Theoretical Underpinnings of Version Spaces

Haym Hirsh
Computer Science Department
Rutgers University
New Brunswick, NJ 08903

Abstract

Mitchell's version-space approach to inductive concept learning has been highly influential in machine learning, as it formalizes inductive concept learning as a search problem—to identify some concept definition out of a space of possible definitions. This paper lays out some theoretical underpinnings of version spaces. It presents the conditions under which an arbitrary set of concept definitions in a concept description language can be represented by boundary sets, which is a necessary condition for a set of concept definitions to be a version space. Furthermore, although version spaces can be intersected and unioned (version spaces are simply sets, albeit with special structure), the result need not be a version space; this paper also presents the conditions under which such intersection and union of two version spaces yields a version space (i.e., representable by boundary sets). Finally, the paper shows how the resulting boundary sets after intersections and unions can be computed from the initial boundary sets, and proves the algorithms correct.

1 Introduction and Motivation

The problem of inductive concept learning—to form general rules from data—has been well-studied in machine learning and artificial intelligence [Buchanan and Mitchell, 1978; Michalski and Chilausky, 1980; Mitchell *el al,* 1983; Quinlan, 1983]. The problem can be stated as follows:

Given:

- *Training Data:* Positive and negative examples of a concept to be identified.
- *Concept Description Language:* A language in which the final concept definition must be expressed.

Determine:

- The desired unknown concept.

Mitchell [1978; 1982] proposed an approach to this problem that maintains all elements of the concept description language that could be the desired unknown concept, namely, those that correctly classify the given training data (i.e., that are consistent with the data); this set is known as a *version space.* However, this set can be cleverly represented by only maintaining the minimal and maximal elements of the set, known as the *boundary sets* of the version space. As further data are obtained, the set of possibilities is refined until ideally only one consistent result remains. This approach has been highly influential to work in concept learning, as it formalizes inductive concept learning as a search problem—to identify some concept definition out of a space of possible definitions.

This paper presents the theoretical underpinnings of version spaces, beginning with the conditions under which a set of concept definitions can be represented by boundary sets. It also presents the conditions under which the intersection and union of two version spaces is a version space, and how the resulting boundary sets can be computed from the boundary sets of the original version spaces.

These questions are important for a number of reasons. First, they clarify Mitchell's own formalization of version spaces and its central term, admissibility [Mitchell, 1978, page 61]. Second, his version-space learning algorithm, the *candidate-elimination algorithm*, can be expressed as a version-space intersection process [Hirsh, 1990a; Hirsh, 1990b], and thus answering the question of when intersections are legal also answers the question of when the candidate-elimination algorithm can be used. Finally, both version-space intersections and unions have proven useful in extending version spaces beyond strict consistency with data [Hirsh, 1990a; Hirsh, 1990c].

2 Background

The traditional scenario for inductive concept learning begins with a set of training data—examples classified by an unknown target concept—and a language in which the desired concept must be expressed (which defines the space of possible generalizations concept learning will search). Mitchell defines a version space to be "the set of all concept descriptions within the given language which are consistent with those training instances" [Mitchell, 1978, page 17]. Mitchell noted that the generality of concepts imposes a partial order that allows more efficient representation of the version space by the boundary sets *S* and *G* representing the most specific and most general concept definitions in the space. The *S*- and G-sets delimit the set of all concept definitions consistent with the given data—the version space contains all concepts as or more general than some element in *S* and as or more specific

than some element in G.

Given a new instance, some of the concept definitions in the version space for past data may no longer be consistent with the new instance. The candidate-elimination algorithm manipulates the boundary-set representation of a version space to create boundary sets that represent a new version space consistent with all the previous instances plus the new one. For a positive example the algorithm generalizes the elements of the 5-set as little as possible so that they cover the new instance yet remain consistent with past data, and removes those elements of the G-set that do not cover the new instance. For a negative instance the algorithm specializes elements of the G-set so that they no longer cover the new instance yet remain consistent with past data, and removes from the 5-set those elements that mistakenly cover the new, negative instance. The unknown concept is determined when the version space has only one element, which in the boundary set representation is when the S- and G-sets have the same single element.

