Sofia University St. Kliment Ohridski
Faculty of mathematics and informatics

SOFIA UNIVERSITY
ST. KLIMENT OHRIDSKI

Some Basic types of Structural Learnability

Name: Monika Ilieva
Faculty number: 25930
Supervisor: Prof. PhD A. Soskova

2023

Prologue

Learnability as a notion in computable theory was introduced by Gold, [Gol67], who defined the first type of learning - learnability in the limit. A Turing machine, fed only with examples of a language, is expected after some finite amount of time (and mind changes) to "prove" its knowledge by giving a correct description of the language. Later other scientists in the field, influenced by Gold, started to use as model the behaviour of a child, who learns a language, and started to adapt that behaviour into the field of computable learnability.

A lot of results about learnability types are considered in the case of grammars, or r.e. functions, whereas we will consider learnability of structures. Couple of results of learnability of structures has emerged. Nikolay Bazhenov, Ekaterina Fokina and Luca San Mauro, in [FKSM19], showed which classes of equivalence structures are InfEx-learnable(or learnable in the limit, when information presentation contains positive and negative information about a structure). Later they, in [BFM20], extended their result to arbitrary classes of computable structures, showing that the InfEx-learnable classes are one which can be differentiated by their Σ_2^{inf}-theories.

In this Master thesis our goal will be to present couple of learnability types and see when a learner (a Turing machine), when fed with information about a structure from a given class, will reach a correct conclusion about the structure. A conclusion for us will mean an index for that structure. Interesting learnability types occur when constraints on the learnability process are presented. As children behave differently, when faced with various challenges trough their learning process, researchers are interested to explore which of those "challenges" weakness or strenghtens the learnability power.

This Master thesis provides some introductionary concepts of structural learnability. First we will define two ways of providing information to a learner. We will be interested in learnbility from text(see def. 2.1.1), and learnability from informant(see def. 2.1.2). When only positive information (or elements from a structure) is given to a learner, the information presentation is said to be from text, whereas when giving a positive and negative information to a learner (or elements from a structure and its complement), the information presentation is said to be from informant. After that we turn our attention to learnability types, based on the number of mind changes allowed to a learner. Those types expand the notion of learnability in the limit, presented by Gold, adding two new notions - finite learnability and behaviourly correct learnability. Here we will present Finite(see def. 2.3.1), Explanatory(see def. 2.3.2), and Behaviourly correct (see def. 2.3.3) learning. In Finite learning, a learner is expected to make conclusion once, whereas an Explanatory one can change its mind finitely many times, before a correct guess is reached. A BC-learner is similar to Ex, with the difference that it can provide more than one index for a c.e. set of an atomic diagram. Next we turn attention to types of learnability on which a constraint over the relation of made conjectures are presented. We will consider Strong-monotonic learnability(see def. 2.3.10), Monotonic learn-

ability(see def. 2.3.10) and Weak-monotonic learnability(see def. 2.3.10). A Strong-monotonic learner is expected to strictly improve its learnability conclusions over time. Monotonic learners can conclude indecies for sets which have some exceptions, but those exceptions are expected to dissapear over time. A Weak-monotonic learner behaves as a strong monotonic one, as long as the made conclusion doesn't contradict the presented information to the learner. After that our focus will turn to the arrangment of data given to a learner. When we are not interested from the order of the data of a specific structure, the learnability type is called Set-driven(see def. 2.3.11), whereas when we add only the length of the data as constraint, the new criteria is called Rearrangment-independent learning(see def. 2.3.12). We introduce three notions of different hypothesis spaces in the case of arrangment of data. We consider Exact learnability, Class-preserving learnability and Class-comprising learnability(see def. 2.1.4). In the case of exact learning we expect that the hypothesis space coincides with the class being learnt, whereas in the class-preserving the hypothesis space is expected to contain the class under consideration with some suitably chosen enumeration. In the last case the hypothesis space contains at least the class under consideration, with some appropriate enumeration. Next we move on memory limitation and we show that finite learners have the same learnability power as memory limited learners. Lastly we will consider learnability type which show the weakness of behaviourly correct learning. We will consider a type called Non-U-Shaped learnability(see def. 2.3.14), where a learner is forbidden of going trough a correct-incorrect-correct behaviour.

The Master thesis is divided in two parts:
I part gives some basic notions of computable structure theory. We will look at what a structure is, and how two (or more) structures can be represented by one another. We will present the notion of Atomic diagram of a structure. We will look at some notions concerned with relations and connect the notion of computable structure theory with relations and then structures. We will give the notion of jump of a structure, and finally we will define the theory of a structure.
II part will present some learnability types, where the focus will be on learnability of structures. Here we will define the notion of learnability, present some learnbility types and make a comparisson between them.

Contents

1 Basic notions **1**
1.1 Computable structure theory 1
1.1.1 Presentations 2
1.1.2 Relations 3
1.1.3 Relativization 7
1.1.4 Jump of a structure 7
1.1.5 Theory of a structure 8

2 Learnability **9**
2.1 Basic notions 10
2.2 An Example 13
2.3 Types of learnability 14
2.3.1 [Finite, Explanatory, Behaviourly Correct] 14
2.3.2 [Monotonic, Strong-monotonic, Weak-monotonic] 26
2.3.3 [Set-driven, Rearrangment independent] 37
2.3.4 [Memory limitation] 50
2.3.5 [NUShapedBC] 53

References **57**

1 Basic notions

1.1 Computable structure theory

Our first task will be to give some basic notions from mathematical logic. When using $\langle . \rangle$ brackets we will denote an ordered n-tuple, for some $n \in \omega$, while when using $\lceil . \rceil$ we will mean a Gödel number for the content under consideration.

Definition 1.1.1. *(Signature) A signature (or a language) L is an n-ordered tuple, for $n \in \omega$, where $L = \langle \boldsymbol{Func}, \boldsymbol{Const}, \boldsymbol{Pred}, \# \rangle$, and*

- ***Func** is a set of function symbols,*
- ***Const** is a set of constants,*
- ***Pred** is a set of predicate symbols,*
- *$\#$ is an arity function, s.t. $\# : \boldsymbol{Func} \cup \boldsymbol{Pred} \to \omega$. The arity function, for a given functional or predicate symbol, returns the number of arguments which the function or predicate accepts.*

Definition 1.1.2. *(Structure) Let $L = \langle \{f_i\}_{i \in \omega}, \{c_i\}_{i \in \omega}, \{P_i\}_{i \in \omega} \rangle$ be a language. A structure $\mathfrak{A}$ with language L, is defined as a tuple*

$$\mathfrak{A} = \langle \mathbb{A}, \{f_i^{\mathfrak{A}}\}_{i \in \omega}, \{c_i^{\mathfrak{A}}\}_{i \in \omega}, \{P_i^{\mathfrak{A}}\}_{i \in \omega} \rangle,$$

where:

- *$\mathbb{A} \neq \emptyset$, is a non-empty set, called an universum(domain) of $\mathfrak{A}$,*
- *Each $f_i^{\mathfrak{A}}, c_i^{\mathfrak{A}}, P_i^{\mathfrak{A}} \in L$ is interpreted in the universum of $\mathfrak{A}$*

Structures will be denoted by $\mathfrak{A}, \mathfrak{B}$, and etc. When universum of a structure is not specified, we will use their respective uppercase bolded letter symbols in latin alphabet. For example for a structure $\mathfrak{A}$ we will use the symbol $\mathbb{A}$, for a structure $\mathfrak{B}$ we will use $\mathbb{B}$, and etc.

Definition 1.1.3. *(Substructure) Let $\mathfrak{A}$ be a structure with language L. A substructure of $\mathfrak{A}$ is a structure $\mathfrak{B}$, s.t.:*

- $\mathbb{B} \subseteq \mathbb{A}$
- *for each function f, predicate p or constant c, we have that*
  - $f^{\mathfrak{B}} \subseteq f^{\mathfrak{A}}$
  - $c^{\mathfrak{B}} = c^{\mathfrak{A}}$
  - $p^{\mathfrak{B}} \subseteq p^{\mathfrak{A}}$

1.1.1 Presentations

A way by which we can find a connection between two structures is known as isomorphism.

Definition 1.1.4. *(Isomorphism) Let $\mathfrak{A}, \mathfrak{B}$ be two structures with language L. We say that $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$ if there is a bijection h: $\mathbb{A} \to \mathbb{B}$, s.t. :*

- *For each function $f \in L$, and for each n-tuple $\bar{a} \in \mathbb{A}^n$, we have that $h(f^{\mathfrak{A}}(\bar{a})) = f^{\mathfrak{B}}(h(\bar{a}))$.*
- *For each constant $c \in L$, we have that $h(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$.*
- *For each predicate $p \in L$, and for each n-tuple $\bar{a} \in \mathbb{A}^n$, we have that $\bar{a} \in p^{\mathfrak{A}} \iff \langle h(a_1), h(a_2), ..., h(a_n)\rangle \in p^{\mathfrak{B}}$.*

To denote isomorphism between two structures we will use the relation $\cong$, i.e. if $\mathfrak{A}, \mathfrak{B}$ are isomorphic structures, that will be denoted by $\mathfrak{A} \cong \mathfrak{B}$. An isomorphism between two structures is not necessary computable. If one of the structures is not computable, then there is no computable way by which we can go from one of those structures to the other.

By $\mathbb{O}, \mathbb{E}$ we will denote the set of odd, and even numbers respectively.

