Generalization and specialization strategies for learning r.e. languages

Sanjay Jain
Department of Information Systems and Computer Science
National University of Singapore
Singapore 119260, Republic of Singapore
Email: sanjay@iscs.nus.sg

Arun Sharma
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052, Australia
Email: arun@cse.unsw.edu.au

August 14, 2009

Abstract

Overgeneralization is a major issue in the identification of grammars for formal languages from positive data. Different formulations of generalization and specialization strategies have been proposed to address this problem, and recently there has been a flurry of activity investigating such strategies in the context of indexed families of recursive languages.

The present paper studies the power of these strategies to learn recursively enumerable languages from positive data. In particular, the power of strong-monotonic, monotonic, and weak-monotonic (together with their dual notions modeling specialization) strategies are investigated for identification of r.e. languages. These investigations turn out to be different from the previous investigations on learning indexed families of recursive languages and at times require new proof techniques.

A complete picture is provided for the relative power of each of the strategies considered. An interesting consequence is that the power of weak-monotonic strategies is equivalent to that of conservative strategies. This result parallels the scenario for indexed classes of recursive languages. It is also shown that any identifiable collection of r.e. languages can also be identified by a strategy that exhibits the dual of weak-monotonic property. An immediate consequence of the proof of this result is that if attention is restricted to infinite r.e. languages, then conservative strategies can identify every identifiable collection.

1 Introduction

Consider the identification of formal languages from positive data. A machine is fed all the strings and no nonstrings of a language L, in any order, one string at a time. The machine, as it is receiving strings of L, outputs a sequence of grammars. The machine is said to identify L just in case the sequence of grammars converges to a grammar for L. This is essentially the paradigm of identification in the limit introduced by Gold [Gol67].

Since only strings belonging to the language are available, if a learning machine conjectures a grammar for some superset of the target language, it may not be "rational" for the machine to revise this conjecture as data about the complement of the language is not available. This is the problem of overgeneralization in learning formal languages from positive data.

The notion of monotonic strategies has been proposed to model learning heuristics that gradually refine their hypotheses. Learning machines following these strategies behave in such a way that their successive conjectures are an "improvement" on their previous conjectures. These strategies can be grouped into the following two classes:

Generalization Strategies These strategies model learners that begin by hypothesizing a grammar for a language smaller than the target language and then build to the target language. These strategies may also be thought of as modeling bottom-up or specific to general search in practical machine learning systems (e.g., Muggleton and Feng's Golem [MF90]).

Specialization Strategies Generalization strategies improve upon their successive conjectures by emitting grammars for larger and larger languages. Alternatively, a learner can begin with a grammar for a language larger than the target language and then "cut down" its hypotheses to converge to a grammar for the target language. Such strategies may be thought of as modeling top-down or general to specific search in practical machine learning systems (e.g., Shapiro's MIS [Sha81] and Quinlan's FOIL [Qui90]).

Recently there has been a flurry of results describing the power of these strategies for identifying indexed families of recursive languages (see Lange and Zeugmann [LZ92b, LZ93b, LZ93a, LZ93c, LZ92a], Lange, Zeugmann, and Kapur [LZK92, LZK96], and Kapur [Kap92, Kap93], Kapur and Bilardi [KB92], Kinber [Kin94], Mukouchi [Muk92a, Muk92b], and Mukouchi and Arikawa [MA93]).

The present paper studies and presents a complete picture of the power of these strategies to identify r.e. languages. To facilitate the discussion of these notions and results, we introduce some notation next.

The symbol φ denotes an acceptable programming system, and φ_i denotes the partial computable function computed by the program with index i in the φ system. The language accepted by the φ-program with the index i is denoted W_i. Hence, $W_0, W_1, W_2, \ldots$ is an enumeration of all the recursively enumerable languages, and i may be thought of as a grammar (acceptor) for the r.e. language W_i. A text for an r.e. language L is an infinite sequence of numbers such that each element of L appears at least once in the sequence and no nonelement of L ever appears in the sequence.

The three generalization strategies investigated in the literature are discussed next.

Jantke [Jan91] proposed the notion of *strong monotonic* strategies that upon being fed a text for the language output a chain of hypotheses such that if grammar j is output after grammar i, then $W_i \subseteq W_j$. The consequence of this requirement is that if a learner incorrectly assumes a string to belong to the language being learned then it cannot revise this assumption by emitting a hypothesis that does not include the string.

Wiehagen [Wie90] suggested that the requirement of strong monotonicity is too stringent and proposed the notion of *monotonic* strategies to be those learners that produce more general hypotheses only with respect to the language being identified. More precisely, if grammar j is conjectured after grammar i, then $L \cap W_i \subseteq L \cap W_j$. In other words, a monotonic strategy is allowed to correct its mistaken assumption that certain nonstrings of L belong to L but once it has correctly concluded that a string of L belongs to L it is not allowed to output a hypothesis that contradicts such a conclusion.

Jantke [Jan91], motivated by the work on non-monotonic logics, introduced the notion of *weak-monotonic* strategies to be such that if grammar j is conjectured after grammar i and the set of strings seen by the machine when grammar j is conjectured is a subset of W_i, then $W_i \subseteq W_j$. In other words, a weak monotonic learner can expel strings from its hypothesis only if it encounters strings that cannot be accounted for by its current hypothesis.

Kapur [Kap92], with a view to model specialization strategies, considered the dual of the above three strategies. The above mentioned papers by Kapur, Lange, and Zeugmann have completely derived the relationship between these strategies in the context of identification of indexed families of recursive languages. The present paper does the same for recursively enumerable languages.

Work on these strategies can be traced back to the notion of *conservative* strategies introduced by Angluin [Ang80]. Conservative strategies are such that they do not change their hypotheses unless they encounter a string that cannot be explained by the current hypothesis. Lange and Zeugmann [LZ93b] have shown that in the context of learning indexed families of recursive languages, weak monotonic and conservative strategies have equivalent power. Using new techniques we are able to show that this equivalence also holds for learning recursively enumerable languages.

Another interesting result is that any identifiable collection of r.e. languages can also be identified by a strategy satisfying the dual of weak-monotonicity requirement. We would like to note that this relationship also holds in the context of indexed families of recursive languages if one considers *class comprising strategies* (see Lange, Zeugmann, and Kapur [LZK96]). However, new proof techniques are needed to establish the result for r.e. languages. Additionally, as a consequence of the proof of this result, we are able to show that if attention is restricted to only infinite r.e. languages, then conservative strategies can identify every identifiable collection.

We now give an example where the situation for indexed families of recursive languages is different from that of r.e. languages. The collection of indexed families of recursive languages identified by class comprising monotonic strategies is a strict subset of the collection of indexed families of recursive languages identified by class comprising weak-monotonic strategies. However, in the context of r.e. languages, the two collections are incomparable.

The results presented in the paper yield the complete relationship between strong monotonic, monotonic, weak-monotonic, dual-strong monotonic, dual-monotonic, dual-weak-monotonic, and conservative strategies in the context of identification of r.e. languages.

In what follows we proceed formally. Section 2 introduces the notation and preliminary notions from language learning theory. Section 3 contains the definition of all the strategies considered in the present paper. Results are presented in Section 4, where the relationship between all the strategies is first pictorially summarized in Figure 1.

2 Preliminaries

2.1 Notation

Any unexplained recursion theoretic notation is from [Rog67]. The symbol N denotes the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$. Unless otherwise specified, i, j, k, l, m, n, q, r, s, t, x, and y, with or without decorations[1], range over N. Symbols $\emptyset$, $\subseteq$, $\subset$, $\supseteq$, and $\supset$ denote empty set, subset, proper subset, superset, and proper superset, respectively. Symbols A and S, with or without decorations, range over sets. $\overline{A}$ denotes the complement of set A. D, P, Q, with or without decorations, range over finite sets. Cardinality of a set S is denoted by $\text{card}(S)$. The maximum and minimum of a set are denoted by $\max(\cdot), \min(\cdot)$, respectively, where $\max(\emptyset) = 0$ and $\min(\emptyset)$ is undefined.

$\langle \cdot, \cdot \rangle$ denotes an arbitrary, computable, bijective mapping from $N \times N$ onto N [Rog67].

By φ we denote a fixed *acceptable* programming system for the partial computable functions: $N \rightarrow N$ [Rog58, Rog67, MY78]. By φ_i we denote the partial computable function computed by the program with index i in the φ-system. Symbol $\mathcal{R}$ denotes the set of all total computable functions. By Φ we denote an arbitrary fixed Blum complexity measure [Blu67, HU79] for the φ-system. By W_i we denote the domain of the partial computable function φ_i. W_i is, then, the r.e. set/language ($\subseteq N$) accepted (or equivalently, generated) by the φ-program i. L, with or without decorations, ranges over languages, that is, subsets of N. $\mathcal{L}$, with or without decorations, ranges over sets of languages. $\mathcal{E}$ denotes the set of all r.e. languages. We denote by $W_{i,s}$ the set $\{x \mid x < s \ \wedge \ \Phi_i(x) < s\}$.

[1]Decorations are subscripts, superscripts and the like.

2.2 Language Learning

A *sequence* σ is a mapping from an initial segment of N into $(N \cup \{\#\})$. The *content* of a sequence σ, denoted content(σ), is the set of natural numbers in the range of σ. Intuitively, #'s represent pauses in the presentation of data. The *length* of σ is denoted by $|\sigma|$; the length of the empty sequence is 0. For $n \leq |\sigma|$, the initial sequence of σ of length n is denoted by $\sigma[n]$. We let σ, τ, and γ, with or without decorations, range over finite sequences. SEQ denotes the set of all finite sequences.

Definition 1 A *language learning machine*, $\mathbf{M}$, is an algorithmic device which computes a mapping from SEQ into $N \cup \{\bot\}$ such that if $\sigma \subseteq \tau$ and $\mathbf{M}(\sigma) \neq \bot$ then $\mathbf{M}(\tau) \neq \bot$.

