INDUCTION OF CONCEPTS IN THE PREDICATE CALCULUS

Steven A. Vere*
Department of Information Engineering
University of Illinois at Chicago Circle

Abstract

Positive and negative instances of a concept are assumed to be described by a conjunction of literals in the predicate calculus, with terms limited to constants and universally quantified variables. A graph representation of a conjunction of literals, called a "product graph", is introduced. It is desirable to merge positive instances by generalization, while maintaining discrimination against negative instances. This is accomplished by an induction procedure which operates on the product graph form of these positive and negative instances. The correctness of the procedure is proven, together with several related results of direct practical significance. This work is directed to the goal of providing a formal model for the inductive processes which are observed in artificial intelligence studies in specialized areas.

1. Introduction

Lnduction may be broadly defined as "reasoning from a part to a whole, from particulars to generals, or from the individual to the universal" [10]. Consistent with this, we view "induction" as a computational process, and a "generalization" as a statement computed by that process.

The present work is guided by the belief that general purpose induction procedures can be formulated which would be independent of the artificial intelligence problem domain. It is specifically concerned with the induction of concepts from positive and negative instances described in the predicate calculus, with terms limited to constants and universally quantified variables.

Inductive processes in artificial intelligence have been studied from several aspects: visual analogies [3], propositional logic concept learning [4,9], analogies in predicate calculus theorem proving [S], IQ test completion problems [11], theory formation from a data base [2], visual concept learning from examples [12], and induction as a dual of deductive theorem proving in the predicate calculus [1,6,8], or in a generalizittion of the predicate calculus [7]. The present work may be regarded as a continuance of the concept learning research of Hunt and Towster, amongs others, to the predicate calculus. Its viewpoint and many of the concepts are derived from Plotkin.

Throughout, each positive and negative instance of a concept is assumed to be described by a conjunction of literals in the predicate calculus (not necessarily first order). Section 2 defines terminology to be employed, of which the most significant is the concept of a "maximal, consistent, unifying generalization", and contains some general observations on generalizations. With this background, a problem statement is then given in the defined vocabulary. Section 3 introduces the "product graph", a graph representation of a conjunction of literals, which serves as a convenient medium for the discussion of the induction process. This process is accomplished by straightforward operations on these product graphs. Section 4 considers the application of product graphs to the question of the "consistency" of a generalization in view of negative instances. Section 5 contains concluding remarks.

2. Generalization in the Predicate Calculus

2.1 Introductory Definitions; Properties of Generalizations

An n-ary predicate will be represented by a list of n + 1 elements: $(t_1\ t_2\ \ldots\ t_{n+1})$. t_1 understood to represent the predicate symbol, but is otherwise undistinguished from other elements. Each t_i is a term, which may be either a constant, represented by lower case Roman letters, or a variable, represented by upper case Roman letters. All variables are assumed to be universally quantified. A literal is such a list of terms, optionally prefixed by the logical inverse operator " ~ ". For example, (a X c) and ~(e f) are both literals. A product is a conjunction of literals.

A substitution is a finite set of the form

$$\{t_1/V_1(\ell_1),\ \ldots,\ t_r/V_r(\ell_r)\}$$

where each t_i is a term, each V_i is a variable, and each ℓ_i is a list of positive integers. A consistency restriction is made that if the same term appears to the left of the slash in two different elements of the substitution set, then the associated lists must be disjoint. For example, {a/X(1,4,5), b/Y(2), a/Z(2,3)} is a legitimate substitution but {a/X(1,4,5), a/Z(2,3,4)} is not.

If ρ is a product and θ is such a substitution, $\rho\theta$ represents the product obtained from ρ by simultaneously replacing the occurrences of term t_i, in the positions specified by the list ℓ_i, by the variable V_i. For example, for θ = {a/X(1,2), b/Y(1)}, the English interpretation reads: "Replace the first and second occurrences of a by X and the first occurrence of b by Y." Thus if

$$\rho = (p\ a\ b)\wedge\sim(e\ a)\wedge(g\ a) \quad \text{then}$$

$$\rho\theta = (p\ X\ Y)\wedge\sim(e\ X)\wedge(g\ a)$$

Note especially that the third occurrence of a in ρ remains unchanged.

This work was partially supported by DHEW Grant No. US-PHS-R01-MB-00114-01.

An additional restriction will be placed on substitutions: no variable in ρ may appear to the right of the slash in a substitution applied to ρ. For example, if $\theta_2 = \{a/X(1)\}$, application of θ_2 to $\rho_2 = (p\ a\ X)$ is prohibited since X already occurs in ρ_2. This restriction is necessary to ensure that ρ is always an instance of $\rho\theta$. A useful convention is to allow the list ℓ_i to be omitted if all occurrences of t_i are to be replaced by V_i. It should also be clear that a substitution in general presumes a particular left to right ordering of the literals of a product.

Let $\rho_1 \subseteq \rho_2$ denote that the literals comprising product ρ_1 are a subset of the literals comprising product ρ_2. A product ρ_1 is a generalization of a product ρ_2 iff there exists a substitution θ such that $\rho_1 \subseteq \rho_2\theta$. This will be denoted by $\rho_1 \leq \rho_2$. For example, $(p\ X\ Y) \leq (p\ a\ b) \wedge (q\ a\ c)$ since

$$(p\ X\ Y) \subseteq (p\ X\ Y) \wedge (q\ X\ c) = (p\ a\ b) \wedge (q\ a\ c) \quad \{a/X,\ b/Y\}$$

A product ρ_a is a unifying generalization for a set of products $\{\rho_1, \rho_2, \ldots, \rho_r\}$ iff $\rho_a \leq \rho_i$, $1 \leq i \leq r$. For example, $(p\ a\ X)$ is a unifying generalization for $(p\ a\ b)$, $(p\ a\ c)$, and $(p\ a\ g)$.