Example 1.1.1. *Let $L = \{\leq\}$ be a language. Let $\mathfrak{A}, \mathfrak{B}$ be two structures with language L. Let $\mathfrak{A}$ has universum the set $\mathbb{O}$, and let $\mathfrak{B}$ has universum the set $\mathbb{E}$. We define a bijective function from $\mathfrak{A}$ to $\mathfrak{B}$ as h: $\mathbb{O} \to \mathbb{E}$, where for a number $o \in \mathbb{O}$, $h(o) = o - 1$. Then $h^{-1}(e) = e + 1$, for $e \in \mathbb{E}$. The given function is clearly an isomoprhism.*

- *For each two numbers $o_1, o_2 \in \mathbb{O}$, where $o_1 \neq o_2$, we have that $h(o_1) \neq h(o_2)$, which gives us that h is injective.*
- *For each number $e \in \mathbb{E}$, exists a number $o \in \mathbb{O}$, s.t. $h(o) = e$, which gives us that h is surjective.*

As we are going to work with structures with domain ω a question which arises is how to present a structure with a different countable universum to a structure with domain ω.

Definition 1.1.5. *(ω-presentation) $\mathfrak{A}$ be a countable structure with language L. A structure $\mathfrak{B}$ with language L and domain ω is said to be an ω-presentation of $\mathfrak{A}$, if $\mathfrak{A} \cong \mathfrak{B}$.*

Remark: We will suppose that we are going to work only with relational structures. A relational structure is one in which functions are represented by their graphs.

We need a way by which a structure can be represented computationally and by which we can make conclusions about its computable properties. The notion

which will be used troughout this Master thesis is known as atomic diagram. An atomic diagram of a structure is a set, which contains information about which atomic formulae (or negations of atomic formulae) are true in that structure. The convention which we will use here, will represent the atomic diagram for a structure $\mathfrak{A}$ as a set containing codes, where each code represents an index of an atomic formulae or negation of it from the language of $\mathfrak{A}$ and arguments which make that formulae true in $\mathfrak{A}$.
Let $\mathfrak{A}$ be a structure with language L. Let $\{\phi_e^{at}\}_{e\in\omega}$ be an effective enumeration of all atomic formulae of L, with free variables from the list $\{x_0, x_1, x_2, ...\}$.

Definition 1.1.6. *(Atomic diagram) An atomic diagram for a structure $\mathfrak{A}$ with language L is defined as a subset of ω,*

$$\mathfrak{D}_{\mathfrak{A}} = \{\ulcorner e, \bar{a_n}, 1\urcorner \mid \mathfrak{A} \models \phi_e^{at}[x_j \to j : j \in \bar{a_n}], \bar{a_n} \in \omega^n\} \cup$$
$$\{\ulcorner e, \bar{a_n}, 0\urcorner \mid \mathfrak{A} \not\models \phi_e^{at}[x_j \to j : j \in \bar{a_n}], \bar{a_n} \in \omega^n\}.$$

The atomic diagram for a structure $\mathfrak{A}$ will be denoted by $\mathfrak{D}_{\mathfrak{A}}$. The complexity of a structure $\mathfrak{A}$ will be determined from the complexity of $\mathfrak{D}_{\mathfrak{A}}$, i.e. a structure $\mathfrak{A}$ is computable, if its atomic diagram is. A positive atomic diagram is defined as $\mathfrak{D}_{\mathfrak{A}}^{+} = \{\ulcorner e, \bar{a_n}\urcorner \mid \mathfrak{A} \models \phi_e^{at}[x_j \to j : j \in \bar{a_n}], \bar{a_n} \in \omega^n\}$.

In order to provide a finite information about a structure we need two notions.

Definition 1.1.7. *(Finite approximation of a structure, [Mon21]) Let $\mathfrak{A}$ be a structure with language L. The sequence $\{\mathfrak{A}_s : s \in \omega\}$, of finite substructures of $\mathfrak{A}$, is called a finite approximation of the structure $\mathfrak{A}$.*

Because we are interested in presenting a structure computationally, we will connect the approximation of a structure to approximation of its atomic diagram.
By $\mid \mathbb{A} \mid$ we will denote the size of a set $\mathbb{A} \subseteq \omega$. If $\mathbb{A}$ is finite, then there is $n \in \omega$, s.t. $\mid \mathbb{A} \mid = n$, otherwise $\mid \mathbb{A} \mid = \infty$.

Definition 1.1.8. *(Approximation of Atomic diagram) Llet $\mathfrak{A}$ be a structure with language L. Let $\{\mathfrak{A}_s : \mathfrak{A}_s \subseteq \mathfrak{A}\}$ be an approximation of the structure $\mathfrak{A}$. Approximation of the atomic diagram $\mathfrak{D}_{\mathfrak{A}}$ of $\mathfrak{A}$ is defined as a set of codes $\{\mathfrak{D}_{\mathfrak{A}_s} : \mathfrak{D}_{\mathfrak{A}_s} \subseteq \mathfrak{D}_{\mathfrak{A}}\}$.*

In other words, each $\mathfrak{D}_{\mathfrak{A}_s}$ for a structure $\mathfrak{A}$, contains information about all atomic formulae with indecies $\leq s$ and are true for numbers $\leq s$.

1.1.2 Relations

Computability

Computational properties of a structure depend from computational properties of its predicates and functions. A structure is considered computable if all its functions and predicates are.
Before considering different types of computation, we need to give some basic

notions, which will be needed later.
A partial computable function ϕ is computable by a Turing machine, defined for a proper subset of ω. A computable function is a partial computable with domain omega, i.e. total.
In order to connect computability theory notions with sets, we need to find a way to "compute" them. We connect the notion of a function with the notion of a set, where we observe the computability of sets depending from the problem which they represent. The notion which we will present is called a characteristic function of a set. When there is a computable way by which, for each number from ω, we can say whether the number is from the set under consideration or not, then the characteristic function is called total characteristic function, otherwise the characteristic function is called partial characteristic function.
We define the total characteristic function of a set $\mathbb{A} \subseteq \omega$ as:
$\chi_{\mathbb{A}}(x) = \begin{cases} 1, & \text{if x} \in \mathbb{A} \\ 0, & \text{otherwise} \end{cases}$
The partial characteristic function of a set $\mathbb{A} \subseteq \omega$ is defined in the following way:
$C_{\mathbb{A}}(x) = \begin{cases} 1, & \text{if x} \in \mathbb{A} \\ \uparrow, & \text{otherwise} \end{cases}$
We say that a set $\mathbb{A} \subseteq \omega$ is decidible(computable), if its characteristic function is computable, semidecidable(c.e.) if its partial characteristic function is partial computable. It's accepted that domains of computable functions to be denoted by $\mathbb{W}_e$, where e is the code for a Turing machine computing ϕ_e.

Definition 1.1.9. *(Oracle Turing machine, [Soa16]) An oracle Turing machine is a Turing machine, which contains an extra "read only" tape, called an oracle tape, on which the characteristic function of some set $\mathbb{A} \subseteq \omega$ is written.*

Definition 1.1.10. *(Turing operator, [Soa16]) If an oracle Turing machine with a set $\mathbb{A} \subseteq \omega^{<\omega}$ (or a function) on its oracle tape and input $x \in \omega$ halts with output $y \in \omega$, then we write*
$$\Phi_e^{\mathbb{A}}(x) = y.$$
The function $\Phi_e^{\mathbb{A}}$ is called Turing operator.

An operator Φ is c.e. if Φ is a c.e. subset of $\omega^{<\omega} \times \omega \times \omega$.

Definition 1.1.11. *(Relativizing computability, [Soa16]) A partial computable function θ is Turing computable in a set $\mathbb{A} \subseteq \omega$ (or $\mathbb{A}$-Turing computable), written $\theta \leq_T \mathbb{A}$, if there is a program with index e such that*
$$\forall x \forall y (\Phi_e^{\mathbb{A}}(x) \downarrow = y \iff \theta(x) = y).$$
A set $\mathbb{B} \subseteq \omega$ is Turing reducible to $\mathbb{A}$ ($\mathbb{B} \leq_T \mathbb{A}$) if the characteristic function of the set $\mathbb{B}$, $\chi_{\mathbb{B}} \leq_T \mathbb{A}$.

Definition 1.1.12. *(R.i.c.e. computable, [Mon21]) Let $\mathfrak{A}$ be a structure with language L. A relation R, $R \subseteq \omega \times \omega^n$, for $n \in \omega$, is called relatively intrinsically computably enumerable in the structure $\mathfrak{A}$ (r.i.c.e. for short), if for every copy $(\mathfrak{B}, R^{\mathfrak{B}})$ of $(\mathfrak{A}, R)$, the relation $R^{\mathfrak{B}}$ is c.e. in $\mathfrak{D}_{\mathfrak{B}}$.*

Example 1.1.2. *(Linear ordering, [Mon21]) Let $\mathfrak{A}$ be a linear ordering, $\mathfrak{A} = \langle \mathbb{A}, \leq \rangle$. Two elements in $x, y \in \mathbb{A}$ are considered adjacent, written as Adj(x, y), if there is no element between them and $x < y$. The complement of this relation, $\neg Adj(x, y)$, is c.e. in $\mathfrak{D}_{\mathfrak{A}}$. Let $\mathfrak{B}$ be an arbitrary ω-presentation of $\mathfrak{A}$. At each step s, we observe only first s elements of $\mathfrak{B}$. When observing each two elements, say a, b of $\mathfrak{B}$, and find a third element between them, the pair $\langle a, b \rangle$ is enumerated into $\neg Adj^{\mathfrak{B}}$. Becase we are considering an arbitrary ω-presentation, it follows that $\neg Adj$ is r.i.c.e.*

Definition 1.1.13. *(Uniformly R.i.c.e, [Mon21]) Let $\mathfrak{A}$ be a structure with language L. A relation R, $R \subseteq \omega \times \omega^n$, for $n \in \omega$, is called uniformly r.i.c.e. (u.r.i.c.e. for short) in the structure $\mathfrak{A}$ if there is a c.e. operator $\mathbb{W}$ such that $R^{\mathfrak{B}} = \mathbb{W}^{\mathfrak{D}_{\mathfrak{B}}}$ for all $(\mathfrak{B}, R^{\mathfrak{B}}) \cong (\mathfrak{A}, R)$.*

