newG$. We can repeat this procedure for all places where rules of the form $X_{A,E} \rightarrow E$ appear. In this modified derivation we can replace all the uses of the sequence $X_{A,E} \rightarrow E \rightarrow \epsilon$ with the production $A \rightarrow \epsilon$ and obtain a derivation that is valid in $oldG$ since all the other productions are common between the two grammars. Thus we have proved that for all $\gamma \in N^+_{oldG}$ we can convert a valid derivation from $newG$, $\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in $oldG$ and in particular this is true also for $S \stackrel{*}{\rightarrow} w$ since $S \in N^+_{oldG}$, and therefore $L(newG) \subseteq L(oldG)$. Therefore we have proved that the grammars are equivalent, i.e., $L(oldG) = L(newG)$.

**Step 3**. To summarize: the new grammar has only rules of the Weak-GEN-ASNF type, safe for the production $E \rightarrow \epsilon$ which is the only production that has $E$ in the lhs and there is no other rule that has $\epsilon$ on the rhs (*strong-uniqueness*) and furthermore the only rules that contain $E$ in the rhs are of the form $A \rightarrow E$ (*only renamings to* $E$).

We will eliminate the $\epsilon$-production $E \rightarrow \epsilon$ as follows: Let $\{A_i \rightarrow E\}_{i=1,n}$ be all the productions that have $E$ on the rhs. For all NonTerminals $X \in N \cup T$ introduce productions $\{XA_i \rightarrow X\}_{i=1,n}$ and $\{A_iX \rightarrow X\}_{i=1,n}$ and eliminate $\{A_i \rightarrow E\}_{i=1,n}$, furthermore we also eliminate $E \rightarrow \epsilon$.

The new grammar generates the same language. We will use a similar proof technique as in the previous step. Let $oldG = (N_{oldG}, T, S, R_{oldG})$ denote the old grammar and $newG = (N_{newG}, T, S, R_{newG})$ denote the new grammar. Let $\gamma \in N^+_{oldG}$ and $\gamma \stackrel{*}{\rightarrow} w$ be a derivation in $oldG$. Then whenever we use a rule of the form $A_i \rightarrow E$ in this derivation $\gamma \stackrel{*}{\rightarrow} \alpha A_i \beta \rightarrow \alpha E \beta \stackrel{*}{\rightarrow} w$ let $\gamma \stackrel{*}{\rightarrow} \alpha A_i \beta \rightarrow \alpha E \beta \stackrel{*}{\rightarrow} \delta E \eta \rightarrow \delta \eta \stackrel{*}{\rightarrow} w$ be the place where the $E$ that occurs between $\alpha$ and $\beta$ is rewritten(used in the lhs of a production, even if it rewrites to the same symbol) for the first time. Then necessarily the $E \rightarrow \epsilon$ rule is the one that applies since it is the only one that has $E$ in the lhs. Furthermore, as a consequence of Lemma 1 we have that $\alpha \stackrel{*}{\rightarrow} \delta$ and $\beta \stackrel{*}{\rightarrow} \eta$ are valid derivations in $oldG$. Therefore we can bring the production $E \rightarrow \epsilon$ right before the use of $A_i \rightarrow E$ as follows $\gamma \stackrel{*}{\rightarrow} \alpha A_i \beta \rightarrow \alpha E \beta \rightarrow \alpha \beta \stackrel{*}{\rightarrow} \delta \beta \stackrel{*}{\rightarrow} \delta \eta \stackrel{*}{\rightarrow} w$ and still have a valid derivations in $oldG$. We can repeat this procedure for all places where rules of the form $A_i \rightarrow E$ appear. In this modified derivation we can replace all the uses of the sequence $A_i \rightarrow E \rightarrow \epsilon$ with the production $XA_i \rightarrow X$ if $\alpha \neq \epsilon$ and $XA_i \rightarrow X$ if $\beta \neq \epsilon$ and obtain a derivation that is valid in $newG$ since all the other productions are common between the two grammars, (e.g., if $\alpha \neq \epsilon$, $\alpha = \alpha_1 X$ replace $\alpha A_i \beta = \alpha_1 X A_i \beta \rightarrow \alpha_1 X E \beta \rightarrow \alpha_1 X \beta = \alpha \beta$ which is valid in $oldG$ with $\alpha A_i \beta = \alpha_1 X A_i \beta \rightarrow \alpha_1 X \beta = \alpha \beta$ which is valid in $newG$ and similarly for $\beta \neq \epsilon$). In this way since all the other productions are common between the two grammars we can convert a derivation $\gamma \rightarrow^*_{oldG} w$ into a derivation $\gamma \rightarrow^*_{newG} w$. Note that it is not possible for both $\alpha$ and $\beta$ to be equal to $\epsilon$ because this will imply that the derivation $\gamma \stackrel{*}{\rightarrow} w$ is $\gamma \stackrel{*}{\rightarrow} A_i \rightarrow E \rightarrow \epsilon$ which contradicts the hypothesis that $w \in T^+$. Thus we have proved that for all $\gamma \in N^+_{oldG}$ we can convert a valid derivation from $oldG$, $\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in $newG$ and in particular this is true also for $S \stackrel{*}{\rightarrow} w$ since $S \in N^+_{oldG}$, and therefore $L(oldG) \subseteq L(newG)$.

Conversely, let $\gamma \in N^+_{oldG}$ and $\gamma \stackrel{*}{\rightarrow} w$ be a derivation in $newG$ . In this derivation we can replace all the uses of the production $XA_i \rightarrow X$ with the sequence $A_i \rightarrow E \rightarrow \epsilon$ and get a

derivation that is valid in *oldG* since all the other productions are common between the two grammars (e.g., we replace $\alpha A_i\beta = \alpha_1 X A_i\beta \rightarrow \alpha_1 X\beta = \alpha\beta$ which is valid in *newG* with $\alpha A_i\beta = \alpha_1 X A_i\beta \rightarrow \alpha_1 X E\beta \rightarrow \alpha_1 X\beta = \alpha\beta$ which is valid in *oldG*). We proceed similarly with the productions of the form $A_i X \rightarrow X$ and replace them with the sequence $A_i \rightarrow E \rightarrow \epsilon$ and get a derivation that is valid in *oldG*. Thus we have proved that for all $\gamma \in N^+_{oldG}$ we can convert a valid derivation from *newG*, $\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in *oldG* and in particular this is true also for $S \stackrel{*}{\rightarrow} w$, therefore $L(newG) \subseteq L(oldG)$. Hence we have proved that the grammars are equivalent, i.e., $L(oldG) = L(newG)$.

□

**Lemma 1.** *Let $G = (N, T, S, R)$ be a grammar and let $\alpha\mu\beta \stackrel{*}{\rightarrow} \delta\mu\eta$, $\mu \in (N \cup T)^+$ a valid derivation in G that does not rewrites the $\mu$ (uses productions whose lhs match any part of $\mu$ even if it rewrites to itself), occurring between $\alpha$ and $\beta$, then $\alpha \stackrel{*}{\rightarrow} \delta$, $\beta \stackrel{*}{\rightarrow} \eta$ are valid derivations in G.*

*Proof.* Because $\alpha\mu\beta \stackrel{*}{\rightarrow} \delta\mu\eta$ does not rewrites any part of $\mu$ (even to itself) it follows that the lhs of any production rule that is used in this derivation either matches a string to the left of $\mu$ or to the right of $\mu$. If we start from $\alpha$ and use the productions that match to the left in the same order as in $\alpha X\beta \stackrel{*}{\rightarrow} \delta X\eta$ then we necessarily get $\alpha \stackrel{*}{\rightarrow} \delta$, a valid derivation in $G$. Similarly, by using the productions that match to the right of $\mu$ we get $\beta \stackrel{*}{\rightarrow} \eta$ valid in $G$.

□

**Remark.**

1. We will call the rules of type 4 Reverse SuperStructuring (RSS).

In the next theorem we will strengthen our results by allowing only the Renamings (REN) to be non unique. First we introduce more exactly what we mean by uniqueness and then we proceed to state and prove a lemma that will allow us to strengthen the Weak-GEN-ASNF by imposing uniqueness on all the productions safe renamings.

**Definition (*strong-uniqueness*).** We will say that a production $\alpha \rightarrow \beta$ respects *strong-uniqueness* if this is the only production that has the property that it has $\alpha$ in the lhs and also this is the only production that has $\beta$ on the rhs.

**Lemma 2.** *Let $G = (N, T, S, R)$, $\epsilon \notin G$ a grammar such that all its productions are of the form:*

1. $A \to B$
2. $A \to \zeta$ , $\zeta \notin N$
3. $\zeta \to B$ , $\zeta \notin N$

*Modify the the grammar into a new grammar $G' = (N', T, S', R')$obtained as follows:*

1. *Introduce a new start symbol $S'$ and the production $S' \to S$.*
2. *For each $\zeta \notin N$ that appears in the rhs of one production in $G$ let $\{A_i \to \zeta\}_{i=1,n}$ all the the productions that contain $\zeta$ in the rhs of a production. Introduce a new NonTerminal $X_\zeta$ and the productions $X_\zeta \to \zeta$ and $\{A_i \to X_\zeta\}_{i=1,n}$ and eliminate the old productions $\{A_i \to \zeta\}_{i=1,n}$.*
3. *For each $\zeta \notin N$ that appears in the lhs of one production in $G$ let $\{\zeta \to B_j\}_{j=1,m}$ all the the productions that contain $\zeta$ the lhs of a production. Introduce a new NonTerminal $Y_\zeta$ and the productions $\zeta \to Y_\zeta$ and $\{Y_\zeta \to B_j\}_{j=1,m}$ and eliminate the old productions $\{\zeta \to B_j\}_{j=1,m}$.*

*Then the new grammar $G'$ generates the same language as the initial grammar $G$ and all the productions of the form $A \to \zeta$ and $\zeta \to B$ , $\zeta \notin N$ respect strong-uniqueness. Furthermore if the initial grammar has some restrictions on the composition of the $\zeta \notin N$ that appears in the productions of type 2 and 3, they are still maintained since $\zeta$ is left unchanged in the productions of the new grammar and the only other types of productions introduced are Renamings which do not belong to either type 2 or 3.*

*Proof.* To show that the two grammars are equivalent we will use a similar proof technique as in the previous theorem. Let $\gamma \in N^+$ and $\gamma \xrightarrow{*} w$ be a derivation in $G$. In this derivation we can replace all the uses of the production $A_i \to \zeta$ with the sequence $A_i \to X_\zeta \to \zeta$ and the uses of the production $\zeta \to B_j$ with the sequence $\zeta \to Y_\zeta \to B_j$ and get a derivation that

is valid in $G$, since all the other productions are common between the two grammars. Thus we have proved that for all $\gamma \in N^+$ we can convert a valid derivation from $G$, $\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in $G'$ and in particular this is true also for $S \stackrel{*}{\rightarrow}_G w$, which can be converted into $S \stackrel{*}{\rightarrow}_{G'} w$ and which furthermore can be converted into $S' \rightarrow S \stackrel{*}{\rightarrow}_{G'} w$ and therefore $L(G) \subseteq L(G')$.

Conversely, let $\gamma \in N^+$ and $\gamma \stackrel{*}{\rightarrow} w$ be a valid derivation in $G'$ then whenever we use the rule $A_i \rightarrow X_\zeta$ in this derivation $\gamma \stackrel{*}{\rightarrow} \alpha A_i \beta \rightarrow \alpha X_\zeta \beta \stackrel{*}{\rightarrow} w$ let $\gamma \stackrel{*}{\rightarrow} \alpha A_i \beta \rightarrow \alpha X_\zeta \beta \stackrel{*}{\rightarrow} \delta X_\zeta \eta \rightarrow \delta \zeta \eta \stackrel{*}{\rightarrow} w$ be the place where the $X_\zeta$ that occurs between $\alpha$ and $\beta$ is rewritten (used in the lhs of a production, even if it rewrites to the same symbol) for the first time. Then necessarily the $X_\zeta \rightarrow \zeta$ rule is the one that applies since it is the only one that has $X_\zeta$ in the lhs. Furthermore, as consequence of Lemma 1 we have that $\alpha \stackrel{*}{\rightarrow} \delta$ and $\beta \stackrel{*}{\rightarrow} \eta$ are valid derivations in $G'$. Therefore we can bring the production $X_\zeta \rightarrow \zeta$ right before the use of $A_i \rightarrow X_\zeta$ as follows $\gamma \stackrel{*}{\rightarrow} \alpha A_i \beta \rightarrow \alpha X_\zeta \beta \rightarrow \alpha \zeta \beta \stackrel{*}{\rightarrow} \delta \zeta \beta \stackrel{*}{\rightarrow} \delta \zeta \eta \stackrel{*}{\rightarrow} w$ and still have a valid derivation in $G'$. We can repeat this procedure for all places where rules of the form $A_i \rightarrow X_\zeta$ appear. Similarly for the uses of the productions of the type $\zeta \rightarrow Y_\zeta$ in a derivation $\gamma \stackrel{*}{\rightarrow} w$ we can bring the production that rewrites $X_\zeta$ ($Y_\zeta \rightarrow B_j$) right after as follows: change $\gamma \stackrel{*}{\rightarrow} \alpha \zeta \beta \rightarrow \alpha Y_\zeta \beta \stackrel{*}{\rightarrow} \delta Y_\zeta \eta \rightarrow \delta B_j \eta \stackrel{*}{\rightarrow} w$ into $\gamma \stackrel{*}{\rightarrow} \alpha \zeta \beta \rightarrow \alpha Y_\zeta \beta \rightarrow \alpha B_j \beta \stackrel{*}{\rightarrow} \delta B_j \beta \stackrel{*}{\rightarrow} \delta B_j \eta \stackrel{*}{\rightarrow} w$, because from Lemma 1 we have that $\alpha \stackrel{*}{\rightarrow} \delta$ and $\beta \stackrel{*}{\rightarrow} \eta$. We can repeat this procedure for all places where rules of the form $\zeta \rightarrow X_\zeta$ appear. In the new modified derivation we can replace all the uses of the sequence $A_i \rightarrow X_\zeta \rightarrow \zeta$ with the production $A_i \rightarrow \zeta$ and the sequence $\zeta \rightarrow Y_\zeta \rightarrow B_j$ with the production $\zeta \rightarrow B_j$ and obtain a derivation that is valid in $G$ since all the other productions are common between the two grammars. Thus, we have proved that for all $\gamma \in N^+$ we can convert a valid derivation from $G'$ ,$\gamma \stackrel{*}{\rightarrow} w$ into a valid derivation in $G$. For a derivation and $S' \rightarrow^*_{G'} w$ we have that necessarily $S' \rightarrow S \rightarrow^*_{G'} w$ since $S' \rightarrow S$ is the only production that rewrites $S'$ but since $S \in N^+$, it follows that we can use the previous procedure in order to convert $S \rightarrow^*_{G'} w$ into a derivation $S \rightarrow^*_G w$ which proves that $L(G') \subseteq L(G)$. Hence, we have proved that the grammars are equivalent, i.e., $L(G) = L(G')$.

It is obvious that all the productions of the form $A \rightarrow \zeta$ and $\zeta \rightarrow B$ , $\zeta \notin N$ respect *strong-*

*uniqueness*. Furthermore if the initial grammar has some restrictions on the composition of the $\zeta \notin N$ that appear in the productions of type 2 and 3 they are still maintained since $\zeta$ is left unchanged in the productions of the new grammar and the only other types of productions introduced are Renamings which do not belong to either type 2 or 3.

□

By applying Lemma 2 to the previous two Weak-ASNF theorems we can obtain Strong versions of these theorems which enforce *strong-uniqueness* in all the productions safe the Renamings.

**Theorem 3 (Strong-CFG-ASNF).** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a Context Free Grammar. Then there exists a Context Free Grammar $G'$ such that $L(G) = L(G')$ and $G'$ contains only rules of the following type:*

1. $A \rightarrow B$
2. $A \rightarrow BC$ *- and this is the only rule that has $BC$ in the rhs and this is the only rule that has $A$ in the lhs (strong-uniqueness).*
3. $A \rightarrow a$ *- and this is the only rule that has $a$ in the rhs and this is the only rule that has $A$ in the lhs (strong-uniqueness).*

**Theorem 4 (Strong-GEN-ASNF).** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a general (unrestricted) grammar. Then there exists a grammar $G'$ such that $L(G) = L(G')$ and $G'$ contains only rules of the following type:*

1. $A \rightarrow B$
2. $A \rightarrow BC$ *- and this is the only rule that has $BC$ in the rhs and this is the only rule that has $A$ in the lhs (strong-uniqueness).*
3. $A \rightarrow a$ *- and this is the only rule that has $a$ in the rhs and this is the only rule that has $A$ in the lhs (strong-uniqueness).*
4. $AB \rightarrow C$ *- and this is the only rule that has $C$ in the rhs and this is the only rule that has $AB$ in the lhs (strong-uniqueness).*

**Remark.** After enforcing strong uniqueness the only productions that contain choice are those of type 1 - Renamings(REN).

Given that we proved the main theorem we proceed in the next section to introduce the concept of abstraction and prove some additional results.

## 5.4 Additional concepts and results

### 5.4.1 Abstractions And Reverse Abstractions

Despite the fact that the names of the theorems contain the word Abstraction we have not shown yet where the Abstractions are. They are buried into the relations among Renamings and their unearthing is the role of this subsection.

**Definitions (Abstractions Graph).** Given a grammar $G = (N, T, S, R)$ which is in an ASNF from any of the Theorems 1 - 4 we call an *Abstractions Graph of the grammar* $G$ and denote it by $AG(G)$ a Directed Graph $G = (N, E)$ whose nodes are the NonTerminals of the grammar $G$ and whose edges are constructed as follows: we put a directed edge starting from $A$ and ending in $B$ iff $A \rightarrow B$ is a production that occurs in the grammar. Without loss of generality we can assume that the graph has no self loops, i.e., edges of the form $A \rightarrow A$, as if this is the case these productions can be eliminated from the grammar without changing the language. In such a directed graph a node $A$ has a set of out-edges, the edges that point out from it, and a set of in-edges, the edges that point into it. We will call a node $A$ along with its out-edges the *Abstraction at* $A$ and denote it $ABS(A) = \{A, OE_A = \{(A, B)|(A, B) \in E\}\}$. Similarly, we will call a node $A$ along with its in-edges the *Reverse Abstraction at* $A$ and denote it $RABS(A) = \{A, IE_A = \{(B, A)|(B, A) \in E\}\}$.

### 5.4.2 Minimality

We can ask the question whether we can find even simpler types of structural elements that can generate the full power of Turing Machines. Two of our operators, RSS and SS require that the size of the production ($|lhs| + |rhs|$) is 3. We can ask whether we can do it with size 2. This is answered negatively by the following proposition.

**Proposition (Minimality)** *If we impose the restriction that* $|lhs| + |rhs| \leq 2$ *for all the productions then we can only generate languages that are finite sets Terminals, with the possible addition of the empty string* $\epsilon$*.*

*Proof* . The only possible productions under this constraint are:

1. $A \rightarrow a$
2. $A \rightarrow \epsilon$
3. $A \rightarrow B$
4. $AB \rightarrow \epsilon$
5. $Aa \rightarrow \epsilon$

All the derivations that start from $S$, either keep on rewriting the current NonTerminal, because there is no production which increases the size, or rewrite the current NonTerminal into a Terminal or $\epsilon$.

□

Another possible question to ask is whether we can find a smaller set of structural elements set or at least equally minimal. The following theorem is a known result by Savitch [14].

**Theorem 5 (Savitch Alternative).** *(Savitch 1973 [14]) Let* $G = (N, T, S, R)$*,* $\epsilon \notin L(G)$ *be a General Grammar then there exists a grammar* $G'$ *such that* $L(G) = L(G')$ *and* $G'$ *contains only rules of the following type:*

1. $AB \rightarrow \epsilon$
2. $A \rightarrow BC$
3. $A \rightarrow a$

The Savitch theorem has three types of rules TERMINAL (type 3), SS (type 2) and ANNIHILATE2 (type 1). However no *strong-uniqueness* has been enforced and were it to be enforced then one more type of rule, REN will be needed. Furthermore if we would not insist on uniqueness all the Renamings in the ASNF could be eliminated too (Renamings elimination is a well

known technique in the theory of formal languages [13]) . However this will make it very difficult to make explicit the Abstractions. Therefore it can be argued that ASNF and the Savitch Normal Form have the same number of fundamental structural elements with the crucial difference between the two being the replacement of the RSS with ANNIHILATE2. The Reverse SuperStructuring seems a more intuitive structural element however, at least from the point of view of this paper.. Next we present for completeness the version of Savitch theorem where *strong-uniqueness* is enforced for rules of type 2 and 3.

**Theorem 6 (Strong-GEN-ASNF-Savitch).** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a General Grammar then there exists a grammar $G'$ such that $L(G) = L(G')$ and $G'$ contains only rules of the following type:*

1. $A \rightarrow B$
2. $A \rightarrow BC$ *- and this is the only rule that has $BC$ in the rhs and this is the only rule that has $A$ in the lhs. (strong-uniqueness)*
3. $A \rightarrow a$ *- and this is the only rule that has $a$ in the rhs and this is the only rule that has $A$ in the lhs. (strong-uniqueness)*
4. $AB \rightarrow \epsilon$ *- and this is the only rule that has $AB$ on the lhs. (uniqueness)*

*Proof.* By the original Savitch theorem [14] any grammar can we written such that it only has rules of the type

1. $AB \rightarrow \epsilon$
2. $A \rightarrow BC$
3. $A \rightarrow a$

The rules of the form $AB \rightarrow \epsilon$ observe uniqueness by default since the only rules that have more than one symbol in the lhs are of the form $AB \rightarrow \epsilon$.

Then we can use the constructions in Lemma 2 on this grammar in order to enforce *strong-uniqueness* for SuperStructuring (SS) $A \rightarrow BC$ and Terminals (TERMINAL) $A \rightarrow a$ at the potential expense of introducing Renamings (REN).

□

In conclusion our set of structural elements is quite minimalistic, and based on intuitive structural elements. The alternative provided by Savitch theorem is a possible counter-candidate, but we prefer ASNF for the intuitiveness of the Reverse SuperStructuring.

In the next subsection we will prove two theorems that show that if the grammar is in one of the Strong-ASNF forms we can order a derivation (explanation) in order to use productions that grow the size of the intermediate in the first phase and then use only productions that shrink the size of the intermediate in the second phase followed by rewritings into Terminals only in the third phase.

