Abstraction Super-structuring Normal Forms: Towards a Theory of Structural Induction[⋆]

Adrian Silvescu and Vasant Honavar

Department of Computer Science, Iowa State University, Ames, IA, USA

Abstract. Induction is the process by which we obtain predictive laws or theories or models of the world. We consider the structural aspect of induction. We answer the question as to whether we can find a finite and minimalistic set of operations on structural elements in terms of which any theory can be expressed. We identify *abstraction* (grouping *similar* entities) and super-structuring (combining topologically e.g., spatio-temporally close entities) as the essential structural operations in the induction process. We show that only two more structural operations, namely, *reverse abstraction* and *reverse super-structuring* (the duals of abstraction and super-structuring respectively) suffice in order to exploit the full power of Turing-equivalent generative grammars in induction. We explore the implications of this theorem with respect to the nature of hidden variables, radical positivism and the 2-century old claim of David Hume about the principles of *connexion* among ideas.

1 Introduction

The logic of induction, the process by which we obtain predictive laws, theories, or models of the world, has been a long standing concern of philosophy, science, statistics and artifical intelligence. Theories typically have two aspects: structural or qualitative (corresponding to concepts or variables and their relationships, or, in philosophical parlance, *ontology*) and numeric or quantitative (corresponding to parameters e.g., probabilities). Once the qualitative aspect of a certain law is fixed, the quantitative aspect becomes the subject of experimental science and statistics. Induction is the process of inferring predictive laws, theories, or models of the world from a stream of observations.

Under the *computationalistic assumption* (i.e., the Church-Turing thesis, which asserts that any expressible theory can be described by a Turing Machine [18]), one way to solve the induction problem is to enumerate all the Turing machines (and run them in parallel - dovetailing in order to cope with the countably infinite number of them) and pick one that strikes a good balance

[⋆] This work was supported in part by a grant (IIS 0711356) from the National Science Foundation and in part by the Iowa State University Center for Computational Intelligence, Learning, and Discovery. The work of Vasant Honavar was supported by the National Science Foundation, while working at the Foundation. Any opinion, finding, and conclusions contained in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation.

between the predictability (of the finite experience stream) and size (complexity) [16], [17], [15], [20] of the predictive model, or within a Bayesian setting, using a weighted vote among the predictions of the various models [7] (see [2] and references therein). In the general setting, the number of types of possible structural laws that can be postulated is infinite which makes it difficult to design a general purpose induction strategy. We ask whether a finite and minimalistic set of fundamental structural operations suffice to construct *any* set of expressible laws. The existence of such a set would mean that the learner can work with a small *finite* set of possible operations as opposed to an infinite one which in turn would make it easier to conceive of a general purpose induction strategy.

Because Turing machines are rather opaque from a structural standpoint, we use the alternative, yet equivalent, mechanism of generative grammars[1]. This allows us to work with theories that can be built recursively by applying structural operations drawn from a finite set. The intuition behind this approach is that induction involves incrementally constructing complex structures using simpler structures (e.g., using super-structuring, also called *chunking*), and simplifying complex structures when possible (e.g., using abstraction). Such a compositional approach to induction offers the advantage of increased transparency over the enumerate-and-select approach pioneered by Solomonoff [16], [17]. It also offers the possibility of reusing intermediate structures as opposed to starting afresh with a new Turing machine at each iteration, thereby replacing enumeration by a process akin to dynamic programming or its heuristic variants such as the A* algorithm.

We seek laws or patterns that explain a stream of observations through successive applications of operations drawn from a small finite set. The induced patterns are not necessarily described solely in terms of the input observations, but may also use (a finite number of) additional internal or hidden (i.e., not directly observable) entities. The role of these internal variables is to simplify explanation. The introduction of internal variables to aid the explanation process is not without perils [12][2]. One way to preclude the introduction of internal variables is to apply the following *demarcation criterion*: If the agent cannot distinguish possible streams of observations based on the values of an internal variable, then the variable is non-sensical (i.e., independent of the data or "senses") [3]. The direct connection requirement restricts the no-nonsense theories to those formed out empirical laws [1] (i.e., laws that relate only measurable quantities). However several scientists, including Albert Einstein, while being sympathetic to the positivist's ideas, have successfully used in their theories, hidden variables that have at best indirect connection to observables. This has led to a series of

[1] See [10] for a similarly motivated attempt using *Lambda calculus.*

[2] Consider for example, a hidden variable which stands for the truth value of the sentence: "In heaven, if it rains, do the angels get wet?"