Definition 1.1.14. *(R.i. computable, [Mon21]) Let $\mathfrak{A}, \mathfrak{B}$ be structures with language L. A relation R, $R \subseteq \omega \times \omega^n$, for $n \in \omega$, is called relatively intrinsically computable in the structure $\mathfrak{A}$ (r.i. computable for sort) if $R^{\mathfrak{B}}$ is computable in the atomic diagram of the structure $\mathfrak{B}$, i.e. $\mathfrak{D}_{\mathfrak{B}}$, whenever $(\mathfrak{B}, R^{\mathfrak{B}})$ is a copy of $(\mathfrak{A}, R)$.*

Syntatical characterization

Here we will show how to connect computational properties of a relation and the notion of arithmetical hierarchy. The arithmetical hierarchy is a way by which we can classify the problem which a set represents by a logical formula. The level in the hierarchy, at which a problem falls in, shows how hard a solution for that problem can be found. We divide each level by classes, where it's accepted those classes to be denoted by greek letters $\Sigma_n, \Pi_n, \Delta_n$. The subscript n $\in \omega$ shows the level in the arithmetical hierarchy, where those classes reside. The bottom level of the hierarchy, $n = 0$, contains all sets which are decidible and can be described by a quantifier free formulae. A Σ_n class, for n > 0, contains those sets which are definable by a formula which start with $\exists$-quantifier, followed by alternations of $n - 1$ $\forall, \exists$ quantifiers. A Π_n, for n > 0, class contains those sets which can be defined by a formulae which start with $\forall$-quantifier, followed by $n - 1$ altenration of quantifiers $\exists, \forall$ quantifiers. A Δ_n class contains those sets which are definable by a Σ_n, and Π_n formula.

Definition 1.1.15. *(Σ_n, Π_n-formulae) The defition will be made by induction on $n \in \omega$:*

- *$n = 0$, then $\Sigma_0 = \Pi_0$ are all quantifier-free formulae.*
- *$n > 0$, then*
  Σ_n are all formulae starting with at least one $\exists$ quantifier(s), followed by a Π_β formulae, for $\beta < n$,
  β_n are all formulae starting with at least one $\forall$ quantifier(s), followed by a Σ_α formula, for $\alpha < n$.

Our focus in this Master thesis will be over computation of relations of structures. A computable relation is one for which a Turing machine (or program for computing it) exists.

Definition 1.1.16. *(Infinitary language, [Mon18]) The infinitary language $L_{\omega_1,\omega}$ is defined as a set of formulae in the following way:*

- *All first order L-formulae are in $L_{\omega_1,\omega}$.*
- *If $\{\phi_0, \phi_1, ...\} \subseteq L_{\omega_1,\omega}$ and altogether they use only finitely many free variables, then $\bigwedge_{i\in\omega} \phi_i$ and $\bigvee_{i\in\omega} \phi_i \in L_{\omega_1,\omega}$*
- *If $\phi \in L_{\omega_1,\omega}$ then $\forall x\phi_i \in L_{\omega_1,\omega}$ and $\exists x\phi_i \in L_{\omega_1,\omega}$.*

Definition 1.1.17. *(Infinitary formulae, [Mon18])We will make an induction on $n \in \omega$.*

- *$n = 0$*
  The Σ_0^{inf} and Π_0^{inf} formulae are quantifier free formulae.
- *$n > 0$*
  The Σ_n^{inf} formulae are defined as $\bigvee_{i\in\omega} \exists\bar{x}\phi_i(\bar{x}, \bar{y})$, where $\phi_i(\bar{x}, \bar{y})$ is Π_m^{inf} formula for some $m < n$.
  The Π_n^{inf} formulae are defined as $\bigwedge_{i\in\omega} \forall\bar{x}\phi_i(\bar{x}, \bar{y})$, where $\phi_i(\bar{x}, \bar{y})$ is Σ_m^{inf} formula for some $m < n$.

A formula $\phi \in L_{\omega_1,\omega}$ is a computable $\Sigma_n^c(\Pi_n^c)$ if all its conjunctions or disjunctions are c.e.
Now we will see what properties a relation should have in order to be Σ_1^c-definable.

Definition 1.1.18. *(Σ_1^c-definability, [Mon21]) Let $\mathfrak{A}$ be a structure with language L. A relation R, $R \subseteq \omega \times \omega^n$, for $n \in \omega$, is Σ_1^c-definable in the structure $\mathfrak{A}$ with parameters, if there is a tuple $\overline{p_m} \in \omega^m$ and a computable sequence of Σ_1^c-formulae $\psi_{i,n}(x_1, ..., x_{\overline{m}}, y_1, ..., y_n)$, for $i \in \omega$, such that*

$$R = \{\lceil i, \bar{b}\rceil \in \omega \times \omega^n : \mathfrak{A} \models \psi_{i,|\bar{b}|}(\overline{p}_m, \bar{b})\}.$$

Theorem 1.1.1. *(Ash, Knight, Manasse, and Slaman [Ash+89]; Chisholm [Chi90]) Let $\mathfrak{A}$ be a structure with language L, and let R,*
$R \subseteq \omega \times \omega^n$, for $n \in \omega$, a relation on it. The following are equivalent:

- R is r.i.c.e. computable
- R is Σ_1^c-definable in $\mathfrak{A}$ with parameters.

Proof: A nice proof is given in Theorem 2.1.16, [Mon21]. □

Corollary 1.1.1. *(Corollary 2.1.19, [Mon21]) Let $\mathfrak{A}$ be a structure with language L, and let $R, R \subseteq \omega \times \omega^n$, for $n \in \omega$, a relation on it. The following are equivalent:*

- R is uniformly r.i.c.e. computable
- R is Σ_1^c-definable in $\mathfrak{A}$ without parameters.

Proof: see Corollary 2.1.19, [Mon21] □

1.1.3 Relativization

We know from computable theory that relativization is a notion used for showing how a solution for a problem, which a set represents, can be found with the help of another set. The difference between both theories - computable and computable structure theory, is that structures by themselves are more complicated objects, defined over a(n) (in)finite language.

Let $\mathfrak{A}$ be a structure with language L.

Definition 1.1.19. *(Relativizing r.i.c.e., [Mon21]) Given a relation $R \subseteq \omega \times \omega^n$ and a relation $Q \subseteq \omega \times \omega^n$, we say that R is r.i.c.e. in Q if R is r.i.c.e. in the structure $(\mathfrak{A}, Q)$, i.e. $R^{\mathfrak{B}}$ is c.e. in the atomic diagram $\mathfrak{D}_{\mathfrak{B} \bigoplus Q^{\mathfrak{B}}}$ for every copy $(\mathfrak{B}, R^{\mathfrak{B}}, Q^{\mathfrak{B}})$ of $(\mathfrak{A}, R, Q)$. R is r.i. computable in Q, written as $R \leq_{rT} Q$ (rT - relatively Turing) if R is r.i. computable in the structure $(\mathfrak{A}, Q)$.*

We define the operation $\bigoplus$ for two relations R, Q by: $R \bigoplus Q = \{\langle 2x, 2y\rangle \mid \langle x, y\rangle \in R\} \cup \{\langle 2x+1, 2y+1\rangle \mid \langle x, y\rangle \in Q\}$. If R and Q contain multiple arguments we can apply the same idea to them.

Definition 1.1.20. *(Relativizing r.i., [Mon21]) Given a relation $R \subseteq \omega \times \omega^n$ and a relation $Q \subseteq \omega \times \omega^n$, for $n \in \omega$, we say that R is r.i.c.e. in Q if R is r.i.c.e. in the structure $(\mathfrak{A}, Q)$, i.e. $R^{\mathfrak{B}}$ is c.e. in $\mathfrak{D}_{\mathfrak{B} \bigoplus Q^{\mathfrak{B}}}$ for every copy $(\mathfrak{B}, R^{\mathfrak{B}}, Q^{\mathfrak{B}})$ of $(\mathfrak{A}, R, Q)$. R is r.i. computable in Q, written as $R \leq_{rT} Q$ (rT - relatively Turing) if R is r.i. computable in the structure $(\mathfrak{A}, Q)$.*

Next example demnostrates the usage of relative intrinsic computability.

Example 1.1.3. *([Mon21]) Let $\mathfrak{A} = \{\mathbb{A}, \leq\}$ be a linear ordering, consider the relation given by the pairs of elements which have at least two elements in between:*

$$T = \{\langle a, b\rangle \in \mathbb{A}^2 : a < b \wedge \exists c, d(a < c < d < b)\}$$

Then $T \leq_{rT} Adj$. This follows from the following fact: Let $\langle a, b\rangle \in \mathbb{A}^2$, where $a < b$, and we want to decide whether $\langle a, b\rangle \in T$ by using Adj. First we check whether Adj(a,b) and if so, then we know that $\langle a, b\rangle \notin T$. If the condition is not satisfied then we search for an element $c \in \mathbb{A}$ for which we have that Adj(a, c) and Adj(c, d). If the last condition is satisfied then we again know that $\langle a, b\rangle \notin T$. In all other cases we know that $\langle a, b\rangle \in T$.

1.1.4 Jump of a structure

In computable structure theory, jump of a structure is defined by adding to the structure a complete list of Σ^c_n or Π^c_n-relations. A complete Σ^c_n (or Π^c_n) class of relations is a class C of relations, where all Σ^c_n (or Π_n) are C-computable. Couple of ways defining the jump emerged trough time. A. Soskova and I. Soskov, in [SS09], define the jump of a structure connecting the notion of degree

spectra and jump spectra. A. Montalban, in [Mon09], adds a complete list of Π^c_n relations to a structure. Later, in [Mon21], Montalban defines the notion for first jump, by adding to a structure a complete list of Σ^c_1-definable relations. S. Vatev, in [Vat11], defines the jump by using conservative extensions.
Here we are going to use the notion of jump defined in [Mon21], who extends a structure with a complete list of Σ^c_1-definable relations.

