The Common Order-theoretic Structure of Version Spaces and ATMS's*
(*Extended Abstract*)

Carl A. Gunter
University of Pennsylvania

Teow-Hin Ngair
University of Pennsylvania

Prakash Panangaden
McGill University

Devika Subramanian
Cornell University

1. Introduction

This paper arose out of the observation that the version space algorithm and the ATMS label-update algorithms operate on very similar structures. The version space algorithm learns concept descriptions from examples. Central to this algorithm is the notion of all concept descriptions consistent with a given set of positive and negative examples. The assumption-based truth maintenance system for recording dependencies during reasoning maintains labels for a proposition which encode all environments in which that proposition is true.

This gives rise to two questions: one, what is the precise nature of the relationship between the structures employed by these algorithms taken from two different problem areas, and second: are the *computations* performed by the two algorithms related, and what special properties of these structures do they depend on? Our aim is to find a common mathematical basis in order to determine applicability conditions for the algorithms, and to cast the computations done in a form that reveals new, more efficient implementations.

We first show that the version space of a concept is a special case of a convex space. The mathematics of these structures is then brought to bear on the question of ensuring that finite representability is preserved under the merging operation. For this, we derive a necessary and sufficient condition, called the $\mathcal{MW}$ property. This can help in determining the class of *admissible* concept languages for which finite observation version spaces are finitely representable. We present the first result on admissibility that captures both finite and infinite concept languages.

We then recast the label-update algorithms in the ATMS as operations on a convex space. The chief result is a new algorithm for computing labels that handles complex disjunctions such as $choose(\{A, B, C\}, \{D, E, F\})$ which stands for either A, B, and C are true, or D, E, and F are true. This algorithm is a natural extension of de Kleer's original ATMS algorithm and does not require hyper-resolution rules to compute minimal, consistent, sound and complete labels.

The paper is structured as follows. In Section 2 we introduce the mathematics of convex spaces: the finite representability and $\mathcal{MW}$ properties. The version space is formulated in terms of convex spaces in Section 3, and we present an admissibility result for concept languages. In Section 4, we perform a similar analysis of the ATMS algorithm and show that the label computation performed by the disjunction-free ATMS is akin to the boundary set updates of the version space algorithm. We extend the class of disjunctions expressible within the ATMS and derive a new label-update algorithm which is more efficient in general, and does not rely on hyper-resolution rules.

2. Convex Spaces

Our goal is to develop the basic mathematical theory of convex spaces with a view to extract the common order theoretic structure of version spaces and ATMSs. This order-theoretic structure formally captures one's intuitions of "consistent" partially ordered knowledge.

The basic issues that we address can be roughly described as follows: (i) for what sorts of concept languages can one have a compact (finite) representation of the relevant knowledge? and (ii) what operations can one do on these representations while preserving representability? The first issue motivates the singling out of convex spaces. If a subset of a poset is convex we have a hope of representing it by its "fringes". Convexity by itself, however, is far from enough and we need a deeper analysis in order to describe the situations in which finite representability holds. With regard to the

*Gunter's work was supported in part by NSF grant CCR-8912778 and by a Young Investigator Award from the Office of Naval Research. Ngair's work was supported by the Institute of Systems Science, Singapore. Panangaden's work was supported by NSF grant CCR-8818979 and by NSERC. Subramanian's work was supported by NSF grant IRI-8902721.

second issue, we show that for languages that satisfy some simple properties, the lattice-theoretic operations of join and meet preserve the finite representability.

We consider a set P with a partial order $\preceq$ defined on it. In relation to the version space theory, P represents the set of all patterns/concepts, and in relation to the ATMS, P is the set of all possible environments which a problem solver may consider. In both of these cases, the elements of P are sets of individuals and the partial order on P is defined by the set inclusion. It is worth noting, however, that the theory in this section can be applied to any domain as long as there is a partial ordering defined on P.