[3] This is a radical interpretation of an idea that shows up in the history of Philosophy from Positivism through the empiricists and scholastics down to Aristotle's *"Nihil est in intellectu quod non prius fuerit in sensu"* (There is nothing in the mind that was not previously in the senses).

revisions of the positivist's doctrine culminating in Carnap's attempt to accommodate hidden variables in scientific explanations [3]. The observables and the internal variables in terms of which the explanation is offered can be seen as the ontology[4] - i.e., the set of concepts and their interrelationships found useful by the agent in theorizing about its experience. In this setting, structural induction is tantamount to ontology construction.

The rest of the paper is organized as follows: Section 2 introduces Abstraction Super-structuring Normal Forms (ASNF) that correspond to a general class of Turing-equivalent generative grammars that can be used to express theories about the world; and shows that: *abstraction* (grouping *similar* entities) and super-structuring (combining topologically e.g., spatio-temporally close entities) as the essential structural operations in the induction process; Only two more structural operations, namely, *reverse abstraction* and *reverse super-structuring* (the duals of abstraction and super-structuring respectively), suffice in order to exploit the full power of Turing-equivalent generative grammars in induction. Section 3 interprets the theoretical results in a larger context the nature of hidden variables, radical positivism and the 2-century old claim of David Hume about the principles of *connexion* among ideas. Section 4 concludes with a summary.

2 Abstraction Super-Structuring Normal Forms

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 process of induction and the formalism of generative grammars and motivate the quest for a minimalistic set of fundamental structural operations. We then get to the main results of the paper: a series of characterization theorems of two important classes of Generative Grammars: Context-Free and General Grammars, in terms of a small set of fundamental structural operations.

2.1 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 $(l \to r)$ 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 $(l \to r)$. The language generated by a grammar is defined by $L(G) = \{w \in T^* | S \overset{*}{\to} w\}$ where $\overset{*}{\to}$ stands for the reflexive transitive closure of the rules from R. Furthermore $\overset{+}{\to}$ 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

[4] The ontology in this case is not universal as it is often the case in philosophy; it is just a set of concepts and interrelations among them that afford the expression of theories.

set of transitions $\alpha \xrightarrow{*} \beta$ is called a derivation. If we want to distinguish between derivations in different grammars we will write $\alpha \xrightarrow{*}_G \beta$ or mention it explicitly. We denote by ϵ 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 ϵ 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 such a sentence an $\epsilon - Amendment$.
4. G is **general grammar (GG)** if all the rules $(l \rightarrow r) \in R$ have no additional restrictions.

Theorem 1. *The set of General Grammars is 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 vice versa.*

Proof. This theorem is a well known result. See for example [13] for a proof[5]. □

2.2 Structural Induction and Generative Grammars

Before proceeding with the main results of the paper we examine the connections between the setting of generative grammars and the problem of structural induction. 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 $S \xrightarrow{*} w$. 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 this correspondence, structural induction aims 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

[5] Similar results of equivalence exist for transductive versions of Turing machines and grammars as opposed to the recognition versions given here (See e.g., [2] and references therein). Without loss of generality, we will assume the recognition as opposed to the transductive setting.

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 that 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 summarized in Table 1.

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

Table 1. Correspondence between Structural Induction and Generative Grammars

Thus, in general, structural induction may invent any rewrite rule of the form $l \to r$, 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 operations which is finite and minimalistic, such that all the rules (or more precisely sets of rules) can be expressed in terms of these operations. This would establish a normal form in terms of a finite set of operations and then the problem of generating laws will be reduced to making appropriate choices from this set without sacrificing completeness. In the next subsection we will attempt to decompose the rules $l \to r$ into a small finite set of fundamental structural elements which will allow us to design better structure search mechanisms.

2.3 ASNF (Abstraction SuperStructuring Normal Form) 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 $G = (N \cup \{S\}, T, S, R = R' \cup \{S \to \epsilon, S \to S'\})$ where $S \notin N'$ will also be considered in that certain Normal Form despite the fact that the two productions $\{S \to \epsilon, S \to S'\}$ 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 $\epsilon - Construction$. We will call the following statement the $\epsilon - Amendment$: Let $G = (N, T, S, R)$ be a grammar, if ϵ 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 state a weak form of the Abstraction SuperStructuring Normal Form for Context Free Grammars.