A substitution θ is an alphabetic variant substitution, written θ^{av}, iff each element of the substitution has the form V_{i1}/V_{i2}, that is θ^{av} merely changes the names of the variables in a product in a consistent manner. If ρ_1 and ρ_2 are two arbitrary products, three elementary and easily demonstrable properties of generalizations are:

P1: $(\rho_1 \leq \rho_2) \wedge (\rho_2 \leq \rho_1) \leftrightarrow (\rho_1 = \rho_2\theta^{av})$ for some θ^{av}. In this case ρ_1 is a trivial generalization of ρ_2 and conversely.

P2: If $\rho_1 \leq \rho_2$ and no θ^{av} exists such that $\rho_1 = \rho_2\theta^{av}$, either $\rho_1 \subset \rho_2\theta_1$ for some θ_1, or $\rho_1 = \rho_2\theta_2$, where θ_2 is not an alphabetic variant substitution. This situation will be denoted by $\rho_1 < \rho_2$.

P3: (transitivity) $(\rho_1 \leq \rho_2) \wedge (\rho_2 \leq \rho_3) \rightarrow (\rho_1 \leq \rho_3)$

As examples of the above,

$(p\ a\ b) < (p\ a\ b) \wedge (q\ a\ c)$

$(p\ X\ b) < (p\ a\ b)$

$(p\ X\ b) \wedge (q\ Y\ c) \cong (p\ U\ b) \wedge (q\ Z\ c)$ $\{U/X,\ Z/Y\}$,

where $\{U/X, Z/Y\}$ is an alphabetic variant substitution.

If $\gamma_1 \leq \rho_1$ and $\gamma_1 \leq \rho_2$, γ_1 is a maximal unifying generalization of ρ_1 and ρ_2 iff no product γ_2 exists such that $\gamma_1 < \gamma_2$, $\gamma_2 \leq \rho_1$, and $\gamma_2 \leq \rho_2$. There may be a number of distinct maximal unifying generalizations of two products, ignoring variants caused by alphabetic variant substitutions. Let $MG(\rho_1, \rho_2)$ represent the set of all generalizations γ such that:

1) γ is a maximal unifying generalization of ρ_1 and ρ_2;
2) no generalization γ' and θ^{av} exist such that $\gamma' \in MG(\rho_1, \rho_2)$ and $\gamma = \gamma'\theta^{av}$.

Theorem 1: A product γ_1 is a unifying generalization of products ρ_1 and ρ_2 iff there exists a product $\gamma_m \in MG(\rho_1, \rho_2)$ such that $\gamma_1 \leq \gamma_m$.

Proof: ($\rightarrow$) If γ_1 is a unifying generalization of ρ_1 and ρ_2, it must be either maximal or non-maximal. If it is maximal, there is a product $\gamma_m \in MG(\rho_1, \rho_2)$ such that $\gamma_1 = \gamma_m\theta^{av}$. Hence, $\gamma_1 \leq \gamma_m$. If γ_1 is non-maximal, it is because another unifying generalization γ_2 exists such that $\gamma_1 < \gamma_2$. γ_2 must have more literals or fewer variables than γ_1, or both. No unifying generalization may have more literals or fewer variables than the extrema of these parameters for ρ_1 and ρ_2, which are of course finite. Consequently, there will in general be a finite chain of generalizations $\gamma_1 < \gamma_2 < \ldots < \gamma_m$, where γ_m is a maximal unifying generalization. By the transitivity property, $\gamma_1 \leq \gamma_m$. From the definition of $MG(\rho_1, \rho_2)$, γ_m is either an element of this set or an alphabetic variant of an element.
($\leftarrow$) γ_m is a maximal unifying generalization of ρ_1 and ρ_2, and $\gamma_1 \leq \gamma_m$. Since $\gamma_m \leq \rho_1$ and $\gamma_m \leq \rho_2$, transitivity gives $\gamma_1 \leq \rho_1$ and $\gamma_1 \leq \rho_2$, and so γ_1 is a unifying generalization of ρ_1 and ρ_2.

Thus any of the potentially large number of all unifying generalizations of ρ_1 and ρ_2 can be obtained by generalizing some element of $MG(\rho_1, \rho_2)$, a much smaller set.

2.2 Problem Statement--Simplification of Concepts by Induction

A concept may be defined "in extension" by exhibiting positive and negative instances of the concept. It is assumed here that each instance is described by a product in the subset of the predicate calculus defined in subsection 2.1. Henceforth, let ρ_i represent a product describing a positive instance, and ν_i a product describing a negative instance. Then the following disjunctive normal form expressions can represent a concept K:

$$\bigvee_i \rho_i \rightarrow K, \quad \bigvee_i \nu_i \rightarrow \sim K \qquad (1)$$

If for some indices i_1 and i_2, $\rho_{i_1} = \nu_{i_2}$, the representation is inconsistent. Each ρ_i is a positive product; each ν_i is a negative product. On the other hand, a negative literal is a predicate prefixed by " ~ "; a positive literal is a predicate not not prefixed by " ~ ".

In the above context, a useful generalization will be one which unifies at least two positive products or two negative products, since the concept representation is thereby simplified. However, it is essential that this generalization not destroy discrimination between K and ~K. Because of problem symmetry, henceforth only generalizations of positive products will be discussed; identical arguments will apply to generalization of negative products. If γ is a generalization of a product ρ_{i1}, γ is consistent iff $\gamma \not\leq \nu_i$ for all i.