Definition 1.1.21. *(Kleene's complete r.i.c.e. relation, [Mon21]) The complete r.i.c.e. relations on $\mathfrak{A}$ is the set of all Σ^c_1-definable relations:*

$$\vec{K} = \{\langle i, \bar{b}\rangle : \mathfrak{A} \models \phi_{i,|\bar{b}|}(\bar{b})\} \subseteq \omega \times \omega^n, \text{ for } n \in \omega.$$

Definition 1.1.22. *(Jump of a structure, [Mon21]) We define the jump of $\mathfrak{A}$ by adding to the structure the complete set of r.i.c.e. relations. That is, we let*

$$\mathfrak{A}' = (\mathfrak{A}, \vec{K}^{\mathfrak{A}}).$$

1.1.5 Theory of a structure

A theory of a structure is nothing more than a set containing all sentences true in the given structure. In the arithmetical hierarchy the theory of a structure is splitted by the levels of the hierarchy.

Definition 1.1.23. *Let $\mathfrak{A}$ be a structure with language L. The theory of $\mathfrak{A}$, for $n > 0$ is defined as a set of closed formulae:*

- *$n = 1$*
- *Σ_1-$Th(\mathfrak{A}) = \{\exists \overline{x}\phi \mid \mathfrak{A} \models \exists \overline{x}\phi\}$, where ϕ is quantifier free formula.*
- *Π_1-$Th(\mathfrak{A}) = \{\forall \overline{x}\phi \mid \mathfrak{A} \models \forall \overline{x}\phi\}$, where ϕ is quantifier free formula.*
- *$n > 1$*
  - *Σ_n-$Th(\mathfrak{A}) = \{\exists \overline{x}\phi \mid \mathfrak{A} \models \exists \overline{x}\phi\}$ where ϕ are Π_β-formula, for some $\beta < n$.*
  - *Π_n-$Th(\mathfrak{A}) = \{\forall \overline{x}\phi \mid \mathfrak{A} \models \forall \overline{x}\phi\}$ where ϕ are Σ_β-formula, for some $\beta < n$.*

For a sructure $\mathfrak{A}$ with langage L, by Σ^c_n -Th and Π^c_n-Th, for n $\in \omega$, we will abbreviate all Σ_n, or Π_n sentences true in $\mathfrak{A}$.

2 Learnability

In a typical scenario of learnability, the process of learning involves a concept to be learned, a learner, who will learn the concept, and a teacher, who will introduce information to the learner about the concept. The learnability process is considered successful, if the learner reaches a moment at which it masters the concept. A closer look at the learnability process, shows that the success of a learner depends on its abailities to grasp new information, the way information is presented to the learner, what restrictions it faces, and etc.
Influenced by those aspects of learnability, Gold [Gol67], adapted the notion of learning in computational theory. He introduced learning in the limit of different kinds of languages, where the learner is a mechanical device (a Turing machine). The learner receives information about a target language and based on that information is expected to make hypothesis about it. Wrong guesses are allowed, but it's expected to reach a moment after which all hypotheses are correct. The notion today is known as explanatory learning.
Later other theorists, from different branches of science followed his example, and introduced learnability into their field of interest.
In order to introduce some results, which emerged as a result from Gold's model of learnability, we will answer some questions. As a model of learnability we will focus on learnability of classes of structures instead of languages.
The first question is concerned with the so called hypothesis space, or in what way a Turing machine will introduce a description of a structure as answer. For us a hypothesis space will be a c.e. set containing indecies of atomic diagrams of structures. We differentiate between three cases of learnability concerned with hypothesis space. When a hypothesis space coincides with the class being learnt, the learnablity type is called Exact learnability(see def. 2.1.4). When a hypothesis space contains at least the class of structures under consideration, where its enumeration may varies, the learnability type is called Class-comprising learnability(see def. 2.1.4). When a hypothesis space contains suitably chosen enumeration of the class of structures under consideration, the learnability type is called Class-preserving learnability (see def. 2.1.4).
In this Master thesis we will expect that our learner will give as guess an index for a c.e. set, representing the atomic diagram of a structure under consideration.
The next question is concerned with how we will present information to a learner. At each time a finite information about a structure (finite piece of its atomic diagram) will be presented to a learner. The information can be either positive information (or finite part of the atomic diagram of the structure), or combination of positive and negative information (elements from the atomic diagram, and elements from its complement). If only a positive information for a structure is provided, the information presentation is said to be from text(see def. 2.1.1, Txt for short), if a positive and negative information is provided for a structure, then the information presentation is said to be from informant(see def. 2.1.2, Inf for short).

Next we focus on the behaviour of a learner.
When enough information is provided to a learner, and there is a restriction on the number of mind changes which the learner can make, the learnability type can be either Finite(Fin, see def. 2.3.1), Explanatory(Ex, see def. 2.3.2), or Behaviourly correct(BC, see def. 2.3.3). A Fin-learner is allowed to make guess once, when enough information about a structure is given to it and is forbidden of making any mind changes from that moment on. An Ex-learner can change its mind finitely many times, but it's expected to reach a moment after which it will output only a correct guess. A BC-learner can change its mind infinitely often over an infinite set of indecies describing the same structure under consideration.
Then we can observe learner's behaviour depending on the previously made conjectures. Here we will look at Strong-monotonic learnability(see def. 2.3.10), Monotonic learnability(see def. 2.3.10) and Weak-monotonic learnability(see def. 2.3.10). In the strong case a learner is expected to improve its gueses in increasing manner. A monotonic learner can make conjectures for structures containing elements not in the target structure, but is expected wrong elements to dissapear over time. A weakly monotonic learner is expected to act as a strong monotonic learner as long as the information available to it doesn't contradict the made guess.
A type of learnaility concerned with memory limitation adds the notion of Memory limited learnability(see def. 2.3.13).
A question concerned with order of the data adds two new notions to the field - Set driven learning (see def. 2.3.11), and Rearrangement independent learning(see def. 2.3.12). In both types of learnability the order of data doesn't matter, where the difference between both types is that a rearrangement independent learner is interested from the length of data available to it.
The last type of learnability which we are going to look at is connected with behaviourly correct learnability and the behaviour of a learner when is forbidden to change its mind from a correct to incorrect behaviour and turn back to the correct behaviour, which is known as non-U-shaped behaviourly correct learnability(see def. 2.3.14).
By combining different types from questions mentioned above, one gets interesting properties and conclusions about each class of structures learnable under the resulted type. Other interesting criteria, and results about connection between them can be found in [AS16], [OSW86].

2.1 Basic notions

We turn our attention to learnability of structures. There are two major ways of information presentation, which we will present here and use troughtout this Master thesis. When only information for a structure is given to a learner, the information presentation is said to be from text(Txt for short). When information provided to a learner contains elements from a structure as well as elements from its complement, the information presentation is said to be from

informant(Inf for short). We will suppose that information provided to a learner can contain elements given to the learner at some time before, i.e. repetition of elements is allowed.
Sometimes no new information will be given to a learner, in this case a special symbol is introduced #. We call the symbol a pause symbol, and when is given to a learner it will mean that no new information is given at that moment. Another use case for # symbol is the possibility of learning the empty structure.

Definition 2.1.1. *(Text, [AS16]) Text is a computable function*
$$t\colon \omega \to \omega \cup \{\#\}.$$

In other words text is a function which at each step n, $n \in \omega$, generates a code, a finite approximation of the positive atomic diagram of a structure, or gives a pause symbol for no new data.
A content for a text t is defined by $content(t) = range(t) \setminus \{\#\}$. A text for a structure $\mathfrak{A}$ then is defined as $content(t) = \mathfrak{D}_{\mathfrak{A}}$, where $\mathfrak{D}_{\mathfrak{A}}$ is the atomic diagram of the structure $\mathfrak{A}$. If $\mathfrak{D}_{\mathfrak{A}}$ for a structure $\mathfrak{A}$ is empty,i.e. there is no positive information for the structure, then the text will contain only #.

Definition 2.1.2. *(Informant, [BFM20]) Informant is a computable function*
$$I\colon \omega \to \omega \cup \{\#\}.$$

In other words an Informant is a function, which at each step n, $n \in \omega$, generates a finite approximation of the full atomic diagram of a structure.

Remark: When we work with texts, we will suppose that for us $\mathfrak{D}_{\mathfrak{A}} = \mathfrak{D}^{+}_{\mathfrak{A}}$, whereas when we work with informant we will suppose that we will work with the full diagram (with positive and negative information).

Definition 2.1.3. *(Hypothesis space, **HS**) Let*
$\mathfrak{L} = \{\mathfrak{A}_i\}_{i\in\omega}$ *be a class of computable structures with language L. We define a hypothesis space of the class* $\mathfrak{L}$ *as*
$$\boldsymbol{HS} = \{e : \mathbb{W}_e = \mathfrak{D}_{\mathfrak{A}_i} \wedge \mathfrak{A}_i \in \mathfrak{L}\}.$$

Gold, in [Gol67], gives two ways by which a language can be found in a given hypothesis space. The first way, which calls a "generator", uses a function which generates the language under consideration. The second way, which he calls a "tester", represents the characteristic function of the language. In our case we can use either generator, with range the set of an atomic diagram for a structure under consideraton, or the characteristic function of the set of an atomic diagram of that structure.

Observation 2.1.1. *As mentioned in [BFM20], p.6 , there are classes of structures for which it was shown that it can be really hard to be enumerated upto isomorphism. This rises problems when considering the exact learnability, in which a class of structures is enumerated depending from the structures in the class.*

A class of structures is considered to be an indexed family iff there exists a computable function h, s.t. $h(e, \lceil i, x \rceil) = 1 \iff \lceil i, x \rceil \in \mathfrak{D}_{\mathfrak{A}_e}$. We use the idea represented in [LZ93a] in order to give the next definition.