The symbol $\bot$ denotes a nonnumeric element. The output of machine $\mathbf{M}$ on evidential state σ is denoted by $\mathbf{M}(\sigma)$, where "$\mathbf{M}(\sigma) = \bot$" means that $\mathbf{M}$ does not issue any hypothesis on σ. The reader should note the further requirement that once a machine $\mathbf{M}$ on evidential state σ outputs a program (that is, $\mathbf{M}(\sigma) \neq \bot$), $\mathbf{M}$ continues to do so on all extensions of σ. The reader should also note that this additional requirement does not affect the collections of languages identified by the strategies introduced in the present paper.

We let $\mathbf{M}$, with or without superscripts, range over language learning machines. ($\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2, \ldots$ represents a special enumeration of of machines; see Proposition 5 below.)

A *text* T for a language L is a mapping from N into $(N \cup \{\#\})$ such that L is the set of natural numbers in the range of T. The *content* of a text T, denoted content(T), is the set of natural numbers in the range of T. $T[n]$ denotes the finite initial sequence of T with length n. We next introduce Gold's seminal notion of identification in the limit of r.e. languages.

We say that $\mathbf{M}$ *converges* on text T to i (written $\mathbf{M}(T)\!\downarrow = i$) if $(\overset{\infty}{\forall} n)[\mathbf{M}(T[n]) = i]$. We say that $\mathbf{M}$ *converges* on text T (written $\mathbf{M}(T)\!\downarrow$) if there exists an i, such that $\mathbf{M}$ on T converges to i; otherwise we say that $\mathbf{M}$ *diverges* on T (written $\mathbf{M}(T)\!\uparrow$).

Definition 2 [Gol67]

(a) $\mathbf{M}$ $\mathbf{TxtEx}$-*identifies* L (written: $L \in \mathbf{TxtEx}(\mathbf{M})$) $\iff$ ($\forall$ texts T for L) ($\exists i \mid W_i = L$) $(\overset{\infty}{\forall} n)[\mathbf{M}(T[n]) = i]$.

(b) $\mathbf{TxtEx} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

Some of our proofs depend on the notion of locking sequences which we describe next.

Definition 3 [BB75, Ful90] Let $\mathbf{M}$ be a learning machine and $L \subseteq N$.

(a) A sequence σ is said to be a *stabilizing sequence* for $\mathbf{M}$ on L just in case content(σ) $\subseteq L$ and $(\forall \tau \mid \sigma \subseteq \tau \ \wedge \ \text{content}(\tau) \subseteq L)[\mathbf{M}(\tau) = \mathbf{M}(\sigma)]$.

(b) A sequence σ is said to be a *locking sequence* for $\mathbf{M}$ on L just in case σ is a stabilizing sequence for $\mathbf{M}$ on L and $W_{\mathbf{M}(\sigma)} = L$.

Following two lemmas are also used in some of our proofs.

Lemma 1 [BB75, OSW82] *If* $\mathbf{M}$ **TxtEx**-*identifies* L*, then there exists a locking sequence for* $\mathbf{M}$ *on* L.

Lemma 2 [Ful90] *From any learning machine* $\mathbf{M}$ *one may effectively construct* $\mathbf{M}'$ *such that the following hold:*

(a) $\mathbf{TxtEx}(\mathbf{M}) \subseteq \mathbf{TxtEx}(\mathbf{M}')$.

(b) If $L \in \mathbf{TxtEx}(\mathbf{M}')$*, then all texts for* L *contain a locking sequence for* $\mathbf{M}'$ *on* L.

We also compare the power of strategies discussed in the present paper with a restricted version of identification in the limit in which a learning machine is allowed to make only one conjecture which is required to be correct. This criterion of success is referred to as finite identification and was also considered by Gold [Gol67].

Definition 4 [Gol67]

(a) $\mathbf{M}$ **TxtFin**-*identifies* L (written: $L \in \mathbf{TxtFin}(\mathbf{M})$) $\iff$ ($\forall$ texts T for L) $(\exists i \mid W_i = L)$ $(\exists n_0)[(\forall n \geq n_0)[\mathbf{M}(T[n]) = i] \ \wedge \ (\forall n < n_0)[\mathbf{M}(T[n]) = \perp]]$.

(b) $\mathbf{TxtFin} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtFin}(\mathbf{M})]\}$.

3 Generalization and Specialization Strategies

Before we consider the three generalization and the three specialization strategies described in the introductory section, we discuss a technical point first.

In many cases, the effect of a strategy may be dependent on the scope of the constraints embodied in the strategy. We illustrate this issue and the notation about strategies with the help of an example.

Consider the constraint on learners that requires them to conjecture hypotheses consistent with the data. Formally, learners satisfying this requirement have the property that on sequence σ, they output the grammar of a language that contains content(σ). This requirement can be applied in two different ways, each application rendering possibly different collections of languages identifiable. These applications are:

(a) *Global consistency:* A collection of languages $\mathcal{L}$ is identifiable by a globally consistent learner just in case there exists an $\mathbf{M}$ that is consistent on all $\sigma \in$ SEQ and $\mathbf{M}$ identifies $\mathcal{L}$.

(b) *Class consistency:* A collection of languages $\mathcal{L}$ is identifiable by a class consistent learner just in case there exists an $\mathbf{M}$ that is consistent on all sequences, σ, drawn from languages in $\mathcal{L}$ and $\mathbf{M}$ identifies $\mathcal{L}$.

Wiehagen and Leipe [WL76] have shown that the above two notions of consistency are not equivalent.

3.1 Generalization Strategies

Taking into account the above discussion, we consider two versions for each of the three generalization strategies: global and class. The global version requires the generalization properties to hold on all languages whereas the class version requires the generalization properties to hold only on the languages in the class of languages being identified. In many cases the global and the class versions turn out to be equivalent.

We first consider strong-monotonicity.

Definition 5 [Jan91]

(a) $\mathbf{M}$ is said to be *strong-monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge \ \text{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \perp \ \vee \ W_{\mathbf{M}(\sigma)} \subseteq W_{\mathbf{M}(\tau)}]$.

(b) $\mathbf{M}$ is said to be *strong-monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is strong-monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *strong-monotonic* just in case $\mathbf{M}$ is strong-monotonic on each $L \subseteq N$.

We now define the collections of languages identifiable by the global version (referred to as **G-SMON**) and the class version (referred to as **C-SMON**) of strong-monotonic strategies.

Definition 6 *(a)* $\mathbf{G\text{-}SMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ *is strong-monotonic and* $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(b) $\mathbf{C\text{-}SMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ *is strong-monotonic on* $\mathcal{L}$ *and* $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

The next proposition shows that **G-SMON** = **C-SMON**.

Proposition 3 **G-SMON** = **C-SMON**.

PROOF. Clearly, **G-SMON** $\subseteq$ **C-SMON**. We now show that **C-SMON** $\subseteq$ **G-SMON**. Suppose $\mathcal{L} \in$ **C-SMON** as witnessed by $\mathbf{M}$. We now informally describe a learner $\mathbf{M}'$ that behaves as follows: $\mathbf{M}'$, fed σ, simulates $\mathbf{M}$ on each initial subsequence of σ (including σ), and outputs a grammar for the union of all the languages whose grammars are output by $\mathbf{M}$ on initial subsequences of σ (including σ). It is easy to verify that $\mathbf{M}'$ is strong-monotonic and $\mathbf{M}'$ identifies $\mathcal{L}$. ▮

Hence, we only consider the global version of strong-monotonicity in the sequel, and refer to the collections of languages identifiable by strong-monotonic learners as simply **SMON**.

The next definition is about monotonic strategies.

Definition 7 [Wie90]

(a) $\mathbf{M}$ is said to be *monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge \ \mathrm{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \ \vee \ W_{\mathbf{M}(\sigma)} \cap L \subseteq W_{\mathbf{M}(\tau)} \cap L]$.

(b) $\mathbf{M}$ is said to be *monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *monotonic* just in case $\mathbf{M}$ is monotonic on each $L \subseteq N$.

(d) $\mathbf{G\text{-}MON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) $\mathbf{C\text{-}MON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is monotonic on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

It is easy to show that the global version of monotonicity is equivalent to the requirement of strong monotonicity (i.e., **G-MON** = **SMON**). Therefore, we consider only the class version of monotonicity in the sequel. In the literature **C-MON** is usually referred to as **MON**.

The next definition introduces weak-monotonicity.

Definition 8 [Jan91]

(a) $\mathbf{M}$ is said to be *weak-monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge \ \mathrm{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \ \vee \ [\mathrm{content}(\tau) \subseteq W_{\mathbf{M}(\sigma)} \Rightarrow W_{\mathbf{M}(\sigma)} \subseteq W_{\mathbf{M}(\tau)}]]$.

(b) $\mathbf{M}$ is said to be *weak-monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is weak-monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *weak-monotonic* just in case $\mathbf{M}$ is weak-monotonic on each $L \subseteq N$.

(d) $\mathbf{G\text{-}WMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is weak-monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) $\mathbf{C\text{-}WMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is weak-monotonic on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

Clearly, **G-WMON** $\subseteq$ **C-WMON**. Now, using Theorem 6 and Theorem 21, it will be shown, in Corollary 22, that **G-WMON** = **C-WMON**. Since the two classes turn out to be equivalent, we will often refer to them as just **WMON**.

3.2 Specialization Strategies

Kapur [Kap92] considered the dual of the above generalization strategies to introduce three specialization strategies. The next three definitions introduce dual-strong-monotonicity, dual-monotonicity, and dual-weak-monotonicity.

Definition 9 [Kap92]

(a) $\mathbf{M}$ is *dual-strong-monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge \ \text{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \ \vee \ W_{\mathbf{M}(\sigma)} \supseteq W_{\mathbf{M}(\tau)}]$.

(b) $\mathbf{M}$ is *dual-strong-monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is dual-strong-monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *dual-strong-monotonic* just in case it is dual-strong-monotonic on each $L \subseteq N$.