### 5.4.3 Grow & Shrink theorems

**Theorem 7.** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a General Grammar in the Strong-GEN-ASNF-Savitch, i.e., all the productions are of the following form:*

1. $A \rightarrow B$
2. $A \rightarrow BC$ *- and this is the only rule that has $BC$ in the rhs and this is the only rule that has $A$ in the lhs. (strong-uniqueness)*
3. $A \rightarrow a$ *- and this is the only rule that has $A$ on the lhs and there is no other rule that has $a$ on the rhs. (strong uniqueness)*
4. $AB \rightarrow \epsilon$ *- and this is the only rule that has $AB$ on the lhs. (uniqueness)*

*Then for any derivation $w$ such that $\gamma \stackrel{*}{\rightarrow} w$ , in $G$, $\gamma \in N^+$ there exists a derivation $\gamma \stackrel{*}{\rightarrow} \mu \stackrel{*}{\rightarrow} \nu \stackrel{*}{\rightarrow} w$ such that $\mu \in N^+$, $\nu \in N^*$ and $\gamma \stackrel{*}{\rightarrow} \mu$ contains only rules of type 1 and 2 (REN, SS), $\mu \stackrel{*}{\rightarrow} \alpha$ contains only rules of the type 4 (ANNIHILATE2) and $\nu \stackrel{*}{\rightarrow} w$ contains only rules of type 3 (TERMINAL).*

*Proof.* Based on Lemma 3 (presented next) we can change the derivation $\gamma \overset{*}{\rightarrow} w$ into a derivation $\gamma \overset{*}{\rightarrow} \nu \overset{*}{\rightarrow} w$ such that the segment $\nu \overset{*}{\rightarrow} w$ contains only rules of type 3 (TERMINAL) and $\gamma \overset{*}{\rightarrow} \nu$ contains only rules of type 1, 2 or 4 (REN, SS, ANNIHILATE2) . Therefore the only thing we still have to prove is that we can rearrange $\gamma \overset{*}{\rightarrow} \nu$ into $\gamma \overset{*}{\rightarrow} \mu \overset{*}{\rightarrow} \nu$ such that $\gamma \overset{*}{\rightarrow} \mu$ contains only rules of type 1 or 2 (REN, SS) and $\mu \overset{*}{\rightarrow} \nu$ contains only rules of the type 4 (ANNIHILATE2).

Let $\gamma \in N^+$, and $w$ such that $\gamma \overset{*}{\rightarrow} \nu \overset{*}{\rightarrow} w$ is a derivation in $G$ , the segment $\nu \overset{*}{\rightarrow} w$ contains only rules of type 3 (TERMINAL) and $\gamma \overset{*}{\rightarrow} \alpha$ contains only rules of type 1, 2 or 4 (REN, SS, ANNIHILATE2). If the derivation already satisfies the condition of the lemma then we are done. Otherwise examine in order the productions in $\gamma \overset{*}{\rightarrow} \nu$ from end to the beginning until we encounter the first rule of type 4: $AB \rightarrow \epsilon$ that violates the condition required by the theorem and at least one production of the type 1 or 2 has been used after it (otherwise we have no violation). More exactly, prior to it only rules of type 1 or 2 were used and at least one such rule was used. That is $\gamma \overset{*}{\rightarrow} \alpha AB\beta \rightarrow \alpha\beta \overset{+}{\rightarrow} \mu' \overset{*}{\rightarrow} \nu$ and only rules of the type 1 or 2 have been used in $\alpha\beta \overset{+}{\rightarrow} \mu$, and only rules of the type 4 have been used in $\mu' \overset{*}{\rightarrow} \nu$. Because only rules of the type 1 or 2 (which never have more than one symbol in the lhs) have been used in $\alpha\beta \overset{+}{\rightarrow} \mu$ it follows that there exists $\mu_1, \mu_2 \in N^*$such that $\alpha \overset{*}{\rightarrow} \mu_1$ and $\beta \overset{*}{\rightarrow} \mu_2$ and $\mu = \mu_1\mu_2$. Therefore we can rearrange the rewriting $\gamma \overset{*}{\rightarrow} \alpha AB\beta \rightarrow \alpha\beta \overset{+}{\rightarrow} \mu$ into $\gamma \overset{*}{\rightarrow} \alpha AB\beta \overset{*}{\rightarrow} \mu_1 AB\beta \overset{*}{\rightarrow} \mu_1 AB\mu_2 \rightarrow \mu_1\mu_2 = \mu$ . In this way we have obtained a derivation for $\mu$ in $G$ that violates the conclusion of the lemma in one place less than the initial derivation. Since there is a finite number of steps in a derivation and therefore a finite number of places where the constraints can be violated it can be inferred that after a finite number of applications of the above-described "swapping" procedure we will obtain a derivation which satisfies the rules of the theorem.

□

**Lemma 3.** *Let $G = (N, T, S, R)$ be a general (unrestricted) grammar that contains only rules of the form:*

1. $\alpha \rightarrow \beta$, $\alpha \in N^+$,$\beta \in N^*$

2. $A \rightarrow a$

*Then for any derivation $w$ such that $\gamma \xrightarrow{*} w$ , in $G$, $\gamma \in N^+$ there exists a derivation $\gamma \xrightarrow{*} \nu \xrightarrow{*} w$ such that $\nu \in N^+$ and $\gamma \xrightarrow{*} \nu$ contains only rules of type 1 and $\nu \xrightarrow{*} w$ contains only rules of type 2 (TERMINAL).*

*Proof.* Let $w$ such that $\gamma \xrightarrow{*} w$ *in* $G$, $\gamma \in N^+$. If the derivation already satisfies the condition of the lemma then we are done. Otherwise examine in order the productions $\gamma \xrightarrow{*} w$ until we encounter a rule of type , say it is $A \rightarrow a$ , such that there are still rules of type 1 used after it $\gamma \xrightarrow{*} \alpha A \beta \rightarrow \alpha a \beta \xrightarrow{*} w$. Because none of the rules in the grammar contain terminals in their lhs it follows that there exists $w_1, w_2 \in T^*$ such that $\alpha \xrightarrow{*} w_1$ and $\beta \xrightarrow{*} w_2$ and $w = w_1 a w_2$. Therefore we can rearrange the rewriting $\gamma \xrightarrow{*} \alpha A \beta \rightarrow \alpha a \beta \xrightarrow{*} w$ into $\gamma \xrightarrow{*} \alpha A \beta \xrightarrow{*} w_1 A \beta \xrightarrow{*} w_1 A w_2 \rightarrow w_1 a w_2 = w$. In this way we have obtained a derivation for $w$ in $G$ that violates the conclusion of the lemma in one place less than the initial derivation. Since there is a finite number of steps in a derivation and therefore a finite number of places where the constraints can be violated it can be inferred that after finite number of application of the above-described "swapping" procedure we will obtain a derivation which satisfies the rules of the lemma.

□

**Theorem 8.** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a General Grammar. Then we can convert such a grammar into the Strong-GEN-ASNF i.e., such that all the productions are of the following form:*

1. $A \rightarrow B$
2. $A \rightarrow BC$ *- and this is the only rule that has $BC$ in the rhs and this is the only rule that has $A$ in the lhs. (strong-uniqueness)*
3. $A \rightarrow a$ *- and this is the only rule that has $A$ on the lhs and there is no other rule that has $a$ on the rhs. (strong uniqueness)*
4. $AB \rightarrow C$ *- and this is the only rule that has $C$ in the rhs and this is the only rule that has $AB$ in the lhs. (strong-uniqueness)*

*And furthermore for any derivation $w$ such that $\gamma \xrightarrow{*} w$ , in $G$, $\gamma \in N^+$ there exists a derivation $\gamma \xrightarrow{*} \mu \xrightarrow{*} \nu \xrightarrow{*} w$ such that $\mu \in N^+$, $\nu \in N^*$ and $\gamma \xrightarrow{*} \mu$ contains only rules of type 1 and 2 (REN, SS), $\mu \xrightarrow{*} \alpha$ contains only rules of the type 1, more particularly only Reverse Abstractions and type 4 (REN(RABS), RSS) and $\nu \xrightarrow{*} w$ contains only rules of type 3 (TERMINAL).*

*Proof Sketch.* By Theorem 6 we can convert the grammar $G$ into a grammar $G'$ in Strong-GEN-ASNF-Savitch. Then we can convert such a grammar into a grammar $G''$ in Strong-GEN-ASNF as follows: for all $\{A_iB_i \rightarrow \epsilon\}_{i=1,n}$ introduce new NonTerminals $\{X_{A_iB_i}\}_{i=1,n}$ and add $A_iB_i \rightarrow X_{A_iB_i}$ and $X_{A_iB_i} \rightarrow E$ and eliminate the original productions $\{A_iB_i \rightarrow \epsilon\}_{i=1,n}$. Furthermore, for all NonTerminals $X \neq E$ in the new grammar, add new NonTerminals $X_{XE}$, $X_{EX}$ and $X_{EE}$ and the production rules $XE \rightarrow X_{XE}$, $EX \rightarrow X_{EX}$, $X_{XE} \rightarrow X$, $X_{EX} \rightarrow X$, $EE \rightarrow X_{EE}$ and $X_{EE} \rightarrow E$. We can easily show using techniques already developed that the new grammar will generate the same language as the previous one and that it respects the *strong-uniqueness* for rules of type 2, 3 and 4.

Furthermore if we take a derivation for a string $w$ in the grammar $G'$ in the Strong-GEN-ASNF-Savitch $S \rightarrow^*_{G'} w$ then from Theorem 7 we know that we can convert it into a derivation that uses first only REN and SS in the phase 1, then only ANNIHILATE2 in phase 2 and finally only TERMINAL in phase 3. We can take such a derivation and replace the usage of the ANNIHILATE2 productions in phase 2 with productions as above in order to get a derivation in $G''$. Note that the productions introduced above are meant to transform the ANNIHILATE2 into rules that follow the Strong-GEN-ASNF, and the only types of rules that we have introduced are RSS with strong-uniqueness holding and REN of the RABS type. In this way we have proved the statement of the theorem.

□

We have therefore proved that for each General Grammar $G$ we can transform it both in a Strong-GEN-ASNF-Savitch and a Strong-GEN-ASNF such that the derivation (explanation in Structural Induction terminology) of any terminal string $w$ can be organized in three phases such that: In Phase 1 we use only productions that grow (or leave the same) the size of the intermediate string; In Phase 2 we use only productions that shrink (or leave the same) the

size of the intermediate string and in Phase 3 we use only TERMINAL productions.

Naively, at first sight, it may seem that this is a way to solve the halting problem and therefore maybe some mistake has been made in the argument. However this is not the case, as the key question is when to stop expanding the current string and start shrinking and this problem still remains. In a certain sense these two theorems are a way to give a clear characterization of the issue associated with solving the halting problem: namely that of knowing when to stop expanding. In the case of Grammars in arbitrary forms the issue is a little bit more muddled as we can have a succession of grow and shrink phases but we have shown that if the form is constrained in a certain ways then we only need one grow and one shrink. Note also that during the grow phase in both theorems we are only using context free productions.

So far we have introduced SuperStructuring, Abstraction and related concepts, and proved various results regarding normal forms, minimality and the grow-shrink forms of derivations in Grammar terminology. We are now ready to reinterpret them in the language of structural induction and tie them to related concepts in this more general context.

## 5.5 Structural Induction and the fundamental structural elements

In this section we will review the Structural Induction process in the light of the concepts and results obtained for Generative Grammars and discuss the role of each of the operators. Then we move on to make some more connections with already existing concepts in Statistics and Philosophy and show how the ASNF affords for very precise characterizations of these concepts.

In the context of Generative Grammars Structural Induction is concerned with the following question: Given a sequence of observations $w$ we attempt to find a theory (grammar) that explains $w$ and simultaneously also the explanation (derivation) $S \xrightarrow{*} w$. In a local way we may think that whenever we have a production rule $l \rightarrow r$ that $l$ explains $r$. In a bottom up - data driven way we may proceed as follows: First introduce for every Observable $a$ a production $A \rightarrow a$. These are just convenience productions that bring the observables into the realm of internal variables in order to make everything more uniform. The association is unique (one

to one and onto) and once we have done it we can forget about the existence of Observables (Terminals). This is only the role of the the TERMINAL productions, and for this reason we will not mention them in future discussions as they are not true structural operations. With this in mind if we are to construct a theory in the GEN-ASNF we can postulate any of the following laws:

1. SS - $A \rightarrow BC$ - SuperStructuring. Takes two internal variables $B$ and $C$ that occur within proximity of each other (adjacent) and labels the compound. From now on the shorter name $A$ can be used instead on the compound. This is the sole role of SuperStructuring - to give a name to a bigger compound in order to facilitate shorter explanations at latter stages.

2. ABS - $A \rightarrow B|C$ - Abstraction. Makes a name for the occurrence of either of the variables $B$or $C$. This may allow for the potential bundling of two productions, the first one involving $B$ and the other one involving $C$ while otherwise being the same, into a production involving only $A$. The role of Abstraction is to give a name to a group of entities (we have chosen two for simplicity) in order to facilitate more general explanations at latter stages which in turn will produce more compact theories.

3. RSS - $AB \rightarrow C$ - Reverse SuperStructuring - invent up to two new internal variables (may also use already existing ones) which if they occur within proximity of each other together they "explain" the internal variable $C$.

4. RABS - $A \rightarrow C$, $B \rightarrow C$ - Reverse Abstraction - invent new internal variables (may also use already existing ones) such that either of them can "explain" the internal variable $C$ (we have chosen two variables for simplicity).

In the next subsection we review two possible reasons for creating hidden variables and we identify them with Reverse Abstraction and Reverse SuperStructuring. Because in our GEN-ASNF we do not have other types of reasons for creating Hidden Variables as all the other production introduce convenience renamings only we can infer under the Computationalistic

Assumption that these two are the essential reasons for postulating Hidden Variables and any other reasons must be derivative. This produces a definite structural characterization of the rationales behind inventing Hidden Variables.

### 5.5.1 Reasons for postulating Hidden Variables

There are at least two types of reasons for creating Hidden Variables:

1. (**OR** type) - [multiple alternative hidden causes] The OR type corresponds to the case when some visible effect can have multiple hidden causes $H1 \rightarrow Effect$, $H2 \rightarrow Effect$ . This case corresponds in our formalism to the notion of Reverse Abstraction. One typical example of this is: The grass is wet, hence either it rained last night or the sprinkler was on. In the Statistical literature the models that use these types of hidden variables are known as mixture models [9].

2. (**T-AND** type) - [multiple concurrent hidden causes] The T-AND type, i.e., topological AND type, of which the AND is a particular case. This corresponds to the case when one visible effect has two hidden causes that both have to occur within proximity (hence the Topological prefix) of each other in order for the visible effect to be produced. $H1H2 \rightarrow Effect$. This corresponds in our formalism to the case of Reverse SuperStructuring. One example of this case is the Annihilation of an electron and a positron into a photon. $e^+e^- \rightarrow \gamma$ as illustrated by the following Feynman diagram. In the diagram the Annihilation is followed by a Disintegration which in our formalism will be represented by a SuperStructure $\gamma \rightarrow e^+e^-$. Note that the electron positron annihilation is a RSS and not an ANNIHILATE2 as in the Savitch type of theorems. In a certain sense one of the arguments for using RSS versus ANNIHILATE2 is also the fact that physical processes always have a byproduct despite carrying potentially misleading names such as annihilation, or the byproduct being energetic rather than material. Nevertheless, since our reasons for preferring GEN-ASNF versus GEN-ASNF-Savitch alternative are at best aesthetic / intuitive reasons it should always be kept in mind as a possible alternative. In the Statistical / Graphical Models literature the particular case of AND hidden explana-

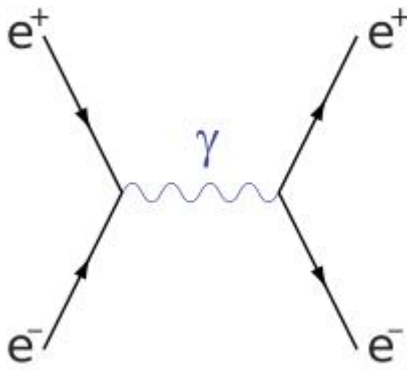



Figure 5.1 Feynman diagram for the electron positron Annihilation into a photon followed by photon Disintegration into an electron positron pair again

tions is the one that introduces edges between hidden variables in the depedence graph [5], [9], [11].

Our analysis of the Turing equivalent formalism of Generative Grammars written in the ASNF has evidenced them as the only needed types and we can infer under the Computationalistic Assumption that these are the only two essential reasons for postulating Hidden Variables and any other reasons must be derivative.

### 5.5.2 Radical Positivism

Since RSS and RABS involve the postulation of hidden variables, and we have discussed the perils associated with it, one alternative is to choose to use only Abstraction and SuperStructuring. We propose that this in fact this characterizes the radical positivist [1] stance which allows for empirical laws only. After we rule out RSS and RABS since they may introduce Hidden Variables we are left with ABS and SS and this produces our proposed characterization of the radical positivist position and what we mean by empirical laws under the Computationalistic Assumption. The internal variables created by Abstraction and SuperStructuring are going to be just convenient notations for aggregates of input data but nothing else: SuperStructuring is just convenient labeling of already existing structures for the sake of brevity and Abstraction on the other hand aggregates a group of variables into a more general type so that we can

produce more encompassing laws but with coarser granularity. An explanation of a stream of observations $w$ in the Radical Positivist theory of the world (or least a piece of it) will look like the picture in Figure 5.2 - top. In this picture the atoms are the observations and the theory of the universe is mainly a theory of how the observables are grouped into classes (Abstractions) and how smaller chunks of observations are tied together into bigger ones (SuperStructures). The laws of the radical positivist theory are truly empirical laws as they only address relations among observations.

However using only these two operators we cannot attain the power of Turing Machines. More precisely, the resulting types of grammars will be a strict subset of Context Free Grammars, (since CFG contain SS, and REN(ABS+RABS)). Next we will examine how any theory of the world may look like from the most general perspective when we do allow Hidden Variables.

### 5.5.3 General theories of the world

An explanation of a stream of observations $S \stackrel{*}{\rightarrow} w$ in the more general hidden variable theory of the world is illustrated in Figure 5.2 - bottom. The atoms are the hidden variables and their organization is again in turn addressed by the Abstraction and SuperStructuring but also Reverse Abstraction. This part is basically a context free part since all the productions are context free. Observations are a derivative byproduct obtained from a richer hidden variable state description by a reduction: either of size - performed by Reverse SuperStructuring or of information - performed by Reverse Abstraction.

The hidden variables theory of the world picture is an oversimplification of the true story, in general we may have a set of alternations of REN+SS and RABS+RSS rather than just one. However as we have shown in Theorem 8 we can turn any grammar into a grammar in Strong-GEN-ASNF such that any explanation done in three phases only (as illustrated in Figure 5.2):

1. a growth phase where we use only REN+SS

2. a shrink phase where we use only RABS+RSS

3. a phase where we only rewrite into Terminals.

Whether additional separations can be made, e.g., if we can push all the RABS from the first phase into the second phase and make the first phase look like the one in the radical positivist story (i.e., using only ABS+SS) is a topic of further investigation. We take this opportunity to proposed it as a conjecture.

**Conjecture.** *Let $G = (N, T, S, R)$, $\epsilon \notin L(G)$ be a General Grammar. Then we can convert such a grammar into the Strong-GEN-ASNF i.e., such that all the productions are of the following form:*

1. $A \to B$
2. $A \to BC$ *- and this is the only rule that has $BC$ in the rhs and this is the only rule that has $A$ in the lhs (strong-uniqueness).*
3. $A \to a$ *- and this is the only rule that has $A$ on the lhs and there is no other rule that has $a$ on the rhs. (strong uniqueness)*
4. $AB \to C$ *- and this is the only rule that has $C$ in the rhs and this is the only rule that has $AB$ in the lhs (strong-uniqueness).*

*And furthermore for any derivation $w$ such that $\gamma \xrightarrow{*} w$ , in $G$, $\gamma \in N^+$ there exists a derivation $\gamma \xrightarrow{*} \mu \xrightarrow{*} \nu \xrightarrow{*} w$ such that $\mu \in N^+$,$\nu \in N^*$ and $\gamma \xrightarrow{*} \mu$ contains only rules of type 1, more particularly only Abstractions, and type 2 (ABS,SS), $\mu \xrightarrow{*} \alpha$ contains only rules of the type 1, more particularly only Reverse Abstractions, and type 4 (RABS,RSS) and $\nu \xrightarrow{*} w$ contains only rules of type 3 (TERMINAL).*

In a certain sense the conjecture can be seen as way to try to salvage as much as we can from the Radical Positivist position by saying that in principle it is right with the caveat that the atoms should be internal(hidden) variables rather than observations. If we were God (here used in the scientifico-philosophical sense - i.e., somebody which knows the laws of the universe and has access to the full hidden state of it) then we would be able to stop our explanation of the current state after phase 1, which would use only Abstractions and SuperStructures (the

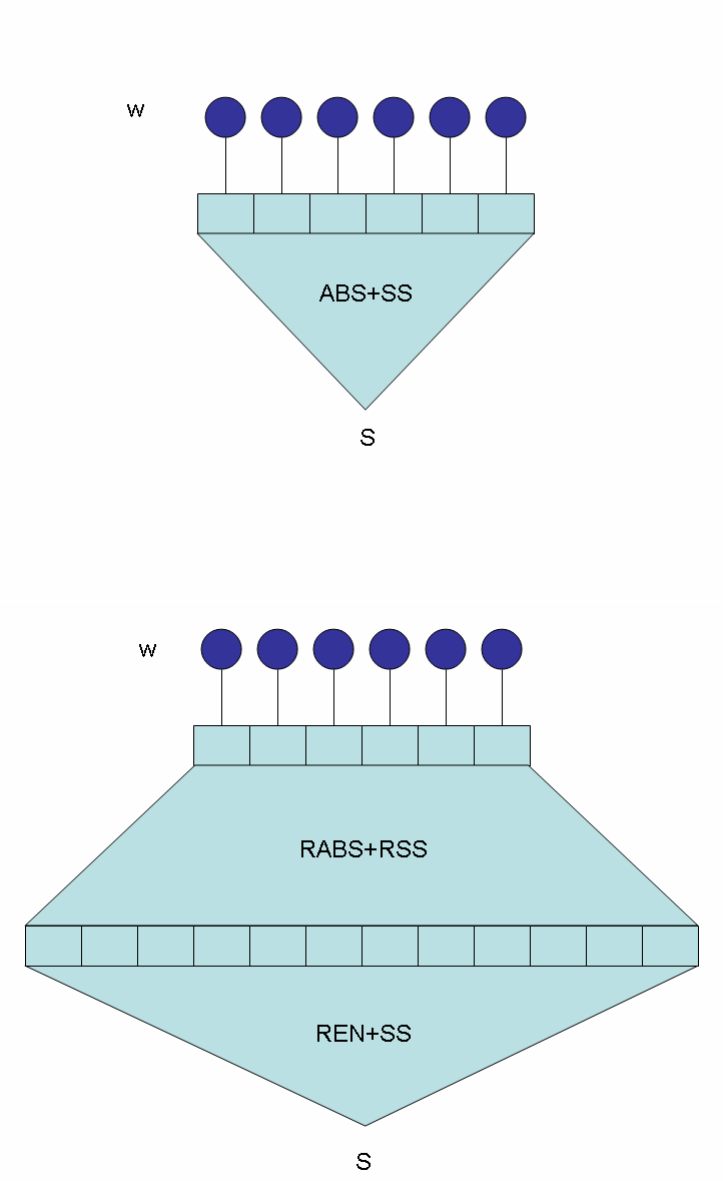



Figure 5.2 Theory of the world: (top) - Radical Positivist, (bottom) - with Hidden Variables

Radical Positivist position). However since we are mere mortals and all we are able to perceive are Observables which are just a simplified small reflection of the current hidden state of the world there is a the need for reduction operations: reduction in size - performed by Reverse SuperStructuring and reduction of information - performed by Reverse Abstraction. This is then followed by the one to one correspondence between some internal variable and Observables (Terminals in grammar terminlogy).