Problem: Given a concept definition in extension, as in (1), how can consistent, unifying generalizations of the positive products be computed in order to simplify the definition?

It is to this specific problem that the following discussions are addressed.

3. The Product Graph as an Induction Medium

In this section an induction procedure is formulated with respect to a graph representation of products, called a "product graph". Unlike some earlier graph representations of relational information, the product graph is not limited to binary relations. It represents literals of arbitrary dimension in a uniform manner. The overall effect of the induction procedure is to merge pairs of positive products. It may be iteratively applied tinti 1 all positive products have been merged, or until no maximal consistent generalizations exist for any pair of products. This procedure is precisely described and its correctness is proven.

The product graph representation has properties that make it useful as an induction medium. If a product p_1 is a generalization of another product p_2, the product graph of p_1 is isomorphic to a subgraph of the product graph of p2. A "greatest common subgraph" of two product graphs is the graph of a maximal unifying generalization of those two products. Thus the problem of computing maximal unifying generalizations is transformed to the computation of maximal isomorphic subgraphs.

3.1 Product Graphs

A product graph is a 6-tuple system $(L, C, \delta, \sigma, \kappa, \alpha)$, where

1. L is a finite set of literal nodes;
2. C is a finite set of constant nodes;
3. δ is the literal dimension function: $L \to Z_+$, where Z_+ is the set of positive integers;
4. σ is the literal sign function: $L \to \{+, -\}$;
5. κ is the literal content partial function: $L \times Z_+ \to C$, such that if $(\ell, i, c) \in \kappa$, then $i \in \{1, 2, \ldots, \delta(\ell)\}$
6. α is the literal adjacency relation, a finite subset of $Z_+ \times Z_+ \times L \times L$, such that
   a) $(i_1, i_2, \ell_1, \ell_2) \in \alpha$ iff $(i_2, i_1, \ell_2, \ell_1) \in \alpha$ (a symmetry property);
   b) if $(i_1, i_2, \ell_1, \ell_2) \in \alpha$ and $(i_2, i_3, \ell_2, \ell_3) \in \alpha$ then $(i_1, i_3, \ell_1, \ell_3) \in \alpha$ (a transitivity property);
   c) for all $\ell_1, \ell_2 \in L$ and $i_1, i_2 \in Z_+$, if $(\ell_1, i_1, c_1) \in \kappa$ and $(\ell_2, i_2, c_2) \in \kappa$ then $(i_1, i_2, \ell_1, \ell_2) \in \alpha$ iff $c_1 = c_2$ (a consistency property).

Each literal in a product will be represented by a unique node in L. Each distinct constant appearing in a product will be represented by a unique element of C. Variables appearing in a pro duct will have no explicit representation. Recall that the variables are universally quantified and thus are dummy variables: their names have no significance. The literal dimension function ? specifies the number of terms in each literal. For example, if λ_1 = (a b c), 6(A) = 3. The literal sign function o serves to distinguish positive and negative literals. For example, with A as above and $\lambda_2 = \sim(e\ f)$, $\sigma(\lambda_1) = +$ and $\sigma(\lambda_2) = -$.

The purpose of the content function κ is to specify the identity and position of constants which appear in the literals of a product. Specifically, if $\lambda_1 \in L$ and $c_1 \in C$, $\kappa_i(\lambda_1) = c_1$ iff the constant c_1 appears in the ith position in λ_1. For example, λ_1 = (c Y e) causes the following specification of κ: $\kappa_1(\lambda_1) = c$ and $\kappa_3(\lambda_1) = e$.

If the same term appears in two literals of a product, these literals are adjacent in the product graph. For example, the two literals in the product (a b c) $\wedge$ (c d e) are adjacent, since the constant c appears in both. To describe precisely the adjacency of two literals, the position of the common term in each literal must be specified. If λ_1 and λ_2 are adjacent literals, the adjacency relation $\alpha_{i,j}(\lambda_1, \lambda_2)$ represents the fact that a term which appears in the ith position in λ_1 also appears in the jth position in λ_2. By the symmetry property of adjacencies, $\alpha_{i,j}(\lambda_1, \lambda_2) \leftrightarrow \alpha_{j,i}(\lambda_2, \lambda_1)$ for all $i, j, \lambda_1, \lambda_2$. Consequently, it is possible to disregard adjacencies for which $i > j$ as redundant information.

A literal may be adjacent to itself, causing a sling in the product graph. If the same term appears in position i and position j in literal λ_1, $\alpha_{i,j}(\lambda_1, \lambda_1)$ is significant. However, if $i = j$, no information is provided, since this relation holds universally; it sill be ignored.

These adjacency relations are the arcs between literal nodes in the product graph. In an obvious way, these arcs can partition the literal nodes into disjoint blocks, with the nodes in each block forming a connected subgraph of the complete product graph. These connected subgraphs will be important in the concept of a "vacuous" generalization, which is discussed in subsection 3.4.

3.2 Product Graph Example

As an example, Figure 1 shows the product graph for the product (g X Y c) $\wedge$ ~(c Y e) $\wedge$ (c f), with the individual literals identified by λ_1, λ_2, and λ_3 respectively. Constant nodes are shown as squares; literal nodes are shown as circles. This example incidentally demonstrates that an adjacency relation due to a common constant in two literals is redundant information, which could be derived from the content function information. This mild and occasional redundancy contributes to the desirable properties of product graphs which are next discussed.

