Generalising Boundary Sets

E.N. Smirnov    H.J. van den Herik    I.G. Sprinkhuizen-Kuyper

IKAT/Infonomics,
Department of Computer Science,
Maastricht University,
P.O.Box 616, 6200 MD Maastricht,
The Netherlands

e-mail: {smirnov, herik, kuyper}@cs.unimaas.nl

Abstract

This report proposes adaptable boundary sets as a new version-space representation. It is shown that adaptable boundary sets can be adjusted to a version-space representation between boundary sets [5, 6, 7] and instance-based boundary sets [11, 12, 13]. This allows the choice of proper version-space representations to be realised dynamically during the learning phase.

1 Introduction

Version spaces are an approach to concept learning [5, 6, 7]. They are defined as sets of descriptions in concept languages that correctly classify training instances. When concept languages are partially ordered, version spaces can be represented by boundary sets. Boundary sets are sets of minimal and maximal descriptions in version spaces. It was proven that they correctly represent version spaces [5, 11].

An analysis of boundary sets shows that their size can grow exponentially in the number of training instances [1]. To overcome this problem alternative version-space representations were introduced in [2, 3, 4, 8, 9, 10, 11, 12, 13, 14]. They extend the scope of concept languages for which version spaces are efficiently applicable.

One of the main difficulties with the alternative version-space representations is that in some cases they are less computationally efficient than boundary sets although their size is still polynomial in the number of training instances. This makes the choice of version-space representations a serious implementation problem. To solve this problem we propose a new version-space representation called adaptable boundary sets. Depending on the memory requirements the representation can be adjusted to a version-space representation between boundary sets and instance-based boundary sets [11, 12, 13]. Thus, the choice of proper version-space representations can be realised dynamically during the learning phase.

The report is organised as follows. Section 2 provides a formalisation of concept learning. The new version-space representation is introduced in section 3. Its learning, merging and other useful algorithms are given in sections 4, 5, and 6, respectively. Section 7 presents an example of the representation application. Finally, in section 8 conclusions are given.

2 Formalisation

Formalisation starts by determining the elements of concept learning. Let \( I \) be a set of descriptions of all possible entities in some domain. A concept \( C \) is defined as a subset of \( I \). Concepts are represented in a concept language \( L_c \). The language \( L_c \) is defined as a set of descriptions \( c \) s.t. (1) every concept is represented by at most one description \( c \), and (2) every description \( c \) represents exactly one concept.
The elements of concepts are called instances. They are related to concept descriptions by a cover relation $M$. The cover relation $M(c, i)$ holds for $c \in Lc$ and $i \in I$ if and only if the instance $i$ is a member of the concept represented by $c$. A description $c \in Lc$ is said to cover an instance $i \in I$ if and only if the cover relation $M(c, i)$ holds.

As a rule any target concept $C$ is incompletely defined by sets $I^+ \subseteq I$ and $I^- \subseteq I$ of positive and negative training instances s.t. $I^+ \subseteq C$ and $I^- \cap C = \emptyset$. Hence, the concept-learning task in this case is to find descriptions of $C$ in $Lc$.

To find the descriptions of a target concept, we specify them by the consistency criterion: a concept description $c$ is consistent if and only if $c$ correctly classifies training instances. The set of all the consistent descriptions in a concept language is called the version space [5].

**Definition 1 (Version Space)** Given a set $I$ of all possible entities, a concept language $Lc$, a cover relation $M$, and training sets $I^+$ and $I^-$, the version space $VS(I^+, I^-)$ is defined as follows:

$$VS(I^+, I^-) = \{c \in Lc | (\forall i \in I^+)M(c, i) \land (\forall i \in I^-)\neg M(c, i)\}.$$ 

In order to learn version spaces, they have to be represented. The key to find a good version-space representation is to observe that concept languages can be ordered. The order can be based on a relation “more specific” [5, 6, 7].

**Definition 2 (Relation “more specific” ($\leq$))** Consider a set $I$ of all possible entities, a concept language $Lc$, and a cover relation $M$. Then:

$$(\forall c_1, c_2 \in Lc)((c_1 \leq c_2) \leftrightarrow (\forall i \in I)(M(c_1, i) \rightarrow M(c_2, i))).$$

When the relation “$\leq$” is defined on a concept language $Lc$, a description $c_1 \in Lc$ is said to be more specific than a description $c_2 \in Lc$ if and only if $c_1 \leq c_2$. By duality, a description $c_1 \in Lc$ is said to be more general than a description $c_2 \in Lc$ if and only if $c_2 \leq c_1$.

The relation “$\leq$” is a partial order [5]. Hence, if “$\leq$” is defined on a concept language $Lc$, then $Lc$ is partially-ordered. One class of partially ordered languages was extensively used for defining version-space representations. This is the class of admissible concept languages. It is introduced using the notion of bounded sets. Bounded sets are given after we state what minimal and maximal sets are.

**Definition 3 (Minimal Sets)** If $C$ is a partially-ordered set, then:

$$MIN(C) = \{c \in C | (\forall c' \in C)((c' \leq c) \rightarrow (c' = c))\}.$$ 

**Definition 4 (Maximal Sets)** If $C$ is a partially-ordered set, then:

$$MAX(C) = \{c \in C | (\forall c' \in C)((c \leq c') \rightarrow (c = c'))\}.$$ 

The minimal and maximal sets of version spaces are known as minimal and maximal boundary sets [5, 6, 7]. To refer to them we introduce the following notations.

**Notation 5** $MIN(VS(I^+, I^-))$ is denoted by $S(I^+, I^-)$.

**Notation 6** $MAX(VS(I^+, I^-))$ is denoted by $G(I^+, I^-)$.

\footnote{For the sake of brevity the sets $I^+$ and $I^-$ are called training sets.}
Definition 7 (Bounded Sets) A partially ordered set $C$ is bounded if and only if:

$$\forall c \in C)((\exists s \in \text{MIN}(C))(s \leq c) \land (\exists g \in \text{MAX}(C))(c \leq g)).$$

Definition 8 (Admissible Concept Languages) A partially-ordered concept language $L_c$ is admissible if and only if every non-empty subset $C \subseteq L_c$ is bounded.

Most admissible concept languages, used in practice, exhibit one of the two dual properties below.