3 Terminology and Notation

Throughout this paper *"CDL"* is used to refer to the (potentially infinite) set of concept definitions describable in the concept description language and considered in the concept learning task. There is a space of possible objects (instances), and each concept definition divides a set of objects into those objects it covers and those it does not. The subset of the entire space of possible objects that a concept definition covers is known as its *extension.* Concept definitions are partially ordered by generality, "$\preceq$." "$C_1 \preceq C_2$" should be read as "Ci is less general than or equal to *Ci"* which means that the extension of *C*\ is a subset of the extension of *Ci.* "$C_1 \prec C_2$," "$C_1 \succeq C_2$," and "$C_1 \succ C_2$" have similar, obvious meanings.

Theorem 1 (Mitchell, 1978) *The relation $\preceq$ is a partial ordering.*

Proof: The reflexivity, asymmetry, and transitivity fall out of the definition of $\preceq$ using subset. Since the subset relation is a partial ordering, so, too, is $\preceq$ a partial ordering. □

The concept learning problem is to identify one concept definition out of the set of potential concept definitions in the *CDL,* given information from some outside source about the nature of the unknown concept. Usually this information is of the form of positive and negative examples of this taiget concept, that is, classified training data of the concept.

4 Criteria for Representability by Boundary Sets

This section presents the first major results of this paper, namely, criteria that, if satisfied by a subset of a concept description language, make that subset representable by boundary sets. First, however, it is necessary to define two terms, convexity and definiteness.[1] This section will show that a subset of a concept description language is representable by boundary sets if and only if it is convex and definite.

[1] Earlier versions of this work [Hirsh, 1990a] used the terms "closure" and "boundedness". Convexity and definiteness better reflect existing terminology in lattice theory.

4.1 Convexity

A set *C* of concept definitions in the *CDL* is said to be *convex* if, given any two elements of C, any element of the concept description language between them (in the partial order) is also in G. This basically says that there are no "holes" in the set. More formally:

Definition 1 *A subset $C \subseteq CDL$ is said to be convex if and only if for all $C_1, C_2 \in C$, $C \in CDL$, $c_1 \preceq c_3 \preceq c_2$ implies $c_3 \in C$.*

Examples:

1. If the *CDL* is the set of all closed ranges over a single real-valued attribute *x* of the form "{x | *a* < *x* < *b}"* where *a* and *b* are reals (in less formal terms, this corresponds to the case where data are described using a single real-valued feature, and concept descriptions can only constrain the possible values for this feature to various closed intervals):
   (a) If *C* = *CDL,* then *C* is convex.
   (b) If *C* = *{c* ∈ *CDL* | a,b are integers}, then *C* is not convex, since between any two concept definitions there will be another whose range delimiters *a* and *b* are reals but not integers.
   (c) If $C = \{c \in CDL \mid a \geq 0, b \leq 1000\}$, then *C* is convex.
2. Given any *CDL,* if *C* is the set of all concept definitions consistent with a set of classified instances, *C* is convex.

4.2 Definiteness

lb define definiteness, it is necessary to state what the minimal and maximal elements of a partially ordered set are:

Definition 2 *The set of minimal elements of C is written* Min(C), *and is defined by*

$$\mathbf{Min}(C) = \{c \in C \mid \neg\exists c' \in C \text{ such that } c' \prec c\}.$$

Similarly, the set of maximal elements of C, Max(C), *is*

$$\mathbf{Max}(C) = \{c \in C \mid \neg\exists c' \in C \text{ such that } c' \succ c\}.$$

C is said to be *definite* if all elements of *C* are greater than or equal to some element of Min(C) and less than or equal to some element of Max(C). More formally:

Definition 3 *C is definite if and only if for all $c \in C$ there exists some $s \in$* Min(C) *and some $g \in$* Max(C) *such that $s \preceq c \preceq g$.*

Min(C) is often referred to as C's 5-set, and Max(C) is Cs G-set, and *S* and *G* should be viewed as shorthand for Min(C) and Max(C). Note that this definition of definiteness docs not constrain either 5 or G to be finite.