(d) $\mathbf{G\text{-}DSMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is dual-strong-monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) $\mathbf{C\text{-}DSMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is dual-strong-monotonic on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

Again as is the case with generalization strategies, the next proposition shows that the global and the class versions of dual-strong-monotonicity are equivalent. The proof of the proposition is similar to the proof of Proposition 3 (the only change required is that the machine $\mathbf{M}'$ outputs the grammar for the *intersection* of the languages whose grammars are output by $\mathbf{M}$ on the initial subsequences).

Proposition 4 $\mathbf{G\text{-}DSMON} = \mathbf{C\text{-}DSMON}$.

Because of the above proposition, we only consider the global version of dual-strong-monotonicity in the sequel, and refer to the class as simply **DSMON**. We next consider dual-monotonic strategies.

Definition 10 [Kap92]

(a) $\mathbf{M}$ is *dual-monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge \ \text{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \ \vee \ \overline{W_{\mathbf{M}(\sigma)}} \cap \overline{L} \subseteq \overline{W_{\mathbf{M}(\tau)}} \cap \overline{L}]$.

(b) $\mathbf{M}$ is *dual-monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is dual-monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *dual-monotonic* just in case it is dual-monotonic on each $L \subseteq N$.

(d) $\mathbf{G\text{-}DMON} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is dual-monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) $\mathbf{C\text{-}DMON} = \{\mathcal{L} \subseteq \mathcal{E} \mid (\exists \mathbf{M})[\mathbf{M}$ is dual-monotonic on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

The reader should note that the global version of dual-monotonicity is essentially the same as the requirement of dual-strong-monotonicity if one also allows the grammars output by dual-strong-monotonic machines to additionally contain the elements which have *already* appeared in the input. For this reason, we feel that the class **G-DMON** does not yield any new insight into the nature of specialization strategies. Thus we will consider only **C-DMON** in the sequel. In literature **C-DMON** is usually referred to as **DMON**.

We next consider dual-weak-monotonicity.

Definition 11 [Kap92]

(a) $\mathbf{M}$ is *dual-weak-monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge\ \mathrm{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \ \vee\ [\mathrm{content}(\tau) \subseteq W_{\mathbf{M}(\sigma)} \Rightarrow W_{\mathbf{M}(\sigma)} \supseteq W_{\mathbf{M}(\tau)}]]$.

(b) $\mathbf{M}$ is *dual-weak-monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is dual-weak-monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *dual-weak-monotonic* just in case it is dual-weak-monotonic on each $L \subseteq N$.

(d) **G-DWMON** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is dual-weak-monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) **C-DWMON** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is dual-weak-monotonic on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

It will be shown in Theorem 19 that **G-DWMON** = **TxtEx**, that is, every identifiable collection of languages is also identifiable by a globally dual-weak-monotonic learner. Now since **G-DWMON** $\subseteq$ **C-DWMON**, we have that the global and the class versions of dual-weak-monotonicity are equivalent. For this reason, we only consider the global version in the sequel and refer to the class as simply **DWMON**.

3.3 Strategies with Both Requirements

Requiring both the normal and dual conditions yields three additional strategies. We give the definition for strong monotonicity; the other two can be defined similarly.

Definition 12 [LZK92]

(a) $\mathbf{M}$ is *both strong and dual-strong-monotonic* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge\ \mathrm{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \ \vee\ W_{\mathbf{M}(\sigma)} = W_{\mathbf{M}(\tau)}]$.

(b) $\mathbf{M}$ is *both strong and dual-strong-monotonic* on $\mathcal{L}$ just in case $\mathbf{M}$ is both strong and dual-strong-monotonic on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *both strong and dual-strong-monotonic* just in case $\mathbf{M}$ is both strong and dual-strong-monotonic on each $L \subseteq N$.

(d) **G-BSMON** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is both strong and dual-strong-monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) **C-BSMON** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is both strong and dual-strong-monotonic on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

It is easy to show that **G-BSMON** = **C-BSMON**, and we refer to these classes as simply **BSMON** in the sequel. Similarly, we can define **G-BMON**, **C-BMON** and **G-BWMON** and **C-BWMON**. Also, **G-BWMON** = **C-BWMON** (using Theorem 6 and Corollary 22 to Theorem 21); hence, we often refer to these two classes as simply **BWMON**.

Also, similar to the case of **G-DMON**, we do not feel that **G-BMON** gives much insight into the nature of monotonic strategies. We will thus consider only **C-BMON** in the sequel. In the literature **C-BMON** is usually referred to as **BMON**.
Remark: It should be noted that except for the classes **C-BMON** and **BSMON** we do not need the machines to output $\perp$. This is because for **SMON**, **C-MON**, **WMON**, **G-BWMON**, **C-BWMON**, **G-BMON** a machine can just output a grammar for $\emptyset$ instead of $\perp$ and for **DSMON**, **C-DMON**, **G-DMON**, **C-DWMON**, **G-DWMON**, the machine can output a grammar for N instead of $\perp$. We do not know at this point whether not allowing $\perp$ will make a difference in the class **C-BMON**. For the class **BSMON**, not allowing $\perp$ essentially means that machine has to output its conjecture on any input (even null input) and which should be a grammar for the input language. This means that a machine following such a strategy can identify at most one language (thus leading to a result such as **BSMON** $\subset$ **TxtFin**).

Finally, we define Angluin's [Ang80] notion of conservative strategies.

Definition 13 (a) $\mathbf{M}$ is *conservative on* L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge\ \mathrm{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \perp \ \vee\ [\mathrm{content}(\tau) \subseteq W_{\mathbf{M}(\sigma)} \Rightarrow \mathbf{M}(\sigma) = \mathbf{M}(\tau)]]$.

(b) $\mathbf{M}$ is *conservative on* $\mathcal{L}$ just in case $\mathbf{M}$ is conservative on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *conservative* just in case $\mathbf{M}$ is conservative on each $L \subseteq N$.

(d) **G-CONSV** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is conservative and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) **C-CONSV** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is conservative on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

As a consequence of Theorem 6 and Theorem 21, in Corollary 22 it is shown that **C-CONSV** = **G-CONSV**. Thus we often refer to these classes as simply **CONSV**.

In some of our theorems, we require a special effective enumeration of language learning machines as given by the following (folklore) proposition.

Proposition 5 *There exists an recursive enumeration* $\mathbf{M}_0, \mathbf{M}_1, \ldots$ *of language learning machines, such that the following is satisfied. Let* $\mathbf{I}$ *be any collection of language classes identified by an strategy introduced in this paper. Then,* $(\forall \mathcal{L} \in \mathbf{I})\ (\exists j)[\mathcal{L} \in \mathbf{I}$ *as witnessed by* $\mathbf{M}_j]$.

A proof of the above proposition can be worked out on similar lines to the proof for $\mathbf{I} = \mathbf{TxtEx}$ case (see for example [OSW86]). We let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ be one such enumeration of machines.

4 Results

It is easy to see that for

$$\mathbf{I} \in \{\mathbf{MON}, \mathbf{DMON}, \mathbf{BMON}, \mathbf{WMON}, \mathbf{DWMON}, \mathbf{BWMON}\},$$

G-I $\subseteq$ **C-I**. Before we present the theorems that imply the relationship between these classes, it is helpful to summarize the relationship between the classes in the following figure.

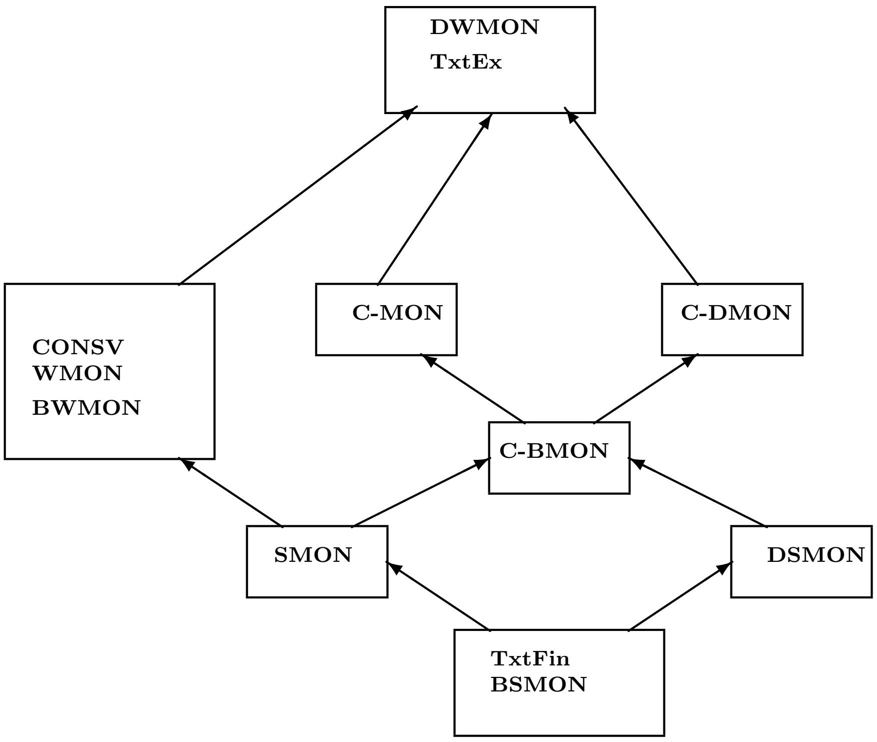

Fig. 1: Relationship between various strategies (classes appearing in the same box are equivalent; $\rightarrow$ denotes strict inclusion; absence of a directed path between two classes means that the two classes are incomparable)

We now present a series of theorems and corollaries that derive the above picture. The following theorem follows from the definition of various strategies.

Theorem 6 *(a)* **TxtFin** = **BSMON**.

(b) **BSMON** $\subseteq$ **G-CONSV**.