Property 1 Each non-empty set $C \subseteq L_c$ has greatest lower bound $\text{glb}(C)$ s.t.:

$$\forall i \in I)((\forall c \in C)M(c, i) \leftrightarrow M(\text{glb}(C), i)).$$

Property 2 Each non-empty set $C \subseteq L_c$ has least upper bound $\text{lub}(C)$ s.t.:

$$\forall i \in I)((\exists c \in C)M(c, i) \leftrightarrow M(\text{lub}(C), i)).$$

An admissible concept language $L_c$ exhibits property 1 when for each nonempty subset $C \subseteq L_c$ there exists the greatest lower bound $\text{glb}(C)$ such that an instance $i \in I$ is covered by all the concept descriptions $c \in C$ if and only if $i$ is covered by $\text{glb}(C)$. It was proven in [11] that when property 1 holds the size of the set $S(I^+, I^-)$ is equal to one for all $I^+ \subseteq I$ and $I^- \subseteq I$.

By duality we can explain when the language exhibits property 2.

3 Adaptable Boundary Sets

Adaptable boundary sets are a new version-space representation with the important property that its size can be adjusted. The formal definition of the representation is based on the notion of a special covering of a set, given below.

Definition 9 (Special Covering) $P$ is a special covering of a set $S$ if and only if:

1. $P = \{\emptyset\}$ if $S = \emptyset$, and
2. $P$ is a covering of $S$ if $S \neq \emptyset$.

Notation 10 The set of all special coverings of a set $S$ is denoted as $SP(S)$.

Definition 11 (Adaptable Boundary Sets (ABSs)) If $L_c$ is an admissible concept language, then adaptable boundary sets of a version space $VS(I^+, I^-)$ are an ordered pair of indexed families of sets $\{S(I^+_p, I^-_p)\}_{I^+_p \in P^+}, \{G(I^+_p, I^-_p)\}_{I^-_p \in P^-}$, where $P^+ \in SP(I^+)$ and $P^- \in SP(I^-)$.

To prove that ABSs are a correct version-space representation for the class of admissible concept languages, we consider two rules. The first one states that a version space $VS(I^+_1, I^-_1)$ is a subset of a version space $VS(I^+_2, I^-_2)$ if and only if every description in $VS(I^+_1, I^-_1)$ is consistent with the sets $I^+_2$ and $I^-_2$. The rule is proven in theorem 12.

Theorem 12 $VS(I^+_1, I^-_1) \subseteq VS(I^+_2, I^-_2)$ if and only if

$$\forall c \in VS(I^+_1, I^-_1)((\forall p \in I^+_2)M(c, p) \land (\forall n \in I^-_2)\neg M(c, n)).$$
Proof.

\[ VS(I_1^+, I_1^-) \subseteq VS(I_2^+, I_2^-) \iff (\forall c \in VS(I_1^+, I_1^-))(c \in VS(I_2^+, I_2^-)) \quad (definition \ 1) \]

(\forall c \in VS(I_1^+, I_1^-))(\forall p \in I_2^-)M(c, p) \land (\forall n \in I_2^-)\neg M(c, n) \quad \Box

The second rule has two dual parts. More formally:

- if \( c \in L_c \) is more general than an element of the set \( S(I_1^+, I_1^-) \) for each training set \( I_p^+ \in Q^+ \), then \( c \) is consistent with each set \( I_p^+ \), where \( Q^+ \) is a nonempty set and \( Q^+ \subseteq P^+ \);

- if \( c \in L_c \) is more specific than an element of the set \( G(I^+, I^-) \) for each training set \( I_p^- \in Q^- \), then \( c \) is consistent with each set \( I_p^- \), where \( Q^- \) is a nonempty set and \( Q^- \subseteq P^- \).

The first and second parts of the rule are proven in lemma 13 and 14, respectively.

**Lemma 13** Consider a nonempty set \( Q^+ \subseteq P^+ \). If \( L_c \) is an admissible concept language, then:

\[ (\forall c \in L_c)((\forall I_p^+ \in Q^+)(\exists s \in S(I_p^+, I^-))(s \leq c) \rightarrow (\forall I_p^+ \in Q^+)(\forall i \in I_p^+))M(c, i).\]

**Proof.** Consider an arbitrarily chosen description \( c \in L_c \) and set \( I_p^+ \in Q^+ \). Take an element \( s \in S(I_p^+, I^-) \) s.t. \( s \leq c \). Since \( L_c \) is admissible, according to definition 2 \( 2 \leq c \) and \( (\forall i \in I_p^+))M(s, i) \) imply \( (\forall i \in I_p^+))M(c, i) \). The last derivation holds for all \( I_p^+ \in Q^+ \). Thus, \( (\forall I_p^+ \in Q^+))(\forall i \in I_p^+))M(c, i). \quad \Box \)

**Lemma 14** Consider a nonempty set \( Q^- \subseteq P^- \). If \( L_c \) is an admissible concept language, then:

\[ (\forall c \in L_c)((\forall I_p^- \in Q^-)(\exists g \in G(I^+, I^-))(c \leq g) \rightarrow (\forall I_p^- \in Q^-)(\forall i \in I_p^-))\neg M(c, i).\]

**Proof.** The proof is dual to that of lemma 13. \( \Box \)

**Theorem 15 (Correctness of Adaptable Boundary Sets)** Consider a version space \( VS(I^+, I^-) \) represented by \( \text{ABSs} : \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \). If the concept language \( L_c \) is admissible, then:

\[ (\forall c \in L_c)((c \in VS(I^+, I^-)) \leftrightarrow ((\forall I_p^+ \in P^+)(\exists s \in S(I_p^+, I^-))(s \leq c) \land (\forall I_p^- \in P^-)(\exists g \in G(I^+, I_p^-))(c \leq g))).\]

**Proof.** \( (\rightarrow) \) Consider an arbitrarily chosen description \( c \in VS(I^+, I^-) \). Since \( (\forall I_p^+ \in P^+)(\forall I_p^- \in I^-) \) and \( (\forall I_p^- \in P^-)(\forall I_p^- \in I^-) \), by theorem 12:

\[ (\forall I_p^+ \in P^+)(VS(I_p^+, I^-) \subseteq VS(I^+, I^-)) \]
\[ (\forall I_p^- \in P^-)(VS(I^+, I_p^-) \subseteq VS(I^+, I^-)).\]