Given an arbitrary subset of P, one can define the least convex space containing it. This and many other operations which interest us here are examples of *closure operations* on a set. Let us denote the power-set of any set X by $\mathcal{P}(X)$. A function $f : \mathcal{P}(P) \longrightarrow \mathcal{P}(P)$, f is called a *closure operation* if it is monotone and $f(f(C)) = f(C) \supseteq C$ for any $C \in \mathcal{P}(P)$. The elements in $\mathcal{CL}(f) = \{f(C) \mid C \in \mathcal{P}(P)\}$ are called the *closed* sets of f.

The properties of closure operations have been studied by many researchers [?] and some of their results are relevant to our discussion here. In particular, it is worth noting that the *range* of a closure operation (*i.e.* $\mathcal{CL}(f)$) is the same as its set of fixed points. Furthermore, one can recover the closure operation from its set of fixed points. Thus one can often think of a closure operation in these two ways. In particular, there is an operation $c : \mathcal{P}(P) \longrightarrow \mathcal{P}(P)$ defined by $c(C) = \{p \in P \mid \exists p_1, p_2 \in C,\ s.t.\ p_1 \preceq p \preceq p_2\}$ called the *convex closure.* Its fixed points are called *convex spaces*. It can be shown that the set of closed sets of a closure operation forms a complete lattice under set inclusion.

In actual applications of convex spaces, it is usually not practical to represent and perform computation on entire spaces due to their sizes. A solution to this problem is to represent the spaces by their boundary sets and restrict the computations to these sets only. This is the basic data representation insight in the work which we will discuss below. Given any subset $C \in \mathcal{P}(P)$, let us denote the set of minimal elements of C by $\mathcal{MIN}(C)$ and the set of maximal elements of C by $\mathcal{MAX}(C)$. We say that C is *representable by boundary sets* if

$$C = \{p \in P \mid \exists s \in \mathcal{MIN}(C), \exists g \in \mathcal{MAX}(C),\ \ s.t.\ s \preceq p \preceq g\}.$$

Furthermore, if $\mathcal{MIN}(C)$ and $\mathcal{MAX}(C)$ are both finite, then C is said to be *finitely representable.*

The most common operations applied to version spaces and convex spaces are the set union, set intersection, join and meet operations. Our goal is to present a criterion for the pattern language so that the finite representability can be preserved under some of these commonly used operations. For simplicity, we restrict ourselves to posets P which are finitely representable.

Definition: Given $p_1, p_2 \in P$ and $S, G \subseteq P$ both finite, we define:

$$\tilde{\bigvee}(p_1, p_2, G) = \mathcal{MIN}(\{p \in P \mid p_1 \preceq p, p_2 \preceq p, \forall g \in G, p \preceq g\}),$$

$$\tilde{\bigwedge}(p_1, p_2, S) = \mathcal{MAX}(\{p \in P \mid p_1 \succeq p, p_2 \succeq p, \forall s \in S, p \succeq s\}).$$

The poset P is said to have the $\mathcal{MW}$ property if, for all $p_1, p_2 \in P$:

1. $\tilde{\bigvee}(p_1, p_2, \phi)$ and $\tilde{\bigwedge}(p_1, p_2, \phi)$ are finite,
2. $\forall p, p \succeq p_1, p \succeq p_2 \Rightarrow \exists p' \in \tilde{\bigvee}(p_1, p_2, \phi),\ s.t.\ p \succeq p'$,
3. $\forall p, p \preceq p_1, p \preceq p_2 \Rightarrow \exists p' \in \tilde{\bigwedge}(p_1, p_2, \phi),\ s.t.\ p \preceq p'$.