Theorem 2 (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 $\epsilon \in L(G)$ then a rule of the form $S \rightarrow \epsilon$ is allowed and S does not appear in the RHS of any other rule ($\epsilon - Amendment$). Since we have assumed that $\epsilon \notin L(G)$ we do not need to deal with $\epsilon - Amendment$ and hence the proof.

□

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 state the the Weak ASNF theorem for the general case.

Theorem 3 (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. $A \rightarrow B$
2. $A \rightarrow BC$
3. $A \rightarrow a$
4. $AB \rightarrow C$

Proof . See Appendix of [19].

Remark. We will call the rules of type 4 Reverse Super-Structuring (RSS).

In the next theorem we will strengthen our results by allowing only the renamings (REN) to be non unique. First we define 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 α in the LHS and also this is the only production that has β 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. $A \rightarrow B$
2. $A \rightarrow \zeta$, $\zeta \notin N$
3. $\zeta \rightarrow B$, $\zeta \notin N$

Modify the the grammar G to obtain $G' = (N', T, S', R')$ as follows:

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

Then the new grammar G' generates the same language as the initial grammar G and all the productions of the form $A \rightarrow \zeta$ 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 appears in the productions of type 2 and 3, they are respected since ζ is left unchanged in the productions of the new grammar and the only other types of productions introduced are renamings that are of neither type 2 nor type 3.

Proof. See Appendix of [19].

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

Theorem 4 (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).*

Proof. Apply Lemma 2 to the grammar converted into Weak-CFG-ASNF$\square$

Theorem 5 (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).*

Proof. Apply Lemma 2 to the grammar converted into Weak-GEN-ASNF$\square$

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

In the light of this theorem we proceed to introduce the concept of abstraction and prove some additional results.

2.4 Abstractions And Reverse Abstractions

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 $A \rightarrow A$; If such self-loops exist, the corresponding productions can be eliminated from the grammar without altering the language. In such a directed graph a node A has a set of outgoing edges and a set of incoming edges which we refer to as out-edges and in-edges respectively. 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 $RABS(A) = \{A, IE_A = \{(B, A)|(B, A) \in E\}\}$.

2.5 Grow Shrink Theorem

Theorem 6. *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. See Appendix of [19].

We have therefore proved that for each General Grammar G we can transform it in 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: Phase 1 uses only productions that grow (or leave unchanged) the size of the intermediate string; Phase 2 uses only productions that shrink (or leave unchanged) the size of the intermediate string; and Phase 3 uses only TERMINAL productions[6]. In the case of grammars that are not in the normal form as defined above, the situation is a little more complicated because of successive applications of grow and shrink phases. However, we have shown that we can always transform an arbitrary grammar into one that in the normal form. Note further that the grow phase in both theorems use only context free productions.

We now proceed to examine the implications of the preceeding results in the larger context including the nature of hidden variables, radical positivism and the David Hume's principles of *connexion* among ideas.

3 The Fundamental Operations of Structural Induction

Recall that our notion of structural induction entails: 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 $l \to r$ that l explains r. In a bottom up - data driven way we may proceed as follows: First introduce for every observable a a production $A \to a$. The role of these productions is simply to bring the observables into the realm of internal variables. The resulting association is between the observables and the corresponding internal variables unique (one to one and onto) and hence, once this association is established, we can forget about the existence of observables (Terminals). Since establishing these associations is the only role of the TERMINAL productions, they are not true structural operations. With this in mind, if we are to construct a theory in the GEN-ASNF we can postulate laws of the following form:

1. $A \to BC$ - Super-structuring (SS) which takes two internal variables B and C that occur within proximity of each other (adjacent) and labels the compound. Henceforth, the shorter name A can be used instead for BC. This is the sole role of super-structuring - to give a name to a composite structure to facilitate shorter explanations at latter stages.
2. $A \to B|C$ - Abstraction (ABS). Introduces a name for the occurrence of either of the variables B or C. This allows for compactly representing two

[6] At first sight, it may seem that this construction offers a way to solve the halting problem. However, this is not the case, since we do not answer the question of deciding when to stop expanding the current string and start shrinking, which is key to solving the halting problem.

productions that are identical except that one uses B and the uses C by a single production using A. The role of Abstraction is to give a name to a group of entities (we have chosen two only for simplicity) in order to facilitate more general explanations at latter stages which in turn will produce more compact theories.
3. $AB \to C$ - Reverse Super-structuring (RSS) which introduces up to two existing or new internal variables that are close to each other (with respect to a specified topology) that together "explain" the internal variable C.
4. $A \to C$, $B \to C$ - Reverse Abstraction (RABS) which uses existing or new internal variables A and B as alternative explanations of the internal variable C (we have chosen two variables only for simplicity).