Definition 2.1.4. *(Learnabilty, based on the Hypothesis space) Let $\mathfrak{L} = \{\mathfrak{A}_i\}_{i\in\omega}$ be an indexed family of c.e. structures with language L. Let **HS** be a hypothesis space.*

- *(Exact-learnability) We say that $\mathfrak{L}$ is Exact-learnable, if $\mathfrak{L} =$ **HS**, i.e. the indexing of the class $\mathfrak{L}$ coincides with the hypothesis space.*
- *(Class-preserving learnability) We say that $\mathfrak{L}$ is Class-preserving-learnable, if **HS** $= \{\mathfrak{G}_j\}_{j\in\omega}$ and $\mathfrak{G}_j$ describes a structure from $\mathfrak{L}$ and a learner M learns the class $\mathfrak{L}$ with respect to the hypothesis space **HS**. In this case a learner is required to describe a structure from the class $\mathfrak{L}$, but the hypothesis space can define or enumerate differently each structure in the original class.*
- *(Class-comprising learnability) We say that $\mathfrak{L}$ is Class-comprising-learnable, if **HS** contains at least a description of each structure in the class $\mathfrak{L}$.*

For a text t (or informant I), when we refer to a t_n (or I_n), for some $n \in \omega$ we will mean the first n elements from a given text(or informant), available to a learner. When we use $n + k$ as index for a text t (or informant I), we will mean t_{n+k}, for some $k \in \omega$, i.e. we proceed with stages of revealing the text to the learner. When we use a subscript n for an index of a structure, we will mean an index found which agrees with the data seen so far at stage n.
A segment of a text t (or informant I) is defined as a finite part of a text (or informant). It's accepted segments to be denoted by lower greek letters, i.e. σ, τ, and etc. The set of all segments of all texts (or informants) is denoted by **SEQ**. The length of a segment will be denoted by $ln(\sigma)$. For $\sigma, \tau \in$ **SEQ** for a structure $\mathfrak{A}$ with language L, $\sigma\hat{}\tau$ is defined as a new segment $\sigma' \in$ **SEQ** which contains σ and τ in this order. When $\sigma \in$ **SEQ** is an initial segment for a text t, that will be denoted by $\sigma \sqsubseteq t$. A proper initial segment σ for a text t, is an initial segment σ' of t, s.t. $\sigma' = t_m$, and $\sigma'' = t_n$, for $m, n \in \omega, m < n$, and $t_m \neq t_n$. If $\sigma \in$ **SEQ** is a proper initial segment of a text t, that will be denoted by $\sigma \sqsubset t$.

Definition 2.1.5. *(Learner, [AS16]) A learner M is a computable function, s.t. M: $\omega \to \omega \cup \{?\}$.*

In other words a learner M is a Turing machine, which at every step $n \in \omega$ gets as input a code for a sequence $\sigma \in$ **SEQ**, and based on that information tries to find an index for the structure under consideration.

Definition 2.1.6. *(Locking sequence, Text) Let $\mathfrak{A}$ be a structure with language L and let M be a learner for the structure $\mathfrak{A}$. Let t be a text for $\mathfrak{A}$. Let $\sigma \in$ **SEQ**, s.t. $\sigma \sqsubset t$. The segment σ is a locking sequence for M of $\mathfrak{A}$, if*

i $content(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}}$

ii $\mathbb{W}_{M(\lceil\sigma\rceil)} = \mathfrak{D}_{\mathfrak{A}}$

iii $\forall\tau \in \boldsymbol{SEQ}(\tau \sqsubset t \rightarrow content(\tau) \subseteq \mathfrak{D}_{\mathfrak{A}} \wedge M(\lceil\sigma\,\hat{}\,\tau\rceil) = M(\lceil\sigma\rceil))$ *holds.*

2.2 An Example

Here we are going to show a basic example of learnability of structures. The example is taken from [BFM20]. But before giving it, we will try to explain, in an abstract way, how a Turing machine (learner) works trough the stages of the learnability process.

Let M be a learner and let $\mathfrak{L}$ be a class of structures.
Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$. Let $\sigma \in \mathbf{SEQ}$, where $\sigma \sqsubset t$, be a finite initial segment available to M.
Let Alg: $\omega \rightarrow \omega \cup \{?\}$ be a computable function. Here Alg will be a function which applies certain steps over σ and outputs an index for an atomic diagram of a structure, or will output $'?'$ if such index is not found.
Let Pr be a type of learnability.
Let's give an abstract description of a way in which a learner behaves troughout the learnability process.
M = "On input $\sigma \in \mathbf{SEQ}$
guess $\leftarrow Alg(\sigma)$
output guess as answer
proceed with computation depending from Pr"

We will suppose that our learner behaves in the way expected from the learnability type under consideration.

Let's turn our attention to the example from [BFM20]. The example represents an TxtEx-learning(Informant def. 2.1.2, Explanatory def. 2.3.2). In other words when a learner receive positive information(Text, 2.1.1) for a strucutre, can the learner give a correct guess after some finite amount of time, provided that the learner is permitted to make incorrect guesses before a correct one is reached(Explanatory learning, 2.3.2). We can never say when our guess will be correct, but the "built in" idea of Ex-learning provide us with the idea that if a learner behaves in the expected way, then such moment of "truthness" exists.

Example 2.2.1. *Let* $\mathfrak{L} = \{\mathfrak{G}_i\}_{i\in\omega}$ *be a class of structures, where* $i > 2$*, and each structure is defined as an undirected graph over a language* $L = \{Edge\}$*, where Edge is a binary predicate. Each structure contains a simple cycle of size* i *and eventually cycles of size* $< i$*. Let* t *be a text for a structure* $\mathfrak{G}_i \in \mathfrak{L}; i \in \omega$*. The atomic diagram of* $\mathfrak{G}_i$ *is defined as* $\mathfrak{D}_{\mathfrak{G}_i} = \{\lceil e, \lceil a, b\rceil\rceil \mid \mathfrak{G}_i \models \phi_e^{at}(a, b)\}$
Where $\phi_e^{at}(a, b)$*, for* $a,\ b \in \mathbb{G}_i$*, is an atomic formula representing the predicate Edge.*

An algorithm (learner) M which TxtEx-learns the class $\mathfrak{L}$ can be defined in the following way.

$M(\lceil t_x \rceil)$ = "$j \leftarrow SizeOfLongestCycle(t_x)$
if $j = \emptyset$
output $'?'$, *and request next input*
otherwise, output j *and request next input*"

$SizeOfLongestCycle(t_x)$ = "
$cycles \leftarrow \emptyset$
$checked \leftarrow \emptyset$
foreach Edge(a, b) in t_x
if $\neg checked(Edge(a, b))$
$cycles \leftarrow GetCycle(Edge(a, b), t_x)$
$checked \leftarrow Edge(a, b)$
$i \leftarrow MaxSizeCycle(cycles)$
output i

The algorithm SizeOfLongestCyclesfinds all simple cycles from the information provided by t_x, in the form of edges between elements from the universum of a given structure $\mathfrak{G}_i$, and get the one with a maximum size. We can change our mind a finitely many times until we find the maximum one. This shows that the learner is TxtEx-learner.

2.3 Types of learnability

2.3.1 [Finite, Explanatory, Behaviourly Correct]

We will start with learnability types based on the behaviour of a learner, when a restriction over number of mind changes is presented.
Suppose that a learner has received enough information to take a decision. If a learner is allowed to make only one guess, the learnability type is called Finite(Fin). If a learner can change its mind only finitely many times, the learnability type is called Explanatory (Ex). A BC-learner is similar to Ex, with the difference that it can provide more than one index for a c.e. set of an atomic diagram.

Definition 2.3.1. *(Fin-learnability, [AS16])*
A learner M TxtFin(InfFin)-learns a sructure $\mathfrak{A}$, from a text t for $\mathfrak{A}$ (or an informant I for $\mathfrak{A}$), if there is a number $n_0 \in \omega$, s.t.
$\forall n < n_0 (M(\lceil t_n \rceil) = ?) \wedge$
$\forall n \geq n_0 (M(\lceil t_n \rceil) = M(\lceil t_{n_0} \rceil) \in \omega) \wedge \mathfrak{D}_{\mathfrak{A}} = \mathbb{W}_{M(\lceil t_{n_0} \rceil)}$.

Definition 2.3.2. *(Ex-learnability, [AS16])*
A learner M TxtEx(InfEx)-learns a structure $\mathfrak{A}$, if t is a text for $\mathfrak{A}$(or an infor-

mant I for $\mathfrak{A}$) , and there is a number $n_0 \in \omega$, s.t.
$\forall n \geq n_0(M(\lceil t_n \rceil) = M(\lceil t_{n_0} \rceil) \in \omega) \wedge \mathfrak{D}_{\mathfrak{A}} = \mathbb{W}_{M(\lceil t_{n_0} \rceil)}$

Definition 2.3.3. *(BC-learnability, [AS16])*
A learner M TxtBC(InfBC)-learns a structure $\mathfrak{A}$, if t is a text for $\mathfrak{A}$(I is an informant for $\mathfrak{A}$), and there is a number $n_0 \in \omega$, s.t.
$\forall n \geq n_0(M(\lceil t_n \rceil) \in \omega \wedge \mathfrak{D}_{\mathfrak{A}} = \mathbb{W}_{M(\lceil t_n \rceil)})$

Definition 2.3.4. *Let Pr be a type of learnability, and let $X \in \{Txt, Inf\}$.*

- *A learner M XPr-learns a class of structures $\mathfrak{L}$, if M XPr-learns each structure $\mathfrak{A} \in \mathfrak{L}$.*
- *A class of structures $\mathfrak{L}$ is XPr-learnable if there is a XPr-learner M which XPr-learns the class $\mathfrak{L}$.*

Properties of learnability types

Here we will consider some properties of $[Fin, Ex, BC]$-learnability types. Properties given in this subsection are taken from [AS16], and modified in the case for structures.
We will define couple of classes of structures for later use. Here, for simplicity, we will focus on structures over a language with one unary relation. Let $\mathfrak{A}$ be a structure.