Theorem 1 *A poset P has the $\mathcal{MW}$ property iff the finitely representable convex spaces form a sublattice of the lattice of convex spaces.* □

For a poset P, let $F(P)$ be the set of finitely representable convex subsets of P ordered by *superset* inclusion, that is, $C \preceq C'$ iff $C \subseteq C'$. It is possible to provide an abstract representation of $F(P)$ in terms of pairs of finite sets using some order-theoretic concepts taken from work on operators which are called *powerdomains* in the semantics of programming languages (for example, see [?] and the references there). Suppose U and V are finite subsets of P. We define three binary relations as follows:

- $U \preceq^{\sharp} V$ iff for every $x \in U$ there is a $y \in V$ such that $x \preceq y$,
- $U \preceq^{\flat} V$ iff for every $y \in V$ there is a $x \in U$ such that $x \preceq y$,
- $U \preceq^{\natural} V$ iff $U \preceq^{\sharp} V$ and $U \preceq^{\flat} V$

In a poset P, an *anti-chain* A is a subset of P with the property that, whenever $p, q \in A$ and $p \preceq q$, then $p = q$.

Definition: Let P be a poset. A *fringe* is a pair (S, G) such that

1. S, G are finite anti-chains of P, and
2. $S \preceq^{\natural} G$. □

We define the poset $G(P)$ to be the set of fringes of P under the ordering

$$(S,G) \preceq (S',G') \quad \text{iff} \quad S \preceq^{\sharp} S' \text{ and } G' \preceq^{\flat} G$$

and we have the following:

Theorem 2 *For any poset P, the poset $F(P)$ of finitely representable convex subsets is isomorphic to the poset $G(P)$ of fringes.* □

showing that it is possible to represent the operations on convex spaces via corresponding operations in the space of fringes.

3. Version Spaces

The version space algorithm, which was introduced by Mitchell [?], can be formulated using the ideas of the previous section. Our goal in this section is to characterize the order-theoretic conditions under which the version space representation is legitimate. Since it is not the case that every concept space supports the version space learning technique, it is desirable to provide some simple condition which will certify, for a given concept space, that the algorithm is sound. Such conditions have been proposed in several discussions of version spaces including the original work [?] and a more recent textbook account [?]. However, the admissibility conditions which have been given are *sufficient* conditions which are too weak to support many of the examples of concept spaces for which the version space algorithm is sound (and, indeed, efficient).

For the rest of this note, we write P for the collection of sets of individual data (observations), and UP ($\equiv \bigcup P$) for the set of individual data. The partial ordering on P is defined by set inclusion $\subseteq$. The notion of a *Version Space* is invented to facilitate the learning of a concept (pattern) when we know a set of its positive instances and a set of its negative instances. The notations given above can be interpreted in this context by taking P as the set of concepts, and UP is the set of all possible instances. We define an operation $\mathcal{K}_P$ (the subscript P is omitted when P is obvious from the context) by taking

$$\mathcal{K}(\Gamma, \Delta) = \{p \in P \mid \Gamma \subseteq p \subseteq \overline{\Delta}\}$$

where $\Gamma, \Delta \subseteq \mathit{UP}$ and $\overline{\Delta}$ is the complement of Δ in UP. Here Γ represents the positive instances and Δ the negative instances.

Definition: A subset C of P is called a *version space* if there exist $\Gamma, \Delta \subseteq \mathit{UP}$, such that $C = \mathcal{K}(\Gamma, \Delta)$.

The two subsets P and ϕ are version spaces, which arise respectively when $\Gamma = \Delta = \phi$ and $\Gamma \cap \Delta \neq \phi$. A version space is an instance of a convex space. Moreover, if we define a mapping $d : \mathcal{P}(P) \longrightarrow \mathcal{P}(P)$ by

$$d(C) = \{p \in P \mid \bigcap C \subseteq p \subseteq \bigcup C\}$$

then we have the following:

Theorem 3 *d is a closure operation whose fixed points are exactly the version spaces.*

Since it can be shown that version spaces are convex spaces and the meet operation on version spaces is the same as that on convex spaces, several results that are true for convex spaces are also true for version spaces. In particular, we have the following:

Theorem 4 *A poset P has the $\mathcal{MW}$ property if and only if the meet of every two finitely representable version spaces is finitely representable.* □

In a concepts learning system using the version space representation, the new version space after the addition of some new observations is the same as the intersection (merging) of the current version space with the version space representing the new observations. Thus, the $\mathcal{MW}$ property is a necessary and sufficient condition for ensuring the preservation of finite representability in version space merging.