3.1 Reasons for Postulating Hidden Variables

Recall that 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 $H1 \to Effect$, $H2 \to Effect$. In our setting, this case corresponds to Reverse Abstraction. One typical example of this is: The grass is wet, and hence either it rained last night or the sprinkler was on. In the statistical and machine learning literature the models that use this type of hidden variables are called 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 sepcial case. This corresponds to the case when one visible effect has two hidden causes both of which have to occur within proximity of each other (with respect to a specified topology) in order to produce the visible effect. $H1H2 \to Effect$. In our setting, this corresponds to Reverse Super-structuring. In the Statistical / Graphical Models literature the particular case of AND hidden explanations is the one that introduces edges between hidden variables in the dependence graph [5], [9], [11].

The preceding discussion shows that we can associate with two possible reasons for creating hidden variables, the structural operations of Reverse Abstraction and Reverse Super Structuring respectively. Because these are the only two types of productions that introduce hidden variables in the GEN-ASNF, this provides a characterization of the rationales for introducing hidden variables.

3.2 Radical Positivism

If we rule out the use of RSS and RABS, the only operations that involve the postulation of hidden variables, we are left with only SS and ABS which corresponds to the radical positivist [1] stance under the computationalist assumption. An explanation of a stream of observations w in the Radical Positivist theory of the world is mainly a theory of how the observables in the world are grouped into classes (Abstractions) and how smaller chunks of observations are tied together into bigger ones (Super-Structures). The laws of the radical positivist theory are truly empirical laws as they only address relations among observations. However, structural induction, if it is constrained to using only ABS and SS, the class of theories that can be induced is necessarily a subset of the set of theories that can be described by Turing Machines. More precisely, the resulting 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.

3.3 General theories of the world

If structural induction is allowed to take advantage of RSS and RABS in addition to SS and ABS, the resulting theories can make use of hidden variables. 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. Note that, while in general, structural induction can alternate several times between REN+SS and RABS+RSS, we have shown that three phases suffice: a growth phase (REN+SS); a shrink phase (RABS+RSS); and a Terminal phase. Whether 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 stance (only ABS+SS) remains an open question (See Appendix of [19] for a Conjecture to this effect).

3.4 Hume's principles of connexion among ideas

We now examine, against the backdrop of GEN-ASNF theorem, a statement made by philosopher David Hume more that 2 centuries ago: *"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" [6].* If we substitute Resemblance with Abstraction (since abstraction is triggered by resemblance or similarity), Contiguity in time or place with Super-Structuring (since proximity, e.g., spatio-temporal proximity drives Super-Structuring) and Cause and Effect with the two types of explanations that utilize hidden variables, it is easy to see that the GEN-ASNF theorem is simply a precise restatement of Hume's claim under the computationalist assumption.

4 Summary

We have shown that *abstraction* (grouping *similar* entities) and super-structuring (combining topologically e.g., spatio-temporally close entities) as the essential structural operations in the induction process. A structural induction process that relies only on abstraction and super-structuring corresponds to the radical positivist stance. We have shown that only two more structural operations, namely, *reverse abstraction* and *reverse super-structuring* (the duals of abstraction and super-structuring respectively) (a) suffice in order to exploit the full power of Turing-equivalent generative grammars in induction; and (b) operationalize two rationales for the introduction of hidden variables into theories of the world. The GEN-ASNF theorem can be seen as simply a restatement, under the computationalist assumption, of Hume's 2-century old claim regarding the principles of connexion among ideas.

References

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*. Basic Books, 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.
19. A. Silvescu and V. Honavar. Abstraction Super-structuring Normal Forms: Towards a Computationalist Theory of Structural Induction. Technical Report. Department of Computer Science. Iowa State University. 2011. *www.cs.iastate.edu/~silvescu/papers/trasnf/trasnf.pdf*
20. C.S. Wallace and D. Dowe, Minimum Message Length and Kolmogorov complexity, Computer Journal. Vol 42, No. 4, pp 270-283. 1999.