3.3 Properties of Product Graphs

Two product graphs g_1 and g_2 are isomorphic, written $g_1 \cong g_2$, iff they are identical except for their literal node identifiers. That g_1 is a subgraph of g_2 will be denoted by $g_1 \subseteq g_2$. That g_1 is isomorphic to a subgraph or a proper subgraph of g_2 will be denoted by $g_1 \tilde{\subseteq} g_2$ or $g_1 \tilde{\subset} g_2$ respectively. $G(\rho)$ will denote the graph of product ρ.

Product graphs have the following easily demonstrable properties:

P4: $G(\rho_1) \cong G(\rho_2)$ iff $\rho_1 = \rho_2\theta^{av}$ for some θ^{av}, i.e., if product graphs are isomorphic, the corresponding products are alphabetic variants;

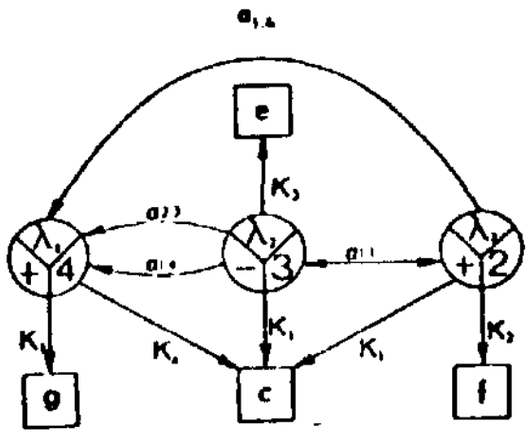
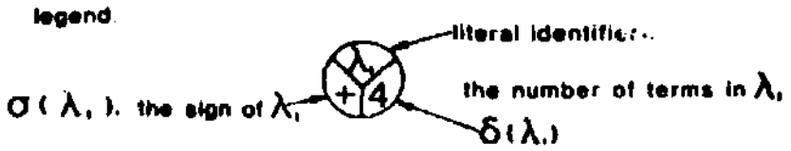


Figure 1

Product graph of (gXYc) ∧ ~ (cYe) ∧ (cf)

legend



P5: $G(\rho_1) \tilde{\subseteq} G(\rho_2)$ iff $\rho_1 \leq \rho_2$
P6: $G(\rho_1) \tilde{\subseteq} G(\rho_2)$ iff $\rho_1 \leq \rho_2$

P7: A product γ is a maximal, unifying generalization of ρ_1 and ρ_2 iff:
1. $G(\gamma) \tilde{\subseteq} G(\rho_1)$
2. $G(\gamma) \tilde{\subseteq} G(\rho_2)$
3. no graph g_4 exists such that
   a. $G(\gamma) \tilde{\subset} g_4$
   b. $g_4 \tilde{\subseteq} G(\rho_1)$
   c. $g_4 \tilde{\subseteq} G(\rho_2)$

The first two conditions specify that γ is a unifying generalization; the third specifies that γ is maximal.

The properties P5, P6, and P7 are significant for showing that generalizations may be computed with product graphs, as elaborated below.

3.4 Computing Maximal Unifying Generalizations

The procedure for obtaining the graph of a maximal unifying generalization requires a few additional concepts. Two literal nodes λ_1 and λ_2 are potentially similar, written psim (λ_1, λ_2), iff they meet these conditions:

C1: $\sigma(\lambda_1) = \sigma(\lambda_2)$, i.e., λ_1 and λ_2 have the same sign;

C2: $\delta(\lambda_1) = \delta(\lambda_2)$, i.e., λ_1 and λ_2 have the same number of terms;

C3: λ_1 and λ_2 appear in separate products.

C3 affirms that only literal nodes in separate product graphs shall be eligible: we do not wish to merge literals in the same product.

Two literal nodes which are potentially similar will be called a pair. Two pairs which have a node in common are alternative pairs. For example, $\rho_1 = (\lambda_1, \lambda_2)$ and $\rho_2 = (\lambda_1, \lambda_3)$ are alternative pairs since λ_1 is a constituent of both. Let $P(\rho_1, \rho_2)$ denote the set of all pairs for two arbitrary products ρ_1 and ρ_2. Certain subsets of $P(\rho_1, \rho_2)$ will be of special importance. We define $s \in CP(\rho_1, \rho_2)$ iff:

1. $s \subseteq P(\rho_1, \rho_2)$
2. no two pairs in s are alternative pairs.

Stated differently, each element of $CP(\rho_1, \rho_2)$ represents a possible 1-1 mapping between literal nodes of ρ_1 and ρ_2 subject to the restriction that matched nodes have the same sign and dimension.

There is a class of generalizations which will be here called "vacuous". A vacuous generalization γ is one for which a connected subgraph of $G(\gamma)$ contains no constant nodes. For example, if $\rho_1 = (a\ b\ c) \wedge (d\ a)$ and $\rho_2 = (c\ a\ q) \wedge (e\ c)$, the only maximal unifying generalization is $(X\ Y\ Z) \wedge (U\ V)$. It seems desirable to exclude such generalizations. Their exclusion will also speed the computation of the remaining non-vacuous generalization. For these advantages, we will weed out additional elements of $CP(\rho_1, \rho_2)$ with the concept of "similarity".