- $\mathfrak{L}^+_{\mathfrak{A}} = \{\mathfrak{A}_i\}_{i\in\omega} \cup \{\mathfrak{A}\}$, s.t. $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}} \cup \{\lceil e, x \rceil\}$, where $i = \lceil e, x \rceil$, i.e. $\mathfrak{A}_i \models \phi_e(x)$.
- $\mathfrak{L}^-_{\mathfrak{A}} = \{\mathfrak{A}_i\}_{i\in\omega} \cup \{\mathfrak{A}\}$, where for each i, $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}} \setminus \{\lceil e, x \rceil\}$, for $i = \lceil e, x \rceil$, i.e. $\mathfrak{A}_i \not\models \phi_e(x)$.
- $\mathfrak{L}^*_{\mathfrak{A}} = \{\hat{\mathfrak{A}} =^* \mathfrak{A}\}$, where $\mathfrak{A}$ is an arbitrary c.e. structure. The class contains all finite variants(see def. 2.3.9) of the structure $\mathfrak{A}$.
- $\mathfrak{L}_{\mathfrak{A}} = \mathfrak{L}^+_{\mathfrak{A}} \cup \mathfrak{L}^-_{\mathfrak{A}}$.

Theorem 2.3.1. *(Theorem 3, [LZ92a]) Let $\mathfrak{L}$ be a c.e. class of computable structures with language L. Then $\mathfrak{L} \in TxtFin$ if and only if there is a c.e. family $(T_j)_{j\in\omega}$ of finite non-empty sets, s.t.*

- *$T_i \subseteq \mathfrak{D}_{\mathfrak{A}_i}$, for all $i \in \omega$, for $\mathfrak{A}_i \in \mathfrak{L}$.*
- *For all $k, j \in \omega$, if $T_k \subseteq \mathfrak{D}_{\mathfrak{A}_j}$, then $\mathfrak{D}_{\mathfrak{A}_k} = \mathfrak{D}_{\mathfrak{A}_j}$, where $\mathfrak{A}_k, \mathfrak{A}_j \in \mathfrak{L}$.*

One important result, from [Ang80], is concerned with learnability from text. While information presentation from informant is shown to be more powerful, because it gives positive and negative information for a structure, the same doesn't remain true for information presentation from text. We can't rely on locking sequences every time when we are learning from text, as we can have a class containing substructures. In order to choose the correct structure from a class, we can rely on the following fact.

Theorem 2.3.2. *(Theorem 1, [Ang80]) Let $\mathfrak{L}$ be a c.e. class of nonempty computable structures. Then $\mathfrak{L} \in TxtEx$, if and only if there exists a c.e. family $(T_j)_{j\in\omega}$ of finite non-empty sets, which satisfy the following conditions:*

1. *$T_i \subseteq \mathfrak{D}_{\mathfrak{A}_i}$, for $\mathfrak{A}_i \in \mathfrak{L}$, and*
2. *for all $j \geq 1$, if $T_i \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ then $\mathfrak{D}_{\mathfrak{A}_j} \not\subset \mathfrak{D}_{\mathfrak{A}_i}$, where $\mathfrak{A}_i, \mathfrak{A}_j \in \mathfrak{L}$.*

Proof: See thm. 1, [Ang80]. □

Lemma 2.3.1. *(Monotonicity Lemma, Lemma 5.13, [AS16])*
Let $Y \in \{Txt, Inf\}$, and let $X \in \{YFin, YEx, YBC\}$ and let $\mathfrak{L}_i$ and $\mathfrak{L}_j$ be c.e. families of computable structures such that $\mathfrak{L}_i \subseteq \mathfrak{L}_j; i, j \in \omega$. If a learner M X-learns $\mathfrak{L}_j$ then M X-learns $\mathfrak{L}_i$. Hence, if $\mathfrak{L}_j$ is X-learnable then $\mathfrak{L}_i$ is X-learnable too.

Proof: Assume otherwise, that there is a structure in $\mathfrak{L}_i$, which can not be X-learnt.
We have that $\mathfrak{L}_i$ is contained in $\mathfrak{L}_j$, and the class $\mathfrak{L}_j$ is X-learnable. So each structure $\mathfrak{A} \in \mathfrak{L}_i$ is X-learnable. A contradiction. □

Lemma 2.3.2. *(Lemma 5.15, [AS16]) Let $\mathfrak{A}$ and $\mathfrak{B}$ be two computable structures with language L, and let $\mathfrak{A} \subset \mathfrak{B}$. Then $\{\mathfrak{A}, \mathfrak{B}\} \notin TxtFin$.*

Proof: Let $\mathfrak{L} = \{\mathfrak{A}, \mathfrak{B}\}$, where both structures are defined over a language L.
Suppose t_0, t_1 are both texts for $\mathfrak{A}, \mathfrak{B}$ respectively.
Let M be a TxtFin-learner which learns the class $\mathfrak{L}$. Then M when fed t_0, t_1 is expected to learn both structures, depending from the given input.
Suppose that M is fed with t_0. Then there is a stage n_0 at which M will start to output an index for $\mathfrak{A}$.
Let's define a new text t based on the following idea:

$$\forall n < n_0(t_n = t_0 \upharpoonright n) \wedge \forall n \geq 0(t_{n_0+n} = t_1 \upharpoonright n).$$

When M receives t as input, at moment n_0 it will start to output an index for $\mathfrak{A}$ instead of $\mathfrak{B}$. Clearly M will fail to identify the class $\mathfrak{L}$. □

Corollary 2.3.1. *Let $\mathfrak{L} = \{\mathfrak{A}_i\}_{i\in\omega}$ be a c.e. class of computable structures with language L, s.t. there are at least two structures $\mathfrak{A}_i, \mathfrak{A}_j$, for $i, j \in \omega$, s.t. $\mathfrak{A}_i \subseteq \mathfrak{A}_j$. Then $\mathfrak{L} \notin TxtFin$.*

In the case of languages, an informant provides information about elements from a language as well as elements not in it (see [AS16]). In our case, when we use informant, we just provide full information (positive and negative) about a struture, i.e. arguments which make a structure true in some relation, or false in it. Because of this, the next lemma, while for the case of languages is not true, in our case it is.

Lemma 2.3.3. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be two computable structures with language L, and let $\mathfrak{A} \subset \mathfrak{B}$. Then $\{\mathfrak{A}, \mathfrak{B}\} \notin InfFin$.*

Proof: In order to prove this lemma, we will use the same idea as in lemma 2.3.2, but in the case of informant.
Let $\mathfrak{L} = \{\mathfrak{A}, \mathfrak{B}\}$ be a class of computable structures, where both structures are defined over a language L.
Suppose I_0, I_1 are both informants for $\mathfrak{A}, \mathfrak{B}$ respectively.
Let M be an InfFin-learner for the class $\mathfrak{L}$. When M receives information from I_0, I_1, it's expected to learn both structures, depending from the input which receives.
Suppose M receives information from I_0. Then at some stage, say n_0, M will start to output an index for $\mathfrak{A}$.
Based on that, we will define a new informant I', with the idea when M receives it as input, M to make a wrong guess. Let's define such informant I':

$$\forall n < n_0(I_n = I_0 \upharpoonright n) \wedge \forall n \geq 0(I_{n_0+n} = I_1 \upharpoonright n)$$

In other words, when M receives I' as input, M will wrongly guess that the structure provided to it is $\mathfrak{A}$, but not $\mathfrak{B}$. As M is InfFin-learner, M is forbidden of changing is mind. Then M will fail to identify the class $\mathfrak{L}$. □

Definition 2.3.5. *(Translation function, [AS16]) Let $g_i : \omega \to \mathbb{A}_i (i \in \{0, 1\})$ be a numbering, where $\mathbb{A}_i$ be a set of countable objects. Then g_0 is reducible to g_1 (or vice versa), denoted by $g_0 \leq g_1$, if there is a computable function $f : \omega \to \omega$, s.t. $g_0 = g_1 \circ f$.*

In other words if we have two functions g_0, g_1, s.t. $g_0 \leq g_1$ by a computable function f, and x $\in dom(g_0)$, then f translates x to an element n $\in dom(g_1)$, and $g_1(n) = y$, then $g_0(x) = g_1(f(x)) = g_1(n) = y$. A computable function f with such property is called translation function from g_0 to g_1.
A cannonical index of a finite set is a computable function $\gamma : \omega \to \omega$, defined as $\gamma(\langle a_0, a_1, ..., a_n\rangle) = \sum_{i=0}^{n} 2^{a_i}$, where $\bar{a} \in \omega^n$ and $a_0 < a_1 < ... < a_n$.
By the S_n^m theorem there is a computable function f such that $\mathbb{W}_{f(e)} = D_e$, where D_e is the finite set with cannonical index e.
Recall that $\mathfrak{L}^+_{\mathfrak{A}} = \{\mathfrak{A}_i\}_{i\in\omega} \cup \{\mathfrak{A}\}$, s.t. $\mathfrak{A} \subseteq \mathfrak{A}_i$ for an arbitrary c.e. structure $\mathfrak{A}$, where $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}} \cup \{\lceil e, x\rceil\}$, and $i = \lceil e, x\rceil$.