Thus,

\[ (\forall I_p^+ \in P^+)(c \in VS(I_p^+, I^-)) \]
\[ (\forall I_p^- \in P^-)(c \in VS(I^+, I_p^-)).\]
Since $L_c$ is admissible, the version spaces $VS(I^+_p, I^-)$ and $VS(I^+, I^-_p)$ are bounded for all $I^+_p \in P^+$, $I^-_p \in P^-$, $I^+ \subseteq I$ and $I^- \subseteq I$. Thus, according to definition 7 using notations 5 and 6 we have that:

$$(\forall I^+_p \in P^+)(\exists s \in S(I^+_p, I^-))(s \leq c)$$

$$(\forall I^-_p \in P^-)(\exists g \in G(I^+, I^-_p))(c \leq g).$$

$$(\leftarrow)$$ Consider an arbitrarily chosen description $c \in L_c$ s.t.:

$$(\forall I^+_p \in P^+)(\exists s \in S(I^+_p, I^-))(s \leq c)$$

$$(\forall I^-_p \in P^-)(\exists g \in G(I^+, I^-_p))(c \leq g).$$

By lemma 13 formula (1) implies $(\forall I^+_p \in P^+)((\forall i \in I^+_p)M(c, i))$ and by lemma 14 formula (2) implies $(\forall I^-_p \in P^-)((\forall i \in I^-_p)\neg M(c, i))$. Thus, $(\forall i \in I^+)M(c, i)$ and $(\forall i \in I^-)\neg M(c, i)$. The last two derivations and $c \in L_c$ imply according to definition 1 that $c \in VS(I^+, I^-)$.

Theorem 15 states that ABSs correctly represent each version space $VS(I^+, I^-)$ for the class of admissible concept languages. The descriptions in $VS(I^+, I^-)$ are exactly those that are (1) more general than an element of each set $S(I^+_p, I^-)$, and (2) more specific than an element of each set $G(I^+, I^-_p)$. Thus, all the elements of $VS(I^+, I^-)$ are considered and are not explicitly enumerated.

Given training sets $I^+$ and $I^-$, all the special coverings $P^+ \in SP(I^+)$ and $P^- \in SP(I^-)$ determine a set $R(I^+, I^-)$ of all possible ABSs of $VS(I^+, I^-)$. Two elements in $R(I^+, I^-)$ are well-known in machine learning. They are as follows:

1. if $P^+ = \{I^+\}$ and $P^- = \{I^-\}$, then ABSs are defined as $\langle\{S(I^+, I^-)\}, \{G(I^+, I^-)\}\rangle$. Thus, according to $[5, 6, 7]$ they are equal to boundary sets.

2. if $(\forall I^+_p \in P^+)(\exists i \in I^+)(I^+_p = \{i\})$ and $(\forall I^-_p \in P^-)(\exists i \in I^-)(I^-_p = \{i\})$, ABSs are defined as $\langle\{S((p), I^-)\}_{p \in P^+}, \{G(I^+, (n))\}_{n \in P^-}\rangle$. Thus, according to $[11, 12, 13]$ they are equal to instance-based boundary sets.

Boundary sets and instance-based boundary sets are boundaries of the set $R(I^+, I^-)$ if it is ordered. Hence, all the adaptable boundary sets in the set $R(I^+, I^-)$ can be considered between these two version-space representations.

### 4 Learning Algorithm

The learning algorithm updates the ABSs of a version space, when a new training instance is given. It is proposed for the class of admissible concept languages and consists of two parts.

The first part of the algorithm computes the new special covering of one of the training sets, given a new training instance $i \in I$. More precisely, if the instance $i$ is positive, the algorithm adds $i$ to the training set $I^+$. The special covering $P'^+$ of the updated set $I'^+ = I^+ \cup \{i\}$ is formed from the special covering $P^+$ of the set $I^+$ s.t. the instance $i$ is added to some elements (training subsets) of $P^+$ and/or the set $\{i\}$ is added to $P^+$. If the instance $i$ is negative, the formation of the updated set $I'^-$ and its special covering $P'^-$ is realised analogously.

Formally, the special coverings $P'^+$ and $P'^-$ are called extensions of the special coverings $P^+$ and $P^-$. The notion of extension is given below.

**Definition 16 (Extension of a Special Covering)** Let $P$ and $P'$ be special coverings of sets $S$ and $S \cup \{i\}$. $P'$ is an extension of $P$ if and only if:

$$((\forall I'_p \in P')(\exists I_p \in P)(I'_p = I_p) \lor (\exists I_p \in P)(I'_p = I_p \cup \{i\})) \lor (I'_p = \{i\}).$$
Note that the first part of the algorithm is not completely specified. In practice it is determined s.t. the size of the ABSs, based on the new covering, is minimised.

The second part of the learning algorithm computes the ABSs of the version space that is consistent with the processed training data and the new training instance $i$. More precisely, if the instance $i$ is positive, the algorithm computes the ABSs: $\langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$ of the version space $VS(I^+, I^-)$, and a version space $VS(I^+, I^-)$ given by $\langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$, where $I^+ = I^+ \cup \{ i \}$, $P^{++} \in SP(I^+)$, and $P^{++}$ is an extension of $P^+$. If $L_c$ is an admissible concept language, then:

$$G(I^{++}, I^-) = \{ g \in G(I^+, I^-) \} \mid M(g, i) \} \text{ for } I^- \in P^-$$
$$S(I^+_p, I^-) = S(I^+_p, I^-) \text{ for } I^+_p \in P^+ \text{ if } (\exists I^- \in P^+)(I^+_p = I^-)$$
$$S(I^+_p, I^-) = \text{MIN}\{ \{ c \in VS(I^+_p, I^-) \mid M(c, i) \} \} \text{ for } I^+_p \in P^+ \text{ if } (\exists I^- \in P^+)(I^+_p = I^- \cup \{ i \})$$
$$S(i, I^-) = \text{MIN}(VS(\{ i \}, I^-)) \text{ if } i \in P^{++}.$$ 

**Proof.** The first and third parts of the theorem follow from the theorem for revising boundary sets given a positive instance [5]. The second and fourth parts of the theorem follow from definition 11. □

Theorem 18 Consider a version space $VS(I^+, I^-)$ given by $\langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$, and a version space $VS(I^+, I^-)$ given by $\langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$, where $I^- = I^- \cup \{ i \}$, $P^- \in SP(I^-)$, and $P^-$ is an extension of $P^-$. If $L_c$ is an admissible concept language, then:

$$S(I^+_p, I^-) = \{ s \in S(I^+_p, I^-) \mid \neg M(s, i) \} \text{ for } I^- \in P^-$$
$$G(I^+, I^-) = G(I^+, I^-) \text{ for } I^- \in P^- \text{ if } (\exists I^- \in P^-)(I^+_p = I^-)$$
$$G(I^+, I^-) = \text{MAX}\{ \{ c \in VS(I^+, I^-) \mid M(c, i) \} \} \text{ for } I^- \in P^- \text{ if } (\exists I^- \in P^-)(I^+_p = I^- \cup \{ i \})$$
$$G(I^+, \{ i \}) = \text{MAX}(VS(I^+, \{ i \})) \text{ if } i \in P^-.$$ 