Let $s = \{p_1, p_2, \ldots, p_m\}$ be an element of $CP(\rho_1, \rho_2)$. Let $p \triangleq (\lambda_1, \lambda_2)$ represent an arbitrary pair in s. λ_1 and λ_2 are similar, written $sim(\lambda_1, \lambda_2)$, iff these conditions hold:
$C4 \wedge (C5 \vee C6)$, where

C4: psim(λ_1, λ_2)
C5: $\exists i\ \kappa_i(\lambda_1) = \kappa_i(\lambda_2)$, i.e., λ_1 and λ_2 have a common term in some position i.
C6: $\exists i, j$ and $(\lambda_3, \lambda_4) \in s$ such that $\alpha_{i,j}(\lambda_1, \lambda_3) \wedge \alpha_{i,j}(\lambda_2, \lambda_4) \wedge sim(\lambda_3, \lambda_4)$, i.e., λ_1 and λ_2 have an identical adjacency to similar nodes.

Now let $SP(\rho_1, \rho_2)$ be defined such that $s \in SP(\rho_1, \rho_2)$ iff
1. $s \in CP(\rho_1, \rho_2)$
2. $\forall i\ (p_i \in s) \rightarrow sim(p_i)$, i.e., every pair of literal nodes in s is similar;
3. no s' exists such that $s' \in SP(\rho_1, \rho_2)$ and $s \subset s'$.

From each aggregate of pairs s in $SP(\rho_1, \rho_2)$, a product graph g can be obtained by this construction:
1. Each pair in s is a literal node of g.
2. For all pairs $p_i = (\lambda_1, \lambda_2) \in s$ if $\exists j\ \kappa_j(\lambda_1) = \kappa_j(\lambda_2)$, then $\kappa_j(\lambda_1)$ is a constant node of g, and the content function $\kappa_j(p_i) = \kappa_j(\lambda_1)$ is defined.
3. For all pairs $p_a = (\lambda_1, \lambda_2)$, $p_b = (\lambda_3, \lambda_4)$ in s, if $\alpha_{n1,n2}(\lambda_1, \lambda_3) \wedge \alpha_{n1,n2}(\lambda_2, \lambda_4)$ then the adjacency relation $\alpha_{n1,n2}(p_a, p_b)$ is defined for g.

In this way g reflects all common features of the pairs in s. Let $G'(SP(\rho_1, \rho_2))$ denote the set of graphs g constructed in this way from the pair aggregates of $SP(\rho_1, \rho_2)$, and let $G(SP(\rho_1, \rho_2))$ denote a reduced subset of $G'(SP(\rho_1, \rho_2))$ such that $g \in G(SP(\rho_1, \rho_2))$ iff:
1. $g \in G'(SP(\rho_1, \rho_2))$, and
2. no $g' \in G'(SP(\rho_1, \rho_2))$ exists such that $g \tilde{\subseteq} g'$.

Theorem 2: A product γ is a maximal, non-vacuous, unifying generalization of products ρ_1 and ρ_2 iff $G(\gamma) \tilde{=} g_1$ for some $g_1 \in G(SP(\rho_1, \rho_2))$.

Proof: ($\rightarrow$) Define $G'(\rho_1)$ and $G'(\rho_2)$ such that $G(\gamma) \tilde{=} G'(\rho_1) \subseteq G(\rho_1)$ and $G(\gamma) \tilde{=} G'(\rho_2) \subseteq G(\rho_2)$

Then of course $G'(\rho_1) \tilde{=} G'(\rho_2)$. Let $G(\gamma)$, $G'(\rho_1)$ and $G'(\rho_2)$ be partitioned into their connected subgraphs:

$G(\gamma) = G_1(\gamma) \cup G_2(\gamma) \cup \ldots \cup G_r(\gamma)$
$G'(\rho_1) = G'_1(\rho_1) \cup G'_2(\rho_1) \cup \ldots \cup G'_r(\rho_1)$
$G'(\rho_2) = G'_1(\rho_2) \cup G'_2(\rho_2) \cup \ldots \cup G'_r(\rho_2)$.

It follows that the number of subgraphs in $G(\gamma)$, $G'(\rho_1)$, and $G'(\rho_2)$ are identical, as indicated,

and $\forall i \quad G_i^!(\gamma) \cong G_i^!(\rho_1) \cong G_i^!(\rho_2)$. (2)
Let L(g) represent the literal nodes of an arbitrary product graph g.

Let $L(G_i(\gamma)) = \{\lambda 0_{i,1}, \lambda 0_{i,2}, \ldots, \lambda 0_{i,n_i}\}$,
$L(G_i^!(\rho_1)) = \{\lambda 1_{i,1}, \lambda 1_{i,2}, \ldots, \lambda 1_{i,n_i}\}$, and
$L(G_i^!(\rho_2)) = \{\lambda 2_{i,1}, \lambda 2_{i,2}, \ldots, \lambda 2_{i,n_i}\}$.
Because of (2), it must be possible to establish a 1 to 1 correspondence between the nodes of these connected subgraphs. Let the binary predicate m (for match) indicate that its arguments are established in this correspondence. Then $\forall\, i,j\ m(\lambda 0_{i,j}, \lambda 1_{i,j}) \wedge m(\lambda 0_{i,j}, \lambda 2_{i,j}) \wedge m(\lambda 1_{i,j}, \lambda 2_{i,j})$.
Since γ is non-vacuous, each $G_i(\gamma)$ must contain at least one constant node, and so $\forall i\ \exists j_1, j_2$ $\kappa_{j_1}(\lambda 0_{i,j_2}) = \kappa_{j_1}(\lambda 1_{i,j_2}) = \kappa_{j_1}(\lambda 2_{i,j_2})$ Then by condition C4 for similarity, $\forall i\ sim(\lambda 1_{i,j_1}, \lambda 2_{i,j_2})$. Since each of these subgraphs is connected, condition C5 will cause the similarity to propagate throughout the subgraphs, giving

$$\forall i,j \quad sim(\lambda 1_{i,j}, \lambda 2_{i,j}) \leftrightarrow m(\lambda 1_{i,j}, \lambda 2_{i,j}). \tag{3}$$