(c) **BSMON** $\subseteq$ (**SMON** $\cap$ **DSMON**).

(d) **C-BMON** $\subseteq$ (**C-MON** $\cap$ **C-DMON**).

(e) **C-BWMON** $\subseteq$ (**C-WMON** $\cap$ **C-DWMON**).

(f) **G-BWMON** $\subseteq$ (**G-WMON** $\cap$ **G-DWMON**).

(g) **C-CONSV** $\subseteq$ **C-BWMON**.

(h) **G-CONSV** $\subseteq$ **G-BWMON**.

(i) **SMON** $\subseteq$ (**C-MON** $\cap$ **G-WMON**).

(j) **DSMON** $\subseteq$ (**C-DMON** $\cap$ **G-DWMON**).

(k) **BSMON** $\subseteq$ (**C-BMON** $\cap$ **G-BWMON**).

The next two results can also be proved easily.

Theorem 7 *(a)* **SMON** $\subseteq$ **C-BMON**.

(b) **DSMON** $\subseteq$ **C-BMON**.

PROOF. We show that **SMON** $\subseteq$ **C-BMON**. It can similarly be shown that **DSMON** $\subseteq$ **C-BMON**.

Suppose $\mathbf{M}$ is given such that $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})$ and $\mathbf{M}$ is strong-monotonic. We claim that $\mathbf{M}$ is both monotonic and dual-monotonic on each $L \in \mathcal{L}$. Note that since $\mathbf{M}$ is strong monotonic we only need to verify that $(\forall L \in \mathcal{L})(\forall \sigma \subseteq \tau \mid \text{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \bot \;\vee\; \overline{L} \cap \overline{W_{\mathbf{M}(\tau)}} \supseteq \overline{L} \cap \overline{W_{\mathbf{M}(\sigma)}}]$.

But this follows from the fact that $(\forall L \in \mathcal{L})(\forall \sigma \mid \text{content}(\sigma) \subseteq L)[\mathbf{M}(\sigma) = \bot \;\vee\; W_{\mathbf{M}(\sigma)} \subseteq L]$ (which is true since $\mathbf{M}$ is strong-monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})$). ▮

Let FIN $= \{L \mid \text{card}(L) < \infty\}$.

Theorem 8 **SMON** $-$ **DSMON** $\neq \emptyset$.

PROOF. It is easy to see that FIN $\in$ **SMON**. We claim that FIN $\notin$ **DSMON**. Suppose $\mathbf{M}$ is given such that FIN $\subseteq \mathbf{TxtEx}(\mathbf{M})$. We will show that $\mathbf{M}$ cannot be dual-strong-monotonic. Let σ be such that content$(\sigma) = \{0\}$, and $W_{\mathbf{M}(\sigma)} = \{0\}$. Let $\sigma' \supseteq \sigma$ be such that content$(\sigma') = \{0, 1\}$ and $W_{\mathbf{M}(\sigma')} = \{0, 1\}$. (Note that there exist such σ, σ' since $\mathbf{M}$ **TxtEx**-identifies FIN.) But then, $W_{\mathbf{M}(\sigma)} \subset W_{\mathbf{M}(\sigma')}$, and thus $\mathbf{M}$ is not dual-strong-monotonic. ▮

Theorem 9 **DSMON** $-$ **WMON** $\neq \emptyset$.

PROOF. Consider the following definitions.

$L_j = \{\langle j, x\rangle \mid x \in N\}$.

$L_j^m = \{\langle j, x\rangle \mid x < m\}$.

Let T_j be a text for L_j such that content$(T_j[m]) = L_j^m$.

$S_j = \{\langle m, n\rangle \mid m > 0 \;\wedge\; \{\langle j, x\rangle \mid x \leq m\} \subseteq W_{\mathbf{M}_j(T_j[m]), n}\}$.

$\mathcal{L} = \{L_j \mid S_j = \emptyset\} \cup \{L_j^m \mid S_j \neq \emptyset \;\wedge\; (\exists n)[\langle m, n\rangle = \min(S_j)]\}$.

We claim that $\mathcal{L} \in \mathbf{DSMON} - \mathbf{WMON}$.

Let G_N be a grammar for N. Let $G_\emptyset$ be a grammar for $\emptyset$. Let G_j denote a grammar for L_j which can be effectively obtained from j. Let G_j^m denote a grammar for L_j^m, which can be obtained effectively from j, m. Note that S_j is recursive. Let $S_j^s = S_j \cap \{x \mid x \le s\}$.

$$\mathbf{M}(T[s]) = \begin{cases} G_N, & \text{if } \mathrm{content}(T[s]) = \emptyset; \\ G_j, & \text{if } \mathrm{content}(T[s]) \neq \emptyset \wedge \\ & \mathrm{content}(T[s]) \subseteq L_j \wedge \\ & S_j^s = \emptyset; \\ G_j^m, & \text{if } \mathrm{content}(T[s]) \neq \emptyset \wedge \\ & \mathrm{content}(T[s]) \subseteq L_j \wedge \\ & S_j^s \neq \emptyset \wedge \\ & (\exists n)[\langle m, n\rangle = \min(S_j^s)]; \\ G_\emptyset, & \text{otherwise.} \end{cases}$$

It is easy to verify that $\mathbf{M}$ is dual strong monotonic and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})$.

We now show that $\mathcal{L} \notin \mathbf{WMON}$. Suppose that $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M}_j)$. We will show that in this case, $\mathbf{M}_j$ cannot be weak-monotonic. Consider L_j. Clearly, $S_j \neq \emptyset$ (otherwise $L_j \in \mathcal{L}$ and $\mathbf{M}_j$ does not $\mathbf{TxtEx}$-identify L_j). Suppose $\langle m, n\rangle = \min(S_j)$. Now $L_j^m \in \mathcal{L}_j$. Suppose σ is an extension of $T_j[m]$, such that $\mathrm{content}(\sigma) = L_j^m$ and $W_{\mathbf{M}_j(\sigma)} = L_j^m$ (note that there exists such a σ since $L_j^m \in \mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M}_j)$). Now $W_{\mathbf{M}_j(\sigma)} = L_j^m \not\supseteq W_{\mathbf{M}_j(T_j[m])} \supseteq \mathrm{content}(\sigma) = L_j^m$. Thus, $\mathbf{M}_j$ is not weak-monotonic on $L_j^m \in \mathcal{L}$. ▮

Corollary 10

(a) $\mathbf{DSMON} - \mathbf{SMON} \neq \emptyset$.

(b) $\mathbf{C\text{-}DMON} - \mathbf{WMON} \neq \emptyset$.

(c) $\mathbf{G\text{-}DMON} - \mathbf{WMON} \neq \emptyset$.

(d) $\mathbf{C\text{-}BMON} - \mathbf{WMON} \neq \emptyset$.

(e) $\mathbf{C\text{-}MON} - \mathbf{WMON} \neq \emptyset$.

As an aside to the above results, it is interesting to observe the following theorem which says that there are collections of languages that can be identified by either strong-monotonic learners or by dual-strong-monotonic learners, but which do not belong to **TxtFin**.[2] The reader should note that this result also nicely contrasts with Theorem 6 (a).

Theorem 11 *$(\mathbf{DSMON} \cap \mathbf{SMON}) - \mathbf{TxtFin} \neq \emptyset$.*

[2]This fact was brought to our attention by one of the referees.

Proof. Let $L_x^1 = \{2x\}$, and $L_x^2 = \{2x, 2x+1\}$. Let A be a Σ_2-complete set.

Let $\mathcal{L} = \{L_x^1 \mid x \in A\} \cup \{L_x^2 \mid x \notin A\}$.

We first show that $\mathcal{L} \notin \mathbf{TxtFin}$. Let T_x denote a text for $\{2x\}$. Suppose by way of contradiction that $\mathcal{L} \in \mathbf{TxtFin}$ as witnessed by $\mathbf{M}$.

$$\begin{array}{rl} x \in A & \Rightarrow\ L_x^1 \in \mathbf{TxtFin}(\mathbf{M}) \\ & \Rightarrow\ (\forall n)[\mathbf{M}(T_x[n]) = \perp\ \vee\ 2x+1 \notin W_{\mathbf{M}(T_x[n]),n}] \\ & \Rightarrow\ L_x^2 \notin \mathbf{TxtFin}(\mathbf{M}) \\ & \Rightarrow\ \neg(x \notin A) \end{array}$$

Thus $x \in A \iff (\forall n)[\mathbf{M}(T_x[n]) = \perp\ \vee\ 2x+1 \notin W_{\mathbf{M}(T_x[n]),n}]$.

This contradicts Σ_2 completeness of A. Thus, $\mathcal{L} \notin \mathbf{TxtFin}$.

We now show that $\mathcal{L} \in \mathbf{SMON} \cap \mathbf{DSMON}$. Clearly, $\mathcal{L} \in \mathbf{SMON}$. We show that $\mathcal{L} \in \mathbf{DSMON}$.

Let P be a recursive predicate such that $x \in A$ iff $(\exists y)(\forall z)[P(x,y,z)]$. Let H be a recursive function such that $W_{H(x,y)}$ may be defined as follows.

$$W_{H(x,y)} = \begin{cases} \{2x\}, & \text{if } (\forall z)[P(x,y,z)]; \\ \{2x, 2x+1\}, & \text{if } (\exists z)[\neg P(x,y,z)]. \end{cases}$$

It is easy to verify that

(1) $W_{H(x,y)} \in \{L_x^1, L_x^2\}$, and

(2) $x \in A \Leftrightarrow (\exists y)[W_{H(x,y)} = L_x^1]$.

Thus if $x \notin A$, then $(\forall y)[W_{H(x,y)} = L_x^2 = \{2x, 2x+1\}]$. On the other hand if $x \in A$, then there exists a y such that $(\forall j < y)[W_{H(x,j)} = L_x^2]$ and $[W_{H(x,y)} = L_x^1 \subset L_x^2]$. It is this property of H that the following construction of $\mathbf{M}$ exploits.