**Proof.** The proof is dual to that of theorem 17. □

We describe the second part of the learning algorithm. Given a new positive instance $i \in I$, the ABSs $\langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$ of a version space $VS(I^+, I^-)$, and a special covering $P^{++}$ of the training set $I^+ = I^+ \cup \{ i \}$, the algorithm executes four steps. In the first step for each $I^- \in P^-$ the algorithm forms the set $G(I^+, I^-)$. The set is formed from those elements of the set $G(I^+, I^-)$ that cover the instance $i$. In the second step for each $I^+_p \in P^+$ for which there exists a set $I^- \in P^-$ s.t. $I^+_p = I^- \text{ the algorithm takes the set } S(I^+_p, I^-) \text{ from the ABSs } \langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$. In the third step for each $I^+_p \in P^+$ for which there exists a set $I^- \in P^-$ s.t. $I^+_p = I^- \cup \{ i \}$, the algorithm forms the set $S(I^+_p, I^-)$. The set is formed from the minimal elements of the version space $VS(I^+, I^-)$ that cover the instance $i$. In the fourth step the algorithm generates the set $S(\{ i \}, I^-)$, if the set $\{ i \}$ is in $P^+$. After execution of all the four steps the ABSs $\langle \{ S(I^+_p, I^-) \} \rangle_{I^+_p \in P^+}, \{ G(I^+, I^-) \} \rangle_{I^- \in P^-}$ of the version space $VS(I^+, I^-)$ is returned.

The algorithm’s second-part performance for a negative instance is dual to that for a positive instance; hence, it is analogous in form.

\(^2\text{Note that } I^+ = I^+ \cup \{ i \} \text{ and } P^{++} \text{ is a special covering of } I^+\).
5 Merging Algorithms

The merging algorithms change ABSs of a version space by merging some elements (training subsets) in the special coverings they are based on. In this section we consider several merging algorithms that differ in their concept-language applicability.

5.1 General Merging Algorithm

The general merging algorithm is proposed for the class of admissible concept languages. It consists of two parts. To describe the first part of the algorithm we introduce the notion of merged regroupment.

Definition 19 (Merged Regroupment) Let P and P' be special coverings of a set S. P' is a merged regroupment of P if and only if:

\[(\forall I'_p \in P')(\exists Q \subseteq P)(I'_p = \bigcup_{I_p \in Q} I_p).\]

Given the ABSs \(\langle \{S(I^+_p, I^-)\}_{I^+_p \in P^+}, \{G(I^+_p, I^-)\}_{I^-_p \in P^-}\rangle\) of a version space \(VS(I^+, I^-)\), the first part of the algorithm computes regroupments \(P^{+'}\) and \(I'_{P^-}\) of the special coverings \(P^+\) and \(P^-\), respectively. Note that this computation is not specified. In practice it is determined s.t. the size of the merged ABSs, based on the new special coverings \(P^{+'}\) and \(I'_{P^-}\), is minimised.

The algorithm’s second part computes the merged ABSs \(\langle \{S(I^+_p, I^-)\}_{P^{+'}}, \{G(I^+_p, I^-)\}_{I'_{P^-}}\rangle\) of the version space \(VS(I^+, I^-)\), given the ABSs \(\langle \{S(I^+_p, I^-)\}_{P^+}, \{G(I^+_p, I^-)\}_{I^-_p \in P^-}\rangle\) the same version space, and the special coverings \(P^{+'}\) and \(I'_{P^-}\). This part of the algorithm is based on theorem 22 given below. The proof of the theorem uses two dual lemmas 20 and 21:

Lemma 20 Consider a nonempty set \(Q^+ \subseteq P^+\). If \(Lc\) is an admissible concept language, then:

\[(\forall c' \in VS(\bigcup_{I^+_p \in Q^+} I^+_p, I^-))(c' \in \{c \in Lc | (\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\}).\]

Proof. Consider an arbitrary description \(c' \in VS(\bigcup_{I^+_p \in Q^+} I^+_p, I^-)\). Since the set \(Q^+\) is a nonempty and \(Q^+ \subseteq P^+\), the set \(Q^+\) is a special covering of the set \(\bigcup_{I^+_p \in Q^+} I^+_p\). Thus, the admissibility of \(Lc\) implies by theorem 15 that:

\[(\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c').\]

Since \(c' \in VS(\bigcup_{I^+_p \in Q^+} I^+_p, I^-)\), according to definition 1 we have that \(c' \in Lc\). Thus,

\[c' \in \{c \in Lc | (\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\}. \Box\]

Lemma 21 Consider a nonempty set \(Q^- \subseteq P^-\). If \(Lc\) is an admissible concept language, then:

\[(\forall c' \in VS(I^+, \bigcup_{I^-_p \in Q^-} I^-_p))(c' \in \{c \in Lc | (\forall I^-_p \in Q^-)(\exists g \in G(I^+, I^-_p))(c \leq g)\}).\]

Proof. The proof of the lemma is dual to that of lemma 20. \Box
Theorem 22 Consider a version space $VS(I^+, I^-)$ given by $\{(S(I^+_p, I^-))_{I^+_p \in P^+}, \{G(I^+, I^-))_{I^-_p \in P^-}\}$ and $\{(S(I^+_p, I^-))_{I^+_p \in P^+}, \{G(I^+, I^-))_{I^-_p \in P^-}\}$ such that $P^+$ is a merged regroupment of $P^+$ and $P^-$ is a merged regroupment of $P^-$. If $Lc$ is an admissible concept language, then:

$$S(I^+_p, I^-) = \{c \in MG(\{S(I^+_p, I^-))_{I^+_p \in Q^+}\})(\forall n \in I^-)\neg M(c,n)\} \text{ for } I^+_p \in P^+$$

$$G(I^+, I^-') = \{c \in MS(\{G(I^+, I^-))_{I^-_p \in Q^-}\})(\forall p \in I^+)M(c,p)\} \text{ for } I^-_p \in P^-$$

where:

$$Q^+ \in SP(I^+_p) \text{ and } Q^- \subseteq P^+$$
$$Q^- \in SP(I^-_p) \text{ and } Q^- \subseteq P^-$$
$$MG(\{S(I^+_p, I^-))_{I^+_p \in Q^+}\) = MIN(\{c \in Lc|(\forall I^+_p \in Q^+)\exists s \in S(I^+_p, I^+)) (s \leq c)\})$$
$$MS(\{G(I^+, I^-))_{I^-_p \in Q^-}\) = MAX(\{c \in Lc|(\forall I^-_p \in Q^-)\exists g \in G(I^+, I^-)) (c \leq g)\}).$$

Proof. (I) The proof of the first part of the theorem. It consists of two sub-parts.