Note, however, that the $\mathcal{MW}$ property does not imply the version spaces that we are interested in are finitely representable. It is possible that there exist some observations that cannot be represented by finitely representable version spaces and hence the preservation of finite representability in version space merging does not help at all. Therefore, we need to find out another important property about a concept language, that is whether or not every set of finite observations can be captured by a finitely representable version space. We call this the admissibility property. In this subsection, we give the formal definition of admissibility and show two different criteria for ensuring the admissibility of a concept language.

Definition: A given P is said to have the $\mathcal{S}$ property if for all $x \in \mathit{UP}$, $\mathcal{K}(\{x\}, \phi)$ and $\mathcal{K}(\phi, \{x\})$ are finitely representable. Furthermore, P is said to be *admissible*, if given any $\Gamma, \Delta \subseteq \mathit{UP}$, with $\Gamma \cup \Delta$ finite and nonempty, $\mathcal{K}(\Gamma, \Delta)$ is finitely representable.

The following lemma allows us to check the admissibility of a pattern language by verifying that if observations are all positive or negative, the version space is finitely representable.

Lemma 5 *A poset P is admissible if and only if for all non-empty finite $\Gamma, \Delta \subseteq \mathit{UP}$, both $\mathcal{K}(\Gamma, \phi)$ and $\mathcal{K}(\phi, \Delta)$ are finitely representable.*

The next theorem allows one to check for admissibility by checking finite representability in some special cases.

Theorem 6 *Given a P, if P has the $\mathcal{MW}$ and $\mathcal{S}$ properties, then P is admissible.*

Note that every finite P has $\mathcal{MW}$ and $\mathcal{S}$ properties. Hence, every finite P is admissible. Furthermore, every convex space in such a P is finitely representable.

In some problem domains, the notion of concept consistent with an observation may be defined differently from the instance inclusion definition (see [?]). However, with the $\mathcal{S}$ property suitably modified, we can derive the same sufficient conditions as those of Theorem **??** to guarantee that the set of concepts consistent with any non-empty finite observations is finitely representable. This latter property can be viewed as the generalized definition of admissibility.

To appreciate the point of having a condition for verifying admissibility, we consider an example adapted from [?]. Let $I = (0, 1) \times (0, 1)$ be the open unit rectangle in the two-dimensional real plane. A *real interval* is defined to be a set of real numbers having one of the following forms:

$$[l, u] = \{x \mid l \leq x \leq u\}$$
$$(l, u) = \{x \mid l < x < u\}$$
$$[l, u) = \{x \mid l \leq x < u\}$$
$$(l, u] = \{x \mid l < x \leq u\}$$

We define the *rectangular* concept space P as the set of subsets $p \subseteq I$ such that p is the product of a pair of intervals (rectangles) or p is the empty set ϕ. The set of observations is a subset of P of the form: $[x, x] \times [y, y]$ (points). Given a set of positive and negative observations, the learning task is to find the set of concepts in P that are consistent with the observations.

For each positive observation $[x, x] \times [y, y]$, its version space is the convex closure of $\{I\}$ and $\{[x, x] \times [y, y]\}$. For each negative observation $[x, x] \times [y, y]$, its version space is the convex closure of $\{(x, 1) \times (0, 1), (0, x) \times (0, 1), (0, 1) \times (y, 1), (0, 1) \times (0, y)\}$ and $\{\phi\}$. Hence, P has property $\mathcal{S}$.