$\mathbf{M}(T[n])$ is defined as follows.

$$\mathbf{M}(T[n]) = \begin{cases} \perp, & \text{if content}(T[n]) = \emptyset; \\ H(x,0), & \text{if content}(T[n]) = \{2x\}\ \wedge\ \mathbf{M}(T[n-1]) = \perp; \\ H(x,j), & \text{if content}(T[n]) = \{2x\}\ \wedge\ \mathbf{M}(T[n-1]) = H(x,j)\ \wedge \\ & 2x+1 \notin W_{H(x,j),n}; \\ H(x,j+1), & \text{if content}(T[n]) = \{2x\}\ \wedge\ \mathbf{M}(T[n-1]) = H(x,j)\ \wedge \\ & 2x+1 \in W_{H(x,j),n}; \\ H(x,0), & \text{if content}(T[n]) = \{2x, 2x+1\}\ \wedge\ \mathbf{M}(T[n-1]) = \perp; \\ H(x,j), & \text{if content}(T[n]) = \{2x, 2x+1\}\ \wedge\ \mathbf{M}(T[n-1]) = H(x,j); \\ \mathbf{M}(T[n-1]), & \text{otherwise.} \end{cases}$$

It is easy to verify, using properties of H discussed above, that $\mathbf{M}$ is dual strong monotonic in nature. To verify that $\mathbf{M}$ $\mathbf{TxtEx}$-identifies $\mathcal{L}$, suppose T is a text for $L \in \mathcal{L}$. Let x be such that $L \subseteq L_x^2$. Now if $L = L_x^1$ (and thus $x \in A$), then $\mathbf{M}(T)$ would converge to the least j such that $W_{H(x,j)} = L_x^1$. on the other hand if $L = L_x^2$, then $\mathbf{M}(T)$

would converge to $H(x, j)$, for some j (which by properties (1) and (2) of H above is a grammar for L_x^2). Thus, **M TxtEx**-identifies $\mathcal{L}$. ▮

We now compare the power of conservative learners with those of monotonic and dual-monotonic learners.

Theorem 12 **CONSV** $-$ (**C-MON** $\cup$ **C-DMON**) $\neq \emptyset$.

PROOF. The proof of this theorem adopts a technique used by Lange and Zeugmann [LZ93b]. Consider the following languages.

$L_1 = \{\langle 0, x\rangle \mid x \in N\}$;

$L_2^m = \{\langle 0, x\rangle \mid x \leq m\} \cup \{\langle 1, x\rangle \mid x > m\}$;

$L_3^{m,n} = \{\langle 0, x\rangle \mid x \leq m \ \vee \ x > n\} \cup \{\langle 1, x\rangle \mid m < x \leq n\}$.

Let $\mathcal{L} = \{L_1\} \cup \{L_2^m \mid m \in N\} \cup \{L_3^{m,n} \mid m < n\}$.

It is easy to see that $\mathcal{L} \in$ **CONSV**.

We show that $\mathcal{L} \notin$ (**C-MON** $\cup$ **C-DMON**).

Consider any machine **M** which **TxtEx**-identifies each $L \in \mathcal{L}$.

Let σ be such that content$(\sigma) = \{\langle 0, x\rangle \mid x \leq m\}$, for some $m \in N$, and $W_{\mathbf{M}(\sigma)} = L_1$. (Note that there exists such a σ since $L_1 \in$ **TxtEx**(**M**).)

Let $\sigma' \supseteq \sigma$ be such that, for some $n > m$, content$(\sigma') = \{\langle 0, x\rangle \mid x \leq m\} \cup \{\langle 1, x\rangle \mid m < x \leq n\}$ and $W_{\mathbf{M}(\sigma')} = L_2^m$. (Note that there exists such a σ' since $L_2^m \in$ **TxtEx**(**M**).)

Let $\sigma'' \supseteq \sigma'$ be such that content$(\sigma'') \subseteq L_3^{m,n}$ and $W_{\mathbf{M}(\sigma'')} = L_3^{m,n}$. (Note that there exists such a σ'' since $L_3^{m,n} \in$ **TxtEx**(**M**).)

We claim that **M** is neither monotonic nor dual-monotonic on $L_3^{m,n} \in \mathcal{L}$.

M is not monotonic on $L_3^{m,n}$ since $\langle 0, n+1\rangle \in L_3^{m,n} \cap W_{\mathbf{M}(\sigma)} \cap W_{\mathbf{M}(\sigma'')}$ but $\langle 0, n+1\rangle \notin W_{\mathbf{M}(\sigma')}$.

M is not dual-monotonic on $L_3^{m,n}$ since $\langle 1, n+1\rangle \notin L_3^{m,n} \cup W_{\mathbf{M}(\sigma)} \cup W_{\mathbf{M}(\sigma'')}$ but $\langle 1, n+1\rangle \in W_{\mathbf{M}(\sigma')}$. ▮

Corollary 13

(a) **CONSV** $-$ **SMON** $\neq \emptyset$.

(b) **CONSV** $-$ **DSMON** $\neq \emptyset$.

(c) **CONSV** $-$ **C-BMON** $\neq \emptyset$.

Theorem 14 **C-DMON** $-$ **C-MON** $\neq \emptyset$.

PROOF. Consider the following languages.

$L_j = \{\langle j, x\rangle \mid x \in N\}$.

$L_j^m = \{\langle j, x\rangle \mid x < m\}$.

Let T_j be a text for L_j such that content$(T_j[m]) = L_j^m$.

$S_j = \{\langle m, n\rangle \mid m > 0 \ \wedge \ \{\langle j, x\rangle \mid x \leq m\} \subseteq W_{\mathbf{M}_j(T_j[m]),n}\}$.

$\mathcal{L} = \{L_j \mid j \in N\} \cup \{L_j^m \mid S_j \neq \emptyset \ \wedge \ (\exists n)[\langle m, n\rangle = \min(S_j)]\}$.

Note that this $\mathcal{L}$ is slightly different from that used in the proof of Theorem 9. In this we have included all L_js!

We claim that $\mathcal{L} \in$ **C-DMON** $-$ **C-MON**.

Let G_N be a grammar for N. Let $G_\emptyset$ be a grammar for $\emptyset$. Let G_j denote a grammar for L_j which can be effectively obtained from j. Let G_j^m denote a grammar for L_j^m, which can be obtained effectively from j, m. Note that S_j is recursive. Let $S_j^s = S_j \cap \{x \mid x \leq s\}$.

$$\mathbf{M}(T[s]) = \begin{cases} G_N, & \text{if content}(T[s]) = \emptyset; \\ G_j, & \text{if content}(T[s]) \neq \emptyset \ \wedge \\ & \text{content}(T[s]) \subseteq L_j \ \wedge \\ & [S_j^s = \emptyset \ \vee \ L_j^m \not\supseteq \text{content}(T[s]), \\ & \text{where } \langle m, n \rangle = \min(S_j^s)]; \\ G_j^m, & \text{if content}(T[s]) \neq \emptyset \ \wedge \\ & \text{content}(T[s]) \subseteq L_j^m \ \wedge \\ & S_j^s \neq \emptyset \ \wedge \\ & (\exists n)[\langle m, n \rangle = \min(S_j^s)]; \\ G_\emptyset, & \text{otherwise.} \end{cases}$$

It is easy to verify that $\mathbf{M}$ is dual-monotonic on $\mathcal{L}$ and **TxtEx**-identifies each $L \in \mathcal{L}$.

Now suppose $\mathbf{M}_j$ **TxtEx**-identifies each $L \in \mathcal{L}$. We will show that $\mathbf{M}_j$ cannot be monotonic on $\mathcal{L}$.

Clearly, since $L_j \in \mathbf{TxtEx}(\mathbf{M}_j)$, we have $S_j \neq \emptyset$. Suppose $\langle m, n \rangle = \min(S_j)$.

Let σ be an extension of $T_j[m]$ such that content$(\sigma) = L_j^m$ and $W_{\mathbf{M}_j(\sigma)} = L_j^m$. (Note that there exists such a σ since $L_j^m \in \mathbf{TxtEx}(\mathbf{M}_j)$.)

Let σ' be an extension of σ such that content$(\sigma') \subseteq L_j$, and $W_{\mathbf{M}_j(\sigma')} = L_j$. (Note that there exists such a σ since $L_j \in \mathbf{TxtEx}(\mathbf{M}_j)$.)

$\mathbf{M}_j$ is not monotonic on L_j since content$(\sigma') \subseteq L_j$, $\langle j, m \rangle \in W_{\mathbf{M}_j(\sigma')} \cap W_{\mathbf{M}_j(T_j[m])}$ but $\langle j, m \rangle \notin W_{\mathbf{M}_j(\sigma)}$. ∎

Corollary 15

(a) **C-DMON** $-$ **C-BMON** $\neq \emptyset$.

(b) **C-DMON** $-$ **SMON** $\neq \emptyset$.

We now briefly consider **G-DMON**. Note that **DSMON** $\subseteq$ **G-DMON** $\subseteq$ **C-DMON**. Also,

(i) $\{L \mid (\forall i, x)[\langle i, x \rangle \in L \Leftrightarrow \langle i, 0 \rangle \in L] \ \wedge \ \text{card}(\{i \mid \langle i, 0 \rangle \in L\}) < \infty\}$ is in **SMON** but not in **G-DMON**.

(ii) a slight modification of the proof of Theorem 14, shows that **G-DMON** $-$ **C-MON** $\neq \emptyset$.

From the above, and Theorem 9, it follows that **G-DMON** is incomparable to **C-MON**, **SMON**, **WMON**, and **C-BMON**.

We now briefly consider **G-BMON**.

(i) **G-BMON** $\subseteq$ **SMON** $\cap$ **G-DMON** since, **G-MON** = **SMON**;

(ii) $\{L \mid (\forall i, x)[\langle i, x\rangle \in L \Leftrightarrow \langle i, 0\rangle \in L] \ \wedge\ \mathrm{card}(\{i \mid \langle i, 0\rangle \in L\}) < \infty\}$ is in **SMON** but not in **G-DMON** (and thus not in **G-BMON**).