$\subseteq$: Consider an arbitrarily chosen description $s' \in S(I^+_p, I^-)$. Since $Q^+ \in SP(I^+_p)$ and $Q^- \subseteq P^+$, $I^+_p = \cup_{I^+_p \in Q^+} I^+_p$ s.t. $(\forall I^+_p \in Q^+)(I^+_p \in P^+)$. Thus, the admissibility of $Lc$ implies by lemma 20 that:

$$s' \in \{c \in Lc|(\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\}.$$ 

Since $Lc$ is admissible, each set in $Lc$ is bounded. Thus, there exists a description $c' \in MIN(\{c \in Lc|(\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\})$ such that $c' \leq s'$.

By lemma 13 $(\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c')$ implies:

$$(\forall I^+_p \in Q^+)(\forall p \in I^+)M(c', p).$$

Since $I^+_p = \cup_{I^+_p \in Q^+} I^+_p$, this is equivalent to:

$$(\forall p \in I^+)M(c', p). \quad (3)$$

According to definition 2 $c' \leq s'$ and $(\forall n \in I^-)\neg M(s', n)$ implies:

$$(\forall n \in I^-)\neg M(c', n). \quad (4)$$

From formulas (3) and (4) and the fact that $c' \in Lc$ we conclude using definition 1 that $c' \in VS(I^+_p, I^-)$. According to definition 3 $c' \leq s'$ and $c' \in VS(I^+_p, I^-)$ imply that $c' = s'$. Thus, $s' \in MIN(\{c \in Lc|(\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\})$. This is equivalent to:

$$s' \in MG(\{S(I^+_p, I^-))_{I^+_p \in Q^+}\).$$

Thus, since $(\forall n \in I^-)\neg M(s', n)$, the $\subseteq$ sub-part of the first part of the theorem is proven.

$\supseteq$: Consider an arbitrarily chosen description $c' \in Lc$ s.t.:

$$c' \in \{c \in MG(\{S(I^+_p, I^-))_{I^+_p \in Q^+}\})(\forall n \in I^-)\neg M(c, n)\}.$$ 

This implies:
By lemma 13 formula (5) implies \((\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c')\) (5)

\((\forall n \in I^-)M(c', n)\) (6)

\(I^+_p = \cup_{I^+_p \in Q^+} I^+_p\), the last derivation is equivalent to:

\((\forall p \in I^+_p)M(c', p)\). (7)

Since \(c' \in L\), formulas (6) and (7) imply according to definition 1 that \(c' \in VS(I^+_p, I^-)\). Thus, according to theorem 15 there exists an element \(s' \in S(I^+_p, I^-)\) s.t. \(s' \leq c'\). By lemma 20 \(s' \in \{c \in L|((\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\} \) But, \(c' \in MIN\{\{c \in L|((\forall I^+_p \in Q^+)(\exists s \in S(I^+_p, I^-))(s \leq c)\}\). Thus, according to definition 3 \(s' \leq c'\) implies that \(s' = c'\). This means that \(c' \in S(I^+_p, I^-)\) and the \(\geq\) sub-part of the first part of the theorem is proven.

(II) The proof of the second part of the theorem is dual to that of the first part and uses lemma 21. \(\square\)

We describe the second part of the general merging algorithm. Given the ABSs \(\{(S(I^+_p, I^-)\}_{I^+_p \in P^+}, (G(I^+, I^-))_{I^- \in P^-}\) of a version space \(VS(I^+, I^-)\), and the special coverings \(P^{+'}\) and \(P^{-'}\), the algorithm executes two steps. In the first step for each set \(I^{+'}_p \in P^{+'}\) the algorithm forms the set \(S(I^{+'}_p, I^-)\). If \(I^{+'}_p \in P^+\), the set \(S(I^{+'}_p, I^-)\) is taken from the ABSs \(\{(S(I^{+'}_p, I^-)\}_{I^+_p \in P^+}, (G(I^+, I^-))_{I^- \in P^-}\}\) (3).

If \(I^{+'}_p \notin P^+\), the algorithm first computes the set \(MG(S(I^{+'}_p, I^-))_{I^+_p \in Q^+}\) using the nonempty set \(Q^+ \subseteq P^+\) s.t. \(I^{+'}_p = \cup_{I^+_p \in Q^+} I^+_p\). The set \(MG(S(I^{+'}_p, I^-))_{I^+_p \in Q^+}\) is computed as a set of minimal generalisations s.t. each generalisation is more general than at least one element of each set \(S(I^+_p, I^-)\) for all \(I^+_p \in Q^+\). Then the algorithm forms the set \(S(I^{+'}_p, I^-)\) from those elements of the set \(MG(S(I^{+'}_p, I^-))_{I^+_p \in Q^+}\) that do not cover any negative instance \(n \in I^-\). In the second step for each set \(I^{-'}_p \in P^{-'}\), the algorithm forms the set \(G(I^+, I^{-'}_p)\). If \(I^{-'}_p \in P^-\), the set \(G(I^+, I^{-'}_p)\) is taken from the ABSs \(\{(S(I^{+'}_p, I^-)\}_{I^+_p \in P^+}, (G(I^+, I^-))_{I^- \in P^-}\}\). If \(I^{-'}_p \notin P^-\), the algorithm first computes the set \(MS([G(I^+, I^{-'}_p)\}_{I^- \in Q^-}\) using the nonempty set \(Q^- \subseteq P^-\) s.t. \(I^{-'}_p = \cup_{I^- \in Q^-} I^+_p\). The set \(MS([G(I^+, I^{-'}_p)\}_{I^- \in Q^-}\) is computed as a set of minimal specialisations s.t. each specialisation is more specific than at least one element of each set \(G(I^+, I^-)\) for all \(I^- \in Q^-\). Then the algorithm forms the set \(G(I^+, I^{-'}_p)\) from those elements of the set \(MS([G(I^+, I^{-'}_p)\}_{I^- \in Q^-}\) that do cover all the positive instances \(p \in I^+\). After the execution of both steps, the merged ABSs \(\{(S(I^{+'}_p, I^-)\}_{I^+_p \in P^+}, (G(I^+, I^{-'}_p)\}_{I^- \in P^-}\) of the version space \(VS(I^+, I^-)\) is returned.