For all i and j, let $p_{i,j} = (\lambda 1_{i,j}, \lambda 2_{i,j})$, and let s represent this set of pairs. Clearly none of these pairs is an alternative to any other. Before determining if $s \in SP(\rho_1, \rho_2)$, we construct g from s in accordance with the specified construction:

$$\forall i,j \quad \sigma(p_{i,j}) = \sigma(\lambda 1_{i,j})$$

$$\forall i,j \quad \delta(p_{i,j}) = \delta(\lambda 1_{i,j})$$

$$\forall i,j,n \ [\kappa_n(\lambda 1_{i,j}) = \kappa_n(\lambda 2_{i,j})] \rightarrow [\kappa_n(p_{i,j}) = \kappa_n(\lambda 1_{i,j})]$$

$$\forall i, j_1, j_2, n_1, n_2 \quad \alpha_{n_1,n_2}(\lambda 1_{i,j_1}, \lambda 1_{i,j_2}) \wedge \alpha_{n_1,n_2}(\lambda 2_{i,j_1}, \lambda 2_{i,j_2}) \rightarrow \alpha_{n_1,n_2}(p_{i,j_1}, p_{i,j_2})$$

This last fact can be simplified to:

$$\forall i, j_1, j_2, n_1, n_2 \ \alpha_{n_1,n_2}(\lambda 1_{i,j_1}, \lambda 1_{i,j_2}) \rightarrow \alpha_{n_1,n_2}(p_{i,j_1}, p_{i,j_2})$$

since $\forall i\ G_i^!(\rho_1) \cong G_i^!(\rho_2)$.
Thus $g \cong G'(\rho_1)$, and hence $g \cong G(\gamma)$.

To determine if s is in $SP(\rho_1, \rho_2)$, the three conditions for membership must be examined. The first is satisfied, since the nodes matched in confirming isomorphism must necessarily be of the same sign and dimension, and be non-alternative. s was seen to be composed of such pairs. The second condition requires that each pair have the similarity property. This was found to be true in (3) above. To test the third condition, suppose s' exists such that $s \subset s'$ and $s' \subseteq SP(\rho_1, \rho_2)$. Let g' represent the graph constructed from s'. g' is non-vacuous since it is constructed from similar pairs. Since $s \subset s'$, it follows that $g \tilde{\subset} g'$, but $g \cong G(\gamma)$, so $G(\gamma) \tilde{\subset} g'$, which implies that γ is not maximal, contrary to fact. Thus no such s' can exist. Consequently, $s \in SP(\rho_1, \rho_2)$, and hence $g \in G'(SP(\rho_1, \rho_2))$. Since γ is maximal, no g' exists such that $g \tilde{\subset} g'$, and so $g \in G(SP(\rho_1, \rho_2))$ also.
($\leftarrow$) This half of the proof is accomplished by showing that a) g_1 is the graph of a unifying generalization of ρ_1 and ρ_2;
b) no graph g' can exist such that $g \tilde{\subset} g'$ and g' is the graph of a unifying generalization of ρ_1 and ρ_2.
c) g satisfies the requirements for the non-vacuous property.

a) It is necessary to show $g_1 \tilde{\subseteq} G(\rho_1)$ and $g_1 \tilde{\subseteq} G(\rho_2)$. For all i, if $\lambda_i \in L(g_1)$ there exists a pair (λ_j, λ_k), where $\lambda_j \in L(G(\rho_1))$ and $\lambda_k \in L(G(\rho_2))$, and $sim(\lambda_j, \lambda_k)$. This follows from the fact that g_1 is constructed from pairs of literal nodes from $G(\rho_1)$ and $G(\rho_2)$. Then a one to one match can be established $m(\lambda_i, \lambda_j)$, $m(\lambda_i, \lambda_k)$. This match relation then has the property that

$\forall i,j\ m(\lambda_i, \lambda_j) \rightarrow (\sigma(\lambda_i) = \sigma(\lambda_j))$
$\forall i,j\ m(\lambda_i, \lambda_j) \rightarrow (\delta(\lambda_i) = \delta(\lambda_j))$
$\forall i,j,n\ m(\lambda_i, \lambda_j) \wedge (\kappa_n(\lambda_i)$ is defined$) \rightarrow (\kappa_n(\lambda_i) = \kappa_n(\lambda_j))$

Finally, since g_1 is constructed to reflect all identical adjacencies of paired nodes,

$$\forall n_1, n_2, i_1, i_2 \quad \alpha_{n_1,n_2}(\lambda_{i_1}, \lambda_{i_2}) \rightarrow [\exists j_1, j_2\ \alpha_{n_1,n_2}(\lambda_{j_1}, \lambda_{j_2}) \wedge \exists k_1, k_2\ \alpha_{n_1,n_2}(\lambda_{k_1}, \lambda_{k_2})]$$

where $m(\lambda_{i_1}, \lambda_{j_1})$, $m(\lambda_{i_2}, \lambda_{j_2})$, $m(\lambda_{i_1}, \lambda_{k_1})$, and so $g_1 \tilde{\subseteq} G(\rho_1)$ and $g_1 \tilde{\subseteq} G(\rho_2)$.
b) A g' is assumed to exist such that $g \tilde{\subset} g'$, $g' \tilde{\subseteq} G(\rho_1)$, and $g' \tilde{\subseteq} G(\rho_2)$. Then $G''(\rho_1)$ and $G''(\rho_2)$ exist that