(iii) $\{L \mid \mathrm{card}(L) < \infty\}$ is in **G-BMON** but not in **DSMON**.

Thus, **G-BMON** is properly contained in **SMON** and is incomparable to **DSMON**.

Theorem 16 *(**C-MON** $\cap$ **WMON**) – **C-DMON** $\neq \emptyset$.*

PROOF. Consider the following languages.

$L_1 = \{\langle 0, x\rangle \mid x \in N\}$.

$L_2^m = \{\langle 0, x\rangle \mid x \leq m\} \cup \{\langle 1, x\rangle \mid x > m\}$.

$L_3^{m,n} = \{\langle 0, x\rangle \mid x \leq m\} \cup \{\langle 1, x\rangle \mid m < x < n\} \cup \{\langle 2, n\rangle\}$.

Let $\mathcal{L} = \{L_1\} \cup \{L_2^n \mid n \in N\} \cup \{L_3^{m,n} \mid m < n\}$.

It is easy to see that $\mathcal{L} \in$ **C-MON** $\cap$ **WMON**.

We claim that $\mathcal{L} \notin$ **C-DMON**. Suppose **M** **TxtEx**-identifies each $L \in \mathcal{L}$. We will then show that **M** cannot be dual-monotonic on $\mathcal{L}$.

Let σ be such that content$(\sigma) \subseteq L_1$ and $W_{\mathbf{M}(\sigma)} = L_1$. (Note that there exists such a σ since $L_1 \in$ **TxtEx**(**M**).)

Let σ' be an extension of σ such that content$(\sigma') \subseteq L_2^m$, for some m, and $W_{\mathbf{M}(\sigma)} = L_2^m$. (Note that there exists such a σ' since $L_2^m \in$ **TxtEx**(**M**).)

Let σ'' be an extension of σ' such that content$(\sigma'') = L_3^{m,n}$, for some n, and $W_{\mathbf{M}(\sigma)} = L_3^{m,n}$. (Note that there exists such a σ'' since $L_3^{m,n} \in$ **TxtEx**(**M**).)

Now $\langle 1, n\rangle \notin L_3^{m,n} =$ content(σ''), $\langle 1, n\rangle \notin W_{\mathbf{M}(\sigma)} \cup W_{\mathbf{M}(\sigma'')}$ but $\langle 1, n\rangle \in W_{\mathbf{M}(\sigma')}$. Thus **M** is not dual-monotonic on $L_3^{m,n} \in \mathcal{L}$. ▮

Corollary 17

(a) **C-MON** – **C-BMON** $\neq \emptyset$.

(b) **C-MON** – **SMON** $\neq \emptyset$.

(c) **C-MON** – **DSMON** $\neq \emptyset$.

We now introduce a procedure, Proc, that is used in the proof of the next two theorems. $W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ is defined as follows.

```
Begin {$W_{\mathrm{Proc}(\mathbf{M},\sigma)}$}
    Let $j = \mathbf{M}(\sigma)$.
    Go to stage 0.
    Stage $s$
        Let $S = W_{j,s}$.
        **if** there exists a $\tau$ such that
            $|\tau| \leq s$,
                $\sigma \subseteq \tau$,
                content$(\tau) \subseteq S$, and
                $\mathbf{M}(\sigma) \neq \mathbf{M}(\tau)$
        **then** HALT (*i.e.*, $W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ does not enumerate anything else.)
        **else** enumerate $S$ and go to stage $s+1$.
        **endif**
    End stage $s$
End $\{W_{\mathrm{Proc}(\mathbf{M},\sigma)}\}$
```

The following lemma summarizes the properties of Proc.

Lemma 18 *(a) For all* $\mathbf{M}$*,* σ*, and* s*,* $L_s = W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ *enumerated before stage* s *can be effectively (in* $\mathbf{M}$*,* σ*, and* s*) determined.*

(b) For all $\mathbf{M}$*,* σ*, and* s *it can be effectively determined whether* $\mathrm{Proc}(\mathbf{M},\sigma)$ *halts before stage* s*.*

(c) $W_{\mathrm{Proc}(\mathbf{M},\sigma)} \subseteq W_{\mathbf{M}(\sigma)}$*.*

(d) Either $\mathrm{Proc}(\mathbf{M},\sigma)$ *halts or* $\mathrm{content}(\sigma) \not\subseteq W_{\mathbf{M}(\sigma)}$ *or* σ *is a locking sequence for* $\mathbf{M}$ *on* $W_{\mathbf{M}(\sigma)}$*.*

(e) If $\mathrm{content}(\sigma) \not\subseteq W_{\mathbf{M}(\sigma)}$*, then* $W_{\mathrm{Proc}(\mathbf{M},\sigma)} = W_{\mathbf{M}(\sigma)}$ *and* $\mathrm{Proc}(\mathbf{M},\sigma)$ *does not halt.*

(f) If $\mathrm{content}(\sigma) \subseteq W_{\mathbf{M}(\sigma)}$ *and* σ *is a locking sequence for* $\mathbf{M}$ *on* $W_{\mathbf{M}(\sigma)}$*, then* $W_{\mathrm{Proc}(\sigma)} = W_{\mathbf{M}(\sigma)}$*.*

(g) If $\mathrm{content}(\sigma) \subseteq W_{\mathbf{M}(\sigma)}$ *and* σ *is a not a locking sequence for* $\mathbf{M}$ *on* $W_{\mathbf{M}(\sigma)}$*, then* $\mathrm{Proc}(\mathbf{M},\sigma)$ *halts (and thus* $W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ *is finite).*

(h) $W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ *enumerated before stage* s*, is contained in* $W_{\mathbf{M}(\sigma),s-1}$*. Thus, if* $\mathrm{Proc}(\mathbf{M},\sigma)$ *halts in stage* s*, then* $W_{\mathrm{Proc}(\mathbf{M},\sigma)} \subseteq W_{\mathbf{M}(\sigma),s-1}$*.*

(i) Suppose $\sigma \subseteq \tau$*,* $\mathrm{content}(\tau) \subseteq W_{\mathbf{M}(\sigma),s}$ *and* $\mathbf{M}(\sigma) \neq \mathbf{M}(\tau)$*. Then* $\mathrm{Proc}(\mathbf{M},\sigma)$ *halts at or before stage* $\max(\{|\tau|, s\})$*. Moreover, either* $\mathrm{content}(\tau)$ *is contained in* $W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ *enumerated before stage* $|\tau|$ *or* $\mathrm{content}(\tau) \not\subseteq W_{\mathrm{Proc}(\mathbf{M},\sigma)}$*.*

(j) For all σ*,* τ*, and* $\mathbf{M}$ *such that* $\sigma \subseteq \tau$ *and* $\mathbf{M}(\sigma) \neq \mathbf{M}(\tau)$*, either* $\mathrm{content}(\tau) \not\subseteq W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ *or* $\mathrm{content}(\tau) \subseteq W_{\mathrm{Proc}(\mathbf{M},\sigma)}$ *enumerated before stage* $|\tau|$*.*

PROOF. (a) to (h) are easy to see from the definition of Proc. For (i) suppose the hypothesis. Clearly, at or before stage $\max(\{s, |\tau|\})$, the procedure for $\mathrm{Proc}(\mathbf{M},\sigma)$ would detect this mind change and halt. The second clause in the conclusion now follows using part (h). Part (j) follows using parts (c) and (i). ∎

We now consider the following two important simulation results that use Proc and the above properties of Proc (Lemma 18).

Theorem 19 **TxtEx** = **G-DWMON**.

PROOF. Clearly, **G-DWMON** ⊆ **TxtEx**. We show that **TxtEx** ⊆ **G-DWMON**. Suppose $\mathbf{M}$ is given. We assume without loss of generality that $\mathbf{M}$ is such that for every $L \in \mathbf{TxtEx}(\mathbf{M})$, each text T for L has a locking sequence for $\mathbf{M}$ on L as a prefix (Lemma 2).

We define a machine $\mathbf{M}'$ on initial sequences of a text T as follows. Together with $\mathbf{M}'$ we also define a function X. Intuitively, X just keeps track of the last point n' in the text T, where $Proc(\mathbf{M}, T[n'])$ was output by $\mathbf{M}'$.

```
Begin {$\mathbf{M}'(T[n])$, $X(T[n])$}
0.  **if** $n = 0$, **then**
    1.  Let $X(T[n]) = 0$.
    2.  Let $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[n])$.
    **else**
    3.  Let $n' = X(T[n-1])$ (note that $n'$ is such that the last $\text{Proc}(\mathbf{M}, T[s])$ considered
          by $\mathbf{M}'$ on initial segments of $T[n]$ was for $s = n'$.)
    4.  **if** $\text{Proc}(\mathbf{M}, T[n'])$ halts in $\leq n$ stages
        **then**
        5.  **if** $\text{content}(T[n]) \not\subseteq W_{\text{Proc}(\mathbf{M},T[n'])}$,
            **then**
            6.  Let $X(T[n]) = n$.
            7.  Let $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[n])$.
            **else**
            8.  Let $X(T[n]) = n'$.
            9.  Let $\mathbf{M}'(T[n])$ be a grammar (obtained effectively from $\text{content}(T[n])$)
                  for $\text{content}(T[n])$.
            **endif**
    10. **elseif** $\mathbf{M}(T[n']) \neq \mathbf{M}(T[n])$
        **then**
        11. **if** $\text{content}(T[n]) \not\subseteq W_{\text{Proc}(\mathbf{M},T[n'])}$ enumerated by stage $n+1$
            **then**
            12. Let $X(T[n]) = n$.
            13. Let $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[n])$.
            **else**
            14. Let $X(T[n]) = n'$.
            15. Let $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[n'])$.
            **endif**
        **else**
        16. Let $X(T[n]) = n'$.
        17. Let $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[n'])$.
        **endif**
    **endif**
End {$\mathbf{M}'(T[n])$, $X(T[n])$}
```

We claim that $\mathbf{M}'$ is dual-weak-monotonic and **TxtEx**-identifies each L in $\mathbf{TxtEx}(\mathbf{M})$. First note that $X(T[n]) \leq n$ and $X(T[n+1]) \geq X(T[n])$. Also note that $X(T[n]) = n \Leftrightarrow [n = 0 \ \vee \ X(T[n]) \neq X(T[n-1])]$. Furthermore, if $X(T[n+1]) \neq X(T[n])$, then for all $n' \leq n$, $\text{content}(T[n+1]) \not\subseteq W_{\text{Proc}(\mathbf{M},T[X(T[n'])])}$. To see this, note that if $X(T[n+1]) \neq X(T[n])$, then this is because of the execution of step 6 or 12 above for $\mathbf{M}'(T[n+1])$, which can happen only if $\text{content}(T[n+1]) \not\subseteq W_{\text{Proc}(\mathbf{M},T[n'])}$, where $n' = X(T[n])$ (using Lemma 18 and the success of the corresponding If conditions). Also, if $X(T[n]) = n'$, then $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[n'])$ or $W_{\mathbf{M}'(T[n])} = \text{content}(T[n]) \subseteq W_{\text{Proc}(\mathbf{M},T[n'])}$. Hence, $\mathbf{M}'$ is dual-weak-monotonic on each $L \subseteq N$.