5.2 Specialised Merging Algorithms

In this subsection we propose two specialised merging algorithms applicable for the class of admissible concept languages when properties 1 and 2 hold. We show that by investigating the properties, the parts of the algorithms that determine coverings of the merged ABSs can be completely specified. Moreover, we show that computing the merged ABSs is simplified.

5.2.1 Specialised Merging Algorithm: the Case of Property 1

The specialised merging algorithm, that exploits property 1, is based on theorem 24. Given a nonempty set \(Q^- \subseteq P^-\) s.t. \((\forall I^- \in Q^-)G(I^+, I^-) = 1\), the theorem determines how to compute the set \(G(I^+, \cup_{I^- \in Q^-} I^-)\). The proof of the theorem uses lemma 23 given below.

\(3\)Note that this sub-step does not follow theorem 22 for efficiency reasons.
Lemma 23 Consider a set $\cup_{I^-_p \in Q^-} I^-_p$ and a nonempty set $U \subseteq Lc$ s.t. $(\forall i \in I^+) (\forall c \in U) M(c, i)$, and $(\forall I^-_p \in Q^-) (\exists c \in U) (\forall i \in I^-_p) \neg M(c, i)$ where $Q^- \subseteq P^-$. If $Lc$ is an admissible concept language and property 2 holds, then: $\text{glb}(U) \in Lc$, $(\forall i \in I^+) (\forall c \in U) M(c, i)$, and $(\forall i \in I^-_p) \neg M(\text{glb}(U), i)$.

Proof. Since $Lc$ is an admissible concept language s.t. property 1 holds and the set $U \subseteq Lc$ is nonempty, we have that:

\begin{align*}
\text{glb}(U) & \in Lc, \\
(\forall i \in I) (\forall c \in U) M(c, i) & \iff M(\text{glb}(U), i).
\end{align*}

(8)

(9)

Formula (9) and $(\forall i \in I^+) (\forall c \in U) M(c, i)$ imply:

$$
(\forall i \in I^+) M(\text{glb}(U), i).
$$

(10)

From formula $(\forall I^-_p \in Q^-) (\exists c \in U) (\forall i \in I^-_p) \neg M(c, i)$ we have that:

$$
(\forall i \in \cup_{I^-_p \in Q^-} I^-_p) (\exists c \in U) \neg M(c, i).
$$

(11)

Thus, formulas (9) and (11) imply:

$$
(\forall i \in \cup_{I^-_p \in Q^-} I^-_p) \neg M(\text{glb}(U), i).
$$

(12)

From formulas (8), (10), and (12) the lemma is proven. □

Theorem 24 Consider a nonempty set $Q^- \subseteq P^-$ s.t. $(\forall I^-_p \in Q^-) |G(I^+, I^-_p)| = 1$. If $Lc$ is an admissible concept language and property 1 holds, then:

$$
G(I^+, \cup_{I^-_p \in Q^-} I^-_p) = \{ \text{glb}(\cup_{I^-_p \in Q^-} G(I^+, I^-_p)) \}.
$$

Proof. First, we prove that $\text{glb}(\cup_{I^-_p \in Q^-} G(I^+, I^-_p)) \in VS(I^+, \cup_{I^-_p \in Q^-} I^-_p)$. Second, we prove that $\text{glb}(\cup_{I^-_p \in Q^-} G(I^+, I^-_p)) \in G(I^+, \cup_{I^-_p \in Q^-} I^-_p)$. Third, we prove that the set $G(I^+, \cup_{I^-_p \in Q^-} I^-_p)$ has just one element $\text{glb}(\cup_{I^-_p \in Q^-} G(I^+, I^-_p))$.

(I) Consider a set $U = \cup_{I^-_p \in Q^-} G(I^+, I^-_p)$. Since $Q^-$ is not empty, $(\forall I^-_p \in Q^-) |G(I^+, I^-_p)| = 1$ implies that the set $U \neq \emptyset$. But, according to definition 11 for all $I^-_p \in Q^-$: $G(I^+, I^-_p) \subseteq VS(I^+, I^-_p)$. Thus, by the definition of version spaces $VS(I^+, I^-_p)$ the set $U$ has two additional properties:

\begin{align*}
(\forall i \in I^+) (\forall c \in U) M(c, i) & \quad (13) \\
(\forall I^-_p \in Q^-) (\exists c \in U) (\forall i \in I^-_p) \neg M(c, i) & \quad (14)
\end{align*}

Since $Lc$ is an admissible concept language and property 1 holds, $U \neq \emptyset$, formulas (13) and (14) imply by lemma 23 that: $\text{glb}(U) \in Lc$, $(\forall i \in I^+) M(\text{glb}(U), i)$ and $(\forall i \in \cup_{I^-_p \in Q^-} I^-_p) \neg M(\text{glb}(U), i)$. Thus, according to definition 1 $\text{glb}(U) \in VS(I^+, \cup_{I^-_p \in Q^-} I^-_p)$.

(II) Consider an arbitrary description $c \in VS(I^+, \cup_{I^-_p \in Q^-} I^-_p)$ s.t. $\text{glb}(U) \leq c$. By theorem 15 $(\forall I^-_p \in Q^-) (\exists g \in G(I^+, \cup_{I^-_p \in Q^-} I^-_p))(c \leq g)$ \footnote{Note that theorem 15 is applicable since $Q^-$ is a special covering of the set $\cup_{I^-_p \in Q^-} I^-_p$.}. Since $(\forall I^-_p \in Q^-) |G(I^+, I^-_p)| = 1$, $c$ is a lower bound of the set $U$. Thus, by the definition of the greatest lower bound $c \leq \text{glb}(U)$. Since the relation “$\leq$” is a
The specialised merging algorithm, that exploits property 2, is based on theorem 26. The theorem is consisting of the greatest lower bound of the set \( I_p \). The proof is dual to that of lemma 23.

Consider an arbitrary description \( g \in G(I^+, \cup i \in Q^- I_p^-) \). By theorem 15 \((\forall i \in Q^-)(\exists g' \in G(I^+, I_p^-))(g \leq g')\). Since \((\forall i \in Q^-)(G(I^+, I_p^-)) = 1\), \( g \) is a lower bound of the set \( U \). This implies that \( g \leq \text{glb}(U) \). Since \( g \) is arbitrarily chosen, \( g \in G(I^+, \cup i \in Q^- I_p^-) \) and \( g \leq \text{glb}(U) \) imply that \( g = \text{glb}(U) \); i.e., \( \text{glb}(U) \) is the only element of the set \( G(I^+, \cup i \in Q^- I_p^-) \).