Lemma 2.3.4. *(Lemma 5.16, [AS16]) Let $\mathfrak{A}$ be a c.e. structure with language L. Then the class $\mathfrak{L}^+_{\mathfrak{A}} \in TxtBC$.*

Proof: A successful TxtBC-learner M is based on the following idea: When M makes a guess for a structure $\mathfrak{A}_i \in \mathfrak{L}^+_{\mathfrak{A}}$, for $i \in \omega$, we will consider the behaviour of M correct with some finite exceptions.
In other words for a text t, if trough the learning process we find that $\lceil e, x\rceil \in \mathfrak{D}_{\mathfrak{A}}$, then for $i = \lceil e, x\rceil$. $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}}$. Hence for all $n \geq n_0$ we will have that $\mathfrak{D}_{\mathfrak{A}} = \mathfrak{D}_{\mathfrak{A}} \cup content(t_n)$. Otherwise if $\lceil e, x\rceil \notin \mathfrak{D}_{\mathfrak{A}}$, then there should be a least numer n_0, s.t. $\lceil e, x\rceil \in content(t_{n_0})$.
Then for all $n \geq n_0$, $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}} \cup content(t_n)$.
Let's define such a learner M more formally. Let $\mathbb{V}_j \subseteq \omega$ be a c.e. set defined in the following way $\mathbb{V}_j = \mathfrak{D}_{\mathfrak{A}} \cup \mathbb{D}_j$, where $\mathbb{D}_j \subset \omega$ is finite, with canonical index j. By the S_n^m theorem, there is a a computable function f, such that

$\mathbb{V}_j = \mathbb{W}_{f(j)}$. Let g be a computable function, which maps $\sigma \in \mathbf{SEQ}$, where $\sigma \sqsubseteq t$, to the canonical index of content(σ). Then $M = f \circ g$ is computable and M TxtBC-learns the class $\mathfrak{L}^+_{\mathfrak{A}}$. □

By $\mathfrak{F}_{rec}$ we will denote the class of all partial computable functions.

Lemma 2.3.5. *(Padding lemma, Lemma 2.34, [AS16]) Let $\phi_e \in \mathfrak{F}_{rec}$ be a partial computable function. Then there is a one-to-one computable function f: $\omega^{n+1} \to \omega$, such that, for any $e \in \omega$, $\bar{s} \in \omega^n$, $\phi_{f(e,\bar{s})} = \phi_e$.*

Proof: See Lemma 2.34, [AS16]. □

Suppose that M is the learner defined in lemma 2.3.4.
As given in ([AS16], p. 57), we can not argue that the class defined in the lemma 2.3.4 can be TxtEx-learnt by M. Suppose that f is a one-to-one function (assumed by the Padding Lemma, see lemma 2.3.5). For an infinite structure $\mathfrak{A}_j \in \mathfrak{L}^+_{\mathfrak{A}}$, for $j \in \omega$, M can not TxtEx-learn $\mathfrak{L}^+_{\mathfrak{A}}$.
In this case, we will have that for each $i \in \omega$, and any text t for a structure $\mathfrak{A}_i$, there exists an infinite sequence $n_0 < n_1 < n_2 < ...$, s.t.

$content(t_{n_0}) \subset content(t_{n_1}) \subset content(t_{n_2}) \subset ....$

As for each index $i \in \omega$, we will have that $g(\lceil t_{n_i} \rceil)$ are different (we output a canonical index), then $f(g(\lceil t_n \rceil))$ will be different too. In other words M changes its mind infinitely often, and never reaches a moment at which correctly will guess a structure, which contradicts the definition of Ex-learnability. It follows that $\mathfrak{L} \notin TxtEx$.

Lemma 2.3.6. *(Lemma 5.17, [AS16]) Let $\mathfrak{A}$ be a computable structure with language L. Then the class $\mathfrak{L}^+_{\mathfrak{A}} \in TxtEx$.*

Proof: A successful TxtEx-learner M for the class $\mathfrak{L}^+_{\mathfrak{A}}$ first checks computably, of a text t for a structure of $\mathfrak{L}^+_{\mathfrak{A}}$, which structure σ describes.
Let f : $\omega \to \omega$ be a computable function, s.t. $\mathbb{W}_{f(\lceil e,x \rceil)} = \mathfrak{D}_{\mathfrak{A}_j}$, where $j = \lceil e, x \rceil$, and $\mathfrak{D}_{\mathfrak{A}_j} = \mathfrak{D}_{\mathfrak{A}} \cup \{\lceil e, x \rceil\}$.
For each step we check if $content(t_s) \subset \mathfrak{D}_{\mathfrak{A}}$. If it is, then $M(\lceil t_s \rceil) = i$, where i is the fixed index of $\mathfrak{D}_{\mathfrak{A}}$. If there is $\lceil e, x \rceil \in content(t_s)$, which is not from $\mathfrak{D}_{\mathfrak{A}}$($\mathfrak{D}_{\mathfrak{A}}$ is computable), then $M(\lceil t_s \rceil) = f(j)$, where $j = \lceil e, x \rceil$.
□

For a struture $\mathfrak{A}$ and a text t for $\mathfrak{A}$, by $t(n)$, for some n $\in \omega$, we will denote the element which can be found at the n-th position of the text.
Next theorem was presented in [BB75](see Theorem 5.25), and was proven for languages. Here we will prove it for structures. By λ we will denote the empty segment.

Theorem 2.3.3. *Let $\mathfrak{L}$ be a finite TxtEx-learnable family of computable structures with language L. Let M be a TxtEx-learner for the family $\mathfrak{L}$. Then there is a locking sequence for M on each structure $\mathfrak{A} \in \mathfrak{L}$.*

Proof: Let t be a text for a structure $\mathfrak{A}_i \in \mathfrak{L}, i \in \omega$. We will prove the theorem by contradiction. Suppose that there is no locking sequence for M on $\mathfrak{A}_i$.Then for every segment $\sigma \in \mathbf{SEQ}$, s.t. $\sigma \sqsubseteq t$, the following is true:

(*) $content(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}_i} \rightarrow \exists \tau (content(\tau) \subseteq \mathfrak{D}_{\mathfrak{A}_i} \wedge M(\lceil \sigma \rceil) \neq M(\lceil \sigma \hat{} \tau \rceil))$

In other words, if σ is not a locking sequence for M on $\mathfrak{A}_i$, then there is a segment $\tau \in \mathbf{SEQ}$, s.t. $\tau \sqsubseteq t$, and $\sigma \sqsubset \tau$, and $M(\lceil \sigma \hat{} \tau \rceil)$ is an index for $\mathfrak{A}_i$, since M learns TxtEx $\mathfrak{A}_i$.
On the other hand if $M(\lceil \sigma \rceil)$ is an index for $\mathfrak{A}_i$, by condition we have that σ is not a locking sequence for M on $\mathfrak{A}_i$ and such segment τ exists.
We can use (*) to inductively define such segments $\sigma_n, n \in \omega$:

- $\sigma_n \sqsubset \sigma_{n+1}$
- $t(n) \in \sigma_{n+1}$
- $content(\sigma_n) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$
- $M(\lceil \sigma_n \rceil) \neq M(\lceil \sigma \rceil)$ for some $\sigma \in \mathbf{SEQ}$ with $\sigma_n \sqsubseteq \sigma \sqsubseteq \sigma_{n+1}$

Let $\sigma_0 = \lambda$, $\sigma_{n+1} = \sigma_n \hat{} \tau \hat{} t(n)$, where τ is the least segment for which $content(\tau) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$ and $M(\lceil \sigma_{n+1} \rceil) \neq M(\lceil \sigma_n \hat{} \tau \rceil)$.
First three conditions guarantee that indeed t is a text for the structure $\mathfrak{A}_i$. The last one satisfies the requirement from $(*)$. It follows that M does not TxtEx-learn $\mathfrak{L}$ which contradicts the choice of M. □

Theorem 2.3.4. *(Theorem 5.29, [AS16]) Let $\mathfrak{A}$ be a non-computable c.e. structure with language L. Then $\mathfrak{L}^+_{\mathfrak{A}} \notin TxtEx$.*

Proof: We will prove the theorem by contradiction. Assume that M is a TxtEx-learner for the class $\mathfrak{L}^+_{\mathfrak{A}}$. Suppose t is a text for $\mathfrak{A}$. Let $\sigma \in \mathbf{SEQ}$, $\sigma \sqsubseteq t$, be a locking sequence for M on $\mathfrak{A}$. We will prove that $\overline{\mathfrak{D}_{\mathfrak{A}}}$ is c.e. We shall show that for any x $\in \omega$:

(*) $\lceil e, x \rceil \notin \mathfrak{D}_{\mathfrak{A}} \iff \exists n (M(\lceil \sigma \rceil) \neq M(\lceil \sigma \hat{} \lceil e, x \rceil \hat{} (t_n) \rceil)$

($\rightarrow$) Suppose that $\lceil e, x \rceil \notin \mathfrak{D}_{\mathfrak{A}}$. Then $\mathfrak{D}_{\mathfrak{A}} \neq \mathfrak{D}_{\mathfrak{A}} \cup \{\lceil e, x \rceil\}$ and $\sigma \hat{} \lceil e, x \rceil \hat{} t$ is a text for a structure with atomic diagram $\mathfrak{D}_{\mathfrak{A}} \cup \{\lceil e, x \rceil\}$. Let $\mathfrak{A}_j$ be that structure. By definition of $\mathfrak{L}^+_{\mathfrak{A}}$, we have that $\mathfrak{A}_j \in \mathfrak{L}^+_{\mathfrak{A}}$. Furthermore, by assumption, the learner M TxtEx-learns the class. Then there is $n_0 \in \omega$, s.t.
$\mathbb{W}_{M(\lceil \sigma \hat{} \lceil e, x \rceil \hat{} (t_n) \rceil)} = \mathfrak{D}_{\mathfrak{A}_j}$, for all n $\in \omega, n \geq n_0$. We have that σ is a locking sequence for M on $\mathfrak{A}$, then $\mathbb{W}_{M(\lceil \sigma \rceil)} = \mathfrak{D}_{\mathfrak{A}}$. So $M(\lceil \sigma \rceil) \neq M(\lceil \sigma \hat{} \lceil e, x \rceil \hat{} t_n \rceil)$. Hence $\overline{\mathfrak{D}_{\mathfrak{A}}}$ is not c.e.
($\leftarrow$) Assume that $\lceil e, x \rceil \in \mathfrak{D}_{\mathfrak{A}}$. Then for every $n \geq 0$, where $n \in \omega$, $M(\lceil \sigma \rceil) = M(\lceil \sigma \hat{} \lceil e, x \rceil \hat{} (t_n) \rceil)$. □

Corollary 2.3.2. *(Corollary 5.26, Angluin, [AS16]) Let $\mathfrak{L}$ be a finite TxtEx-learnable class of computable structures with language L. Then for any structure $\mathfrak{A} \in \mathfrak{L}$ there is a finite substructure $\mathfrak{F}$, such that $\mathfrak{F} \subseteq \mathfrak{A}$ and*

(*) $\forall \hat{\mathfrak{A}} \in \mathfrak{L} (\mathfrak{F} \subseteq \hat{\mathfrak{A}} \subseteq \mathfrak{A} \rightarrow \hat{\mathfrak{A}} = \mathfrak{A})$ holds.