Given a pair of rectangles $p_1 = [lx, ux] \times [ly, uy]$, $p_2 = [lx', ux'] \times [ly', uy']$, we have:

$$\tilde{\bigvee}(p_1, p_2, \phi) = \{[\min(lx, lx'), \max(ux, ux')] \times \min(ly, ly'), \max(uy, uy')\},$$
$$\tilde{\bigwedge}(p_1, p_2, \phi) = \{[\max(lx, lx'), \min(ux, ux')] \times \max(ly, ly'), \min(uy, uy')\}.$$

Besides being finite, they also satisfy the second and third conditions of the $\mathcal{MW}$ property. Similar results can be derived for different combinations in the types of rectangles. Hence, P has $\mathcal{MW}$ property and it is therefore admissible. Note that the version spaces of P typically include infinite (and even uncountable) chains, so an admissibility condition which precludes such properties in the concept space will fail to cover this example.

4. Assumption-based TMS's

In this section we examine Johan de Kleer's formulation of Truth Maintenance Systems known as *Assumption-based* TMS's [?, ?]. The formulation involves the inclusion, in the nodes of a TMS, *labels* indicating the minimal set of assumptions which would make the datum of the node true. These sets of assumptions must be recalculated as new information is acquired, so they must be represented in a way which makes this recalculation efficient. Sets of assumptions form a finite lattice (under set inclusion), and it is possible to show that sets of assumptions making the datum of a node true form a convex space. Hence such sets of assumptions may be represented using their boundary sets, that is, using the sets of greatest and least elements of the convex space. In fact, the existence of a common lower bound of inconsistent sets of assumptions makes it possible to simplify this representation so that only the greatest elements are required.

Our goal in this section is to look first at the basic ATMS and show how the calculations with label sets can be performed using computations on the boundaries of convex spaces that we have considered in the previous sections. Expressing the basic ATMS in this way is not difficult, but the concept of an *extended* ATMS [?] provides a more interesting challenge. We show how to recalculate a label using a formula involving closure operations. These closure operations can be computed using the meet and join algorithms for convex spaces. The final part of the section discusses how this approach compares to the use of hyper-resolution rules in [?].

Basic ATMS. The basic ATMS [?] consists of a set of nodes representing the problem solving data, a set of Horn clause justifications which describe how some nodes can be derived from other nodes, and a set of assumptions which are positive literals. Assumptions form the foundation to which every datum's support can be traced. The main purpose of the ATMS is to discover for each node, the set of *environments* that will derive the node, where an environment is any set of assumptions. Note that the set of all environments forms a finite boolean lattice and is called the *environment lattice.*

De Kleer classifies the set of nodes into premises, assumptions, assumed and derived nodes. Each of them is represented by the basic ATMS data structure of the form: $<$*datum, label, justifications*$>$, where *label* is a set of environments and *justifications* is a set of Horn clauses of the form $X_1, \ldots, X_n \Rightarrow$ *datum*. Usually, there is a special node in the ATMS denoted by $\perp$ which represents falsity. For the sake of convenience we will adopt the convention of using the same symbol to denote a datum and the node representing it.

Given any node X that is not $\perp$, we want to capture the set of environments that will derive X but not $\perp$, *i.e.* the set of consistent environments that derive X. An important observation by de Kleer is that, given an environment that derives X, every superset of that environment will also derive X. Furthermore, given an inconsistent environment, every superset of that environment is also inconsistent. This implies that the set of consistent environments deriving X is a convex space under the ordering defined by set inclusion.

In view of the convex space theory, we take $\mathcal{UP}$ to be the set of all assumptions and P to be the set of all subsets of $\mathcal{UP}$. However, the partial ordering $\preceq$ on P is defined to be the *reverse* of set inclusion, *i.e.* $\preceq \equiv \supseteq$. This partial order captures the notion of the generality of an environment. Since the set of assumptions is finite, the boolean lattice of environments is also finite and the local $\mathcal{MW}$ property is clearly satisfied. Furthermore, convex spaces can always be represented by their (finite) boundaries. We use the following notation in our discussion: X is a node, $V_\perp$ is the set of inconsistent environments, V_X, where $X \neq \perp$, is the set of consistent environments that derive X.