$g' \cong G''(\rho_1) \subseteq G(\rho_1)$
$g' \cong G''(\rho_2) \tilde{\subseteq} G(\rho_2)$

The literal nodes of $G''(\rho_1)$ and $G''(\rho_2)$ may be paired to form a set $s' \in SP(\rho_1, \rho_2)$. From s' can be constructed a graph $g'' \cong G''(\rho_1) \cong G''(\rho_1) \cong g'$. However if $g \tilde{\subset} g' \cong g''$, then g is not in $G(CP(\rho_1, \rho_2))$, which is contrary to fact. Thus no such g' can exist, and hence γ is maximal.

c) Since $G(\gamma) \cong g_1 \in G(CP(\rho_1, \rho_2)) \subseteq G'(CP(\rho_1, \rho_2))$, and all graphs in $G'(CP(\rho_1, \rho_2))$ are constructed from similar pairs, γ is non-vacuous.

3.5 Example -- Computation of Maximal Unifying Generalizations

For two simple but nontrivial products, the sets $P(\rho_1, \rho_2)$, $CP(\rho_1, \rho_2)$, $SP(\rho_1, \rho_2)$, $G'(SP(\rho_1, \rho_2))$ and $G(SP(\rho_1, \rho_2))$ will be exhibited.
Suppose $\rho_1 \vee \rho_2 \rightarrow K$, where
$\rho_1 = (b\ x2) \wedge (w\ x3) \wedge (c\ x2\ r) \wedge (s\ x3\ x2)$
$\rho_2 = (w\ x6) \wedge (b\ x5) \wedge (s\ x6\ x5) \wedge (d\ x6\ e)$

$P(\rho_1, \rho_2) = \{P1, P2, P3, P4, P5, P6, P7, P8\}$, where

P1 = (b x2), (w x6)
P2 = (b x2), (b x5)
P3 = (w x3), (w x6)
P4 = (w x3), (b x5)
P5 = (c x2 r), (s x6 x5)
P6 = (c x2 r), (d x6 e)
P7 = (s x3 x2), (s x6 x5)
P8 = (s x3 x2), (d x6 e)

Alternative pairs are:
P1/P2, P1/P3, P3/P4, P2/P4,
P5/P6, P6/P8, and P7/8.
Hence $CP(\rho_1, \rho_2)$ consists of the six combinations:

(P1, P4, P5, P7), (P1, P7, P5, P8),
(P1, P4, P6, P7), (P2, P3, P5, P7),
(P2, P3, P5, P8) and (P2, P3, P6, P7),

as well as their subsets. The set CP is always very large, and has significance only as a theoretical construct. In practice, only $SP(\rho_1, \rho_2)$ need be computed. $SP(\rho_1, \rho_2) = \{s_1, s_2\}$, where s_1 = {P2, P3, P7}, and s_2 = {P2, P3, P8}. The product graphs constructed from s1 and s2 are shown in Figures 2a and 2b respectively. Here each literal node has been labelled with the pair name from which it derives. Since 2b is a subgraph of 2a, 2a constitutes the only element of $G(SP(\rho_1, \rho_2))$, and thus represents the only maximal, nonvacuous, unifying generalization of ρ_1 and ρ_2. The linear form of 2a may then be obtained to give:
$(w\ Y) \wedge (b\ Z) \wedge (s\ Y\ Z) \rightarrow K$

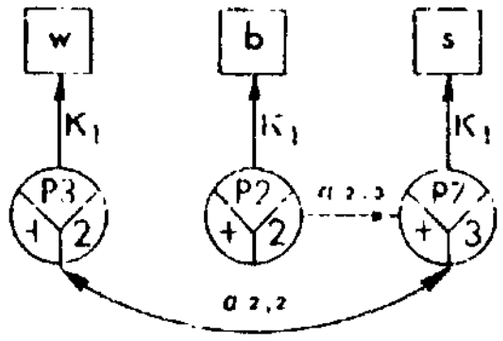


Figure 2a Graph constructed from {P2, P3, P7}

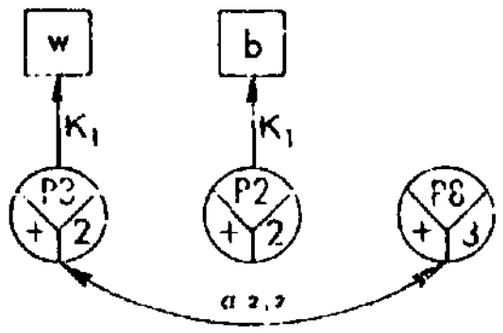


Figure 2b Graph constructed from {P2, P3, P8}

4. Determining the Consistency of a Generalization

Recall from subsection 2.2 that a generalization of positive instances of a concept is inconsistent if it is also a generalization of a negative instance. Property P6 of subsection 3.3 suggests a way to determine consistency.

Theorem 3: If γ is a generalization of positive products, γ is consistent iff $G(\gamma) \not\subseteq G(\nu_i)$ for all i.

Proof: If γ is consistent, no ν_i exists such that $\gamma \leq \nu_i$, and so by property P6 no ν_i exists such that $G(\gamma) \subseteq G(\nu_i)$.

In other words, consistency of a generalization can be determined by verifying that its product graph is not isomorphic to a subgraph of any negative product.

The following lemma and theorem demonstrate the significance of maximal generalizations in determining the existence of any consistent generalization.

lemma 1: If γ_1 is an inconsistent generalization, no generalization of γ_1 is consistent.