Examples:

1. If the *CDL* is the set of closed ranges of the form $\{x \mid a \leq x \leq b\}$ where a and *b* are reals:
   (a) If *C* = *CDL,* then *C* is not definite, since *G* = {} (for every concept definition in the language there is another concept definition more general than it), and thus for every element c ∈ *C* there is no $g \in G$ with $c \preceq g$.

(b) If $C = \{c \in CDL \mid a, b \text{ are integers}\}$, then $G = \{\}$, so C is not definite.

(c) If $C = \{c \in CDL \mid a \geq 0, b \leq 1000\}$, then $S = \{\{0\}, \ldots, \{1000\}\}$, $G = \{\{x \mid 0 \leq x \leq 1000\}\}$, and C is definite.

2. If the *CDL* is the set of all conjunctive expressions over a finite set of boolean features, then all subsets $C \subseteq CDL$ are definite.

This second example is a simple result of the following theorem:

Theorem 2 *If the CDL is finite (i.e., there are only a finite number of concept definitions expressible) then all subsets C of the CDL are definite.*

Proof: By induction on the size of C. D

The concept of definiteness is closely related to Mitchell's (1978) notion of *admissibility?* Admissibility is intended as a property of concept description languages—whether *all* subsets of the language can be represented by boundary sets. The concept of definiteness applies to *subsets* of the concept description language. It is possible to have languages with subsets that are not definite, yet also with some subsets that *are* definite. The important thing to guarantee is that only definite subsets will be used during the particular learning task at hand. Admissibility is a stronger restriction.

4.3 Representability by Boundary Sets

It is now possible to identify which subsets of a *CDL* can be represented by only retaining their minimal and maximal elements (i.e., boundary sets). However, it is first necessary to state precisely what it means to be representable by boundary sets:

Definition 4 *If* $C \subseteq CDL_t$ S = Min(C), *and* G = Max(C), C *is said to be representable by boundary sets if and only if*

$$C = \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\}.$$

This definition states that *C* is representable by boundary sets if the set of elements between the minimal and maximal elements of *C* gives back C.

It can now be shown that any convex, definite subset $C \subseteq CDL$ can be represented by boundary sets:

Theorem 3 *All convex, definite subsets C of a CDL can be represented by boundary sets.*

Proof: Showing that a set *C* can be represented by boundary sets is done by first demonstrating that

$$C \subseteq \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\},$$

(where 5 = Min(C) and *G* = Max(C)) then demonstrating that

$$C \supseteq \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\}.$$

First the $\subseteq$: If $c' \in C$ then there exists an *s* in *S* and *g* in *G* such that $s \preceq c' \preceq g$ (since *C* is definite). Furthermore, since $c' \in C$ and $C \subseteq CDL$, $c' \in CDL$. Therefore

$$c' \in \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\},$$

so

$$C \subseteq \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\}.$$

For the $\supseteq$:

$$c' \in \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\}$$

means that there is an s in S for which $s \preceq c'$ and a g in G where $c' \preceq g$. Since $S, G \subseteq C$, $s, g \in C$. Since C is convex, this means that $c' \in C$, and thus that

$$C \supseteq \{c \in CDL \mid \exists s \in S, g \in G \text{ such that } s \preceq c \preceq g\}.$$

□

The converse can also be shown to be true, namely, that all subsets $C \subseteq CDL$ that can be represented by boundary sets are convex and definite. This demonstrates that convexity and definiteness are both necessary and sufficient conditions on when an arbitrary subset of a *CDL* can be represented by boundary sets.