Observe that each V_X is a convex space, hence it can be represented by finite boundary sets. Note that an environment p is in V_X for $X \neq \perp$ if there does not exist $p' \in \mathcal{MAX}(V_\perp)$ such that $p \preceq p'$ and there exists $p'' \in \mathcal{MAX}(V_X)$ such that $p \preceq p''$. Therefore, to encode the information in V_X, we need only store $\mathcal{MAX}(V_X)$ and $\mathcal{MAX}(V_\perp)$. Since the latter is shared by every X, we can maintain it separately. In an ATMS, the label of X, denoted by L_X, stores $\mathcal{MAX}(V_X)$ and the common $\mathcal{MAX}(V_\perp)$ is stored in the label of the node $\perp$, denoted by $L_\perp$ called the *nogoods*.

Given a convex space C, it is said to be *representable by label* if $C = \{p \in P \mid (\forall p' \in L_\perp,\ p \not\preceq p')$ and $(\exists p'' \in \mathcal{MAX}(C),\ p \preceq p'')\}$. It can be easily verified that given any two convex spaces X and Y that are representable by label, their intersection and union are convex, representable by label and are equivalent to their meet and join respectively. Hence, if S and T are the upper boundary sets of $X \cap Y$ and $X \cup Y$ respectively, then by specializing the algorithms for the meet and join of convex spaces [?], we have the following formulas for computing S and T using only the labels of X and Y:

$$\begin{aligned} S &= \mathcal{MAX}(\{p_X \cup p_Y \mid p_x \in L_X, p_y \in L_Y, \\ &\qquad s.t.\ \neg\exists p' \in L_\perp, p_X \cup p_Y \preceq p'\}) \quad (1) \\ T &= \mathcal{MAX}(L_X \cup L_Y) \quad (2) \end{aligned}$$

where $L_X = \mathcal{MAX}(X)$ and $L_Y = \mathcal{MAX}(Y)$.

One of the most significant operations in a basic ATMS is the addition or retraction of a justification which will trigger a recalculation of the label for each affected node. Let V_i^k be the consistent environments that derive the i_{th} node (in the antecedent) of the k_{th} justification of a node X. To recalculate the label of X, we first compute the intersection of V_i^k for the antecedent nodes of a justification (*i.e.* with k fixed) followed by taking the union of these results over all k: $\bigcup_k \bigcap_i V_i^k = \bigvee_k \bigwedge_i V_i^k$ We will generally need to recompute the above many, but *finite*, times before we arrive at the correct label. Since each application of the intersection and union operation will derive a convex space representable by label, we can make use of the formulas (**??**) and (**??**) to calculate the new label of X.

Extended ATMS. In [?], the basic ATMS is extended with the notion of primitive disjunction of assumptions written as *choose*$(C_1, \ldots, C_n)$, where each $C_i, 1 \leq i \leq n$ is a distinct assumption. The interpretation of the primitive disjunction is that at least one of the C_i must be true in the ATMS. Primitive disjunctions can be used to encode negated assumptions, hence, all propositional expressions can be encoded using Horn clause justifications and primitive disjunctions only. With the introduction of primitive disjunctions, the set $V_\perp$ of inconsistent environments is expanded to include any environment p such that all supersets of p which satisfy every primitive disjunction will derive $\perp$. Similarly, consistent environments deriving a node X, *i.e.* V_X, are now expanded to include any consistent environment p such that all the consistent supersets of p which satisfy every primitive disjunction will derive X.