We describe the specialised merging algorithm under the assumption that the set \( S(I^+, I^-) \) of the version space \( VS(I^+, I^-) \) to be learned consists of only one element. Note that the assumption is plausible since the size of the set is equal to one if property 1 holds. Thus, the input ABSs is \langle S(I^+, I^-), \{G(I^+, I^-)\}_{i \in p} \rangle \) of the version space \( VS(I^+, I^-) \).

The first part of the algorithm determines the new regroupment \( P' \) of the special covering \( P \). This is done in two steps. In the first step the algorithm determines all the sets \( G(I^+, I_p^-) \) which size is equal to one. All the subsets \( I_p^- \) are put in a set \( Q^- \). In the second step the algorithm forms the set \( I_p^- = \cup i \in Q^- I_p^- \). The new regroupment \( P' \) is formed from the special covering \( P \) by excluding all the subsets \( I_p^- \) and adding the set \( I_p^- \).

The second part of the algorithm computes the set \( G(I^+, I_p^-) \). By theorem 24 the set is computed consisting of the greatest lower bound of the set \( \cup i \in Q^- G(I^+, I_p^-) \). After this computation the merged ABSs \langle S(I^+, I^-), \{G(I^+, I_p^-)\}_{i \in p} \rangle \) of the version space \( VS(I^+, I^-) \) is returned.

### 5.2.2 Specialised Merging Algorithm: the Case of Property 2

The specialised merging algorithm, that exploits property 2, is based on theorem 26. The theorem is dual to theorem 24 and can be analogously explained. The proof of the theorem uses lemma 25.

**Lemma 25** Consider a set \( \cup i \in Q^+ I_p^+ \) and a nonempty set \( U \subseteq L_c \) s.t. \((\forall i \in Q^+)(\exists c \in U)(\forall i \in I_p^+)M(c, i) \) and \((\forall i \in I^-)(\forall c \in U)\neg M(c, i)\), where \( Q^+ \subseteq P^+ \). If \( L_c \) is an admissible concept language and property 2 holds, then: \( \text{lub}(U) \in L_c \), \( \forall i \in \cup i \in Q^+ I_p^+ \) \( M(\text{lub}(U), i) \), and \( \forall i \in I^- \) \( \neg M(\text{lub}(U), i) \).

**Proof.** The proof is dual to that of lemma 23. □

**Theorem 26** Consider a nonempty set \( Q^+ \subseteq P^+ \) s.t. \((\forall i \in Q^+)|S(I_p^+, I^-)| = 1\). If \( L_c \) is an admissible concept language and property 2 holds, then:

\[
S(\cup i \in Q^+ I_p^+, I^-) = \{\text{lub}(\cup i \in Q^+ S(I_p^+, I^-))\}.
\]

**Proof.** The proof is dual to that of theorem 24.

Since property 2 is dual to property 1, the algorithm can be explained analogously to the specialised merging algorithm from the previous subsection.

### 6 Other Useful Algorithms

In this section we propose two useful algorithms of ABSs. The first algorithm checks whether version spaces represented by ABSs are empty. It can be applied for the class of admissible concept languages when one of properties 1 and 2 holds. The algorithm is based on theorems 27 and 28.
Theorem 27 If Lc is an admissible concept language and property 1 holds, then:
\[(VS(I^+, I^-) \neq \emptyset) \leftrightarrow (\forall I_p^+ \in P^+)(VS(I^+, I_p^-) \neq \emptyset).\]

Proof. \((\rightarrow)\) This part of the proof follows from theorem 12.

\((-\rightarrow)\) Consider a set \(U = \bigcup_{I_p^- \in P^-} VS(I^+, I_p^-)\). Since \(P^-\) is a special covering, \((\forall I_p^- \in P^-)(VS(I^+, I_p^-) \neq \emptyset)\) implies that the set \(U \neq \emptyset\). By the definition of version spaces \(VS(I^+, I_p^-)\) the set \(U\) has two additional properties:

\[(\forall i \in I^+)(\forall c \in U)M(c, i)\]  
\[(\forall I_p^- \in P^-)(\exists c \in U)(\forall i \in I_p^-)\neg M(c, i).\]  

By lemma 23 \(U \neq \emptyset\), formulas (15) and (16) imply that: \(\text{glb}(U) \in Lc\), \((\forall i \in I^+)M(\text{glb}(U), i)\) and \((\forall i \in I^-)\neg M(\text{glb}(U), i)\). Thus, according to definition 1 \(\text{glb}(U) \in VS(I^+, I^-)\); i.e., \(VS(I^+, I^-) \neq \emptyset\).

Theorem 28 If Lc is an admissible concept language and property 2 holds, then:
\[(VS(I^+, I^-) \neq \emptyset) \leftrightarrow (\forall I_p^+ \in P^+)(VS(I_p^+, I^-) \neq \emptyset).\]

Proof. The proof is dual to that of theorem 27. □

The algorithm considered is described as follows. Assume that the version space \(VS(I^+, I^-)\) is given by \(\text{ABSs}\) \(\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle\). The algorithm checks whether \(VS(I^+, I^-) = \emptyset\) by testing the sets \(S(I_p^+, I^-)\) and \(G(I^+, I_p^-)\). If at least one of these sets is empty, then by theorem 27 or 28 \(VS(I^+, I^-) = \emptyset\). Hence, the algorithm returns true; otherwise, it returns false.

The second useful algorithm is the classification algorithm of ABSs. It implements the rule of unanimous voting [5, 6, 7]: an instance is classified if all the descriptions in a version space agree on an instance classification. The algorithm is based on theorems 29 and 30.

Theorem 29 \((\forall i \in I)((\forall c \in VS(I^+, I^-))M(c, i) \rightarrow VS(I^+, I^- \cup \{i\}) = \emptyset)\)

Theorem 30 \((\forall i \in I)((\forall c \in VS(I^+, I^-))\neg M(c, i) \rightarrow VS(I^+ \cup \{i\}, I^-) = \emptyset)\).