Our main claim so far is that we can rewrite any General Grammar in GEN-ASNF, that is using only ABS, SS, RABS and RSS. In the next section we will examine a statement that was made by the philosopher David Hume more that 200 years ago in the light of the GEN-ASNF theorem and propose that they are more or less equivalent under the Computationalistic Assumption. We then examine the developments that occured in the meantime in order to facilitate our proof of Hume's claim.

### 5.5.4 Hume principles of connexion among ideas

*"I do not find that any philosopher has attempted to enumerate or class all the principles of association [of ideas]. ... To me, there appear to be only three principles of connexion among ideas, namely,* ***Resemblance****,* ***Contiguity*** *in time or place, and* ***Cause*** *and* ***Effect"*** *- David Hume, Enquiry concerning Human Understanding, III(19), 1748. [6]*

If we are to substitute Resemblance for Abstraction (as resemblance/similarity is the main criterion for abstraction), Contiguity in time or place for SuperStructuring (as the requirement for SuperStructuring is proximity - in particular spatial or temporal) and Cause and Effect for the two types of hidden variable explanations then the GEN-ASNF theorem is the proof of a more the two hundred years old claim. The main developments that have facilitated this result are:

1. The Church-Turing thesis -1936 [18] (i.e., the Computationalistic Assumption) which allowed us to characterize what a theory of the world is, i.e., an entity expressed in a Turing equivalent formalism.

2. The Generative Grammars - 1957 [4] the development of the Compositional formalism of Generative Grammars which is Turing equivalent.

3. Developments in understanding the structure of Generative Grammars - The Kuroda Normal Form 1964 [8] and General Kuroda Normal Form [13].

4. The GEN-ASNF theorems proved in this paper.

Furthermore the elucidation of the two types of causal explanation (alternative - Reverse Abstraction (RABS) and topological conjunctive - Reverse SuperStructuring (RSS)) is an additional merit of GEN-ASNF theorem. It should be mentioned also that we became aware of Hume's claim after we have already stated the GEN-ASNF theorem but prior to it's proof in the full generality. We were led to it in a certain sense by similar intuitions and investigations into the nature of the Structural Induction process as David Hume's. Once aware of it, this became an additional supporting argument for the fact that the proof may be a feasible enterprise.

## 5.6 Conclusions

In this paper we have shown that Abstraction and SuperStructuring are essential structural elements in the induction process. We have shown that only two more structural elements are needed in order to obtain the full blown capacity of Turing Machines - the dual versions: Reverse Abstraction and Reverse SuperStructuring. These two additional structural elements are the constructs behind the two rationales for introducing hidden variables: alternative hidden causes, and concurrent "proximal" hidden causes. Furthermore we have also shown that given one theory of the world (expressed as a Generative Grammar) we can always convert it into a theory of the world such that any explanation (derivation) of a stream of observations in it can be reordered such that it occurs in three phases: a growth phase, where all the productions used are of the context free type (REN(ABS+RABS)+SS), a shrink phase where we only use (RABS+RSS) and a terminal phase where we only use TERMINAL productions. Whether the RABS can be extricated from the first phase is a matter of future research and can be viewed as a proposed conjecture. Furthermore we have also proposed that the radical positivist

position of postulating only empirical laws restricts theories to the usage of Abstraction and SuperStructuring only by avoiding altogether rules that postulate the existence of "hidden" explanations. We have also shown that our GEN-ASNF can be seen as a proof, under the Computationalistic Assumption, of a more than 200 years old claim of the philosopher David Hume about the principles of connexion among ideas.

## Bibliography

[1] A.J. Ayer, *Language, Truth, and Logic*. London: Gollancz. (2nd edition, 1946), 1936.

[2] M. Burgin. *Super-Recursive Algorithms*. Springer, 2005.

[3] R. Carnap. *An introduction to the Philosophy of Science*. 1966.

[4] N. Chomsky. *Syntactic Structures*. The Hague: Mouton. 1957.

[5] G. Elidan and N. Friedman. Learning Hidden Variable Networks: The Information Bottleneck Approach. *Journal of Machine Learning Research (JMLR)*, 6:81-127, 2005.

[6] D. Hume. *An Enquiry Concerning Human Understanding* . Hackett Publ Co. 1993.

[7] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. EATCS, Springer, 2005.

[8] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2): 207–223, 1964.

[9] S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[10] T. Oates, T. Armstrong, J. Harris and M. Nejman. On the Relationship Between Lexical Semantics and Syntax for the Inference of Context-Free Grammars. *Proceedings of the 19th National Conference on Artificial Intelligence ({AAAI})*, 431–436, 2004.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, 1988.

[12] K. R. Popper. *The Logic of Scientific Discovery,* Basic Books (English ed. 1959), 1934.

[13] A. Salomaa. *Computation and Automata.* Cambridge University Press, 1985.

[14] W. Savitch. How to make arbitrary grammars look like context-free grammars. *SIAM Journal on Computing,* 2:174-182, 1973.

[15] J. Schmidhuber, J. Zhao and M. Wiering Shifting Bias with Success Story Algorithm. *Machine Learning,* 28:105-130, 1997.

[16] R. Solomonoff. A Formal Theory of Inductive Inference, Part I. *Information and Control*, 7(1):1-22, 1964.

[17] R. Solomonoff. A Formal Theory of Inductive Inference, Part II. *Information and Control*, 7(2):224-254, 1964.

[18] A. Turing. On computable numbers with an application to the Entscheuidungs-problem, *Proc. Lond. Math. Soc.,* Ser.2, 42:230-265, 1936.

# CHAPTER 6. COMBINING ABSTRACTION AND SUPERSTRUCTURING

Modified from a paper to be submitted to *Neural Information Processing Systems*

Adrian Silvescu[1] and Vasant Honavar

## Abstract

Finding an appropriate set of concepts in terms of which to develop a model is a very important problem as the concepts in terms of which we choose to make the modeling can have a big impact both on the accuracy and tractability (e.g., as measured by size) of the final model. In this paper we study a way to construct new concepts / features based on the following operations: Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into one unit topologically close entities - in particular spatio-temporally close). We propose an algorithm to construct features by combining the complementary powers of Abstraction and SuperStructuring. SuperStructuring affords better modeling accuracy at the expense of increasing model size, while Abstraction reduces model size and improves the statistical estimates of too complicated models by shrinkage. In our experiments by using the combination of Abstraction and SuperStructuring we achieve significantly lower model sizes (1-3 orders of magnitude) for a minor drop in accuracy over the pure SuperStructuring approach and in some cases we even achieve better accuracy while simultaneously lowering the model size. We also show the superiority of Abstraction as opposed to the alternative approach of

[1]Primary researcher and author - designed and implemented the methods, made the theoretical arguments and wrote the paper.

using Feature Selection in order to reduce the size of the model created by SuperStructuring.

## 6.1 Introduction

Given a certain domain, the problem of finding an appropriate set of concepts in terms of which to develop a model pertaining to it is a nontrivial one. The concepts in terms of which we choose describe the model can have a big impact both on the accuracy and tractability (e.g., as measured by size) of the final model.

In previous papers [12], [26], [2] we have studied two important ways to construct new concepts / features based on a primitive set (assumed given): Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into one unit topologically close entities - in particular spatio-temporally close).

SuperStructuring in the particular case of Multivariate Statistics corresponds to considering higher order moments [3], [16]. In the Multivariate Statistics the topology is that of a complete graph, namely, it is assumed that any two variables are within proximity of each other and therefore can be grouped into a SuperStructure. The consideration of higher order moments in particular and SuperStructuring in general are well known methods to increase the power of the models (e.g., [20], [1], [8], [10]) and we have also used them for this purpose in [26], [2] in the spatial domain(sequences) and temporal domain [25]. The increase in modeling power however comes with the drawback that the size of the new extended model also increases and consequently the model also becomes less comprehensible. One conceivable way to cope with this increase is to perform Feature Selection [7], in order to reduce back the number of variables. Another alternative is to use Abstraction (grouping similar features under one overarching category). Abstraction aside from being capable of reducing model size can also be viewed as some sort of regularization of the shrinkage type [17], [7]. By using more abstract features which group together a set of less abstract features we effectively are able to increase the sample size for that feature and thus have the potential to get better estimates and better generalization due to the reduction in variance.

The main problem with using Abstraction versus Feature Selection is that we need to have

a source of Abstractions at hand to be able to use it. Sometimes such sources may exist in the form of taxonomies over the base features but may not exist over the SuperStructures. In order to alleviate this problem an algorithm that is able to generate Abstraction over arbitrary features is needed. We have proposed a algorithm - Attribute Value Taxonomy Learner [12] that is capable to learn Taxonomies on demand for one attribute at a time in the Multivariate setup. We can in principle extend it to cope with SuperStructures, and also to work in a Multinomial setup.

So far we have studied the two operations of Abstraction and SuperStructuring in isolation and in this paper we propose to study them combination with each other. The pairing is complementary as SuperStructures provides one way to increase modeling accuracy, but at the same time increase model size and Abstraction is a method which decreases model size and also provides some sort of regularization by improving the statistical confidence of the parameter estimates associated with each feature. This is extremely important as powerful models such as the ones that use SuperStructuring are usually plagued by overfitting problems on small sample size data . We will compare the combined approach with the baseline which uses only the primitive features and with the two approaches that use Abstraction only and SuperStructuring only. Additionally, we will also compare our approach with an alternative combination, that of Feature Selection along with SuperStructuring. In this later case Feature Selection will play the role of the operator that attempts to reduce model size.

In our experiments we show that by using the combination of Abstraction and SuperStructuring we achieve significantly lower model sizes (1-3 orders of magnitude) for a minor drop in accuracy over the pure SuperStructuring approach and in some cases we even achieve better accuracy while simultaneously lowering the model size. The Abstraction + SuperStructuring combination also dominates the Feature Selection + SuperStructuring combination showing the superiority of Abstraction over Feature Selection at preserving modeling accuracy for the same reduction in model size. Furthermore the combination Abstraction + SuperStructuring is also superior in accuracy to both using the base features alone and using Abstraction over the base features for the same model size (number of features used).

The remainder of paper is organized as follows: In the next section we present our approach to obtain features by combining Abstraction and SuperStructuring. Then we describe how to use the induced features in a prediction task, namely sub-cellular localization for protein sequences. Then we describe the experimental setup and the experimental results. Finally, we conclude.

## 6.2 Combining Abstraction and SuperStructuring

We will test the usefulness of combining Abstraction and SuperStructuring in a sequence classification scenario. More exactly we have a dataset of the form $D = \{d^i = (s^i, c^i)\}_{i=1,n}$ where $s_i$ is a sequence over a finite alphabet $G = \{g_1, ..., g_n\}$, that is $s \in G^*$ and $c^i$ is a class associated with the sequence $s^i$ which belongs to a finite set $C$, $c^i \in C$. Given such a dataset $D$ our algorithm is asked to produce a model which is able to predict the class $c$ for a novel sequence $s$.

We will use a filter / model-free approach [29] in order to derive a classification scheme based on features constructed by the combination of Abstraction and SuperStructuring. In the filter / model-free approach the relevance measure that scores the usefulness of the features constructed is defined independently of the learning algorithm.that will use them, as opposed to the wrapper / model-based approach which is dependent on the learning algorithm and scores features based how well they improve its performance. The main advance of filter / model-free versus the wrapper / model-based approaches is in terms of speed..

The main idea behind a filter classification scheme which can use any type of Feature Construction prior to classification is presented in Algorithm 5. The input is a dataset $D$ and a number of features to construct $nf$. In step 1 a set of $features$ of size $nf$ are constructed based on the training set $D$ and the training set is transformed into a new data set $D'$ that uses only these $features$. The data structure $features$ contains the information needed to construct each of the $nf$ $features$ from the instances like the ones appearing in the dataset $D$ (sequences $s$ - without the class as it will not be available at prediction time). Subsequently, in step 2, a $classifier$ is constructed based on the transformed dataset $D'$. The$classifier$ is

**FeatureConstructionClassifierTraining**
Input: $D = \{d_i = (s_i, c_i)\}_{i=1,n}$; $nf$ - number of features to construct
1. $features$, $D'$= ConstructFeaturesAndTransformDataset($nf$, $D$)
2. $classifier$ = ConstructClassifier($D'$)
Output: $classifer$,$features$

**FeatureConstructionClassifierPrediction**
Input: $classifier$, $features$, $s$ - instance to predict
1. $s1$= Transform($s$,$features$)
2. $class$ = Classify(classifier, $s1$)
Output: $class$

Algorithm 5 Feature Construction Classifier

totally agnostic to the fact that a Feature Construction and dataset Transformation took place and any classifier can be used as long as it can handle the given type of features produced by the transformation. The **FeatureConstructionClassifierTraining** then outputs the produced $classifier$ and $features$.

When asked for a prediction on a new instance $s$ the instance is transformed using the $features$ and then is classified with the $classifier$ that have been obtained in the previous step. This is outlined in **FeatureConstructionClassifierPrediction** of Algorithm 5. Note again that the classifier is agnostic as to whether a Feature Construction and data Transformation took place or not.

In the next subsection we detail a ConstructFeaturesAndTransformDataset procedure which combines Abstraction and SuperStructuring. Then in the following subsection we describe the classifier that we have used for our experiments, namely the Naive Bayes Multinomial Classifier. In this way we will produce a full blown instantiation of the Feature Construction Classifier scheme presented in Algorithm 5.

### 6.2.1 ConstructFeatures: Abstraction + SuperStructuring

In this section we will describe one instantiation of the ConstructFeaturesAndTransform-Dataset algorithm whose main purpose is to provide a scheme to construct features by combing Abstraction and SuperStructuring. We first detail what we mean by SuperStructuring in the

case of sequential data; then proceed to outline the algorithm and then address its various details one by one.

The SuperStructures in the case of sequences are just all the contiguous (potentially overlapping) sub-sequences of the initial string of a certain length $k$, called k-grams (see e.g., [4] for more details). For example for the string *abaac* the SuperStructures of size 3 (3-grams) are $\{aba, baa, aac\}$. The topology in this case is a linear topology in the sense that each element has at most two immediate neighbors: the neighbor to the right and the neighbor to the left in the sequence if they do exist.

The basic idea of our algorithm is to first create an extended feature set based on SuperStructures (k-grams in the case of sequences) in order to improve the modeling performance, and then to shrink down this feature set in order to decrease the model size to a fixed number of features $nf$. This is done in two steps as illustrated in Algorithm 6. First we find out all the k-grams (SuperStructures) that appear in the string instances from the dataset $D$ in step 1. Let's say that there are $m$ k-grams$\{kg_1, ..., kg_m\}$ that appear in either of the sequences contained in the dataset $D$. If a k-gram does not appear in the dataset $D$ then it will not be considered as a feature. This is done in order to prevent overfitting [2].

Then in step 2 a transformed dataset is created where each instance $d^i = (s^i, c^i)$ from the original dataset is mapped into an instance $d1^i = (\{[kg_1 : \#kg_1^i], ..., [kg_m : \#kg_m^i]\}, c^i)$ where $\{[kg_1 : \#kg_1^i], ..., [kg_m : \#kg_m^i]\}$ represents how many times each of the $m$ k-grams $kg_j$ can be found in the string $s_i$, and is denoted by $\#kg_j^i$. This representation is also known as a "bag of features( k-grams)" representation. More exactly:

**Definition(bag/multiset).** Let $F = \{f_1, ..., f_m\}$ be a set of elements, called features. We call a bag/multiset a features over the base universe $F = \{f_1, ..., f_m\}$ a function $b : F \to \mathbb{N}$ , where $\mathbb{N}$ is the set of natural numbers including 0. The function $b$ is called the *numerator function* and $b(f_j)$ tells us how many elements of the type $f_j$ are in the bag/multiset (contrast

[2]If the feature has zero support on the data there is no point in fitting it - the values for the parameters associated with it will be non-sensical, i.e., not based on the data $D$ - since the feature has zero support on the data $D$! Note that this is true in the case where we have no additional information on how the feature is related to other features that have non-zero support in the dataset $D$ or even if we have the information our algorithm does not uses it. If this additional information is available additional schemes become conceivable. In this paper however we will not address this issue.

**ConstructFeaturesAndTransformDataset**
Input: $D = \{d_i = (s_i, c_i)\}_{i=1,n}$; $nf$ - number of features to construct; $ks$ the k-gram(SuperStructure) size.
1. $SS$ = ConstructSuperStructures($ks$, $D$)
2. $D1$ = Transform($D$,$SS$)
3. $ABS$ = ConstructAbstractions($nf$,$D1$) / $FSEL$ = FeatureSelection($nf$,$D1$)
4. $D2$= Transform($D1$,$ABS$) / $D2$= Transform($D1$,$FSEL$)
Output: $features: SS + ABS$ / $features: SS + FSEL$, $D2$

Algorithm 6 Features Constructor

this with the more particular *indicator function* for sets $I : F \to \{0, 1\}$). Furthermore we will often write the bag/multiset $b$ by enumerating all the values that the function $b$ takes at each point $\{[f_1 : \#f_1], ..., [f_m : \#f_m]\}$ to stand for $b : F \to \mathbb{N}$ such that $b(f_j) = \#f_j$.

Returning to the description of the algorithm: Given the dataset $D1$ where each instance $d1^i = (b^i = \{[kg_1 : \#kg_1^i], ..., [kg_m : \#kg_m^i]\}, c^i)$ consists in a bag of k-grams over the universe$\{kg_1, ..., kg_m\}$ and a class, in step 3 we reduce this feature set to the desired number of feature $nf$. The reduction can be accomplished in two possible ways:

1. (ABS) by constructing Abstractions over the k-grams - that is partitioning the set of k-grams into $nf$ non-overlapping sets $ABS = \{a_1 : set_1, ..., a_{nf} : set_{nf}\}$ where $a_i$ denotes the label for the $i$-th Abstraction and $set_i$ denotes the set of k-grams which are grouped together into the $i$-th Abstraction. Thus an abstraction is basically identified with the set of k-grams / features that it groups together. Note that because of the partition requirement $set_1 \cup ... \cup set_{nf} = \{kg_1, ..., kg_m\}$ and $\forall 1 \leq i, j \leq nf,\ set_i \cap set_j = \emptyset$.

2. (FSEL) by Feature Selection - more precisely by selecting a set of $nf$ k-grams $FSEL \subseteq \{kg_1, ..., kg_m\}$, $|FSEL| = nf$.

Then in step 4 we transform the dataset $D1$ into a dataset $D2$ that has $nf$ features (aside from the class) as follows:

1. (ABS) given that the set of $nf$ Abstractions is $ABS = \{a_1 : set_1, ..., a_{nf} : set_{nf}\}$, with $set_j \subseteq \{kg_1, ..., kg_m\}, \forall 1 \leq j \leq nf$ an instance $d1^i = (\{[kg_1 : \#kg_1^i], ..., [kg_m : \#kg_m^i]\}, c^i)$

from $D1$ is transformed into an instance $d2^i = (\{[a_1 : \#a_1^i], ..., [a_{nf} : \#a_{nf}^i]\}, c^i)$ where

$$\#a_j^i = \sum_{kg_l \in set_j} \#kg_l^i \; \forall 1 \leq j \leq nf$$

.

2. (FSEL) given that the selected set of features is $FSEL = \{kg_{j_1}, ..., kg_{j_{nf}}\} \subseteq \{kg_1, ..., kg_m\}$ an instance $d1^i = (\{[kg_1 : \#kg_1^i], ..., [kg_m : \#kg_m^i]\}, c^i)$ from $D1$ is transformed into an instance $d2^i = (\{[kg_{j_1} : \#kg_{j_1}^i], ..., [kg_{j_{nf}} : \#kg_{j_{nf}}^i]\}, c^i)$.

In the next two subsections we describe the two reduction procedures used in step 3, namely Abstraction (ABS) and Feature Selection (FSEL).

#### 6.2.1.1 Constructing Abstractions

The procedure to construct Abstractions is illustrated in Algorithm 3. The input of the algorithm is a dataset where each datapoint consists of a bag / multiset of features and a class value $D = \{d^i = (b^i = \{[f_1 : \#f_1^i], ..., [f_m : \#f_m^i]\}, c^i)\}_{i=1,n}$ along with a number $nf \leq m$ that represent the reduced number of Abstractions that is desired.

The algorithm is a typical Hierarchical Agglomerative Clustering [7] approach. We start in step 1 by initializing the abstractions with the trivial grouping that creates as many groups as there are input features, with one element in each group. Then in steps 2 - 4 we recursively merge the given groups / Abstractions into bigger groups until we we obtain $nf$ groups / Abstractions. More exactly $n - nf$ times (Step 2) we find the most "similar" two groups/Abstractions (Step 3), merge them, delete them from the set of *abstractions* and add the newly merged group / Abstraction to the set of *abstractions* (Step 4). Finally, after $n - nf$ such steps have been performed we are left with $nf$ groups / Abstractions and we return them as the final result.

In order to complete the description of the previous algorithm all we need to show is how to compute the similarity measure $D(a_i : set_i, a_j : set_j)$ from step 3. The general criteria for establishing similarity between items and thus deriving useful Abstractions is the following *functionalist* claim: *Similar items occur within similar contexts.* In this spirit one way to define

**ConstructAbstractions**
Input: $D = \{d^i = (b^i = \{[f_1 : \#f_1^i], ..., [f_m : \#f_m^i]\}, c^i)\}_{i=1,n}$; $nf$ - number of abstractions to construct
1. $abstractions = \{a_1 : \{f_1\}, ..., a_m : \{f_m\}\}$
2. **for** $p := m + 1$ **to** $p = 2m - nf$
3. $\quad (im, jm) = \arg\min_{i,j \in abstractions} D(a_i : set_i, a_j : set_j)$
4. $\quad abstractions = abstractions \backslash \{a_{im} : set_{im}, a_{jm} : set_{jm}\} \cup \{a_p : set_{im} \cup set_{jm}\}$
Output: $abstractions$

Algorithm 7 Abstractions Constructor

the similarity distance $D(a_i : set_i, a_j : set_j)$ is to specify what we mean by the context of an abstraction $a_i : set_i$ and then define a distance between these contexts.