With the introduction of primitive disjunctions, the label of a node calculated by the basic ATMS algorithm may no longer be correct. To see why, consider the following example from [?]. Suppose the set of justifications is $\{A \Rightarrow a; B \Rightarrow b; C \Rightarrow c; c, a \Rightarrow \perp; c, b \Rightarrow \perp\}$. The label for the proposition $\perp$ is $\{\{A, C\}, \{B, C\}\}$. Adding *choose*$(\{A\}, \{B\})$ causes this label to change to $\{\{C\}\}$ because one of A or B holds in the new ATMS state. The label propagation algorithm described in the previous section fails to make this correction because it handles Horn clause justifications only, and our choose statement is non-Horn. To solve this problem, de Kleer extended his algorithm to correct the labels by using two similar hyper-resolution rules, one for the $\perp$ node and one for the rest of nodes.

In the following, we will reformulate the problem using the convex spaces as we did in the basic ATMS. However, we extend the choose operation to allow encoding of complex disjunctions which can have any set of assumptions as its argument, *i.e.* any DNF formula of assumptions. This allows greater flexibility and efficiency in the encoding of knowledge in an ATMS. First we need to define an important operation needed in the calculation of the label in an extended ATMS.

Let $(P, \preceq)$ be any lattice. A subset $C \subseteq P$ is *downward closed* if $p \in C$ and $p' \preceq p$ implies $p' \in C$. For any subset $S \subseteq P$, the *downward closure* of S is the set $\downarrow S = \{p \in P \mid \exists p' \in S,\ p \preceq p'\}$. Now, given any subset $T \in \mathcal{P}(P)$, the operation Φ_T is defined on downward closed sets $C \in \mathcal{P}(P)$ as: $\Phi_T(C) = \{p \in P \mid \forall t \in T, p \wedge t \in C\}$. An important result relating to the Φ operation is:

Theorem 7 *Given a set of disjunctions $\{choose(t^i_1, \ldots, t^i_{n_i}) \mid 1 \leq i \leq n\}$, the algorithm for the corresponding extended ATMS can be described in three steps: (1) Apply the basic ATMS algorithm on the set of justifications; (2) let $C_X = V_X \cup V_\perp = V_X \vee V_\perp$; (3) let $V_\perp = \Phi_{T_1} \circ \cdots \circ \Phi_{T_n}(V_\perp)$ and $V_X = \Phi_{T_1} \circ \cdots \circ \Phi_{T_n}(C_X) - V_\perp$, where $T_i = \{t^i_1, \ldots, t^i_{n_i}\}$.*

To calculate Φ, we use the following result:

Theorem 8 *If P is an environment lattice, then given any downward closed C with $\mathcal{MAX}(C) = \{s_1, \ldots, s_m\}$, and any $T = \{t_1, \ldots, t_n\}$, then:*

$$\Phi_T(C) = \bigwedge_{1 \leq j \leq n} \bigvee_{1 \leq i \leq m} E^i_j, \tag{3}$$

where $E^i_j = \{p \in P \mid p \wedge t_j \preceq s_i\}$.

It can be shown that every E^i_j is downward closed and its upper boundary set is $s_i - t_j$. Furthermore, the label set of $\Phi_T(C)$ can be calculated using the modified convex space meet and join procedures, *i.e.* formula (**??**) and (**??**). This means that we can calculate for each node X in the extended ATMS, the new label set L_X that takes disjunctions into consideration.

Note that if Σ is the set of all functions with domain $\{1, \ldots, n\}$ and codomain $\{1, \ldots, m\}$, we can rewrite the formula (**??**) as:

$$\Phi_T(C) \quad = \quad \bigvee_{\sigma \in \Sigma} \bigwedge_{j} E^{\sigma(j)}_j \tag{4}$$

In the case where only primitive disjunctions are considered, it can be shown that de Kleer's extended ATMS algorithm in effect calculates the label of a node using the formula (**??**) with the hyper-resolution rules used to prune away some of the terms that will not produce any new environments. However, in our implementation of the extended ATMS,[1] we make use of an optimized algorithm that is based on the formula (**??**).