Proof: $\gamma_1 \leq \rho_1$, $\gamma_2 \leq \rho_2$, and a negative product ν_{i_1} exists such that $\gamma_1 \leq \nu_{i_1}$. If γ_2 is any generalization of γ_1, $\gamma_2 \leq \gamma_1 \leq \nu_{i_1}$ or $\gamma_2 \leq \nu_{i_1}$, and so γ_2 is also inconsistent.

Theorem 4: If for all $g_k \in G(SP(\rho_1, \rho_2))$, a negative product ν_{i_k} exists such that $g_k \subseteq G(\nu_{i_k})$, then no consistent generalization of ρ_1 and ρ_2 exists.

Proof: Define γ_k such that $G(\gamma_k) = g_k$. By theorem 1, if γ_a is any unifying generalization of ρ_1 and ρ_2, then $\gamma_a \leq \gamma_k$ for some γ_k. However, $G(\gamma_k) = g_k \subseteq G(\nu_{i_k})$ and so by theorem 3 γ_k is inconsistent. Hence by lemma 1, γ_a is also inconsistent.

To continue the example of subsection 3.5, suppose a negative product is known: $\nu_1 \rightarrow \neg K$, where $\nu_1 = (c\ x8\ r) \wedge (b\ x8) \wedge (w\ x9)$. The product graph of ν_1 is given in Figure 3. In light of this additional information, is the generalization graphed in Figure 2a consistent? Yes, since it is readily determined that Figure 2a is not isomorphic to a subgraph of Figure 3.

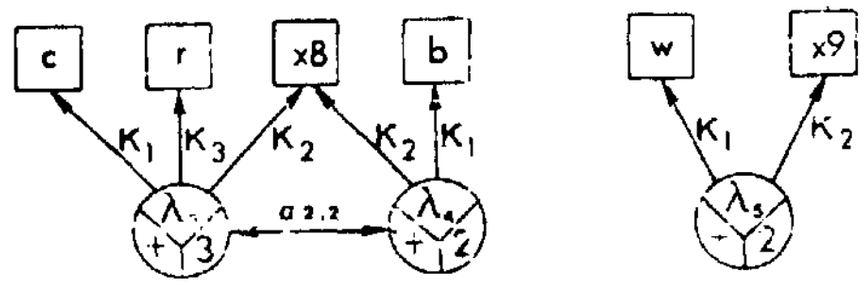


Figure 3 Graph of $\underbrace{(c\ x8\ r)}_{\lambda_3} \wedge \underbrace{(b\ x8}_{\lambda_4} \wedge \underbrace{w\ x9)}_{\lambda_5}$

5. Concluding Remarks

A precise yet mathematically parsimonious induction procedure has been described for concepts in the predicate calculus. The product graph representation accommodates literals having an arbitrary number of terms. The computation of generalizations and the testing of their consistency has been achieved by comparison operations on these product graphs. The information supplied by negative instances of a concept determines if a generalization is consistent, and may serve to select between a number of distinct maximal generalizations. If all maximal generalizations are inconsistent, it has been shown that no consistent generalization exists. If a maximal generalization is consistent, it may then be generalized further, holding short of the point where it would "cover" a negative product. A SNOBOL4 program has been written which implements and confirms the methods of this paper. Exploration of its capabilities is in progress.

Because a generalization replaces just two products, and a number of distinct maximal generalizations may exist, there remains the question of strategies in the application of the induction procedure in an iterative manner to a large number of positive products.

Worthwhile departures from present theory would include concepts which are iterative orpotentially infinite, such as the sequence abbecedddd. .., as well as hierarchial and recursive concepts, all of which seem essential to human perception and thought.

Acknowledgment

The author appreciatively acknowledges a number of helpful criticisms by Professor Roger Conant and the IJCAI-75 referees of an earlier version of this work.

References

1. Banerji, R.B. Theory of Problem Solving, chapter 5, "Learning and Generalization", American Elsevier, 1969, pp. 153-177.
2. Brown, J.S. Steps Toward Automatic Theory Formation. Proceedings of the Third Intl. Joint Conference on Artificial Intelligence, 1973, pp. 121-129.
3. Evans, T.G. A Program for the Solution of Geometric-Analogy Intelligence Test Questions. Semantic Information Processing, M. Minsky (ed.). MIT Press, 1968, pp. 271-353.
4. Hunt, E.B., Marin, J. and Stone, P.J. Experiments in Induction. Academic Press, 1966.
5. Kling, R.F.. Reasoning by Analogy with Applications to Heuristic Problem Solving: A Case Study. Ph.D. thesis, Stanford University, 1971.
6. Meltzer, B. Generation of Hypotheses and Theories. Nature, Vol. 225, March 7, 1970.
7. Michalski, R.S. Learning by Inductive Inference. NATO Seminar on Computer Oriented Learning Processes, Aug. 26-Sept. 7, 1974.
8. Plotkin, G.I). Automatic Methods of Inductive Inference. Ph.D. thesis, Dept. of Machine Intelligence and Perception, University of Edinburgh, 1971.
9. Towster, *t.* Several Methods of Concept-Formation by Computer. Ph.D. thesis, University of Wisconsin, 1970.
10. Webster's Third New Intl. Dictionary of the English Language, Merriam, 1969.
11. Williams, D.S. Computer Program Organization Induced from Problem Examples. Representation and Meaning, H.A. Simon and L. Siklossy (eds.), Prentice-Hall, 1972, pp. 143-205.
12. Winston, P.H. Learning Structural Descriptions from Examples. Ph.D. thesis, Massachusetts Institute of Technology, 1970.