In what follows we will define the notion of the context of an Abstraction in a classification scenario. First we will define what we mean by the context of a feature $f_i$ in a classification scenario and then since an Abstraction is a set of features we will show how to define the context of such a set by aggregating the contexts of the elements.

**Definition (Class Context for Abstraction)**. Given a dataset $D = \{d^i = (b^i = \{[f_1 : \#f_1^i], ..., [f_m : \#f_m^i]\}, c^i)\}_{i=1,n}$ we define the the Class Context of the feature $f_j$ to be the total number of times the feature $f_j$ appears in an instance of each class $c \in C$ counting with multiplicity. That is:

$$[\#[f_j, c]]_{c=1,|C|} := \left[\sum_{i=1}^{n} \delta(c, c^i) \#f_j^i\right]_{c=1,n}$$

where $\delta$ is the Dirac function $\delta(i, j) = 1\ iff\ i = j\ and\ 0\ otherwise$. Alternatively, we can represent the context as a pair:

$$[\#f_j, P(C|f_j)] := \left[\sum_{c \in C} \#[f_j, c], \left[\frac{\#[f_j, c]}{\#f_j}\right]_{c=1,n}\right]$$

That is the number of times that the feature $f_j$ appears in the dataset $D$ counting with multiplicity along with the conditional probability of the class given the feature $f_j$: $P(C|f_j)$. Note that the two representations contain the same information, just in different ways. We will denote this pair with $CContext_D(f_j) := [\#f_j, P(C|f_j)]$ and call it the Class Context of the feature $f_j$ with respect to the data $D$. The $P(C|f_j)$ represents in a certain sense the context

proper and $\#f_j$ is a weight that will be used in order to aggregate appropriately the "contexts" of two or more features.

Next we show how to obtain the context of an Abstraction composed of two features. Let $f_{j1}$and $f_{j2}$ be two features and their contexts be $[\#f_{j1}, P(C|f_{j1})]$ and $[\#f_{j2}, P(C|f_{j2})]$ respectively. We define the class context of the set $\{f_{j1}, f_{j2}\}$ to be

$$CContext_D(\{f_{j1}, f_{j2}\}) := [\#f_{j1} + \#f_{j2}, \pi_1 P(C|f_{j1}) + \pi_2 P(C|f_{j2})]$$

where $\pi_1 := \frac{\#f_{j1}}{\#f_{j1}+\#f_{j2}}$ and $\pi_2 := \frac{\#f_{j2}}{\#f_{j1}+\#f_{j2}}$ .

More generally given a set of $q$ features $\{f_{j_1,...,}f_{j_q}\}$ we define

$$CContext_D(\{f_{j_1,...,}f_{j_q}\}) := \left[\sum_{l=1}^{p} \#f_{j_l}, \sum_{l=1}^{p} \pi_l P(C|f_{j_l})\right] \tag{6.1}$$

where

$$\pi_{_l} := \frac{\#f_{j_l}}{\sum_{l=1}^{p} \#f_{j_l}}$$

Note that if we are to "abstract out" the difference between all the features $\{f_{j_1,...,}f_{j_q}\}$ and replace each of their occurrences with a new feature called for example $a$ then in each datapoint $d^i = (b^i = \{[f_1 : \#f_1^i], ..., [f_m : \#f_m^i]\}, c^i)$ the feature $a$ will appear $\#a^i = \sum_{l=1}^{p} \#f_{j_l}^i$, and furthermore the feature $a$ will appear in the whole dataset $D$ $\#a = \sum_{l=1}^{p} \#f_{j_l}$ and also $\#[a, c] = \sum_{l=1}^{p} \#[f_{j_l}, c]$. Furthermore:

$$\begin{aligned}
P(C|a) &= \left[\frac{\#[a,c]}{\#a}\right]_{c=1,n} \\
&= \left[\frac{\sum_{l=1}^{p} \#[f_{j_l}, c]}{\sum_{l=1}^{p} \#f_{j_l}}\right]_{c=1,n} \\
&= \left[\sum_{l=1}^{p} \frac{\#[f_{j_l}]}{\sum_{l=1}^{p} \#f_{j_l}} \frac{\#[f_{j_l}, c]}{\#[f_{j_l}]}\right]_{c=1,n} \\
&= \left[\sum_{l=1}^{p} \pi_l P(c|f_{j_l})\right]_{c=1,n} \\
&= \sum_{l=1}^{p} \pi_l P(C|f_{j_l})
\end{aligned}$$

Thus we have just proved that

$$CContext_D(\{f_{j_1,...,}f_{j_q}\}) = \left[\sum_{l=1}^{p} \#f_{j_l}, \sum_{l=1}^{p} \pi_l P(C|f_{j_l})\right] = [\#a, P(C|a)] := CContext_D(a)$$

That is the definition of $CContext_D(\{f_{j_1,\dots,}f_{j_q}\})$ from Equation 6.1 coincides with the natural definition of an abstraction based on "abstracting out" the difference between the features $\{f_{j_1,\dots,}f_{j_q}\}$ and considering them as being a single feature $a$. Because of this well definiteness we can identify the two and write

$$CContext_D(a : \{f_{j_1,\dots,}f_{j_q}\}) = \left[\sum_{l=1}^{p} \#f_{j_l}, \sum_{l=1}^{p} \pi_l P(C|f_{j_l})\right] = [\#a, P(C|a)]$$

This will be the formula that we will use in order to compute the *context an abstraction*. Note that in this way we have also proved that $\sum_{l=1}^{p} \pi_l P(C|f_{j_l})$ is actually a probability distribution over the values $c \in C$ because it is the same as $P(C|a)$.

Given this well definiteness and identification we can move on and define the distance between the contexts of two arbitrary Abstractions. This will be used to assess the similarity between those two Abstractions and we will thus finish the description of Algorithm 7.

We first define the Kullback-Leibler Divergence [5] between two probability distributions. Then we define the Weighted Jensen-Shannon Distance [15] between two probability Distributions with weights $w_1, w_2$. Finally, we use the Weighted Jensen-Shannon Distance to define the distance between two Abstractions.

**Definition (Kullback-Leibler Divergence).** Let $C$ be a finite set and $P_1(C)$, $P_2(C)$ be two probability distribution over $C$. We define the Kullback-Leibler Divergence between two probability distributions $P_1(C)$, $P_2(C)$ as:

$$KL(P_1(C)||P_2(C)) = \sum_{c \in C} P_1(c) \log \frac{P_1(c)}{P_2(c)}$$

**Definition (Weighted Jensen-Shannon Distance).** Let $w_1, w_2 \in [0, \infty)$, $C$ be a finite set and $P_1(C)$, $P_2(C)$ be two probability distribution over $C$. Let $\pi_1 = \frac{w_1}{w_1+w_2}$ ,$\pi_2 = \frac{w_2}{w_1+w_2}$ and $P_c(C) = \pi_1 P_1(C) + \pi_2 P_2(C)$. Note that $P_c(C) = \pi_1 P_1(C) + \pi_2 P_2(C)$ is also a probability distribution over $C$ if $P_1(C)$, $P_2(C)$ are probability distributions over $C$and $\pi_1 + \pi_2 = 1$ as it is the case in our definition (a convex combination of probability distributions is also a probability distribution).

We define the *Weighted Jensen-Shannon Distance* between $P_1(C)$ and $P_2(C)$ with weights $w_1$ and $w_2$, respectively as:

$$WJS([w_1, P_1(C)], [w_2, P_2(C)]) := \pi_1 KL(P_1(C)||P_c(C)) + \pi_2 KL(P_2(C)||P_c(C))$$

The weighted Jensen Shannon distance is a generalization of the Jensen Shannon distance between two probability distributions [15] that allows for an asymmetry in the consideration of the two elements (the standard Jensen Shannon distance uses $\frac{1}{2}$ as weights), see also [27] .

Finally we are ready to define the distance between two Abstractions that is needed in order to finalize the description of Algorithm 7.

**Definition (Class Context Distance).** Let $a_i : set_i, a_j : set_j$ two abstractions we define

$$\begin{aligned} D(a_i : set_i, a_j : set_j) &:= WJS(CContext(a_i : set_i), CContext(a_j : set_j)) \\ &= WJS([\#a_i, P(C|a_i), [\#a_j, P(C|a_j)]) \end{aligned}$$

In the next subsection we describe the other reduction method from Algorithm 6, namely Feature Selection.

#### 6.2.1.2 Feature Selection

We will now describe Feature Selection, more precisely: selecting a set of $nf$ features from $FSEL \subseteq \{f_1, ..., f_m\}$, $|FSEL| = nf$. In order to achieve this we will rank all features according to the scoring function $Score$ and then select the top $nf$ ranking features according to the respective $Score$ function. We will use mutual information [5] (a.k.a. information gain in this context [19]) between the probability of the class variable $P(C)$ and the probability of the feature feature $f$, which measures how dependent the two variables are. More exactly, given a dataset $D = \{d^i = (b^i = \{[f_1 : \#f_1^i], ..., [f_m : \#f_m^i]\}, c^i)\}_{i=1,n}$, with $C$ being the set of classes $c^i \in C$ . Let $f \in \{f_1, ..., f_m\}$ , we define the value / $Score$ of the feature $f$ as estimated from data $D$ to be:

$$Score_D(f) = KL(P(f, C)||P(f)P(C))$$

where $P(f, C)$ is estimated from counts gathered from the dataset $D$ and $P(f)$ and $P(C)$ are obtained by marginalization from$P(f, C)$.

The $Score_D(f)$ is also known as the information gain because of the identity:

$$KL(P(f,C)||P(f)P(C)) = H(P(C)) - H(P(C|f)|P(C))$$

So far we have fully described the ConstructFeaturesAndTransformDataset algorithm and we will move on to present next the classifier used in our experiments.

### 6.2.2 Naive Bayes Multinomial Classifier

In this section we describe the Naive Bayes Multinomial Classifier. It is one possible classifier to be constructed in step 2 of the Feature Construction Classifier outlined Algorithm 5. And it is in particular the classifier that we have used in our experiments.

**Definition (Naive Bayes Multinomial Classifier).** Let $D = \{d^i = (b^i = \{[f_1 : \#f_1^i], ..., [f_m : \#f_m^i]\}, c^i)\}_{i=1,n}$ be a dataset, with $C$ being the set of classes $c^i \in C$ and $F = \{f_1, ..., f_m\}$ be the set of features. Then we call a *Naive Bayes Multinomial Classifier* an algorithm that predicts the class of a new instance $b' = \{[f_1 : \#f_1'], ..., [f_m : \#f_m']\}$ as follows:

$$class = \arg\max_{c \in C} P(c) \prod_{j=1}^{m} P(f_j|c)^{\#f_j'}$$

where $P(c)$ and $P(f_j|c)$ are estimated from counts gathered from the dataset $D$.

We have therefore shown how to perform a complete instantiation of the Feature Construction Classifier outlined in Algorithm 5 and we will next describe the experiments performed within this setup. We first examine some related work and then proceed to the description of the experimental setup and results.

### 6.2.3 Related work

The problem of Feature Construction is a widely studied issue in the Machine Literature. It can be found under many names and in various approaches: feature construction [28], [8] or predicate invention [14] in inductive logic programming, Statistical Predicate Invention in Relational Learning [13], constructive / natural induction [30], [20], [21], [18], [11]. The closest work related to ours is Jonyer et al. 2004 [11] where the authors induce a Context-Free Graph

Grammar Induction. Their algorithm attempts to infer Abstractions and SuperStructures even though not explicitly called so. Their approach however is unconditional (not aimed at predicting a target class variable) and also the scoring function is MDL-Based (aimed at compression) rather than statistically based as ours.

Many methods use SuperStructuring in order to improve modeling accuracy starting from using higher order moments in statistics [3] to Bayesian Networks and Markov networks [22], [6], [10], k-grams in linguistics [4], decision trees and rules [24], etc. A variety of methods also have been proposed for clustering features (see [12] and references therein). Slonim and Tishby [27] described a technique (called the agglomerative information bottleneck method) which extended the distributional clustering approach described by Pereira et al. [23], using the Jensen-Shannon divergence for measuring distance between document class distributions associated with words and applied it to a text classification task. The work by Slonim and Tishby [27] agglomerative information bottleneck is very close both in spirit and in form to our proposed method for creating Abstractions but they use the raw features i.e., words rather that SuperStructures (k-grams in our case). To our knowledge our work is the first that provides an examination and comparison of the possible alternatives to combine SuperStructuring with model reduction techniques such as Abstraction and Feature Selection in the class conditional case and makes a strong case for the combination of Abstraction and SuperStructuring.

## 6.3 Experiments

### 6.3.1 Experimental setup

We show experimental results on two subcellular localization prediction datasets, one for eukaryotes and one for prokaryotes. Each dataset consists in a set of protein sequences along with a class associated with each protein sequence which denotes the cellular compartment where the protein is found in the cell. More exactly:

1. **eukaryotes** dataset - The dataset consists of 2427 eukaryotic protein sequences that are composed of 22 aminoacids plus two terminators: BEGIN_SEQ and END_SEQ inserted

by us at the beginning and end of the of each sequence (the base alphabet $G$ thus has 24 letters). Each sequence can belong to one of the following four classes *{nuclearEuk, cytoplasmicEuk, extracellularEuk, mitochondrialEuk}*.

2. **prokaryotes** dataset - The dataset consists of 997 prokaryotic protein sequences that are composed of 20 aminoacids plus two terminators: BEGIN_SEQ and END_SEQ inserted by us at the beginning and end of the of each sequence (the base alphabet $G$ thus has 22 letters). Each sequence can belong to one of the following three classes *{peripalsmicPro, cytoplasmicPro, extracellularPro}*.

The average length of a protein sequence is about 300 aminoacids.

The experiments performed are meant to contrast the combination of Abstraction and SuperStructuring with approaches where we do not use either or both of these methods for Feature Construction and also with the alternative of combining Feature Selection and SuperStructuring. More precisely for both datasets we have performed experiments in the following setups:

1. UNIGRAM - experiments where no Abstraction or SuperStructuring or Feature Selection is done. The features used the unigrams. Basically this means that a bag/multiset based on the base alphabet (aminoacids + terminators) is created and for each symbol in the alphabet the number of times it appears in a sequence is recorded as a value for that feature. This is the most basic scenario where no Feature Construction is done.

2. ABS_ONLY - No SuperStructuring (k-grams) is performed, we use unigrams instead, but a reduction to $nf$ features by Abstraction is done - the features are a bag of $nf$ Abstractions of unigrams.

3. SS_ONLY - Only SuperStructuring is performed (k-grams $k \geq 2$ are constructed) but then no reduction by either Abstraction or Feature Selection is attempted. The features are a bag of k-grams, one feature for each k-gram that appears in the dataset.

4. FSEL+SS - SuperStructuring is performed (k-grams $k \geq 2$ are constructed) and then Reduction by Feature Selection is done down to $nf$ features. The features are a *select* bag of $nf$ k-grams.

5. ABS+SS - SuperStructuring is performed (k-grams $k \geq 2$ are constructed) and then Reduction by Abstraction is done down to $nf$ features. The features are a bag of $nf$ Abstractions of k-grams.

In the next section we present and discuss the experimental results for both the eukaryotes and prokaryotes dataset, for different sizes of k-grams (2 and 3) and for variable number of constructed features $nf$.

### 6.3.2 Experimental results

1. Figure 6.1: eukaryotes dataset 3-grams - UNIGRAM - 24 unigrams, SS_ONLY 3-grams $\sim$ 8000 features ($24 \times 24 \times 24$ - the ones that do not appear)

2. Figure 6.2: blow-out of the previous figure in the [1-40] features range.

3. Figure 6.3: eukaryotes dataset 2-grams - UNIGRAM - 24 unigrams, SS_ONLY 2-grams $\sim$ 400 features ($24 \times 24$ - the ones that do not appear)

4. Figure 6.4: prokaryotes dataset 3-grams - UNIGRAM - 22 unigrams, SS_ONLY 3-grams $\sim$ 8000 features ($22 \times 22 \times 22$ - the ones that do not appear)

5. Figure 6.5: prokaryotes dataset 2-grams - UNIGRAM - 22 unigrams, SS_ONLY 2-grams $\sim$ 400 features ($22 \times 22$ - the ones that do not appear)

On an overwhelming set of points the graphs show that for the same number of features constructed the combination of Abstraction and SuperStructuring - ABS+SS is superior in performance to using the base features - UNIGRAM, Abstraction only on top of these base features - ABS_ONLY, or the alternative combination of Feature Selection and SuperStructuring - FSEL+SS.

When comparing the combination ABS+SS with the SuperStructuring only approach SS_ONLY, the ABS+SS in all cases gets very close in terms of performance to the SS_ONLY with relatively few features (SS_ONLY uses $> 8000$ for 3-grams, and $> 400$ for 2-grams) and in some cases even outperforms SS_ONLY - most prominently in the Eukaryotes 3-gram case (Figures 6.1-6.2) first after 80 features and then consistently after 120 -150 features. Compared with $> 8000$ features for SS_ONLY, this is an almost two orders of magnitude drop in model size for the same performance. Furthermore for a small drop in performance $(1\% - 2\%)$ in all the graphs we have a a reduction of model size by at least one order of magnitude - but in some cases even 2 or almost 3 orders of magnitude. As already stated the ABS+SS is also superior to the alternative possible combination of Feature Selection and SuperStructuring - FSEL+SS.

## 6.4 Conclusions

We have proposed an algorithm to construct features by combining the complementary powers of Abstraction and SuperStructuring. SuperStructuring affords better modeling accuracy at the expense of increasing model size, while Abstraction reduces model size and improves the statistical estimates of too complicated models by shrinkage. We have performed experiments on two datasets and have shown the ability of Abstraction and SuperStructuring combination to produce significantly smaller models (1-3 orders of magnitude) for little or no sacrifice in cross-validation performance or even gain in some cases. We have also further shown the superiority of Abstraction as opposed to the alternative approach of using Features Selection in order to reduce the size of the model created by SuperStructuring. The proposed algorithm is probably the simplest possible scheme of combining Abstraction and SuperStructuring and it represents just a starting point, but even in such a simple scenario a good case can be made for the merits of this marriage.

## Bibliography

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetski-Shapiro, P. Smyth, and R. Uthurusamy,

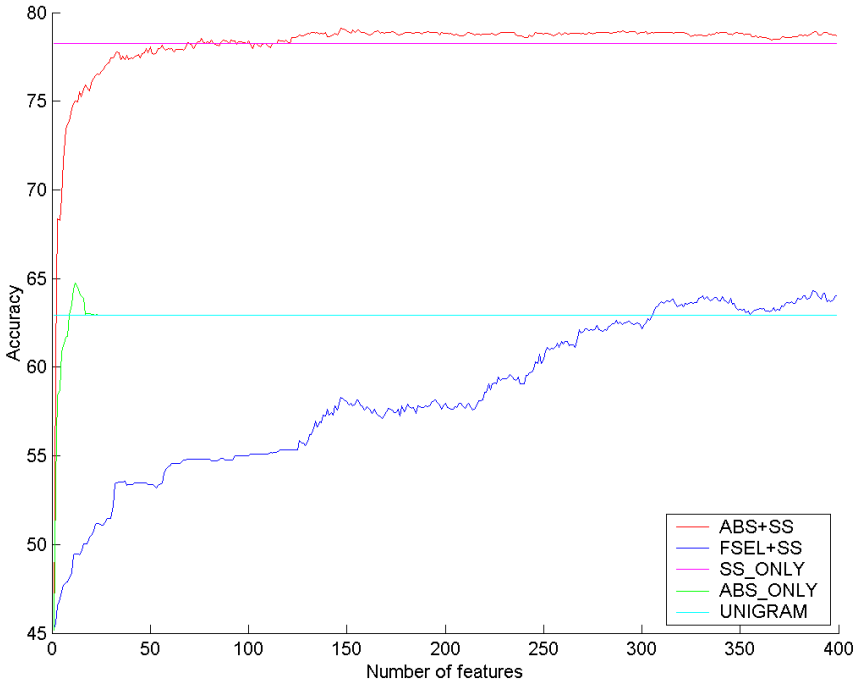



Figure 6.1 Comparison of the Abstraction + SuperStructuring method with other methods on the Eukaryotes dataset 3-grams. The graph represents cross-validation accuracy (5-fold) as a function of the number of features used. UNIGRAM uses 24 features; SS_ONLY 3-grams uses 8000 features. ABS_ONLY varies in the range 1-24 features; ABS+SS and FSEL+SS vary in the range1-400 features.

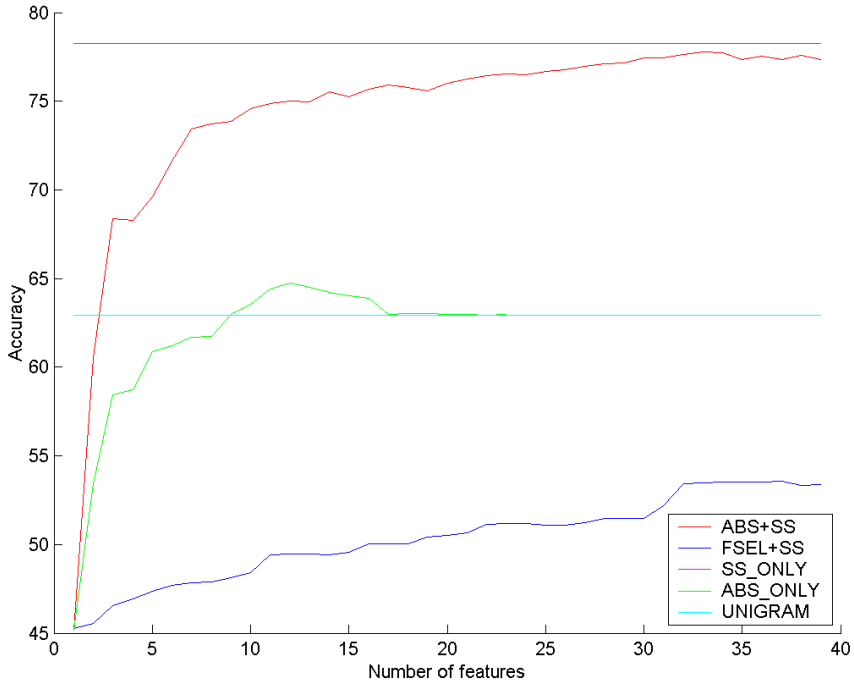



Figure 6.2 Comparison of the Abstraction + SuperStructuring method with other methods on the Eukaryotes dataset 3-grams - blowout of the [1-40] range. The graph represents cross-validation accuracy (5-fold) as a function of the number of features used. UNIGRAM uses 24 features; SS_ONLY 3-grams uses 8000 features. ABS_ONLY varies in the range 1-24 features; ABS+SS and FSEL+SS vary in the range 1-40 features.