When an instance \(i\) has to be classified, the classification algorithm starts by forming the \(\text{ABSs}\) of the version space \(VS(I^+, I^- \cup \{i\})\). If \(VS(I^+, I^- \cup \{i\})\) is empty, by theorem 29 all the descriptions in \(VS(I^+, I^-)\) cover the instance \(i\). Hence, the instance \(i\) is positive w.r.t. \(VS(I^+, I^-)\). If \(VS(I^+, I^- \cup \{i\})\) is not empty, the algorithm forms the \(\text{ABSs}\) of the version space \(VS(I^+ \cup \{i\}, I^-)\). If \(VS(I^+ \cup \{i\}, I^-)\) is empty, by theorem 30 all the descriptions in \(VS(I^+, I^-)\) do not cover the instance \(i\). Hence, the instance \(i\) is negative w.r.t. \(VS(I^+, I^-)\).

The classification algorithm uses the learning algorithm and the algorithm that checks whether version spaces are empty. Since the latter is applicable when one of properties 1 and 2 holds, the classification algorithm is used only for these cases.

7 Example of Adaptable Boundary Sets

Consider a concept-learning task s.t. the set \(I\) and the concept language \(Lc\) are 1-CNF languages with 8 Boolean attributes and \(I \subseteq Lc\). The training instances are given in the left column of table 1. They

\(^5\)Note that property 1 holds for 1-CNF languages with Boolean attributes.
are chosen s.t. $|G(\{i_1^+, i_2^+, i_3^+, i_4^+, i_5^+\})| = 2^{\{i_2^+, i_3^+, i_4^+, i_5^+\}} = 16$; i.e., the size of the set is exponential in the number of negative instances.

To avoid this exponential dependence, the learning algorithm is tuned to form new special coverings $P^+$ and $P^-$ s.t. the size of ABSs is less or equal to the size of instance-based boundary sets. This requirement is imposed since the size of instance-based boundary sets is polynomial in the size of training data for 1-CNF languages with Boolean attributes \[11, 12, 13\].

The ABSs of the target version space is initialised s.t. $S(\emptyset, \emptyset) = \{(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\}$ and $G(\emptyset, \emptyset) = \{?(?, ?, ?, ?, ?, ?, ?)\}$. The trace of the learning algorithm is given in the right column of table 1. It is explained as follows. Before the instance $i_4^+$ the ABSs are equal to boundary sets, since the sizes of boundary sets and instance-based boundary sets are equal in this case. When the instance $i_4^+$ is processed the size of boundary sets becomes larger. Hence, the algorithm creates a new set $G(\{i_1^+, i_2^+, i_3^+, i_4^+\})$ and later updates it by the next instance $i_5^+$ (see set $G(\{i_1^+, i_2^+, i_3^+, i_4^+, i_5^+\})$). Note that $|G(\{i_1^+, i_2^+, i_3^+, i_4^+\})| + |G(\{i_1^+, i_4^+, i_5^+\})| = 8$; i.e., the size of the $G$ part of the ABSs is not exponential in the number of the negative instances. When the positive instance $i_6^+$ is processed, the sets $G(\{i_1^+, i_6^+\})$, $\{i_2^+, i_3^+, i_4^+, i_6^+\}$) and $G(\{i_1^+, i_5^+, i_6^+\})$, $\{i_4^+, i_6^+\}$) are pruned to contain only one element.

The learning algorithm forms the $S$-part of the ABSs consisting of the set $S(\{i_1^+, i_5^+, i_6^+\})$, $\{i_2^+, i_3^+, i_4^+, i_5^+, i_6^+\}$) only, since the size of the set is equal to one \[11\].

The resulting ABSs are processed by the specialised merging algorithm, since $G$ sets contain just one element. Thus, the final ABSs are given by $S(\{i_1^+, i_5^+, i_6^+\})$, $\{i_2^+, i_3^+, i_4^+, i_5^+, i_6^+\}$) = \{(1, ?, 1, ?, 1, ?, 1, ?)\} and $G(\{i_1^+, i_6^+\})$, $\{i_1^+, i_5^+, i_6^+\}) = \{(1, ?, 1, ?, 1, ?, 1, ?)\}$.

### Table 1: Training Instances and the Trace of the Learning Algorithm.

<table>
<thead>
<tr>
<th>Training Instance</th>
<th>Algorithm Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $i_1^+ = (1,1,1,1,1,1,1)$</td>
<td>$S({i_1^+, i_1^+}, \emptyset) = {(1, 1, 1, 1, 1, 1, 1)}$</td>
</tr>
<tr>
<td></td>
<td>$G({i_1^+, i_1^+}, \emptyset) = {(? , ?, ?, ?, ?, ?, ?)}$</td>
</tr>
<tr>
<td>2. $i_2^+ = (0,0,1,1,1,1,1)$</td>
<td>$S({i_2^+, i_2^+}, \emptyset) = {(1, 1, 1, 1, 1, 1, 1)}$</td>
</tr>
<tr>
<td></td>
<td>$G({i_2^+, i_2^+}, \emptyset) = {(? , ?, ?, ?, ?, ?, ?, ?)}$</td>
</tr>
<tr>
<td>3. $i_3^+ = (1,1,0,0,1,1,1)$</td>
<td>$S({i_3^+, i_3^+}, \emptyset) = {(1, 1, 1, 1, 1, 1, 1)}$</td>
</tr>
<tr>
<td></td>
<td>$G({i_3^+, i_3^+}, \emptyset) = {(? , ?, ?, ?, ?, ?, ?, ?, ?)}$</td>
</tr>
<tr>
<td>4. $i_4^+ = (1,1,1,1,0,0,1)$</td>
<td>$S({i_4^+, i_4^+}, \emptyset) = {(1, 1, 1, 1, 1, 1, 1)}$</td>
</tr>
<tr>
<td></td>
<td>$G({i_4^+, i_4^+}, \emptyset) = {(? , ?, ?, ?, ?, ?, ?, ?, ?, ?)}$</td>
</tr>
</tbody>
</table>

8 Conclusions

In this report we presented ABSs as a new version-space representation. We showed that ABSs can be adjusted to each version-space representation between boundary sets and instance-based boundary sets. Hence, the choice of version-space representations can be done dynamically during the learning phase.

We consider ABSs as a first attempt unifying version-space representations.\(^6\) This allows different

\(^6\)Note that boundary sets and instance-based boundary sets are special cases of ABSs.
representations to benefit from each other. For example:

- if the size of boundary sets becomes exponential in the number of training instances, they can be considered as adaptable and learning can continue by the learning algorithm from section 4;
- if the comprehensibility of instance-based boundary sets is not sufficient, they can be considered as adaptable and learning can continue after their optimisation by one of the merging algorithms from section 5.

Future research will focus on unifying other existing version-space representations (cf. [2, 3, 4, 8, 14]). The research, if successful, will lead to a development of a complete practical theory of version-space representations.

References