Theorem 4 ***All subsets C of a CDL that can be represented by boundary sets are convex and definite.***

Proof: The definiteness portion of the proof is trivial. For all $c \in C$ there exists $s \in S$ = Min(C) and $g \in G$ = Max(C) such that $s \preceq c \preceq g$ (since C can be represented by boundary sets). This is for any $c \in C$. Thus C is definite. For convexity: Given $c_1, c_2 \in C$, it is necessary to show for all $c_3 \in CDL$ when $c_1 \preceq c_3 \preceq c_2$, $c_3 \in C$. Since $c_1 \in C$ and C can be represented by boundary sets, there exists $s \in S$ such that $s \preceq c_1$. Similarly there exists $g \in G$ such that $c_2 \preceq g$. Since $c_1 \preceq c_3 \preceq c_2$, $s \preceq c_3 \preceq g$. But by the definition of "representable by boundary sets" this means $c_3 \in C$. □

4.4 Generalizing Version Spaces

Mitchell defined a version space to be the set of all concept descriptions in a prespecified language that are consistent with data. The important quality that such sets have is that they can be represented by their boundary sets. This section has shown that representability by boundary sets is equivalent to convexity plus definiteness. This section can thus can be viewed as generalizing Mitchell's consistency-based version spaces to arbitrary subsets of a concept description language representable by boundary sets, or equivalently, to convex and definite subsets of the concept description language. The results of the remainder of this paper apply to both Mitchell's original notion of version space as well as this more general form. The term "version space" will be used throughout, but can be replaced with "convex definite subsets of the *CDL*"?

5 Version-Space Intersections

This section presents conditions under which the intersection of two version spaces is a version space, an important question since Mitchell's candidate elimination algorithm can be expressed as a version-space intersection process. This section furthermore provides a method for doing this intersection

[2] Mitchell defines a concept description language to be admissible if and only if every chain has a maximum and minimum element

[3] The results presented here apply more generally to *convex spaces;* Gunter *et al.* [1991] have explored this generalization of the work presented here and apply it to ATMSs, where convex spaces also arise.

in boundary-set representation; such version-space intersection forms the basis for an alternative version-space learning algorithm [Hirsh, 1990a; Hirsh, 1990b].

It is first useful to note that the intersection of two version spaces is always convex:

Theorem 5 *Given two version spaces VS_1 and VS_2, their intersection $VS_1 \cap VS_2$ is convex.*

Proof: $VS_1 \cap VS_2$ is convex if for all $c_1 \in VS_1 \cap VS_2$, $c_2 \in VS_1 \cap VS_2$, and $c_3 \in CDL$, $c_1 \preceq c_3 \preceq c_2$ implies $c_3 \in VS_1 \cap VS_2$. Since both c_1 and c_2 are in $VS_1 \cap VS_2$, they are both in VS_1 and VS_2. Since both VS_1 and VS_2 are convex (since they are version spaces), $c_3 \in VS_1$ and $c_3 \in VS_2$. But this means $c_3 \in VS_1 \cap VS_2$. □

A simple corollary of this is

Corollary 1 *The intersection of two version spaces is a version space if and only if the intersection is definite.*

Note that the intersection of two version spaces VS_1 and VS_2 is not necessarily definite. To see this consider the concept description language in which concept definitions are closed ranges of reals over a single variable x of the form "$\{x \mid a \leq x \leq b\}$", but with the term "$\{x \mid 0 \leq x \leq 10\}$" excluded. If VS_1 contains all concept definitions as or more general than "$\{x \mid 0 \leq x \leq 5\}$" and as or less general than "$\{x \mid -1000 \leq x \leq 1000\}$" and VS_2 contains all concept definitions as or more general than "$\{x \mid 5 \leq x \leq 10\}$" and as or less general than "$\{x \mid -1000 \leq x \leq 1000\}$," then their intersection is still a subset of the concept description language but is no longer definite. The problem is that $\text{Min}(VS_1 \cap VS_2)$ is empty. The desired minimal element for the intersection would ideally have been "$\{x \mid 0 \leq x \leq 10\}$," but this is not in the language.