Thus, it only remains to prove that $\mathbf{M}'$ **TxtEx**-identifies every language **TxtEx**-identified by $\mathbf{M}$. For this suppose T is a text for $L \in \mathbf{TxtEx}(\mathbf{M})$. Note that since $\mathbf{M}$ satisfies the condition in Lemma 2, T contains a locking sequence for $\mathbf{M}$ on L. Thus, $\lim_{n\to\infty} X(T[n])$ converges. Let $t = \lim_{n\to\infty} X(T[n])$. Now consider the following two cases.

Case 1: $\text{Proc}(\mathbf{M}, T[t])$ does not halt.

In this case, consider $\mathbf{M}'(T[n])$ for $n > t$. Clearly, if clause at step 4 does not hold. We claim that if clause at step 10 also cannot hold. To see this suppose otherwise. Thus, using Lemma 18 (i), if clause at step 11 must also hold (since $\text{Proc}(\mathbf{M}, T[t])$ does not halt). But then $X(T[n]) \neq X(T[t])$. It follows that for all $n > t$, $\mathbf{M}(T[n]) = \mathbf{M}(T[t])$ and $\mathbf{M}'(T[n]) = \text{Proc}(\mathbf{M}, T[t])$. Also, by Lemma 18(parts (d), (e), (f)) $W_{\text{Proc}(\mathbf{M},T[t])} = W_{\mathbf{M}(T[t])} = L$. Thus $\mathbf{M}'$ **TxtEx**-identifies L.

Case 2: $\text{Proc}(\mathbf{M}, T[t])$ halts.

In this case for all $n > t$, in $\mathbf{M}'(T[n])$, if clause at step 4 succeeds and the if clause at step 5 fails. It follows that L is finite and $\mathbf{M}'$ outputs, in the limit on T, a grammar for $\text{content}(T)$. Thus $\mathbf{M}'$ **TxtEx**-identifies L. ∎

Remark: A careful analysis of the above proof reveals that it *almost* shows that $\mathbf{TxtEx} \subseteq \mathbf{CONSV}$; the only place the dual-weak-monotonicity (instead of conservativeness) is used is at Step 9. Hence if we modified Step 9 in the above proof to simply output $\text{Proc}(\mathbf{M}, T[n'])$ then $\mathbf{M}'$ becomes a conservative machine that identifies every infinite language identified by $\mathbf{M}$ (but maybe unsuccessful on finite languages). Hence, this shows that if attention is restricted to infinite languages, conservative machines are as powerful as general machines. We summarize this observation in the following corollary. Let $\mathcal{E}_\infty$ denote the collection of all infinite r.e. languages.

Corollary 20 $(\forall \mathcal{L} \subseteq \mathcal{E}_\infty)[\mathcal{L} \in \mathbf{CONSV} \iff \mathcal{L} \in \mathbf{TxtEx}]$.

Theorem 21 $\mathbf{C\text{-}WMON} \subseteq \mathbf{G\text{-}CONSV}$.

PROOF. Proof of this theorem is similar to that of Theorem 19. There are two issues one needs to address in the simulation as done by $\mathbf{M}'$ in Theorem 19. First in step 9, one cannot output a grammar for content($T[n]$), because that could violate conservativeness; hence, one outputs Proc($\mathbf{M}, T[n']$) (this is fine since if $\mathbf{M}$ is weak monotonic then it cannot identify any proper subset of $W_{\mathrm{Proc}(\mathbf{M},T[n'])}$.) The second difference arises due to the fact that for identification by a weak-monotonic machine one may not be able to assume that every text contains a locking sequence. Thus in steps 6,7 one needs to do some rearrangement of the input text (this is because one needs to argue that if X does not converge on (rearranged) text T, then so does $\mathbf{M}$.) Rest of the proof is similar to the proof of Theorem 19. We now proceed to give the details. For simplicity of presentation we give a somewhat different description of machine $\mathbf{M}'$ as compared to that corresponding description in the proof of Theorem 19.

We define $\mathbf{M}'$ as follows. Along with $\mathbf{M}'$ we define a function X and a function rearrange. Intuitively, X just keeps track of the "last $Proc(\mathbf{M}, T[n'])$" output by $\mathbf{M}'$ and rearrange is used for the rearrangement of the input text as hinted above. We define rearrange only for the cases when $X(T[n]) = n$ (note that $X(T[n]) = n \Leftrightarrow n = 0 \vee X(T[n]) \neq X(T[n-1])$). Also note that rearrange($T[n]$), if defined, is a rearrangement of $T[n]$.

```
Begin $\mathbf{M}'(T[n])$, $X(T[n])$, rearrange($T[n]$).
0.  **if** $n = 0$, **then**
            Let $X(T[n]) = 0$.
            Let rearrange($T[n]$) = $T[n]$.
            Let $\mathbf{M}'(T[n]) = \mathrm{Proc}(\mathbf{M}, T[n])$.
    **else**
1.          Let $n' = X(T[n-1])$ (note that $n'$ is such that the last Proc($\mathbf{M}$, rearrange($T[s]$))
              considered by $\mathbf{M}'$ on initial segments of $T[n]$ was for $s = n'$.)
2.          **if** there exists a $\sigma \supseteq$ rearrange($T[n']$) such that
                    content($\sigma$) $\subseteq$ content($T[n]$) $\wedge$
                    $|\sigma| \leq 2n + |\mathrm{rearrange}(T[n'])|$ $\wedge$
                    $\mathbf{M}(\sigma) \neq \mathbf{M}(\mathrm{rearrange}(T[n']))$ $\wedge$
                    content($\sigma$) $\not\subseteq W_{\mathrm{Proc}(\mathbf{M},\mathrm{rearrange}(T[n']))}$ enumerated till stage $|\sigma| + 1$.
            **then**
            2.1.    Let $\tau$ denote one such $\sigma$. Let $\tau'$ be an extension of $\tau$ such that content($\tau'$) =
                      content($T[n]$).
            2.2.    Let $X(T[n]) = n$.
            2.3.    Let rearrange($T[n]$) = $\tau'$.
            2.4.    Let $\mathbf{M}'(T[n]) = \mathrm{Proc}(\mathbf{M}, \tau')$.
3.          **else**
            3.1.    Let $X(T[n]) = n'$.
            3.2.    Let $\mathbf{M}'(T[n]) = \mathrm{Proc}(\mathbf{M}, \mathrm{rearrange}(T[n']))$.
            **endif**
    endif
End $\mathbf{M}'(T[n])$, $X(T[n])$, rearrange$(T[n])$.
```

We first show that $\mathbf{M}'$ is conservative. We will then show that $\mathbf{M}'$ **TxtEx**-identifies every language L such that $L \in \mathbf{TxtEx}(\mathbf{M})$ and $\mathbf{M}$ is weak-monotonic on L. This will complete the proof.

First note that $X(T[n]) \leq n$ and $X(T[n+1]) \geq X(T[n])$. Also note that $X(T[n]) = n \Leftrightarrow [n = 0 \ \vee \ X(T[n]) \neq X(T[n-1])]$. Also note that, for n such that $X(T[n]) = n$, rearrange$(T[n])$ is defined and is a rearrangement of $T[n]$. Also, the only conjectures output by $\mathbf{M}'$ are of the form $\text{Proc}(\mathbf{M}, \text{rearrange}(T[n]))$, such that $X(T[n]) = n$. Also $\mathbf{M}'(T[n+1]) \neq \mathbf{M}'(T[n]) \Leftrightarrow X(T[n+1]) = n+1$.

Furthermore, $\mathbf{M}'(T[n+1]) \neq \mathbf{M}'(T[n])$ (equivalently, $X(T[n+1]) \neq X(T[n])$) implies that, $T[n+1] \not\subseteq W_{\text{Proc}(\mathbf{M},\text{rearrange}(T[n']))}$, where $n' = X(T[n])$. To see this note that if $X(T[n+1]) \neq X(T[n])$, then this is because of the execution of step 2.2. But then due to the success of the If condition at step 2, we have that $T[n+1] \not\subseteq W_{\text{Proc}(\mathbf{M},\text{rearrange}(T[n']))}$, where $n' = X(T[n])$ (using Lemma 18, and the requirement for if condition to succeed). Thus $\mathbf{M}'$ behaves conservatively on all $L \subseteq N$.