There are several advantages to using this new algorithm over de Kleer's hyper-resolution approach. The absence of resolution contributes substantially to the performance of our algorithm in comparison to a hyper-resolution based approach. This advantage is similar to the way ATMS's were an improvement over the earlier Truth Maintenance Systems [**?**], because the labeling eliminates the need to re-evaluate some computations multiple times during backtracking. In our case, we eliminate many redundant pattern matching operations. Another advantage is the easier encoding and potential improvement in efficiency because a formula in disjunctive normal form can be asserted as a single *choose* statement.

5. Conclusions

We have formulated a theory of convex spaces of partially ordered sets which includes algorithms for basic operations on finitely representable convex spaces in the presence of a simple assumption on the partial order. Using this theory, we can also describe conditions that could ensure the admissibility of the version space representation of a concept description language.

We then show how convex spaces can be used to describe the label-update algorithm in de Kleer's basic assumption-based truth maintenance systems. This idea suggests a new approach to the label-update algorithm for the extended ATMS. Our approach generalizes the extended ATMS *choose* operations to allow the use of disjunctions such as $choose(\{A, B, C\}, \{D, E, F\})$. This provides additional flexibility in expressing constraints and also contributes to the efficiency of label updating. Our new label-update algorithm does not require the introduction of hyper-resolution rules. Instead, we use an approach which is similar to that employed in the version space algorithm to recalculate labels. This simplifies the description of how labels are updated and makes the extended ATMS label-update algorithm more consistent with the algorithm used for the basic ATMS's.

The convex space treatment of ATMS is more than just a new algorithm for ATMS. In a forthcoming paper [**?**], we show that this leads us to provide a formal semantics for both the basic and extended ATMS. Furthermore, we can prove that if negation is introduced into the ATMS architecture, we can apply the Φ operation to calculate the prime implicants [**?**, **?**] of any set of DNF propositional formulas. This algorithm has been implemented on top of our ATMS implementation.

It is especially our hope that the abstraction of the convex space algorithms which we have discussed will lead to new insights in other areas which do, or could, employ similar structures for the representation of knowledge.

[1] This is currently implemented on top of de Kleer's basic ATMS.

References

[Bir84] Garrett Birkhoff. *Lattice Theory.* American Mathematical Society, 1984.

[dK86a] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, **28**:127–162, 1986.

[dK86b] Johan de Kleer. Extending the ATMS. *Artificial Intelligence*, **28**:163–196, 1986.

[Doy79] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, **12**:231–272, 1979.

[FMH90] Yasushi Fujiwara, Yumiko Mizushima, and Shinichi Honiden. On logical foundations of the ATMS. *Proc. of ECAI-90 Workshop on Truth Maintenance Systems*, 1990.

[GN87] M. R. Geneserth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, 1987.

[GNPS90] Carl A. Gunter, Teow-Hin Ngair, Prakash Panangaden, and Devika Subramanian. *The Common Order-theoretic Structure of Version Spaces and ATMS's.* Technical Report MS-CIS-90-86, University of Pennsylvania, 1990.

[Gun91] Carl A. Gunter. The mixed powerdomian. *Theoretical Computer Science*, 1991. In press.

[Hir90] Haym Hirsh. *Incremental Version Space Merging: A General Framework for Concept Learning.* Kluwer Academic Publishers, 1990.

[KT90] Alex Kean and George Tsiknis. An incremental method for generating prime implicants/implicates. *J. Symbolic Computation*, 185–206, 1990.

[Mit78] Tom Mitchell. *Version Space: An approach to Concept Leaning.* PhD thesis, Stanford University, 1978.

[Mit82] Tom Mitchell. Generalization as search. *Artificial Intelligence*, **18**:203–226, 1982.

[Nga91] Teow-Hin Ngair. ATMS and its extensions. Forthcoming, 1991.

[RdK87] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintence systems: preliminary report. *Proc. of AAAI-87*, 183–188, 1987.