When the intersection of two version spaces is indeed definite, rather than explicitly intersecting the two version spaces by enumerating each of their elements, it is possible to compute the boundary sets for the intersection directly from the boundary sets of the two original version spaces. This is done by finding the minimal common generalizations of pairs of S-set elements for the new S-set and the maximal common specializations of pairs of G-set elements for the new G-set, and removing the overly general elements from the new S-set and overly specific elements from the new G-set. To state this formally it is necessary to present what the most specific common generalizations (MSG) and most general common specializations (MGS) of two concept definitions are:

$$\text{MSG}(A,B) = \{C \mid A \preceq C \text{ and } B \preceq C \text{ and there is no } D \text{ such that } D \prec C \text{ with } A \preceq D \text{ and } B \preceq D\}$$

$$\text{MGS}(A,B) = \{C \mid C \preceq A \text{ and } C \preceq B \text{ and there is no } D \text{ such that } C \prec D \text{ with } D \preceq A \text{ and } D \preceq B\}.$$

It is now possible to state more formally how the boundary sets for the intersection of two version spaces, when it is itself a version space, can be computed:

Theorem 6 (Version-Space Merging Algorithm) *If $VS_1 \cap VS_2$ is definite, then $\text{Min}(VS_1 \cap VS_2) = S_{1\cap2}$ and $\text{Max}(VS_1 \cap VS_2) = G_{1\cap2}$, where $S_{1\cap2}$ and $G_{1\cap2}$ are determined as follows:*

$$S_{1\cap2} = \text{Min}(\{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1 \textit{ and } s_2 \in S_2, \textit{ and } \exists g_1 \in G_1, g_2 \in G_2 \textit{ with } s \preceq g_1 \textit{ and } s \preceq g_2\})$$

and

$$G_{1\cap2} = \text{Max}(\{g \in \text{MGS}(g_1, g_2) \mid g_1 \in G_1 \textit{ and } g_2 \in G_2, \textit{ and } \exists s_1 \in S_1, s_2 \in S_2 \textit{ with } s_1 \preceq g \textit{ and } s_2 \preceq g\}).$$

Proof: This is done for the S-set case (the proof for the G-set case is analogous) by first showing that $S_{1\cap2} \subseteq \text{Min}(VS_1 \cap VS_2)$, then showing $S_{1\cap2} \supseteq \text{Min}(VS_1 \cap VS_2)$. Both cases use the following lemma:

Lemma 1 *If c is in the set $\{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$, then c is in $VS_1 \cap VS_2$.*

Proof: $c \in \{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$ means $c \in \text{MSG}(s_1, s_2)$ for some $s_1 \in S_1$ and $s_2 \in S_2$. This means $s_1 \preceq c$ and $s_2 \preceq c$ (by the definition of MSG), which shows that c has some element from each of the two original S-sets below it. But furthermore $c \in \{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$ means $\exists g_1 \in G_1, g_2 \in G_2$ with $c \preceq g_1$ and $c \preceq g_2$, which shows that c has some element from each of the two original G-sets below it. Therefore $c \in VS_1$ and $c \in VS_2$, and so $c \in VS_1 \cap VS_2$. □

First the $\subseteq$: If $c \in S_{1\cap2}$, then $c \in \{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$, so $c \in VS_1 \cap VS_2$ by Lemma 1. To show that $c \in \text{Min}(VS_1 \cap VS_2)$ it is necessary to show that there is no $c' \prec c$ with $c' \in VS_1 \cap VS_2$. However, to show that there is no such $c' \prec c$, it is sufficient to show that there is no $s \in \text{Min}(VS_1 \cap VS_2)$ such that $s \prec c$ (since the definiteness of $VS_1 \cap VS_2$ means that if there were such a $c' \prec c$, there would be such an s). This will be shown by contradiction.

Suppose there is an $s \in \text{Min}(VS_1 \cap VS_2)$ such that $s \prec c$. $s \in \text{Min}(VS_1 \cap VS_2)$ means $s \in VS_1 \cap VS_2$, so $s \in VS_1$ and $s \in VS_2$. This in turn means there is some $s_1 \in S_1$ and some $s_2 \in S_2$ with $s_1 \preceq s$ and $s_2 \preceq s$. Finally, there is no $s' \prec s$ with $s_1 \preceq s'$ and $s_2 \preceq s'$ (otherwise s' would be in $VS_1 \cap VS_2$, and so s would not be in $\text{Min}(VS_1 \cap VS_2)$). Thus $s \in \text{MSG}(s_1, s_2)$, and since s is in VS_1 and in VS_2 there is some $g_1 \in G_1$ and some $g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2$. Therefore $s \in \{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$, which contradicts the fact that c is a minimal element of this set ($c \in S_{1\cap2}$).