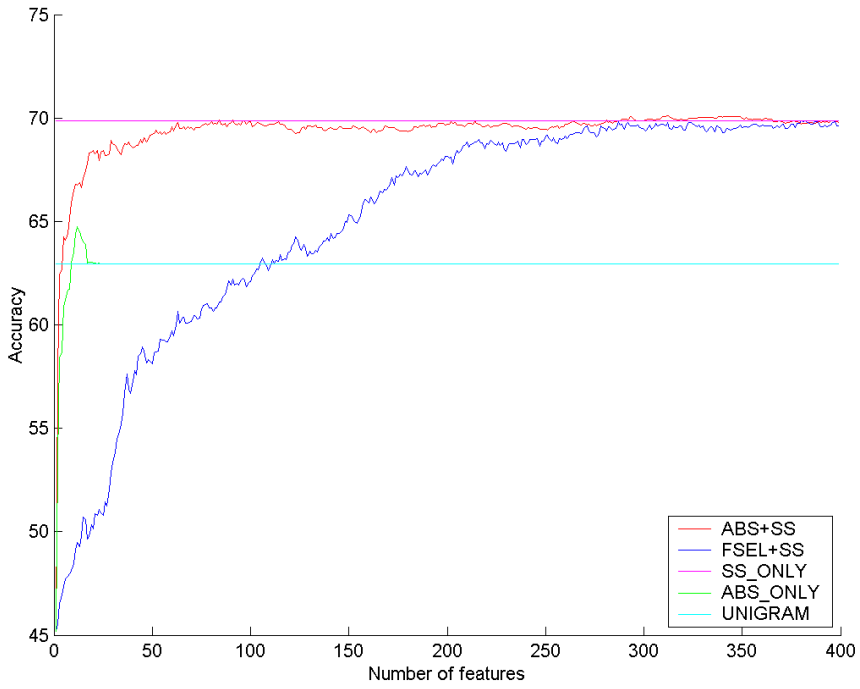



Figure 6.3 Comparison of the Abstraction + SuperStructuring method with other methods on the Eukaryotes dataset 2-grams. The graph represents cross-validation accuracy (5-fold) as a function of the number of features used. UNIGRAM uses 24 features; SS_ONLY 2-grams uses 400 features. ABS_ONLY varies in the range 1-24 features; ABS+SS and FSEL+SS vary in the range 1-400 features.

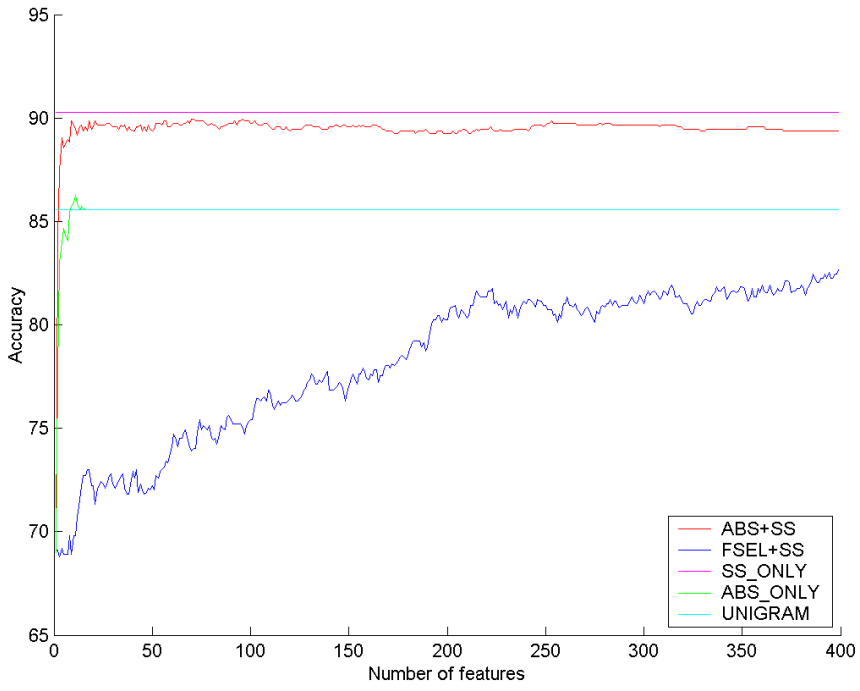



Figure 6.4 Comparison of the Abstraction + SuperStructuring method with other methods on the Prokaryotes dataset 3-grams. The graph represents cross-validation accuracy (5-fold) as a function of the number of features used. UNIGRAM uses 22 features; SS_ONLY 3-grams uses 8000 features. ABS_ONLY varies in the range 1-22 features; ABS+SS and FSEL+SS vary in the range 1-400 features.

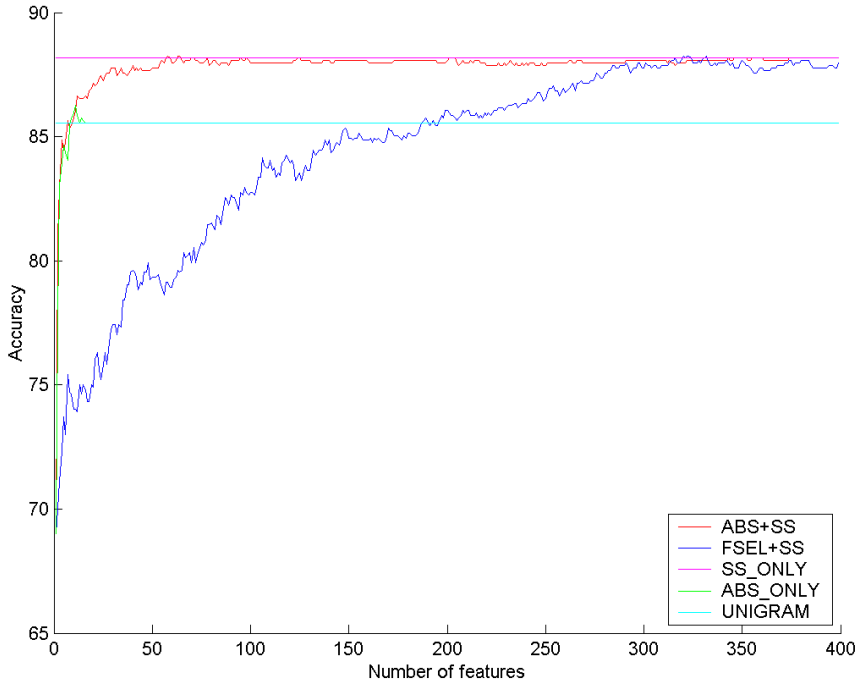



Figure 6.5 Comparison of the Abstraction + SuperStructuring method with other methods on the Prokaryotes dataset 2-grams. The graph represents cross-validation accuracy (5-fold) as a function of the number of features used. UNIGRAM uses 22 features; SS_ONLY 2-grams uses 400 features. ABS_ONLY varies in the range 1-22 features; ABS+SS and FSEL+SS vary in the range 1-400 features.

editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

[2] C. Andorf, A. Silvescu, D. Dobbs, and V. Honavar. Learning Classifiers for Assigning Protein Sequences to Gene Ontology Functional Families. In: *Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004)*. India. 2004.

[3] G. Casella, and R. L. Berger. *Statistical Inference*, Duxbury, 2002.

[4] E. Charniak. *Statistical Language Learning, Cambridge*. MIT Press, 1993.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.

[6] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley. 2001.

[8] S. Della Pietra, V. Della Pietra, and J. Lafferty, "Inducing features of random fields," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380-393, 1997.

[9] F Fleuret. Fast Binary Feature Selection with Conditional Mutual Information, *Journal of Machine Learning Research (JMLR)*, 5:1531–1555, 2004.

[10] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning* 29:131-163, 1997 .

[11] I. Jonyer, L. Holder and D. Cook, MDL-Based Context-Free Graph Grammar Induction and Applications, *International Journal of Artificial Intelligence Tools*, 13(1), pages 45-64, 2004.

[12] D.-K. Kang, A. Silvescu, J. Zhang, and V. Honavar. Generation of Attribute Value Taxonomies from Data for Accurate and Compact Classifier Construction. In: *IEEE International Conference on Data Mining (ICDM 2004)*, Brighton, UK, November 1, 2004.

[13] S. Kok, and P. Domingos. Statistical predicate invention. In *Proceedings of the 24th international Conference on Machine Learning (Corvalis, Oregon, June 20 - 24, 2007).* Z. Ghahramani, Ed. ICML '07, vol. 227. ACM, New York, NY, 433-440, 2007.

[14] S. Kramer. Predicate invention: A comprehensive view. *Technical Report.* Austrian Research Institute for Artificial Intelligence, Vienna, Austria, 1995.

[15] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. on Information Theory*, 37(1):145-151, January 1991.

[16] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis.* Academic Press, 1979.

[17] A. McCallum, R. Rosenfeld, T. Mitchell, and A. N. Ng. Improving text classification by shrinkage in a hierarchy of classes. *Proceedings of the 15th International Conference on Machine Learning - ICML'98.* 1998.

[18] R.S. Michalski. ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction, *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA, April, 2004.

[19] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[20] M. Pazzani. Constructive induction of Cartesian product attributes. *Information, Statistics and Induction in Science.* Melbourne, Australia.

[21] R. Parekh, J. Yang, and V. Honavar. (2000). Constructive Neural Network Learning Algorithms for Multi-Category Pattern Classification. *IEEE Transactions on Neural Networks.* 11(2):436-451.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, 1988.

[23] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. *Proceedings of the 31st ACL*, pp 183–190, 1993.

[24] J. R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[25] A. Silvescu, and V. Honavar. Temporal Boolean Network Models of Genetic Networks and Their Inference from Gene Expression Time Series. In: *Complex Systems*. 13(1):54-75, 2001.

[26] A. Silvescu, C. Andorf, D. Dobbs, and Honavar, V. Inter-element dependency models for sequence classification. *Technical Report*. June 2004.

[27] N Slonim and N Tishby. Agglomerative information bottleneck. In: *Advances in Neural Information Processing Systems (NIPS-12)*. 1999.

[28] A. Srinivasan and R.D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, volume 1314 of Lecture Notes in Artificial Intelligence, pages 89–104. Springer- Verlag, 1996.

[29] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In H. Liu and H. Motoda, editors, *Feature extraction, construction and selection*, pages 118–135. Kluwer Academic Publisher, 1998.

[30] J. M. Zelle and R. J. Mooney. Learning semantic grammars with constructive inductive logic programming. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 817–822). Washington, DC: AAAI, 1993.

[31] J. Zhang, A. Silvescu, and V. Honavar. Ontology-Driven Induction of Decision Trees at Multiple Levels of Abstraction. In: *Proceedings of Symposium on Abstraction, Reformulation, and Approximation (SARA 2002)*. 316-323. Berlin: Springer-Verlag. 2002.

# CHAPTER 7. INDEPENDENCE, DECOMPOSABILITY AND FUNCTIONS WHICH TAKE VALUES INTO AN ABELIAN GROUP

Modified from a paper published in *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*[1]

Adrian Silvescu[2] and Vasant Honavar

## Abstract

Decomposition is an important property that we exploit in order to render problems more tractable. The decomposability of a problem implies the existence of some "independences" between relevant variables of the problem under consideration. In this paper we investigate the decomposability of functions which take values into an Abelian Group. Examples of such functions include: probability distributions, energy functions, value functions, fitness functions, and relations. For such problems we define a notion of conditional independence between subsets of the problem's variables. We prove a decomposition theorem that relates independences between subsets of the problem's variables with a factorization property of the respective function. As particular cases of this theorem we retrieve the Hammersley-Clifford theorem for probability distributions; an Additive Decomposition theorem for energy functions, value functions, fitness functions; and a relational algebra decomposition theorem.

[1]Reprinted (no permission needed as the copyrights were not transfered) from *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics (AIMATH 2006).* Ft Lauderdale, Florida, 2006.

[2]Primary researcher and author - designed the framework and definitions, made the theoretical arguments and wrote the paper.

## 7.1 Introduction

Probabilistic Graphical Models (PGMs) have proved to be an effective way of representing probability distributions in a concise and intuitive form. Compact graphical representations support efficient reasoning and learning algorithms in many cases that arise in practice [12], [4], [8]. The key idea behind PGMs is the notion of probabilistic independence. Independence allows us to "decompose" the probability distribution into smaller parts, thereby substantially reducing the number of independent parameters that we need to know in order to specify the probability distribution.

Given the successful exploitation of independence in PGMs it is natural to ask whether we can define a more general case of independence with similar decomposability properties. Results concerning additive (rather than multiplicative) decomposability in various scenarios have been explored in [2], [9], [1]. Similarly, in the context of relational databases decomposition results such as MVD (Multi-Valued Decomposition) were obtained, see [15] and references therein.

In this paper, we seek a unifying thread of decomposability results. More precisely we identify a class of models more general than probabilities, namely: functions which take values into an Abelian group (denoted $\mathcal{F}_{\cdot \to AG}$) and we examine their decomposability properties. The class of models $\mathcal{F}_{\cdot \to AG}$ includes as particular cases: strictly positive probability distributions; additive decomposable functions and relations among others. In general we show that some essential results already known for probabilities will also hold for the more general class of models $\mathcal{F}_{\cdot \to AG}$. More exactly, we show that for $\mathcal{F}_{\cdot \to AG}$ we can define a general notion of Conditional Independence that is the natural generalization of probabilistic Conditional Independence. This General Conditional Independence will allow us to *"decompose"* a function $f$ from $\mathcal{F}_{\cdot \to AG}$ in the same way as a probability distribution can be "decomposed". More precisely, we prove a generalization of the well known Hammersley-Clifford theorem, which holds for arbitrary functions from $\mathcal{F}_{\cdot \to AG}$. Furthermore, we also prove a completeness theorem.

These results, in addition to unifying a broad class of decomposition results that have been previously proved in particular cases, should hopefully further the crossfertilization of results and techniques developed in settings that represent particular cases of $\mathcal{F}_{\cdot \to AG}$. One

particular case for example, aside from PGMs, is the analysis of fitness functions in Evolutionary Algorithms in order to improve search [10].

The rest of the paper is organized as follows: In Section 7.2 we introduce some definitions regarding Decomposition and Independence in a very general setting. In section 7.3 we shrink our domain of interest and define a General Conditional Independence concept for functions whose ranges are Abelian Groups: $\mathcal{F}_{\cdot \to AG}$. Subsequently, we explore some of the properties of this generalized independence relation. This section also contains the main result of this paper: a factorization property consisting in the natural generalization of the Hammersley-Clifford theorem [3] for arbitrary functions from $\mathcal{F}_{\cdot \to AG}$. We conclude section 7.3 with a completeness theorem that shows that four axioms completely characterize all the independence statements that hold for a function $f \in \mathcal{F}_{\cdot \to AG}$. In section 7.4 we present some important particular cases of decomposable functions which take values into an Abelian Group, such as: probability distributions, additive decomposable functions and relations. Section 7.5 concludes with a summary, discussion, and a brief outline of some directions for further research.

## 7.2 Decomposition and Independence

Decomposition is an key technique that makes the solution of otherwise complex problems tractable by the means of a divide and conquer approach. Decomposition exploits the fact that occasionally a problem can be split (*decomposed*) into subproblems which can be solved in isolation and then the overall solution can be obtained by aggregating the partial solutions. If such is the case, we say that the two parts are *independent* with respect to the problem under consideration.

The subproblems, in practice, are seldom disjoint, but all is not lost in this case, because we can define a weaker notion of independence, namely conditional independence, that is still useful.

Furthermore, if one or more of the subproblems are further decomposable into smaller parts, we can apply the same strategy of aggregating partial solutions, recursively. Thus, in a divide and conquer fashion, we would be able to obtain a solution for our problem by aggregating it

from solutions of smaller and smaller parts.

In what follows we will try to capture these intuitions underlying Decomposition and Independence with more precise definitions.

**Definition (problem):** A problem $P$ is a triple $P = (D, S, sol_P)$ where $D$ is a set called the *domain set*, $S$ is a set called the *solutions set* and $sol_P : D \to S$ is a function that maps an element $d$ from the domain of the problem to its solution $s$.

**Example:** $Determinant\_Computation(\mathcal{M}^2, \mathbb{R}, det)$ where $\mathcal{M}^2$ is the set of all square matrices, and *det* is the function that returns the determinant of a matrix.

**Definition (conditional independence - variable-based)** Given a problem $P = (D, S, sol_P)$ we say that, $D$ is decomposable into $A$ and $B$ conditioned on $C$, or equivalently, that $A$ is independent of $B$ conditioned on $C$, both with respect to the problem $P$, if $D = A \times B \times C$ (or more liberally $D$ is isomorphic with $A \times B \times C$) and there exist two problems $P_1 = (A \times C, S_{AC}, sol_{P_1})$ and $P_2 = (B \times C, S_{BC}, sol_{P_2})$ and an operator $\oplus_P^{P_1,P_2} : S_{AC} \times S_{BC} \to S_D$ such that for all $d = (a, b, c) \in A \times B \times C$ we have:

$$sol_P(d = (a, b, c)) = sol_{P_1}(a, c) \oplus_P^{P_1,P_2} sol_{P_2}(b, c)$$

where $\oplus_P^{P_1,P_2} : S_{AC} \times S_{BC} \to S_D$. ■

The rest of this paper is concerned with variable-based independence. More precisely, we will consider the particular case where the operator $\oplus_P^{P_1,P_2}$ is an operator $\oplus_P^{P_1,P_2} : G \times G \to G$ . That is, the solutions to the problems $P, P_1, P_2$ are elements from the same set $G$ . Furthermore, we will assume that the operator $\oplus_P^{P_1,P_2}$ does not depend on the problems $P, P_1, P_2$ and therefore we can drop them from the notation yielding a single operator: $\oplus : G \times G \to G$. Lastly, we will consider $G$ to be an Abelian Group for reasons that will become apparent later on.

The main reason for studying this more particular scenario is to be able to obtain results characterizing the phenomenon. In the very general setup given by the previous definition, while very interesting in itself, little can be said from the mathematical point of view aside from the divide and conquer strategy already outlined.

## 7.3 Independence & Decomposition in $\mathcal{F}_{\cdot\to AG}$

In this section we consider Independence and Decomposability of functions whose ranges are Abelian Groups: $\mathcal{F}_{\cdot\to AG}$. After defining Abelian Groups (7.3.1), we introduce the definition of Conditional Independence with respect to a function $f$, - $I_f(\cdot,\cdot|\cdot)$ (7.3.2) (Note that for probabilistic independence the function $f$ is the very probability distribution). We then show that $I_f(\cdot,\cdot|\cdot)$ satisfies four properties that are considered as essential/defining for the notion of independence, (e.g., [12]) (7.3.3). These properties are: trivial independence, symmetry, weak union and intersection. We then recapitulate the main results already existing in the literature, (e.g., [6]), regarding Conditional Independence relations that satisfy these four properties (7.3.4). The main target of these results is to establish the equivalence between conditional independence and graph separability (Note: the graph in question is obtained by not drawing an edge between two variables whenever they are independent of each other given the rest of the variables, and drawing one otherwise). We then present the main theorem which allows us to "boil down" a set of pairwise conditional independences of the form $I_f(A,B|C)$ into a global decomposition of the function $f$ over the maximal cliques of the associated independence graph. This theorem is the natural generalization of the Hammersley-Clifford theorem for probability distributions, to the more general case of $\mathcal{F}_{\cdot\to AG}$. We conclude this section with a completeness theorem which states that the four abovementioned properties are a complete axiomatic characterization of independences that hold for a function $f$ whose range is an Abelian group $f\in\mathcal{F}_{\cdot\to AG}$ (7.3.5).

### 7.3.1 Abelian Groups

In this subsection we give the definition of Abelian Groups (a.k.a. commutative groups), followed by some examples that we will use later on as representative particular cases for our theory.

**Definition. (Abelian Group)** An Abelian Group is a quadruple $(G,\oplus,\theta,\ominus)$ where $G$ is a nonempty set, $\oplus : G\times G\to G$ is a binary operation over elements from $G$ that returns an element of $G$, $\theta\in G$ and $\ominus : G\to G$ is a unary operation over the elements of $G$ that returns

an element of $G$, with the following properties:

1. $\oplus$ is associative i.e., $\forall g_1, g_2, g_3 \in G\ (g_1 \oplus (g_2 \oplus g_3)) = ((g_1 \oplus g_2) \oplus g_3)$
2. $\oplus$ is commutative i.e., $\forall g_1, g_2 \in G\ g_1 \oplus g_2 = g_2 \oplus g_1$
3. $\theta$ is an identity element i.e., $\forall g \in G\ g \oplus \theta = \theta \oplus g = g$
4. $\ominus$ is an inversion operator i.e., $\forall g \in G\ \exists! h \in G\ g \oplus h = h \oplus g = \theta$. We will call $\ominus g$ the unique $h$ with the previous property. Subsequently, the inversion property can be written as $\forall g \in G\ \exists! \ominus g \in G\ s.t\ g \oplus \ominus g = \ominus g \oplus g = \theta$ ■

**Examples:** 1. $(\mathbb{R}, +, 0, -)$ is a group, where: $\mathbb{R}$ is the set of real numbers; $+$ is the addition between real numbers; 0 is zero; and $-$is the unary operator minus that returns the inverse of a real number (e.g., $-(7) = -7$ and $-(-6) = 6$). In more common notation we use $-$ as a binary operator where $a - b$ actually stands for $a + -b$.

2. $((0, \infty), \cdot, 1, {}^{-1})$ is a group. where: $\cdot$ is the multiplication between real numbers; 1 is one; and ${}^{-1}$is the inverse of a real number with respect to multiplication (i.e, $a^{-1} = \frac{1}{a}$ ). In more common notation we use fractions as binary operators therefore having expressions such as $\frac{a}{b}$, which stands for $a \cdot b^{-1}$.

3. $(\mathbb{R}, \cdot, 1, {}^{-1})$ is not a group. This is because 0 has no inverse.

4. $\mathbb{Z}_2 = (\{0, 1\}, \otimes, 0, -)$ is a group where $\otimes$ stands for the Exclusive OR (XOR) operation (or equivalently, addition modulo 2). More exactly: $0 \otimes 0 = 0$; $0 \otimes 1 = 1$;$1 \otimes 0 = 1$ and $1 \otimes 1 = 0$, and $-$ is the identity operator, that is: $-0 = 0$; and $-1 = 1$. ■

For the purposes of simplifying notation, for the rest of this section, we will use as operators the standard operations of the Additive Abelian Group instead of the fancier ones that we have introduced in the definition of a group. More precisely, instead of saying: let $(G, \oplus, \theta, \ominus)$ be a group ..., we will say: let $(G, +, 0, -)$ be a group ... . This will make the definitions and proofs look more familiar since they are in additive notation. However the only properties that we will use are those of groups and as a consequence all the results will hold for arbitrary groups, such as, for example, the multiplicative or the $\mathbb{Z}_2$ group. Additionally, we will also use the

shorthand notation of $a \ominus b$ to stand for $a \oplus \ominus b$, which in our familiar additive notation, to be used from now on, will be nothing but $a - b$ (which stands for $a + -b$).