Thus it only remains to prove that $\mathbf{M}'$ **TxtEx**-identifies every language L such that $L \in \mathbf{TxtEx}(\mathbf{M})$ and $\mathbf{M}$ is weak-monotonic on L. For this suppose T is a text for $L \in \mathbf{TxtEx}(\mathbf{M})$. We first claim that $\lim_{n\to\infty} X(T[n])$ converges. To see this suppose otherwise. Let T' be the text $\bigcup_{X(T[t+1]) \neq X(T[t])} \text{rearrange}(T[t+1])$. Note that content$(T')$ = content(T), and that $\mathbf{M}$ on the text T' changes its mind infinitely often (since $X(T[n+1]) \neq X(T[n])$ implies that, there exists a σ, such that for $n' = X(T[n])$, rearrange$(T[n']) \subseteq \sigma \subseteq$ rearrange$(T[n+1])$ such that $\mathbf{M}(\sigma) \neq \mathbf{M}(\text{rearrange}(T[n']))$). A contradiction. Thus $\lim_{n\to\infty} X(T[n])$ converges. Let $t = \lim_{n\to\infty} X(T[n])$. Note that $\mathbf{M}'(T)\!\downarrow = \text{Proc}(\mathbf{M}, \text{rearrange}(T[t]))$.

Now since the if at step 2 for $\mathbf{M}'(T[n])$, does not succeed for any $n > t$, we have that rearrange$(T[t])$ is a locking sequence for $\mathbf{M}$ on L, or $L \subseteq W_{\text{Proc}(\mathbf{M},\text{rearrange}(T[t]))}$. In the former case clearly, $L = W_{\text{Proc}(\mathbf{M},\text{rearrange}(T[t]))}$. In the later case, since $W_{\text{Proc}(\mathbf{M},\text{rearrange}(T[t]))} \subseteq W_{\mathbf{M}(\text{rearrange}(T[t]))}$, it follows that $L \subseteq W_{\mathbf{M}(\text{rearrange}(T[t]))}$, and thus by the weak-monotonicity condition we have that $L = W_{\mathbf{M}(\text{rearrange}(T[t]))}$. Thus $L = W_{\text{Proc}(\mathbf{M},\text{rearrange}(T[t]))}$.

It follows that $\mathbf{M}'$ **TxtEx**-identifies L. ∎

The above proof together with Theorem 6 implies the following corollary which says that the global and the class versions of weak-monotonicity and conservatism are equivalent.

Corollary 22 **G-WMON = C-WMON = G-BWMON = C-BWMON = G-CONSV = C-CONSV**.

Finally, we consider an alternative formulation of the notion of conservative strategy. Angluin's notion of conservatism requires the learner to conjecture the same grammar unless the data presented includes a counterexample to the current hypothesis. We relax this requirement as follows: If the learner encounters an element that is not explained by the current hypothesis then the learner can change its hypothesis, otherwise the learner must output a grammar that is extensionally equivalent to the current hypothesis. The following definition formally introduces this strategy, referred to as *extensional conservatism*.

Definition 14 (a) $\mathbf{M}$ is *extensionally-conservative* on L just in case $(\forall \sigma, \tau \mid \sigma \subseteq \tau \ \wedge \ \mathrm{content}(\tau) \subseteq L)[\mathbf{M}(\sigma) = \perp \ \vee \ [\mathrm{content}(\tau) \subseteq W_{\mathbf{M}(\sigma)} \Rightarrow W_{\mathbf{M}(\sigma)} = W_{\mathbf{M}(\tau)}]]$.

(b) $\mathbf{M}$ is *extensionally-conservative* on $\mathcal{L}$ just in case $\mathbf{M}$ is extensionally-conservative on each $L \in \mathcal{L}$.

(c) $\mathbf{M}$ is *extensionally-conservative* just in case $\mathbf{M}$ is extensionally-conservative on each $L \subseteq N$.

(d) **G-EXT-CONSV** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is extensionally-conservative and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

(e) **C-EXT-CONSV** $= \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ is extensionally-conservative on $\mathcal{L}$ and $\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

It is easy to verify that **G-CONSV** $\subseteq$ **G-EXT-CONSV** $\subseteq$ **C-EXT-CONSV** $\subseteq$ **C-WMON**. But, since **G-CONSV** $=$ **C-WMON** (Theorem 21), the extensional notion of conservatism turns out to be equivalent to the intensional notion of conservatism. The next corollary summarizes these observations.

Corollary 23 G-CONSV = C-CONSV = G-EXT-CONSV = C-EXT-CONSV = WMON = C-WMON = G-BWMON = C-BWMON.

The results presented in this paper are pictorially presented in Figure 1.

5 Acknowledgements

We are pleased to acknowledge the influence of the work of Angluin, Arikawa, Jantke, Kapur, Kinber, Lange, Mukouchi, Wiehagen, and Zeugmann on the present investigation. We also acknowledge helpful comments from Steffen Lange, Rolf Wiehagen, and Thomas Zeugmann. We thank the referees for several helpful comments. In particular, Theorem 11 was pointed out to us by one of the referees. Recently Kinber and Stephan [KS95] have independently obtained some of the results in this paper.

References

[Ang80] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

[Ful90] M. Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[Jan91] K. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.

[JS89] S. Jain and A. Sharma. Recursion theoretic characterizations of language learning. Technical Report 281, University of Rochester, 1989.

[JS94] S. Jain and A. Sharma. Characterizing language learning by standardizing operations. *Journal of Computer and System Sciences*, 49(1):96–107, 1994.

[JS97] S. Jain and A. Sharma. Characterizing language learning in terms of computable numberings. *Annals of Pure and Applied Logic*, 84(1):51–72, 1997. Special issue on Asian Logic Conference, 1993.

[Kap92] S. Kapur. Monotonic language learning. In S. Doshita, K. Furukawa, K. Jantke, and T. Nishida, editors, *Algorithmic Learning Theory: Third International Workshop (ALT '92)*, volume 743 of *Lecture Notes in Artificial Intelligence*, pages 147–158. Springer-Verlag, 1992.

[Kap93] S. Kapur. Uniform characterizations of various kinds of language learning. In *Algorithmic Learning Theory: Fourth International Workshop (ALT '93)*, volume 744 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1993.

[KB92] S. Kapur and G. Bilardi. Language learning without overgeneralization. In *Proceedings of the Ninth Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science.* Springer-Verlag, 1992.

[Kin94] E. Kinber. Monotonicity versus efficiency for learning languages from texts. Technical Report 94-22, Department of Computer and Information Sciences, University of Delaware, 1994.

[KS95] E. Kinber and F. Stephan. Language learning from texts: Mind changes, limited memory and monotonicity. *Information and Computation*, 123:224–241, 1995.

[LZ92a] S. Lange and T. Zeugmann. Monotonic language learning on informant. Technical Report 11/92, GOSLER-Report, FB Mathematik und Informatik, TH Lepzig, 1992.

[LZ92b] S. Lange and T. Zeugmann. Types of monotonic language learning and their characterization. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 377–390. ACM Press, 1992.

[LZ93a] S. Lange and T. Zeugmann. Language learning with bounded number of mind changes. In *Proceedings of the Tenth Annual Symposium on Theoretical Aspects of Computer Science*, pages 682–691. Springer-Verlag, 1993. Lecture Notes Computer Science, 665.

[LZ93b] S. Lange and T. Zeugmann. Monotonic versus non-monotonic language learning. In *Proceedings of the Second International Workshop on Nonmonotonic and Inductive Logic*, volume 659 of *Lecture Notes in Artificial Intelligence*, pages 254–269. Springer-Verlag, 1993.

[LZ93c] S. Lange and T. Zeugmann. On the impact of order independence to the learnability of recursive languages. Technical Report ISIS-RR-93-17E, Institute for Social Information Science Research Report, Fujitsu Laboratories Ltd., 1993.

[LZK92] S. Lange, T. Zeugmann, and S. Kapur. Class preserving monotonic language learning. Technical Report 14/92, GOSLER-Report, FB Mathematik und Informatik, TH Lepzig, 1992.

[LZK96] S. Lange, T. Zeugmann, and S. Kapur. Monotonic and dual monotonic language learning. *Theoretical Computer Science A*, 155:365–410, 1996.

[MA93] Y. Mukouchi and S. Arikawa. Inductive inference machines that can refute hypothesis spaces. In K.P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Algorithmic Learning Theory: Fourth International Workshop (ALT '93)*, volume 744 of *Lecture Notes in Artificial Intelligence*, pages 123–136. Springer-Verlag, 1993.

[MF90] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Workshop on Algorithmic Learning Theory*, pages 368–381. Ohmsa Publishers, 1990. Reprinted by Ohmsa Springer-Verlag.

[Muk92a] Y. Mukouchi. Characterization of finite identification. In K. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the Third International Workshop*, pages 260–267, 1992.

[Muk92b] Y. Mukouchi. Inductive inference with bounded mind changes. In S. Doshita, K. Furukawa, K. Jantke, and T. Nishida, editors, *Algorithmic Learning Theory: Third International Workshop (ALT '92)*, volume 743 of *Lecture Notes in Artificial Intelligence*, pages 125–134. Springer-Verlag, 1992.

[MY78] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms.* North Holland, New York, 1978.

[OSW82] D. Osherson, M. Stob, and S. Weinstein. Learning strategies. *Information and Control*, 53:32–51, 1982.

[OSW86] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press, 1986.

[Qui90] J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Rog58] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967. Reprinted, MIT Press 1987.

[Sha81] E. Shapiro. Inductive inference of theories from facts. Technical Report 192, Computer Science Department, Yale University, 1981.

[Wie90] R. Wiehagen. A thesis in inductive inference. In J. Dix, K. Jantke, and P. Schmitt, editors, *Nonmonotonic and Inductive Logic, 1st International Workshop*, volume 543 of *Lecture Notes in Artificial Intelligence*, pages 184–207. Springer-Verlag, 1990.

[WL76] R. Wiehagen and W. Liepe. Charakteristische eigenschaften von erkennbaren klassen rekursiver funktionen. *Electronische Informationverarbeitung und Kybernetik*, 12:421–438, 1976.