For the $\supseteq$: If $c \in \text{Min}(VS_1 \cap VS_2)$, then $c \in VS_1 \cap VS_2$. This means that $c \in VS_1$ and $c \in VS_2$, and furthermore there is some $s_1 \in S_1$ and some $s_2 \in S_2$ with $s_1 \preceq c$ and $s_2 \preceq c$. Finally, there is no $c' \prec c$ with $s_1 \preceq c'$ and $s_2 \preceq c'$ (otherwise c' would be in $VS_1 \cap VS_2$, and so c would not be in $\text{Min}(VS_1 \cap VS_2)$). Thus $c \in \text{MSG}(s_1, s_2)$, and since c is in VS_1 and in VS_2 there is some $g_1 \in G_1$ and some $g_2 \in G_2$ with $c \preceq g_1$ and $c \preceq g_2$. Therefore $c \in \{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$.

To show that c is a minimal element of this set, and thus in $S_{1\cap2}$, it is necessary to show that there is no $c' \in \{s \in \text{MSG}(s_1, s_2) \mid s_1 \in S_1$ and $s_2 \in S_2$, and $\exists g_1 \in G_1, g_2 \in G_2$ with $s \preceq g_1$ and $s \preceq g_2\}$ with $c' \prec c$. This is true because if there were such a c', it would be in $VS_1 \cap VS_2$ (by Lemma 1), which would mean that c is not in $\text{Min}(VS_1 \cap VS_2)$, which would be a contradiction. □

Note that Theorem 6 presents a method for computing intersections in theory, but to be practical the computations must take a finite amount of time. In particular, computing MSGi(s_1, s_2) must take a finite amount of time. If there are infinite chains in the concept description language, the procedure for computing MSG might never halt if it traverses the partial order. Thus in real applications MSG will often be computed from the syntactic form of 8_1 and 8_2, rather than traversing the partial order. For example, to determine MSGi$(\{x \mid a_1 \leq x \leq b_1\}, \{x \mid a_2 \leq x \leq b_2\})$ a procedure for computing MSG directly from the partial order would never halt, since there are an infinite number of concept definitions more general than each of the concept definitions. However, it is a simple matter to compute their most specific common generalization from their syntactic form: $\{x \mid \min(a_1, a_2) \leq x \leq \max(b_1, b_2)\}$. Typically MSG is defined for specific concept description languages to manipulate the syntactic structures so it need not face infinite-chain difficulties.

6 Version-Space Unions

This section discusses unions of version spaces, and presents conditions under which the union of two version spaces is a version space, as well as a method for doing unions in boundary-set representation. Such version-space union has proven useful for dealing with certain forms of noisy data [Hirsh, 1990a; Hirsh, 1990c].

It is first useful to note that the union of two version spaces is always definite:

Theorem 7 *Given two version spaces VS_1 and VS_2, their union $VS_1 \cup VS_2$ is definite.*

Proof: For any $c \in VS_1 \cup VS_2$, either $c \in VS_1$ or $c \in VS_2$. Without loss of generality, assume $c \in VS_1$. To show that $VS_1 \cup VS_2$ is definite, it is necessary to show that there is some $s \in \text{Min}(VS_1 \cup VS_2)$ with $s \preceq c$. The proof that there is some $g \in \text{Max}(VS_1 \cup VS_2)$ with $c \preceq g$ is analogous.