Abelian groups defined, we are ready to introduce the central concept of the paper: Conditional Independence with respect to a function $f$.

### 7.3.2 Conditional Independence with respect to a function $f$

We now proceed to define a general notion of Conditional Independence with respect to a function $f$, $I_f(\cdot, \cdot|\cdot)$. This independence relation is basically a formalization of the intuition behind decomposition presented in section 7.2: namely, that independence should allow us to decompose a problem into subproblems, solve them separately and then combine the results. We start with some notations (following [11]), and then present the definition and some illustrative examples.

**Notation.** Let $(X_\alpha)_{\alpha \in V}$ stand for a collection of variables that take values into the spaces $(\mathcal{X}_\alpha)_{\alpha \in V}$, where $V$ is a set of indices for these variables. For a subset $A$ of $V$ let $\mathcal{X}_A = \times_{\alpha \in A} \mathcal{X}_\alpha$ and in particular let $\mathcal{X}$ stand for $\mathcal{X}_V$. The collection $(X_\alpha)_{\alpha \in V}$ represents the relevant variables pertaining to our problem. $\mathcal{X}_A$ will stand for the set of all possible configurations for the variables indexed by $A$. Typical elements of $\mathcal{X}_A$ will be denoted as $x_A = (x_\alpha)_{\alpha \in A}$. Similarly, $X_A$ will stand for $(X_\alpha)_{\alpha \in A}$ and $X$ will stand for $X_V$. Given sets of variable indices $A, B, C \subseteq V$ we will assert conditional independence statements regarding the associated subsets of variables $X_A$,$X_B$,$X_C$ such as: the sets of variables $X_A$ and $X_B$ are independent conditioned on $X_C$ and write $I(X_A, X_B|X_C)$. We will actually use the shorthand formulation, $A$ and $B$ are independent conditioned on $C$, and the shorthand notation $I(A, B|C)$ to stand for $I(X_A, X_B|X_C)$.

**Definition (Conditional Independence with respect to a function $f$ - $I_f(\cdot, \cdot|\cdot)$):** Let $(G, +, 0, -)$ be an Abelian Group, $(X_\alpha)_{\alpha \in V}$ be a collection of variables indexed by $V$ and $\mathcal{X} = \times_{\alpha \in V} \mathcal{X}_\alpha$ be the set of configurations for these variables. Let $f : \mathcal{X} \to G$ be a function from the set $\mathcal{X}$ to $G$. Furthermore, let $A, B, C \subseteq V$ a partition of $V$ (hence $\mathcal{X} = \mathcal{X}_A \times \mathcal{X}_B \times \mathcal{X}_C$). Then we say that $A$ is independent of $B$ conditioned on $C$ with respect to the function $f$, and

we write $I_f(A, B|C)$, if there exist two functions $f_1, f_2$ such that:

$$f(X) = f(X_A, X_B, X_C) = f_1(X_A, X_C) + f_2(X_B, X_C)$$

where $f_1 : \mathcal{X}_A \times \mathcal{X}_C \to G$ and $f_2 : \mathcal{X}_B \times \mathcal{X}_C \to G$.

Instead of the previous formula we will use the shorthand notation

$$f(V) = f(A, B, C) = f_1(A, C) + f_2(B, C)$$

■

Note that in our notion of conditional independence just defined, $A, B, C$ is necessarily a partition of $V$ as opposed to the case of probabilistic independence where it is possible that $A, B, C$ do not cover $V$ . In our (general) case, if $A, B, C$ do not cover $V$ then $f(A, B, C)$ is not necessarily defined. In the case of probability distribution there is a natural way to define $f(A)$ when $A \subsetneqq V$ based on $f(V)$. That is, by the means of marginals. In the more general cases that we will study, (e.g., additive independence) the equivalent notion of a marginal is not necessarily present. As a consequence the theory developed in this context will be weaker, and hence more general. Following the terminology of [6] independence statements $I(A, B|C)$ such that $A, B, C$ cover $V$, will be called *saturated* independence statements.

Examples of Conditional Independence with respect to a function $f$ , $I_f(\cdot, \cdot|\cdot)$:

**Probabilistic:** $I_f(\cdot, \cdot|\cdot)$: In the case when the group is $((0, \infty), \cdot, 1, {}^{-1})$ and furthermore the function $f : \mathcal{X} \to (0, \infty)$ is a probability distribution (that is, $\sum_{x \in X} f(x) = 1$, or more generally $\Delta_X \, df = \Delta_X \, f(x)dx = 1$) then our notion of conditional independence $I_f(\cdot, \cdot|\cdot)$ becomes probabilistic conditional independence. Note that this group includes strictly positive probabilities only, in order to satisfy the group property (0 has no inverse element). $I_f(A, B|C)$ in this case is equivalent with:

$$f(A, B, C) = f_1(A, C) \cdot f_2(B, C)$$

obtained by substituting in the definition the original + with the probabilistic group binary operator $\cdot$. This is an alternative definition for probabilistic conditional independence, see [11]. Thus, we have just shown that probabilistic conditional independence is a particular case of

Conditional Independence with respect to a function $f$ , when $f$ happens to be a probability distribution.

**Additive:** $I_f(\cdot,\cdot|\cdot)$: In the case the group is $(\mathbb{R},+,0,-)$ and we have a function $f:\mathcal{X}\to\mathbb{R}$ we obtain the notion of Additive Independence i.e.:

$$f(A,B,C)=f_1(A,C)+f_2(B,C)$$

**Relational:** $I_f(\cdot,\cdot|\cdot)$: In the case the group is $\mathbb{Z}_2=(\{0,1\},\otimes,0,-)$ and we have a function $f:\mathcal{X}\to\mathbb{Z}_2$ (a.k.a. relation) we obtain:

$$f(A,B,C)=f_1(A,C)\otimes f_2(B,C)$$

Given our definition of Conditional Independence $I_f(\cdot,\cdot|\cdot)$, which was derived in an intuitive and model-driven way, it is natural to ask how compatible it is with various axiomatic frameworks that have been proposed to characterize the phenomenon of Conditional Independence - dating back at least since [5]. In the next section we will state and prove the most important axioms/properties that hold for $I_f(\cdot,\cdot|\cdot)$, while postponing a more detailed analysis and comparison with other axiomatic frameworks to the discussion section.

### 7.3.3 Properties of $I_f(\cdot,\cdot|\cdot)$

In this section we prove some properties associated with $I_f(\cdot,\cdot|\cdot)$. These are general properties that researchers [5], [13], [12], [6], [4] have identified as desirable for conditional independence relations because they capture some intuitive notions that pertain to independence. We will show that the our Conditional Independence relation with respect a function $f$ - $I_f(\cdot,\cdot|\cdot)$ satisfies these properties, thus providing supporting evidence that this is the "right" concept .

**Theorem 1 (independence properties):** *Let $(G,+,0,-)$ be an Abelian Group, $(X_\alpha)_{\alpha\in V}$ be a collection of variables indexed by $V$, $f:\mathcal{X}\to G$ be a function from the set $\mathcal{X}$ to the group $G$, and $A,B,C,D$ be subsets of $V$. Then the Conditional Independence relation with respect to $f$, $I_f(\cdot,\cdot|\cdot)$ has the following properties:*

1. **(Trivial Independence)** $I_f(A,\emptyset|B)$ - $\forall A,B$ *a partition of* $V$.

2. **(Symmetry)** $I_f(A, B|C)$ iff $I_f(B, A|C)$ - $\forall A, B, C$ *a partition of* $V$.

3. **(Weak Union)** $I_f(A, B \cup D|C) \Rightarrow I_f(A, B|C \cup D)$ - $\forall A, B, C, D$ *a partition of* $V$.

4. **(Intersection)** $I_f(A, B|C \cup D)$ & $I_f(D, B|C \cup A) \Rightarrow I_f(A \cup D, B|C)$ - $\forall A, B, C, D$ *a partition of* $V$. ■

*Proof.* See Appendix 7.6.1■

Note that: in order to prove **Trivial Independence** we need the identity element property of the Abelian group $(G, +, 0, -)$; to prove **Symmetry** we needed commutativity; and to prove **Intersection** we needed associativity and most importantly the inverse operator. So it seems that we "need" all the Abelian Group properties.

As a consequence of the abovementioned properties holding we can obtain for free some results that are already known to hold in frameworks that use these properties as axioms. The survey of such results, mainly involving the relationship of independence properties with graph theoretic properties that hold in the induced independence graph, will be the purpose of the next subsection. What we cannot get for free are model-based results such as the equivalence of such independence properties with a factorization property of the model (in our case the function $f$), or completeness theorems. Such model-theoretic results depend upon the class of models under consideration and will be the subject of the rest of this section.

The axiomatic frameworks for characterizing Conditional Independence cited at the beggining of this subsection were mainly developed with the probabilistic model in mind. From this point of view the question that we ask in this paper is whether there exist a more general class of models where a notion of Conditional Independence "makes sense". We have already seen that we can talk about additive and relational Conditional Independence in a well defined way as particular cases of Conditional Independence with respect to a function $f$. Our strive for generality however may force us to drop some of the axioms that were holding in the probabilistic case. More exactly we have seen that in the general case we shall restrict ouselves to *saturated* independence statements (i.e., where the variables involved form a partition of the whole set) because non-saturated statements depend upon a well defined notion of marginals

exiting in the model (the function $f$) and such a notion is an idiosyncratic concept pertaining to probabilities but not necessary existent in the case of additive functions, and in general for that matter. Under such constraints we will show that properties 1.-4. interpreted as axioms are a complete characterization of the *saturated* Conditional Independences that hold with respect to a function over an Abelian Group. Such a completeness theorem along with the fact that in order for these properties to hold in the model we needed it to be a function which takes values into an Abelian group make a strong argument for the fact the properties 1.-4. are a strong "axiomatic core" for Conditional Independence in quite a general setup, with functions that take values into an Abelian group as its set of models. Such an argument is furthermore strengthen by the main theorem of this paper: which establishes equivalence of the Markovian graph theoretic properties of the independence graph with a factorization property of the model (the function $f$ which ranges into an Abelian Group) under the assumption of 1.-4..

Therefore the answer to our question will be: Yes, there exists a more general set of models where a notion Conditional Independence "makes sense" (with respect to saturated independence statements) and this set is $\mathcal{F}_{\_\to AG}$ (functions whose ranges are Abelian groups), and 1.-4. are a complete axiomatic characterization of the (saturated) independecnes that hold in these models.

The rest of this section is organized as follows: In the next subsection we survey the results that come for free as a consequence of properties 1.-4. holding. Subsequently, in the next two subsections we present the model dependent results: the factorization theorem and the completeness theorem, respectively.

### 7.3.4 Markovian Properties of Independence

We start with a survey some known results regarding Conditional Independence relations satisfying the above-mentioned four properties [6]. We first introduce some graph terminology, then define different types of Markov properties and subsequently, establish their equivalence. We end with a theorem that states the equivalence between graph separability and conditional independence. All results hold under the assumptions of: trivial independence, symmetry, weak

union and intersection.

**Graph notions**: A *graph* is a pair $\mathcal{G} = (V, E)$ where $V$ is a finite set of vertices and $E$ is a set of edges. That is, $E$ is set of pairs of vertices $E \subseteq V \times V$ . A graph is called *undirected* if it has the property that for every $\alpha, \beta \in V$ $(\alpha, \beta) \in E$ if and only if $(\beta, \alpha) \in E$. Thus for the case of undirected graphs there is no distinction between the edges $(\alpha, \beta)$ and $(\beta, \alpha)$ and we will use them interchangeably to mean the same thing, namely an undirected edge between $\alpha$ and $\beta$. In what follows we will only consider undirected graphs.

A graph $\mathcal{G} = (V, E)$ is called complete iff there is an edge between all of its vertices. A *subgraph* of a graph $\mathcal{G} = (V, E)$ associated with set of vertices $V'$, $V' \subseteq V$, is a graph $\mathcal{G}' = (V', E')$ such that $E' = E \cap (V' \times V')$. A set of vertices $C \subseteq V$ is called a *clique* in the graph $\mathcal{G} = (V, E)$ if the subgraph of $\mathcal{G}$ associated with $C$ is a complete graph. That is, there is an edge between every two vertices in $C$ in the graph $\mathcal{G}$. A clique $C$ is called a *maximal clique* of $\mathcal{G}$ if there is no other clique $C'$ in the graph $\mathcal{G}$ such that $C \subset C'$. Given a graph $\mathcal{G} = (V, E)$ we will use $MaxCliques(\mathcal{G})$ to denote the set of maximal cliques of $\mathcal{G}$.

Given a set $A \subseteq V$ we denote by $\mathcal{N}(A)$ and call *neighbourhood of* $A$ the set of vertices from $V \backslash A$ that share at least one edge with an element in $A$. More precisely, $\mathcal{N}(A) = \{\beta | \beta \notin A \text{ and } \exists \alpha \in A \text{ such that } (\alpha, \beta) \in E\}$.

Given two vertices $\alpha, \beta \in V$ we say that there exists a *path* between $\alpha$ and $\beta$ if there exists a set of vertices $\gamma_1, ..., \gamma_k$, $k \geq 0$ such that $(\alpha, \gamma_1), (\gamma_i, \gamma_{i+1}), (\gamma_k, \beta) \in E$ $\forall 1 \leq i < k$. We will call the sequence $\alpha, \gamma_1, ..., \gamma_k, \beta$ the path from $\alpha$ to $\beta$. Furthermore, given three subsets of vertices $A, B, C \subseteq V$ we say that $C$ *separates* $A$ *from* $B$ *in the graph* $\mathcal{G} = (V, E)$ if there is no path between a vertex in $A$ to a vertex in $B$ that does not contain vertices from $C$. We will use $Sep_{\mathcal{G}}(A, B|C)$ to denote the fact that $C$ separates $A$ from $B$ in the graph $\mathcal{G} = (V, E)$.

**Definition (Markov properties):** [12], [11] Let $\mathcal{G} = (V, E)$ be an undirected graph where $V$ is a set of indices into a collection of variables $(X_\alpha)_{\alpha \in V}$. Then we say that the conditional independence relation has the following properties relative to the graph $\mathcal{G}$ iff:

1. (P) *Pairwise Markov Property* relative to $\mathcal{G}$ iff $(\alpha, \beta) \notin E \Rightarrow I(\alpha, \beta | V \backslash \{\alpha, \beta\})$ .

2. (L) *Local Markov Property* relative to $\mathcal{G}$ iff $\forall \alpha \in V$ $I(\alpha, V \backslash (\{\alpha\} \cup \mathcal{N}(\alpha))) | \mathcal{N}(\alpha))$ .

3. (G) *Global Markov Property* relative to $\mathcal{G}$ iff for any two sets $A, B \subseteq V$ such that $V\backslash(A \cup B)$ separates $A$ and $B$ in the graph $\mathcal{G}$ we have $I(A, B|V\backslash(A \cup B))$. ■

It turns out that the previous three relations are equivalent for any independence relation satisfying properties 1-4 of the previous section (trivial independence, symmetry, weak union and intersection).

**Theorem 2 (Markov properties equivalence):** [13] (G) $\Leftrightarrow$ (L) $\Leftrightarrow$ (P) *for any conditional independence relation* $I(\cdot, \cdot|\cdot)$ *that satisfies* ***Trivial independence, Symmetry, Weak union*** *and* ***Intersection***. ■

*Proof.* This theorem has been proved in[13] and can also be found in [12], [11], [8]. Note that the Global Markov property is slightly weaker in our case because we have only saturated independence and hence we cannot pick arbitrary sets that separate $A$ and $B$ instead of just $X\backslash(A \cup B)$ . Nevertheless the equivalence still holds. See Appendix 7.6.2 for a complete proof. ■

**Corollary.** *In particular:* (G) $\Leftrightarrow$ (L) $\Leftrightarrow$ (P) *for* $I_f(\cdot, \cdot|\cdot)$. ■

**Definition (closure):** Let $(X_\alpha)_{\alpha \in V}$ be a collection of variables indexed by $V$ , $\Sigma$ be an arbitrary set of independence statements of the form $I(A, B|C)$, where $A, B, C$ is a partition of $V$, and $\mathcal{A}$ a set of axioms. We denote by $\Sigma^+$ the set of all independence statements that can be inferred from the independence statements in $\Sigma$ in a finite number of steps by using only axioms from the set $\mathcal{A}$. If such is the case, we call $\Sigma^+$ *the closure of* $\Sigma$ *under the axioms* $\mathcal{A}$.■

**Definition (associated independence graph):** Given a set $\Sigma$ of pairwise conditional independence statements $I(\alpha, \beta|V\backslash\{\alpha, \beta\})$, a graph $\mathcal{G}(\Sigma) = (V, E)$ is called the *associated independence graph* if $(\alpha, \beta) \notin E \Leftrightarrow I(\alpha, \beta|V\backslash\{\alpha, \beta\}) \in \Sigma$. In general given a set $\Sigma$ of not necessarily pairwise conditional independence statements we define the set $\Sigma_{pairwise}$ as the set of of all pairwise independence statements that can be inferred from $\Sigma$ using the ***Trivial independence, Symmetry, Weak union*** *and* ***Intersection*** axioms (i.e., all pairwise Independence statements from the closure of $\Sigma$, - $\Sigma^+$ ). Furthermore we define the associated dependence graph $\mathcal{G}(\Sigma)$ of such a general set of conditional independence statements $\Sigma$ as the associated dependence graph of $\Sigma_{pairwise}$. ■

**Theorem 3 (separability $\Leftrightarrow$ conditional independence):** [6] *Let $\Sigma$ be a set of saturated independence statements over a finite set of variables $(X_\alpha)_{\alpha\in V}$ indexed by elements from $V$. Let $\Sigma^+$ be the closure of $\Sigma$ with respect to saturated trivial independence, symmetry, intersection and weak union. And let $\mathcal{G}(\Sigma^+)$ the dependence graph associated with set of pairwise independence statements in $\Sigma^+$. Then for any $A, B, C$ partition of $V$ we have:*

$$Sep_{\mathcal{G}(\Sigma^+)}(A, B|C) \;\Leftrightarrow\; I(A, B|C) \in \Sigma^+$$

*Proof.* See [6] Theorem 13 for a proof of this theorem. ■

**Corollary:** *In particular $I_f(\cdot,\cdot|\cdot)$ satisfies the equivalence between graph separability and Conditional Independence stated in the previous theorem.*

So far we have seen that any set $\Sigma$ of Conditional Independence statements produces a graph $\mathcal{G}(\Sigma^+)$ such that separability in this graph is equivalent to Conditional Independence in the closure of $\Sigma$. If the Independence relation in question is $I_f(\cdot,\cdot|\cdot)$ we have furthermore that $Sep_{\mathcal{G}(\Sigma^+)}(A, B|C) \Leftrightarrow I_f(A, B|C) \in \Sigma^+ \Leftrightarrow f(A, B, C) = f_1(A, C) + f_2(B, C)$. We will next prove the main result of the paper, namely, a theorem that will allow us to "compile" pairwise decompositions that are implied by conditional independence statements between two sets of variables conditioned on a third one, of the form $I_f(A, B|C) \in \Sigma^+ \Rightarrow f(A, B, C) = f_1(A, C) + f_2(B, C)$ into a "finer" decomposition over the maximal cliques of the associated graph $\mathcal{G}(\Sigma^+)$. In other words, this theorem shows that if the four properties are satisfied, we can "boil down" a set of pairwise decompositions to one "holistic" decomposition over the maximal cliques of the associated graph $\mathcal{G}(\Sigma^+)$.

### 7.3.5 The factorization theorem

We now proceed to state the theorem that ties the Conditional Independence relation with respect to a function $f$, $I_f(\cdot,\cdot|\cdot)$, with a factorization property of the function $f$ over the maximal cliques of the associated graph.

**Definition (factorization property)**: Let $\mathcal{G} = (V, E)$ be an undirected graph, let $(G, +, 0, -)$ be a group, $(X_\alpha)_{\alpha\in V}$ be a collection of variables indexed by $V$ and $f : \mathcal{X} \to G$

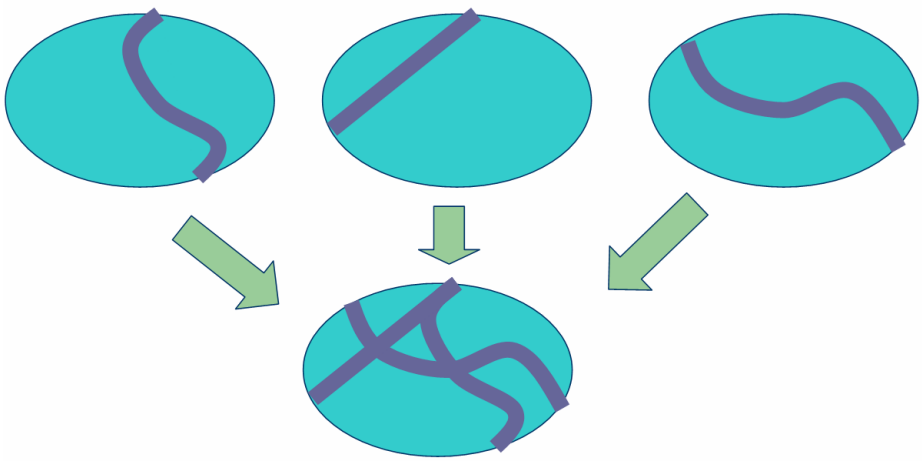


Figure 7.1 Illustration of the intuition behind the factorization theorem: A set of pairwise, rougher decompositions are assembled into one holistic, finer decomposition over the maximal cliques of the induced independence graph.

be a function from the set $\mathcal{X}$ to the group $G$. We say that $f$ satisfies the factorization property (F) with respect to the graph $\mathcal{G}$ iff there exist a collection of functions $\{f_C : \mathcal{X}_C \rightarrow G\}_{C \in MaxCliques(\mathcal{G})}$

$$(F) \qquad f(V) = \sum_{C \in MaxCliques(\mathcal{G})} f_C(C)$$

**Theorem 4 (factorization):** *Let $\mathcal{G} = (V, E)$ be an undirected graph, $(G, +, 0, -)$ be an Abelian Group, $(X_\alpha)_{\alpha \in V}$ be a collection of variables indexed by $V$, $f : \mathcal{X} \rightarrow G$ be a function from the set $\mathcal{X}$ to the group $G$. Let $I_f(\cdot, \cdot|\cdot)$ conditional indepedence relation induced by the function $f$ . Then* (G) $\Leftrightarrow$ (L) $\Leftrightarrow$ (P) $\Leftrightarrow$ (F)*, where all the Markov properties are with respect to $I_f(\cdot, \cdot|\cdot)$.* ■

*Proof.* See the Appendix 7.6.3 for the proof.■

An illustration of the intuition behind the factorization theorem is shown in Figure 1.