Since $c \in VS_1$, there is some $s \in S_1$ with $s \preceq c$. Either $s \in \text{Min}(VS_1 \cup VS_2)$, in which case there is some $s \in \text{Min}(VS_1 \cup VS_2)$ with $s \preceq c$, or else $s \notin \text{Min}(VS_1 \cup VS_2)$. If this latter case is true, there must be some $c' \in VS_1 \cup VS_2$ with $c' \prec s$. Since s is in S_1, c' must be in VS_2. But VS_2 is a version space, and hence definite, so there is some $s' \in S_2$ with $s' \preceq c'$. Since $c' \prec s$, this means $s' \prec s$. But $s \in S_1$, so there is no $c'' \in VS_1$ with $c'' \prec s'$. Also, there is no $c'' \in VS_2$ with $c'' \prec s'$ (since $s' \in S_2$). This means $s' \in \text{Min}(VS_1 \cup VS_2)$. Thus for any $c \in VS_1 \cup VS_2$ there is some $s \in \text{Min}(VS_1 \cup VS_2)$ with $s \preceq c$. □

A simple corollary of this is

Corollary 2 *The union of two version spaces is a version space if and only if the union is convex.*

Note that just as the intersection of two version spaces need not be a version space, similarly the union of two version spaces need not be a version space. To see this consider any concept description language with three elements $c_1 \prec c_2 \prec c_3$. Let $VS_1 = \{c_1\}$ and $VS_2 = \{c_3\}$. The union of the two version spaces is {c\, C3}, but this doesn't include C2, which it must if it is to be convex and a version space.

Just as for intersections, the minimal and maximal elements for the union of two version spaces can be determined directly from the boundary sets for the two version spaces being unioned. These form the boundary sets for the union when the union is convex and thus a version space.

Theorem 8 (Version-Space Union Algorithm) $\text{Min}(VS_1 \cup VS_2) = S_{1\cup2}$ *and* $\text{Max}(VS_1 \cup VS_2) = G_{1\cup2}$, *where* $S_{1\cup2} = \text{Min}(S_1 \cup S_2)$ *and* $G_{1\cup2} = \text{Max}(G_1 \cup G_2)$.

Proof: This is done for $S_{1\cup2}$ by first showing that $S_{1\cup2} \subseteq \text{Min}(VS_1 \cup VS_2)$, then showing that $S_{1\cup2} \supseteq \text{Min}(VS_1 \cup VS_2)$. The proof for $G_{1\cup2}$ is analogous.

For the $\subseteq$: If $c \in S_{1\cup2}$, then $c \in \text{Min}(S_1 \cup S_2)$. This means c is in $S_1 \cup S_2$, which in turn means $c \in S_1$ or $c \in S_2$, and thus $c \in VS_1$ or $c \in VS_2$. Hence $c \in VS_1 \cup VS_2$. To show that c is a minimal element of this set it is necessary to show that there is no $c' \in VS_1 \cup VS_2$ with $c' \prec c$. Assume there is such a c'. Then it is either in VS_1 or VS_2. Without loss of generality assume it is in VS_1. Then there must be some $s_1 \in S_1$ with $s \preceq c'$ (since VS_1 is a version space). This s_1 is in S_1, so it is in $S_1 \cup S_2$. But since $s_1 \preceq c' \prec c$, c cannot be in $\text{Min}(S_1 \cup S_2)$. This is a contradiction. Therefore there is no $c' \in VS_1 \cup VS_2$ with $c' \prec c$, and thus $c \in \text{Min}(VS_1 \cup VS_2)$

For the $\supseteq$: If $c \in \text{Min}(VS_1 \cup VS_2)$, then $c \in VS_1 \cup VS_2$ and there is no $c' \in VS_1 \cup VS_2$ with $c' \prec c$. This means $c \in VS_1$ or $c \in VS_2$. Assume $c \in VS_1$. c must be in S_1, since c is in $\text{Min}(VS_1 \cup VS_2)$, which means there is no $c' \in VS_1$ with $c' \prec c$. The case for $c \in VS_2$ is analogous. Thus $c \in S_1 \cup S_2$. Furthermore, there is no $c'' \in S_1 \cup S_2$ with $c'' \prec c$ (otherwise c would not be in $\text{Min}(VS_1 \cup VS_2)$, since $c'' \in S_1 \cup S_2$ means $c'' \in VS_1 \cup VS_2$). Thus $c \in S_{1\cup2}$. □

7 Concluding Remarks

This paper has presented formal results on the range of applicability of version spaces. It analyzed when a set of concept definitions is a version space, as well as when the intersection and union of version spaces yields a version space. It also gave methods to do such intersection and union in boundary-set representation, and proved that the algorithms to do so are correct.

An interesting open problem is how a version-space-like approach can be used even for arbitrary (nonconvex or nondefinite) sets of concept definitions. One approach would be to represent arbitrary sets as nested differences of version spaces. The intersection and union procedures could utilize facts such as $(A - B) \cap (C - D) = (A \cap C) - (B \cup D)$, as well as simplifications such as $A - B = A$ if $A \cap B = \emptyset$, and $(A - \emptyset) = A$. Ultimately operations will bottom out at intersections and unions of version spaces.

A final issue to consider is the size of boundary sets. The simplest question that should be answered is under what conditions can a version space be represented by *finite* boundary sets. Clearly one sufficient condition for finite boundary-set size is if the version space is itself finite. Gunter *et al.* [1991] generalize the work presented here to give comparable conditions on when a version space is representable by finite boundary sets. Equally important is answering the question of how large the finite boundary-set size is. Haussler [1988] has shown that boundary sets can grow exponentially in the number of instances processed. However, Smith and Rosenbloom [Smith and Rosenbloom, 1990] show that, for tree-structured languages under certain conditions data can be processed to guarantee that boundary sets stay singleton. Understanding under what conditions exponential growth happens and how

to avoid it is important in determining the practical utility of version spaces. These are questions for future work.

Acknowledgments

This paper presents an updated version of Chapter 8 of the author's doctoral dissertation [Hirsh, 1990a], where more lengthy acknowledgments can be found. Johanna Seibt helped with initial formalizations. Dave Smith provided feedback on a very early draft, and William Cohen on a more recent one. Discussions with Bruce Buchanan, Carl Gunter, Jeff Kahn, Bill Landi, Tbm Marlowe, Tom Mitchell, Teow-Hin Ngair, Paul Rosenbloom, and Devika Subramanian were invaluable.

References

[Buchanan and Mitchell, 1978] B. G. Buchanan and T. M. Mitchell. Model-directed learning of production rules. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems,* pages 297-312. Academic Press, New York, 1978.

[Gunterera/., 1991] C. A. Gunter, T.-H. Ngair, P. Panangaden, and D. Subramanian. The common order-theoretic structure of version spaces and ATMS's (extended abstract). In *Proceedings of the National Conference on Artificial Intelligence,* Anaheim, CA, July 1991.

[Haussler, 1988] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence,* 26(2): 177-221, Sept 1988.

[Hirsh, 1990a] H. Hirsh. *Incremental Version-Space Merging: A General Framework for Concept Learning.* Kluwer, Boston, MA, 1990.

[Hirsh, 1990b] H. Hirsh. Incremental version-space merging. In *Proceedings of the Seventh International Conference on Machine Learning,* Austin, Texas, June 1990.

[Hirsh, 1990c] H. Hirsh. Learning from data with bounded inconsistency. In *Proceedings of the Seventh International Conference on Machine Learning,* Austin, Texas, June 1990.

[Michalski and Chilausky, 1980] R. S. Michalski and R. L. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *Policy Analysis and Information Systems,* 4(3):219-244, September 1980.

[Mitchell *et aL,* 1983] T. M. Mitchell, P. E. Utgoff, and R. B. Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristics. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach,* pages 163-190. Morgan Kaufmann, Los Altos, CA, 1983.

[Mitchell, 1978] T. M. Mitchell. *Version Spaces: An Approach to Concept Learning.* PhD thesis, Stanford University, December 1978.

[Mitchell, 1982] T. M. Mitchell. Generalization as search. *Artificial Intelligence,* 18(2):203-226, March 1982.

[Quinlan, 1983] J. R. Quinlan. Learning efficient classification procedures and their application to chess end-games. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach,* pages 463-482. Morgan Kaufmann, Los Altos, CA, 1983.

[Smith and Rosenbloom, 1990] B. D. Smith and P. S. Rosenbloom. Incremental non-backtracking focusing: A polynomial^ bounded generalization algorithm for version spaces. In *Proceedings of the National Conference on Artificial Intelligence,* pages 848-853, Boston, MA, August 1990.