In the particular case when $f$ is a probability distribution the last implication in the previous theorem ((P) $\rightarrow$(F)) is known as the Hammersley-Clifford theorem [3]. The proof technique based on the Moebius Inversion Lemma which we also use was first used for proving the Hammersley-Clifford theorem for probabilities in [7], see also [7], [8]. To the best of our knowledge, the proof presented here is the first proof that holds for the general case of conditional

independence with respect to a function $f$ which ranges into an Abelian Group.

### 7.3.6 Completeness

In this section we state a completness theorem which states that the axioms of *trivial independence, symmetry, intersection and weak union* are a complete characterization for the independencies that hold over functions which take values into an Abelian Group $G$.

**Definition (satisfaction):** Let $f : \mathcal{X} \to G$ be a function that takes values into an Abelian Group $(G, +, 0, -)$ and let $A, B, C$ is a partition of $V$. We say that $f$ satisfies a conditional independence statement $\sigma = I(A, B|C)$ if $I_f(A, B|C)$.

**Definition (Markov Network):** A graph $\mathcal{G} = (V, E)$ is called a Markov Network for a function $f : \mathcal{X} \to G$ iff for all partitions into three sets of $V$, $A, B, C$ we have $Sep_{\mathcal{G}}(A, B|C) \Rightarrow I_f(A, B|C)$.

**Theorem 5 (completeness):** *Let $f : \mathcal{X} \to G$ be a function that takes values into an Abelian Group $(G, +, 0, -)$ . Let $\Sigma$ be a set of saturated independence statements over sets in $V$, and let $\Sigma^+$ be the closure of $\Sigma$ with respect to trivial independence, symmetry, intersection and weak union. Then for every $\sigma \notin \Sigma^+$, there exists a function $f : \mathcal{X} \to G$ such that $f$ satisfies $\Sigma^+$ and does not satisfy $\sigma$.* ■

*Proof.* This is the generalization for Abelian groups of the existing completeness proof for probablity distributions from [6]. See the Appendix 7.6.5 for the proof.■

Note that if the group $G$ has only one element then there exists only one function from any domain to $G$ (the one that maps all the elements of the domain to the unique element in $G$) and therefore the problem of completeness is vacuous.

The **(completeness)** theorem along with the **(separability ⇔ conditional independence)** theorem allow us to determine conditional independence statements relative to the function $f$ by the means of examining separation in the associated graph see [6] for more details.

## 7.4 Particular Cases

We now review some important examples of functions over particular Abelian Groups and the associated factorization theorems.

**Probability Theory** In the case when we consider functions $f : \mathcal{X} \to (0, \infty)$ where the group is $((0, \infty), \cdot, 1, {}^{-1})$ and additionally we impose the constraint that $\sum_{x \in X} f(x) = 1$, or more generally $\Delta_X \, df = \Delta_X \, f(x)dx = 1$, we obtain strictly positive probability distributions and the notion of conditional independence becomes probabilistic conditional independence. By the factorization theorem with respect to an associated graph $\mathcal{G}$ we can decompose the probability distribution in terms of clique potentials $f_C$ as:

$$f(V) = \prod_{C \in MaxCliques(\mathcal{G})} f_C(C)$$

This is precisely the Hammersley-Clifford theorem [3], [7], [11], [8].

**Additive Decomposability / Value Theory** When we consider functions $f : \mathcal{X} \to \mathbb{R}$ where the group is $(\mathbb{R}, +, 0, -)$ we obtain an additive decomposition of the function $f$ over the maximal cliques of the associated graph $\mathcal{G}$.

$$f(V) = \sum_{C \in MaxCliques(\mathcal{G})} f_C(C)$$

This decomposition theorem can be used to decompose value functions or fitness functions. A set of theorems in the same spirit, while not in the same framework are the utility decomposition theorems. See [1] and references therein.

**Relational Algebra** A relation is a function $r : \mathcal{X} \to \{0, 1\}$. If we consider the group $\mathbb{Z}_2 = (\{0, 1\}, \otimes, 0, -)$ we can decompose any relation $r$ in terms of smaller relations defined over subsets of $V$. In this case the factorization theorem with respect to an associated graph $\mathcal{G}$ will be:

$$r(V) = \bigotimes_{C \in MaxCliques(\mathcal{G})} r_C(C)$$

## 7.5 Conclusions and Discussion

**Review**: In this paper, we have introduced a general notion of Conditional Independence / Decomposability. Following the intuitions derived from the general case, we introduced the notion of Conditional Independence relative to a function $f$ which takes values into an Abelian Group, - $I_f(\cdot, \cdot|\cdot)$. We then proved that $I_f(\cdot, \cdot|\cdot)$ satisfies the following four properties: trivial independence, symmetry, weak union and intersection, which are held to be important properties for the notion of independence [12]. As a consequence, we obtained the equivalence of the Global, Local and Pairwise Markov Properties for $I_f(\cdot, \cdot|\cdot)$, as well as the equivalence between Conditional Independence and Graph Separability in the associated graph, based on well known results e.g., [6]. We then proved the main theorem of this paper, which allows us to "boil down" a set of pairwise Conditional Independences (and consequently pairwise factorizations) of the function $f$ to a "global" factorization of $f$ over the maximal cliques of the associated independence graph. This theorem is the natural generalization of the Hammersley-Clifford theorem which holds for probability distributions, to the more general case of functions that take values into an Abelian Group. The theory developed in this paper subsumes: factorization of probability distributions, additive decomposable functions and decomposable relations, as particular cases of functions over Abelian Groups.

**Discussion**: In contrast with the more traditional framework for probabilistic independence i.e., graphoids [12], our notion of independence does not support contraction $[I(A, D|C)\ \&\ I(A, B|C \cup D) \Rightarrow I(A, B \cup D|C)]$ and decomposition $[I(A, B \cup D|C) \Rightarrow I(A, B|C)]$. Decomposition is essentially a way to support non-saturated independence statements and hence it requires marginals, but marginals are not generally available since they appear to be an idiosyncratic feature of probability distributions. Thus it is understandable that we had to drop decomposition in our quest for generality. Contraction had to be dropped as well because one of its premises contains a non-saturated statement. A weaker version however, called Weak Contraction $[I(A \cup B, D|C)\ \&\ I(A, B|C \cup D) \Rightarrow I(A, B \cup D|C)]$ that was used in [6] does not require non-saturated statements and is satisfied by our independence relation $I_f(\cdot, \cdot|\cdot)$. This follows from the observation that Weak Union and Intersection imply Weak Contraction (see

Appendix 7.6.6). The Intersection property is sometimes dropped from probabilistic independence axioms in order to obtain semi-graphoids [12]. Both the graphoid and semi-graphoid sets of axioms however have been shown to be incomplete for non-saturated probabilistic conditional independence statements [14]. In other words [14] shows that any finite set of axioms is an incomplete characterization of non-saturated probabilistic conditional independence statements. In our case, since we use only saturated independence statements due to the unavailability of marginals in the general case, a completeness theorem can be proved. We proved a completeness theorem which states that the axioms of *trivial independence, symmetry, intersection and weak union* are a complete characterization of Conditional Independences that hold for a function $f$ which ranges into an Abelian Group. This theorem is a natural generalization, for functions over Abelian groups, of a completeness theorem for positive probability distributions and the four above-mentioned axioms in a saturated probabilistic conditional independence setup, that was previously obtained by [6].

**Conclusion**: In conclusion we have indentified a larger set of models - functions which range over Abelian groups, for which a well defined notion of *saturated* Conditional Independence "makes sense". (It had to be saturated because nonsaturated statements seem to require the existence of marginals in the model, which we consider as a quite idiosyncratic feature of probabilities). We have also proved that *trivial independence, symmetry, intersection and weak union* consist in a complete axiomatic characterization of independences which hold over that set of models and furthermore it seems that the Abelian group properties of the range are required for the model in order to prove that these four axioms hold. This pieces of evidence give significant support to the claim that these four properties, are indeed an "axiomatic core" for a very general notion of Conditional Independence in the saturated setup, and functions which range over Abelian groups are their natural set of models. Furthermore such a tie is strengthen by the existence of the factorization theorem - the natural generalization of the Hammersly-Clifford known to hold for probabilities.

## 7.6 Proofs of the theorems

### 7.6.1 Proof of Theorem 1 (independence properties)

**Theorem 1 (independence properties):** *Let $(G, +, 0, -)$ be a group, $(X_\alpha)_{\alpha \in V}$ be a collection of variables indexed by $V$, $f : \mathcal{X} \to G$ be a function from the set $\mathcal{X}$ to the group $G$, and $A, B, C, D$ be subsets of $V$. Then the Conditional Independence relation with respect to $f$, $I_f(\cdot, \cdot|\cdot)$ has the following properties:*

1. **(Trivial Independence)** $I_f(A, \emptyset|B)$ - $\forall A, B$ *a partition of* $V$.
2. **(Symmetry)** $I_f(A, B|C)$ iff $I_f(B, A|C)$ - $\forall A, B, C$ *a partition of* $V$.
3. **(Weak Union)** $I_f(A, B \cup D|C) \Rightarrow I_f(A, B|C \cup D)$ - $\forall A, B, C, D$ *a partition of* $V$.
4. **(Intersection)** $I_f(A, B|C \cup D)$ & $I_f(D, B|C \cup A) \Rightarrow I_f(A \cup D, B|C)$ - $\forall A, B, C, D$ *a partition of* $V$.

*Proof.*

1. (Trivial Independence) To show that $I_f(A, \emptyset|B)$ we have to prove that there exist $f_1, f_2$ such that $f(A, B) = f_1(A, B) + f_2(B)$. By taking $f_1(A, B) = f(A, B)$ and $f_2(B) \equiv 0$ (0 is the identity element of $G$) we have the desired conclusion.

2. (Symmetry) is obvious from the definition of $I_f(A, B|C)$ and the commutativity of $+$.

3. (Weak Union)

We want to prove that: $I_f(A, B \cup D|C) \Rightarrow I_f(A, B|C \cup D)$.

$I_f(A, B \cup D|C)$ implies that $f(A, B, C, D) = f_1(A, C) + f_2(B \cup D, C) = f_1(A, C) + f_2(B, C, D)$ where the last equality follows from the fact that $B \cap D = \emptyset$.

To show that $I_f(A, B|C \cup D)$ we need to show that there exists $f_3, f_4$ such that $f(A, B, C, D) = f_3(A, C \cup D) + f_4(B, C \cup D) = f_3(A, C, D) + f_4(B, C, D)$ where the last equality follows from the fact that $C \cap D = \emptyset$.

By taking $f_3(A, C, D) = f_1(A, C)$ and $f_4(B, C, D) = f_2(B, C, D)$ we have that $f(A, B, C, D) = f_1(A, C) + f_2(B, C, D) = f_3(A, C, D) + f_4(B, C, D)$ which by definition implies that $I_f(A, B|C \cup D)$ .

4. (Intersection)

We want to prove that $I_f(A, B|C \cup D)$ & $I_f(D, B|C \cup A) \Rightarrow I_f(A \cup D, B|C)$.

$I_f(A, B|C \cup D)$ implies that $f(A, B, C, D) = f_1(A, C \cup D) + f_2(B, C \cup D) = f_1(A, C, D) + f_2(B, C, D)$ (eq. 1)

$I_f(D, B|C \cup A)$ implies that $f(A, B, C, D) = f_3(D, C \cup A) + f_4(B, C \cup A) = f_3(A, C, D) + f_4(A, B, C)$ (eq. 2). By subtracting the two equations we get

$f(A, B, C, D) - f(A, B, C, D) = 0 = f_1(A, C, D) + f_2(B, C, D) - f_3(A, C, D) - f_4(A, B, C)$.

Let $f_5(A, C, D) = f_1(A, C, D) - f_3(A, C, D)$ and by moving $f_4(A, B, C)$ to the left hand side of the previous formula we get

$f_4(A, B, C) = f_2(B, C, D) + f_5(A, C, D)$. But $D$ does not appears in the left hand side ($f_4$ does not depend on $D$). Let $d \in D$ an arbitrary, but fixed, element of $D$. Then

$f_4(A, B, C) = f_6(B, C, d) + f_7(A, C, d) = f_8(B, C) + f_9(A, C)$. Now by re-substituting $f_4(A, B, C)$ in the equation (eq. 2) we have

$f(A, B, C, D) = f_3(A, C, D) + f_8(B, C) + f_9(A, C)$. Finally by taking $f_{10}(A, C, D) = f_7(A, C) + f_3(A, C, D)$ we have

$f(A, B, C, D) = f_6(B, C) + f_{10}(A, C, D)$ which by definition implies that $I_f(A \cup D, B|C)$.

■

Note that in order to prove Trivial Independence we need the identity element property of the Abelian group $(G, +, 0, -)$, to prove Symmetry we needed commutativity, and to prove Intersection we needed associativity and most importantly the inverse operator $-$. So it seems that we "need" all the Abelian Group properties.

### 7.6.2 Proof of Theorem 2 (Markov properties equivalence)

**Theorem 2 (Markov properties equivalence):** [13] (G) $\Leftrightarrow$ (L) $\Leftrightarrow$ (P) *for any conditional independence relation* $I(\cdot, \cdot|\cdot)$ *that satisfies* ***Trivial independence, Symmetry, Weak union*** *and* ***Intersection****.*

*Proof.*

We will prove that (G) => (L) => (P) => (G).

(G) => (L) is trivial by taking $A = \{\alpha\}$ and $B = V\backslash(\{\alpha\} \cup \mathcal{N}(\alpha)))$.

(L) => (P) We want to prove that $\forall\alpha \in V\ I(\alpha, V\backslash(\{\alpha\} \cup \mathcal{N}(\alpha)))|\mathcal{N}(\alpha))$ implies that $(\alpha, \beta) \notin E \Rightarrow I(\alpha, \beta|V\backslash\{\alpha, \beta\})$. Since $(\alpha, \beta) \notin E$ implies$\beta \notin \mathcal{N}(\alpha)$ it follows that $\beta \in V\backslash(\{\alpha\}\cup\mathcal{N}(\alpha)))$ . By using the weak union property with $A = \{\alpha\}$, $B = \{\beta\}$, $C = \mathcal{N}(\alpha)$,$D = V\backslash(\{\alpha\}\cup\{\beta\}\cup\mathcal{N}(\alpha)))$ and the fact that $(\alpha, \beta) \notin E \Rightarrow \beta \notin \mathcal{N}(\alpha)$ it follows that $I(\alpha, \beta|\mathcal{N}(\alpha) \cup (V\backslash(\{\alpha\} \cup \{\beta\} \cup \mathcal{N}(\alpha))))$ or equivalently $I(\alpha, \beta|V\backslash(\{\alpha\} \cup \{\beta\}))$ q.e.d.

(P) => (G) we want to prove that: (Prop) if $C$ separates $A$ and $B$ in the graph $\mathcal{G}$, then $I(A, B|C)$

We will prove the property (Prop) by induction after cardinality of the separator $C$, $|C|$.

Base case: For the case when $|C| = |V| - 2$ the property follows trivially from the from the Pairwise Markov Property (P). Also for $|C| > |V|-2$ all the independence statements are trivial because at least one of $A$ or $B$ is empty; and since $I(\cdot, \cdot|\cdot)$ satisfies the trivial independence property they are true by default.

Inductive case: We will assume that the property is true for $|C| \geq k$ and prove it for $|C| = k-1$, where $1 \leq k \leq |V|-2$ . Let $|C| = k-1$ , then we want to prove that if $C$ separates $A$ and $B$ in the graph $\mathcal{G}$, then $I(A, B|C)$. Since $k \leq |V|-2$ it follows that $|C| = k-1 \leq |V|-3$ Hence at least one of the sets $A$ and $B$ has at least two elements. Because of the symmetry property we can assume without loss of generality that the set is $A$. Let $\alpha \in A$. From the fact that $C$ separates $A$ and $B$ in the graph $\mathcal{G}$ it follows that $C \cup \{\alpha\}$separates $A\backslash\{\alpha\}$ from $B$ in $\mathcal{G}$ . Using the inductive assumption we get that $I(A\backslash\{\alpha\}, B|C \cup \{\alpha\})$ . Also from the fact that $C$ separates $A$ and $B$ in the graph $\mathcal{G}$ it follows that $C \cup (A\backslash\{\alpha\})$separates $\{\alpha\}$ from $B$ in $\mathcal{G}$ and again using the inductive assumption we get that $I(\alpha, B|C \cup (A\backslash\{\alpha\}))$. Now by the intersection property we have that $I(A\backslash\{\alpha\}, B|C \cup \{\alpha\})$ & $I(\alpha, B|C \cup (A\backslash\{\alpha\})) \Rightarrow I(A, B|C)$.

■

Note that the Global Markov property is slightly weaker in our case because we have only saturated independence and hence we cannot pick arbitrary sets that separate $A$ and $B$ instead of just $X\backslash(A \cup B)$

### 7.6.3 The proof of Theorem 3 (factorization)

**Theorem 3 (factorization):** *Let $\mathcal{G} = (V, E)$ be an undirected graph, $(G, +, 0, -)$ be a group, $(X_\alpha)_{\alpha \in V}$ be a collection of variables indexed by $V$, $f : \mathcal{X} \to G$ be a function from the set $\mathcal{X}$ to the group $G$. Then* (G) $\Leftrightarrow$ (L) $\Leftrightarrow$ (P) $\Leftrightarrow$ (F).

*Proof.* We will prove (F) $\Rightarrow$ (G) and (P) $\Rightarrow$ (F) and this will be enough to prove the theorem because the other equivalences follow from the ***Markov properties*** theorem in the previous section.

(F) $\Rightarrow$ (G) Let $\mathcal{G} = (V, E)$ be a graph and $f : \mathcal{X} \to G$ be a function that satisfies the factorization property with respect to $\mathcal{G}$. Then it follows that:

$$f(V) = \sum_{C \in MaxCliques(\mathcal{G})} f_C(C)$$

Now let $A$, $B$ be two sets such that $V \backslash (A \cup B)$ separates $A$ and $B$ in the graph $\mathcal{G}$. Then

$$\begin{aligned} f(V) &= \sum_{C \in MaxCliques(\mathcal{G}) \,\&\, C \cap A \neq \emptyset} f_C(C) \\ &\quad + \sum_{C \in MaxCliques(\mathcal{G}) \,\&\, C \cap A = \emptyset} f_C(C) \end{aligned}$$

Let $f_1(V \backslash B) = \sum_{C \in MaxCliques(\mathcal{G}) \,\&\, C \cap A \neq \emptyset} f_C(C)$ and

$f_2(V \backslash A) = \sum_{C \in MaxCliques(\mathcal{G}) \,\&\, C \cap A \neq \emptyset} f_C(C)$. To show that $f_1$ and $f_2$ are well defined we have to show that the right hand sides of their definitions contain only variables from $V \backslash B$ and $V \backslash A$ respectively. Obviously $f_2$ contains only variables that are not from $A$. We will show that $f_1$ has variables from $V \backslash B$ only, by contradiction.

Suppose $f_1$ contains variables from $B$. Then it follows that there exists a clique $C$ such that $C \in MaxCliques(\mathcal{G})$, $C \cap A \neq \emptyset$ and also $C \cap B \neq \emptyset$. Let $\alpha \in C \cap A$ and $\beta \in C \cap B$. But since $C$ is a clique in $\mathcal{G}$ it follows that $(\alpha, \beta)$ is an edge in $\mathcal{G}$, which contradicts the fact that $V \backslash (A \cup B)$ separates $A$ and $B$ in the graph $\mathcal{G}$.

Now given that $f_1$ and $f_2$ are well defined we can write:

$$\begin{aligned} f(V) &= f_1(V \backslash A) + f_2(V \backslash B) \\ &= f_1(A, V \backslash (A \cup B)) + f_2(B, V \backslash (A \cup B)) \end{aligned}$$

Which implies, by definition, that $I_f(A, B|V\backslash(A \cup B))$ .

(P) $\Rightarrow$ (F) In order to prove this implication we will use the following helpful lemma:

**Lemma (Moebius inversion):** *Let $f$ and $g$ be two functions defined on the set of all subsets of a finite set $V$ of variable indices, taking values into an Abelian Group $(G, +, 0, -)$ . Then the following two statements are equivalent:*

(1) *for all* $A \subseteq V : g(A) = \sum_{B:B\subseteq A} f(B)$

(2) *for all* $A \subseteq V : f(A) = \sum_{B:B\subseteq A}(-1)^{|A\backslash B|}g(B)$

*where, by $(-1)^k$ we mean $-$ if $k$ is odd and $+$ if $k$ is even. (Note that we need this explicitation because multiplication is not necessarily defined over the elements of G)*

*Proof.* A proof of this lemma can be found in [11] [8]. See also the next subsection in the Appendix 7.6.4. ■

We are now ready to prove the (P)$\Rightarrow$ (F) implication from the **factorization** theorem.

Let $f : \mathcal{X} \to G$ be the function, which induces an Independence relation $I_f(\cdot, \cdot|\cdot)$ over the variables indexed by $V$ and $\mathcal{G} = (V, E)$ the graph with respect to which $I_f(\cdot, \cdot|\cdot)$ has the Pairwise Markov property (P). Let $x^* \in \mathcal{X}$ be an arbitrary, but fixed, element of $\mathcal{X}$. We define for all $A \subseteq V$ the function

$$g_A(x) = f(x_A, x^*_{A^C})$$

where $(x_A, x^*_{A^C})$ is an element $y$ with $y_\gamma = x_\gamma$ if $\gamma \in A$ and $y_\gamma = x^*_\gamma$ if $\gamma \notin A$. Since $x^*$ is fixed, $g_A$ depends on $x$ through $x_A$ only. Now, for all $A \subseteq V$, let

$$f_A(x) = \sum_{B:B\subseteq A} (-1)^{|A\backslash B|} g_B(x)$$

This formula implies that $f_A(x)$ depends on $x$ through $x_A$ only.

By applying the Moebius inversion lemma to the functions $f$ and $g$ we get:

$$f(x) = g_V(x) = \sum_{A:A\subseteq V} f_A(x)$$

We will show next that $f_A(x) \equiv 0$ whenever $A$ is not a clique of $\mathcal{G}$. This fact, along with absorbing $f_A$ into $f_M$ whenever $A$ is not a maximal clique and where $A \subset M \in MaxCliques(\mathcal{G})$ , will prove our factorization property (F) over the maximal cliques of the graph $\mathcal{G}$. (absorption: if $A \subset M \in MaxCliques(\mathcal{G})$ we can redefine $f'_M(x) = f_M(x) + f_A(x)$ and $f'_A(x) \equiv 0$).

To show that $f_A(x) \equiv 0$ whenever $A$ is not a clique of $\mathcal{G}$, let $\alpha, \beta \in A$ such that $(\alpha, \beta) \notin E$ and let $C = A \backslash \{\alpha, \beta\}$. Then we have

$$\begin{aligned} f_A(x) = \sum_{B:B\subseteq C} (-1)^{|C\backslash B|} \quad \{ \quad & g_B(x) - g_{B\cup\{\alpha\}}(x) \\ & - \quad g_{B\cup\{\beta\}}(x) + g_{B\cup\{\alpha,\beta\}}(x)\} \end{aligned}$$

We now want to show that $g_B(x) - g_{B\cup\{\alpha\}}(x) - g_{B\cup\{\beta\}}(x) + g_{B\cup\{\alpha,\beta\}}(x) \equiv 0$ for all $B \subseteq C = A\backslash\{\alpha, \beta\}$, which will prove our claim. $(\alpha, \beta) \notin E$ implies that $I_f(\alpha, \beta | V\backslash\{\alpha, \beta\})$ (by (P)), so there exist $f_1, f_2$ such that

$$f(V) = f_1(\alpha, V\backslash\{\alpha, \beta\}) + f_2(\beta, V\backslash\{\alpha, \beta\})$$

i.e.,

$$f(x_V) = f_1(x_\alpha, x_{V\backslash\{\alpha,\beta\}}) + f_2(x_\beta, x_{V\backslash\{\alpha,\beta\}}) \; \forall x_V \in \mathcal{X}$$

by considering $x_V$ of the form $(x_B, x_\alpha, x_\beta, x_R^*) \; \forall x_B \in \mathcal{X}_B, x_\alpha \in \mathcal{X}_\alpha, x_\beta \in \mathcal{X}_\beta$ where $R = V\backslash(B \cup \{\alpha, \beta\})$ we get

$$\begin{aligned} g_{B\cup\{\alpha,\beta\}}(x) &= f(x_B, x_\alpha, x_\beta, x_R^*) \\ &= f_1(x_B, x_\alpha, x_R^*) + f_2(x_B, x_\beta, x_R^*) \quad (f1) \end{aligned}$$

for all $x_B \in \mathcal{X}_B, x_\alpha \in \mathcal{X}_\alpha, x_\beta \in \mathcal{X}_\beta$. By instantiating $x_\beta$ in the formula $(f1)$ with $x_\beta^*$ we get

$$\begin{aligned} g_{B\cup\{\alpha\}}(x) &= f(x_B, x_\alpha, x_\beta^*, x_R^*) \\ &= f_1(x_B, x_\alpha, x_R^*) + f_2(x_B, x_\beta^*, x_R^*) \end{aligned}$$

for all $x_B \in \mathcal{X}_B, x_\alpha \in \mathcal{X}_\alpha$. Similarly by instantiating $x_\alpha$ in in the formula $(f1)$ with $x_\alpha^*$ we get

$$\begin{aligned} g_{B\cup\{\beta\}}(x) &= f(x_B, x_\alpha^*, x_\beta, x_R^*) \\ &= f_1(x_B, x_\alpha^*, x_R^*) + f_2(x_B, x_\beta, x_R^*) \end{aligned}$$

for all $x_B \in \mathcal{X}_B, x_\beta \in \mathcal{X}_\beta$. And finally, by instantiating both $x_\alpha$ and $x_\beta$ with $x_\alpha^*$ and $x_\beta^*$ respectively, in the formula $(f1)$ we get

$$\begin{aligned} g_B(x) &= f(x_B, x_\alpha^*, x_\beta^*, x_R^*) \\ &= f_1(x_B, x_\alpha^*, x_R^*) + f_2(x_B, x_\beta^*, x_R^*) \end{aligned}$$

for all $x_B \in \mathcal{X}_B$. Now computing the formula $(*) = g_B(x) - g_{B\cup\{\alpha\}}(x) - g_{B\cup\{\beta\}}(x) + g_{B\cup\{\alpha,\beta\}}(x)$ with these alternative expansions we get

$$\begin{aligned}(*) &= f_1(x_B, x^*_\alpha, x^*_R) + f_2(x_B, x^*_\beta, x^*_R) \\ &\quad -f_1(x_B, x_\alpha, x^*_R) - f_2(x_B, x^*_\beta, x^*_R) \\ &\quad -f_1(x_B, x^*_\alpha, x^*_R) - f_2(x_B, x_\beta, x^*_R) \\ &\quad +f_1(x_B, x_\alpha, x^*_R) + f_2(x_B, x_\beta, x^*_R) \\ &= 0\end{aligned}$$

■

In the particular case when $f$ is a probability distribution the last implication in the previous theorem ((P) →(F)) is known as the Hammersley-Clifford theorem [3]. The proof technique based on the Moebius Inversion Lemma was first used for proving the Hammersley-Clifford theorem for probabilities in [7], see also [11], [8]. To the best of our knowledge, the present proof is the first one which holds for the general case of conditional independence with respect to a function $f$ which takes values into an Abelian Group.■

### 7.6.4 The proof of the Moebius Inversion Lemma

(2) => (1)

$$\begin{aligned}\sum_{B:B\subseteq A} f(B) &= \sum_{B:B\subseteq A}\sum_{C:C\subseteq B}(-1)^{|B\backslash C|}g(C) \\ &= \sum_{C:C\subseteq A}\sum_{B:C\subseteq B\subseteq A}(-1)^{|B\backslash C|}g(C) \\ &= \sum_{C:C\subseteq A}\sum_{D:D\subseteq A\backslash C}(-1)^{|D|}g(C)\end{aligned}$$

$\sum_{D:D\subseteq A\backslash C}(-1)^{|D|}g(C)$ is 0 unless $A\backslash C = \emptyset$ , or equivalently $A = C$ (because $C \subseteq A$), and therefore the whole sum will be $g(A)$, which proves the desired equality. To see that if $E = A\backslash C \neq \emptyset$ then $\sum_{D:D\subseteq E}(-1)^{|D|}g(C)$ is 0 we observe that there are as many pluses in this sum as there are minuses and therefore the terms cancel out. $(0 = (1-1)^{|D|} = \sum_{D:D\subseteq E}(-1)^{|D|}$ if $D \neq \emptyset$; if $D = \emptyset$ the sum is by definition $0^0 = 1)$

(1) => (2)

$$\begin{aligned}\sum_{B:B\subseteq A}(-1)^{|A\setminus B|}g(B) &= \sum_{B:B\subseteq A}\sum_{C:C\subseteq B}(-1)^{|A\setminus B|}f(C)\\ &= \sum_{C:C\subseteq A}\sum_{B:C\subseteq B\subseteq A}(-1)^{|A\setminus B|}f(C)\\ &= \sum_{C:C\subseteq A}\sum_{D:D\subseteq A\setminus C}(-1)^{|D|}f(C)\end{aligned}$$

Which by the same argument as before is equal to $f(A)$, which proves the desired equality.

### 7.6.5 The proof of Theorem 5 (completeness)

**Theorem 5 (completeness):** *Let $f : \mathcal{X} \to G$ be a function that takes values into an Abelian Group $(G, +, 0, -)$ . Let $\Sigma$ be a set of saturated independence statements over sets in $V$, and let $\Sigma^+$ be the closure of $\Sigma$ with respect to trivial independence, symmetry, intersection and weak union. Then for every $\sigma \notin \Sigma^+$, there exists a function $f : \mathcal{X} \to G$ such that $f$ satisfies $\Sigma^+$ and does not satisfy $\sigma$.* ■

*Proof.* From Theorem 4 there exists a graph $\mathcal{G}(\Sigma^+)$ (the associated graph) that satisfies $\Sigma^+$ and no other independence statement. So to prove the theorem we need to show that there exists a function $f : \mathcal{X} \to G$ that satisfies the statements that hold in $\mathcal{G}(\Sigma^+)$ and does not satisfy $\sigma$. We will prove this in the next lemma. ■

**Lemma.** *Let $\mathcal{G} = (V, E)$ a graph and let $A, B, C$ a partition of $V$ such that $C$ does not separate $A$ from $B$ in the graph $\mathcal{G}$. Let $(G, +, 0, -)$ a group with at least two elements. Then there exists a function $f : \mathcal{X} \to G$ such that $\mathcal{G}$ is a Markov Network for $f$ and $\neg I_f(A, B|C)$.*

*Proof.* Let $(X_\alpha)_{\alpha\in V}$ be a collection of variables indexed by $V$ such that all variables $X_\alpha$ have the same range $\{a, b\}$. We will construct a function $f : \mathcal{X} = \times_\alpha \mathcal{X}_\alpha \to G$ such that $\mathcal{G}$ is a Markov Network for $f$ and $\neg I_f(A, B|C)$. Because $G$ is a group with at least two elements, let $g \in G$ such that $g \neq 0$. From the fact that $C$ does not separate $A$ from $B$ in the graph $\mathcal{G}$ and the fact that $A, B, C$ is a partition of $V$ it follows that there exists a $\alpha \in A$ and $\beta \in B$ such

that $(\alpha, \beta) \in E$ . Let

$$f(x_V) = h(x_\alpha, x_\beta) \sum_{x_\gamma \in V \setminus \{\alpha,\beta\}} g(x_\gamma)$$

where $g(x) = 0 \; \forall x \in \{a, b\}$and

$$h(x,y) = \begin{cases} 0 & if \quad (x,y) = (a,a) \\ 0 & if \quad (x,y) = (a,b) \\ 0 & if \quad (x,y) = (b,a) \\ g & if \quad (x,y) = (b,b) \end{cases}$$

Note that $h(x, y)$ is not decomposable into smaller functions (i.e., $h(x, y) \neq h_1(x) + h_2(y)$- see a proof next). The function $f$ has the desired properties, that is $I_f(\alpha, \beta|V \setminus \{\alpha, \beta\})$ does not hold because $h(x, y)$ is not decomposable and $f(x_V) = h(x_\alpha, x_\beta)$ hence $f$ cannot be decomposed into functions that separate $x_\alpha$ and $x_\beta$. Furthermore, $Sep_{\mathcal{G}}(A, B|C) \Rightarrow I_f(A, B|C)$ because $I_f(A, B|C)$ holds for all sets $A, B$ that do not separate $\alpha$ and $\beta$ (easy to check).

We will now prove that $h(x, y)$is not decomposable by contradiction. Assume $h(x, y) = h_1(x) + h_2(y)$. Then

$$h(a,a) = 0 = h_1(a) + h_2(a) \tag{7.1}$$

$$h(a,b) = 0 = h_1(a) + h_2(b) \tag{7.2}$$

$$h(b,a) = 0 = h_1(b) + h_2(a) \tag{7.3}$$

$$h(b,b) = g = h_1(b) + h_2(b) \tag{7.4}$$

from equations (7.1) and (7.2) it follows (by simplifying $h_1(a)$) that $h_2(a) = h_2(b)$; Similarly from equations (7.1) and (7.3) it follows (by simplifying $h_2(a)$) that $h_1(a) = h_1(b)$. Let $g_1, g_2 \in G$ such that $g_1 = h_2(a) = h_2(b)$ and $g_2 = h_1(a) = h_1(b)$. Then from equation (7.1) it follows that $g_1 + g_2 = 0$, and from equation (7.4) it follows that $g_1 + g_2 = g$, hence $g = 0$, contradiction.

This completes the proof of the lemma. ■

### 7.6.6 Proof that: Intersection + Weak Union ⇒ Weak Contraction

**Proposition (INT + WU ⇒ WC):** Intersection + Weak Union ⇒ Weak Contraction.

WU: $I(A \cup B, D|C) \Rightarrow I(A, D|C \cup B)$

INT: $I(A, D|C \cup B)$ & $I(A, B|C \cup D) \Rightarrow I(A, B \cup D|C)$

⇒

WC: $I(A \cup B, D|C)$ & $I(A, B|C \cup D) \Rightarrow I(A, B \cup D|C)$

*Proof.*

We want to prove that:

WC: $I(A \cup B, D|C)$ & $I(A, B|C \cup D) \Rightarrow I(A, B \cup D|C)$

Assume $I(A \cup B, D|C)$ (1)and $I(A, B|C \cup D)$ (2)

$I(A \cup B, D|C) \Rightarrow_{WU} I(A, D|C \cup B)$ (3)

(2) & (3) $\Rightarrow_{INT} I(A, B \cup D|C)$ ■

## Bibliography


[1] F. Bacchus and A. Grove. Graphical Models for Preference and Utility. In *Uncertainty in AI*, 11, 1995.

[2] U. Bertele and F. Brioschi. *Nonserial dynamic programming*, Academic Press, New York, 1972.

[3] J. Besag. Spatial Interaction and Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B (Methodological),* 36(2):192-236, 1974.

[4] R. Cowell, P. Dawid, S. Lauritzen and D. Spiegelhalter. *Probabilistic Networks and Experts Systems.* Springer, 1999.

[5] A. P. Dawid. Conditional independence in statistical theory. *J. Roy. Statist. Soc. B,* 41:1-31, 1979.

[6] D. Geiger and J. Pearl. Logical and Algorithmic Properties of Conditional Independence and Graphical Models. *The Annals of Statistics*, 21(4):2001-2021, 1993.

[7] D. Griffeath. Introduction to Random Fields. In J. Kemeny, J. Snell and A. Knapp (Eds), *Denumerable Markov Chains*. New York: Springer-Verlag, 1976.

[8] M. Jordan. *An Introduction to Probabilistical Graphical Models (to appear)*. 2005 .

[9] R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, 1976.

[10] H. Muhlenbein and R. Hons. The Estimation of Distributions and the Minimum Relative Entropy Principle, *Evolutionary Computation* 13(1):1-27, 2005.

[11] S. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, 1988.

[13] J. Pearl and A. Paz. Graphoids: a graph based logic for reasoning about relevancy relations. In *Advances in Artificial Intelligence-II,* 357-363. North-Holland, 1987.

[14] M. Studeny. Conditional independence relations have no finite complete characterization. In *Transactions if the 11th Prague conference on information theory, statistical decision functions and random processes,* 377-396, 1992.

[15] S.K.M. Wong, C.J. Butz, and D. Wu, On the Implication Problem for Probabilistic Conditional Independency, *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 30(6):785-805, 2000.

# CHAPTER 8. GENERAL CONCLUSIONS

## 8.1 Summary

The results presented in the thesis can be divided into two sets: Results that have been obtained prior to the full development of the theoretical part of the *Abstraction + SuperStructuring thesis*; and results obtained afterward.

The results obtained prior to the proof of *Abstraction + SuperStructuring thesis* are exposed in chapters 2, 3 and 4. These prior exploratory results along with earlier less comprehensive theoretical results have at least partially enabled and then further reinforced the belief that the *Abstraction + SuperStructuring thesis* in its full generality may have a chance to be proved at a point in time when this was not a given. They contain in chronological order:

1. an exploration of SuperStructuring in the temporal domain - Temporal Boolean Networks (Chapter 2) - Learning temporal SuperStructures by the means of decision trees in the context of predicting the behavior of Genetic Regulatory Networks.

2. an exploration of SuperStructuring in the spatial domain (sequences) - Naive Bayes k - NB(k) (Chapter 3) - Various extensions of Naive Bayes, which treats input features independently, to the the case of $k$-th order dependencies in the context of function prediction for protein sequences.

3. an exploration of Abstraction learning in the Multivariate case - Learning Attribute Value Taxonomies (Chapter 4) - Learning Taxonomies (a particular way of organizing Abstractions) in the Multivariate case.

In Chapter 5 - Abstraction SuperStructuring Normal Forms - we presented the theoretical argument for the of the *Abstraction + SuperStructuring thesis*. We showed that Abstrac-

tion, SuperStructuring and their duals: Reverse Abstraction and Reverse SuperStructuring respectively are enough in order to produce the Turing equivalent class of General Generative Grammars. Under the Computationalistic Assumption this means that every theory can be expressed in terms of Abstraction, SuperStructuring and their duals: Reverse Abstraction and Reverse SuperStructuring. We also showed that this result can be seen as proving (under the Computationalistic Assumption) a more than 200 years claim of the philosopher David Hume regarding the "principles of connexion among ideas". Furthermore by the means of our Abstractions SuperStructuring Normal Form Theorem we have produced a characterization of the rationales for introducing hidden variables, which are identified with Reverse SuperStructuring and Reverse Abstraction. Subsequently we have also proposed a characterization of the radical positivist setup which allows only empirical laws in terms of Abstraction and SuperStructuring.

In Chapter 6 - Combining Abstraction and SuperStructuring - appropriate extensions of the earlier results from chapters 3 and 4 allowed us to present a baseline solution for acquiring Abstraction and SuperStructuring in combination. The experimental results show that using the two principles of Abstraction and SuperStructuring, either in isolation - Chapters 2,3 and 4, but furthermore even better in combination - Chapter 6, can produce significant improvements in the comprehensibility (as measured by size) and / or accuracy of the models produced. By using a simple combination of Abstraction and SuperStructuring we have shown in Chapter 6 that we can reduce model size by (1-3) orders of magnitude for a small loss or sometimes even a small gain in the accuracy of the models produced when compared with the model that uses SuperStructuring only. Furthermore we have also shown the superiority of using Abstraction as a model reduction technique over using Feature Selection. We have also provided two algorithms (one for the Multivariate case - Chapter 4 and one for the Multinomial case - Chapter 6) to automatically generate Abstractions for a given classification task, which we identified as being the principal impediment in transforming the use of Abstraction into an "off the shelf" procedure.

In Chapter 7 - Independence and Decomposability - we present a theoretical analysis of the notion of (Variable) Independence and how such a structural analysis may impact various

Numerical components, such as value functions, probabilities, relations, etc. Independence is basically the complement of SuperStructuring - that is, if two "proximal" entities are independent then there is little need to consider their union as bigger structural unit - hence no SuperStructuring. In Chapter 7 we explored the notion of independence and how it can be used to facilitate the decomposition / factorization function $f$ which takes values into an arbitrary Abelian group. This allowed us to treat uniformly the additive, multiplicative and relational setups. Our main result shows that we can "boil down" a set of pairwise Conditional Independences (and consequently pairwise factorizations) to a "global" factorization of a function $f$ over the maximal cliques of the associated independence graph. This theorem is the natural generalization of the Hammersley-Clifford theorem which holds for probability distributions, to the more general case of functions that take values into an Abelian Group. This theory subsumes: factorization of probability distributions, additive decomposable functions and decomposable relations, as particular cases of functions over Abelian Groups.

## 8.2 Contributions

In this thesis we have explored the problem of Structural Induction - Automatic Ontology Elicitation. In order to deal with the problem of Structural Induction we need to solve two main problems: 1. We have to say what we mean by Structure (***What?***); and 2. We have to say how to get it (***How?***). In this thesis we give one possible, but very definite, answer to the first question (***What?***) and we also explore how to answer the second question (***How?***) in some particular cases. A comprehensive answer to the second question (***How?***) in the most general setup will involve dealing very carefully with the interplay between the Structural and Numerical aspects of Induction and will represent a full solution to the Induction problem. This is a vast enterprise and we only touch some aspects of this issue.

The contributions of the thesis can be divided into a theoretical and a practical part respectively:

1. On the the theoretical side:

(a) **Abstraction + SuperStructuring thesis** - In Chapter 5, in the form of the Abstraction SuperStructuring Normal Form theorem (ASNF) we have given one possible, but very definite and intuitive, answer to the problem of ***What?*** we mean by structure. Namely: Abstraction (grouping similar entities under one overarching category) and Super-Structuring (grouping into a bigger unit topologically close entities - in particular spatio-temporally close) along with their duals: Reverse Abstraction and Reverse SuperStructuring.

(b) **Hidden Variables Operators: Reverse Abstraction + Reverse SuperStructuring** - In Chapter 5, by examining the types of rules present in the Abstraction SuperStructuring Normal Form we have identified the ones that can introduce "hidden variable explanations", namely Reverse Abstraction (alternative causes) and Reverse SuperStructuring (concurrent "proximal" causes).

(c) **Radical Positivism: Abstraction + SuperStructuring** - In Chapter 5, by eliminating the productions that can introduce hidden explanations: Reverse Abstraction and Reverse SuperStructuring we have proposed the remainder: Abstraction and SuperStructuring, represents a precise characterization of the radical positivist position which allows empirical laws only. (Chapter 5)

(d) **Proof of Hume's claim on the principles of connexion between ideas -** In Chapter 5, the Abstraction SuperStructuring Normal Form theorem (ASNF) under an appropriate interpretation can be seen a a proof, under the Computationalistic Assumption, of a more than 200 years old claim of the philosopher David Hume about the principles of connexion among ideas (Chapter 5)

(e) **General factorization theorem for functions which take values into an Abelian group -** In Chapter 7, we explored the notion of independence and how it can be used to facilitate the decomposition / factorization function $f$ which takes values into an arbitrary Abelian group. The main theorem shows that we can "boil down" a set of pairwise Conditional Independences (and consequently pairwise factorizations) to a "global" factorization of a function $f$ over the maximal cliques of the

associated independence graph. This theorem is the natural generalization of the Hammersley-Clifford theorem which holds for probability distributions, to the more general case of functions that take values into an Abelian Group. This theory subsumes: factorization of probability distributions, additive decomposable functions and decomposable relations, as particular cases of functions over Abelian Groups.

2. On the practical side:

   (a) **Exploration of SuperStructuring -** In Chapters 2 and 3, we have showed the merits of using SuperStructuring in the temporal and spatial domains in contexts of genetic regulatory networks and protein function prediction respectively.

   (b) **Exploration of Abstraction Learning** - In Chapter 4, we have introduced a method for deriving automatically Attribute Value Taxonomies for the Multivariate classification case and showed its merits in producing taxonomies competitive with humanly generate ones, both in providing more comprehensible models and occasionally also in producing more accurate classification results.

   (c) **Exploration of the combination between Abstraction and SuperStructuring** - In Chapter 6, we have made the case for the combination of Abstraction and SuperStructuring as complementary approaches. We have shown that in this way we can produce significant improvements in the comprehensibility (as measured by size) of the models for a small loss or even occasionally a small gain in accuracy. We have also shown the superiority of using Abstraction as a model reduction technique over using Feature Selection. We have also provided two algorithms (one for the Multivariate case - Chapter 4 and one for the Multinomial case - Chapter 6) to automatically generate Abstractions for a given classification task. This eliminates the principal impediment in transforming the use of Abstraction into an "off the shelf" procedure as Feature Selection is.

## 8.3 Future work

In the future additional work can be envisioned in the veins open by this thesis. We outline below a few possible directions:

- Explore additional methods for combining Abstraction and SuperStructuring such as more sophisticated search mechanisms and more complex setups (e.g., allow recursion and also Reverse Abstraction and Reverse SuperStructuring)
- Explore model-based setups for feature evaluation and techniques to scale up such approaches in the context of combining Abstraction and SuperStructuring
- Use an Algebraic Geometry / Algebraic Topology foundation for the models.
- Apply similar techniques or combine with already existing ones in the related areas of Kernel Learning and Spectral Methods
- Develop techniques for automatic object/entity elicitation