Proof: Suppose that M is a $TxtEx$-learner for the class $\mathfrak{L}$.
Let $\mathfrak{A} \in \mathfrak{L}$, let $\sigma \in \mathbf{SEQ}$, be the least locking sequence for M on $\mathfrak{A}$ such that $\mathfrak{D}_{\mathfrak{F}} = content(\sigma)$.
Let $\hat{\mathfrak{A}} \in \mathfrak{L}$ be a structure, such that $\mathfrak{F} \subseteq \hat{\mathfrak{A}} \subseteq \mathfrak{A}$ and let t be a text for $\hat{\mathfrak{A}}$.
By assumption $content(\sigma) = \mathfrak{D}_{\mathfrak{F}} \subseteq \hat{\mathfrak{A}}$, so $\sigma\hat{}t$ is a text for $\hat{\mathfrak{A}}$, too.
As we have that M TxtEx-learns the structure $\hat{\mathfrak{A}}$, then $W_{M(\lceil \sigma\hat{}t_n \rceil)} = \mathfrak{D}_{\hat{\mathfrak{A}}}$ for all sufficiently large n $\in \omega$.
On the other hand, we have that $\hat{\mathfrak{A}} \subseteq \mathfrak{A}$, i.e. $content(t_n) \subseteq \mathfrak{D}_{\mathfrak{A}}$. By assumption σ is a locking sequence for M on $\mathfrak{A}$, so we get that
$\mathfrak{D}_{\mathfrak{A}} = \mathbb{W}_{M(\lceil \sigma \rceil)} = \mathbb{W}_{M(\lceil \sigma\hat{}t_n \rceil)}$. Hence $\mathfrak{A} = \hat{\mathfrak{A}}$. □

By $\mathbf{FIN}_L$ we will denote all finite structures with language L.

Theorem 2.3.5. *(Theorem 5.19, Gold, [AS16]) Let $\mathfrak{A}$ be an infinite structure with language L. Then $\boldsymbol{FIN}_L \cup \{\mathfrak{A}\} \notin TxtBC$.*

Proof: Let's assume that M is a learner which TxtBC-learns the class $\mathbf{FIN}_L$.
We will define a text t for $\mathfrak{A}$ s.t. for infinitely many $k \in \omega$, $M(\lceil t_k \rceil)$ is not an index for $\mathfrak{A}$.
Let f be a one-to-one function, with range the atomic diagram $\mathfrak{D}_{\mathfrak{A}}$. Since M TxtBC-learns any finite structure from $\mathbf{FIN}_L$ we can inductively define numbers $m_n \geq 1$, s.t.

$$\mathbb{W}_{M(\lceil f(0)^{m_0} f(1)^{m_1} \ldots f(n)^{m_n} \rceil)} = \{f(0), f(1), ..., f(n)\}.$$

(Namely, since $f(0)^\omega$ is a text for the finite set $\{f(0)\}$, then $\mathbb{W}_{M(\lceil f(0)^m \rceil)} = \{f(0)\}$ for almost all $m \in \omega$, let m_0 be the least such $m \geq 1$. Similarly given $m_0, m_1, ..., m_{n-1}, f(0)^{m_0} ... f(n-1)^{m_{n-1}} f(n)^\omega$ is a text for the finite set $\{f(0), ..., f(n)\}$, whence $\mathbb{W}_{M(\lceil f(0)^{m_0} \ldots f(n-1)^{m_{n-1}} f(n)^m \rceil)} = \{f(0), ..., f(n)\}$ for almost all $m \in \omega$, and let m_n be the least such $m \geq 1$).
So the text defined as $t = f(0)^{m_0} f(1)^{m_1} ....$ of the atomic diagram of $\mathfrak{A}$ has the desired properties. □

We can convert a TxtEx-learner from Corollary 2.3.2 to a TxtBC-learner in the following way:

- If in thm. 2.3.3 we replace M to a non-computable c.e. learner, the theorem remains true.
- For any TxtBC-learner M, for a family $\mathfrak{L}$, we can convert M to a non-computable c.e. TxtEx-learner $\hat{M}$ in the following way:
  For an input $\sigma \in \mathbf{SEQ}$, for which $M(\lceil \sigma \rceil) \downarrow, \hat{M}(\lceil \sigma \rceil) = M(\lceil \sigma_n \rceil)$, where $n < ln(\sigma)$, s.t. $\mathbb{W}_{M(\lceil \sigma_n \rceil)} = \mathbb{W}_{M(\lceil \sigma \rceil)}$. This ensures that if M semantically converges on a text t to indices for a structure $\mathfrak{A}$, then $\hat{M}$ will (sintatically) converges on t to a fixed index for $\mathfrak{A}$.

Corollary 2.3.3. *(Corollary 5.27, [AS16]) Let $\mathfrak{L}$ be a TxtBC-learnable class of c.e. structures. For any structure $\mathfrak{A} \in \mathfrak{L}$ there is a finite substructure $\mathfrak{F}$ such that $\mathfrak{F} \subseteq \mathfrak{A}$ and the condition in Corollary 2.3.2 holds.*

Proof: By the second observation from above, we don't have necessarily computable learner M which TxtEx-learns $\mathfrak{L}$.
By the first observation from above, for any structure $\mathfrak{A} \in \mathfrak{L}$ there is a locking sequence $\sigma \in \mathbf{SEQ}$ for M on $\mathfrak{A}$. If we let $\mathfrak{D}_{\mathfrak{F}} = \mathbb{W}_{M(\lceil \sigma \rceil)}$ for the least such σ, then $\mathfrak{F} \subseteq \mathfrak{A}$ and we can argue as in thm. 2.3.2 that the condition holds. □

Recall that a class of structures $\mathfrak{L}$, is TxtFin-learnable when for each two structures $\mathfrak{A}$ and $\mathfrak{B}$ from $\mathfrak{L}$ we have that both structures are incomparable (see lemma 2.3.2 and corollary 2.3.1 for more information). Now when we replace information presentation from text with information presentation from informant, we can expand the family of learnable classes under the new type of learnability InfFin. When a class $\mathfrak{L}$ of structures contains structures which have elements in common, but both still contain information with which a difference can be made, an InfFin-learner M will use that information in order to reach a correct conclusion.

Definition 2.3.6. *(Limit point, def. 2.21, [AS16]) Let $\boldsymbol{F}$ be a family of total functions (of type $\omega \to \omega$). A function f is a limit point of the family $\boldsymbol{F}$, if $f \in \boldsymbol{F}$ and $\forall x \geq 0 \exists g \in \boldsymbol{F}(g \neq f \wedge \forall y \leq x[f(y) = g(y)])$ holds.*

Lemma 2.3.7. *(Limit point, lemma 6.7, [AS16]) Let $\mathfrak{L}$ be a InfFin-learnable class of computable structures. Then $\mathfrak{L}$ doesn't have any limit point, i.e. for any structure $\mathfrak{A}_e \in \mathfrak{L}$, there is $n \in \omega$ s.t.*

$$\forall \hat{\mathfrak{A}}_j \in \mathfrak{L}(\mathfrak{D}_{\hat{\mathfrak{A}}_j} \restriction n = \mathfrak{D}_{\mathfrak{A}_e} \restriction n \to \mathfrak{D}_{\hat{\mathfrak{A}}_j} = \mathfrak{D}_{\mathfrak{A}_e})$$

Proof: Let M be an InfFin-learner for $\mathfrak{L}$. Suppose that $\mathfrak{A} \in \mathfrak{L}$ is a limit point of $\mathfrak{L}$ and I is an informant for $\mathfrak{A}$.
Fix n $\in \omega$, s.t. $M(\lceil I_n \rceil)$ is an index for $\mathfrak{A}$.
By assumption $\mathfrak{A}_n \in \mathfrak{L}$. Suppose I' is an informant for $\mathfrak{A}_n$, s.t. $I'_n = I_n$, but $\mathfrak{A}_n \neq \mathfrak{A}$. Then $M(\lceil I'_n \rceil)$ is defined, but is not an index for $\mathfrak{A}_n$. It follows that M doesn't learn $\mathfrak{A}_n$ from informant. But $\mathfrak{A}_n \in \mathfrak{L}$. Then M doesn't InfFin-learn the class $\mathfrak{L}$. A contradiction. □

Theorem 2.3.6. *(Theorem 6.8, [AS16])*

(a) $\mathfrak{L}^+_{\mathfrak{A}} \in TxtFin \iff \mathfrak{D}_{\mathfrak{A}} = \omega$ (in other words all atomic formulae of the atomic diagram of $\mathfrak{A}$ are true for all numbers in ω).

(b) $\mathfrak{L}^+_{\mathfrak{A}} \in TxtEx \iff \mathfrak{D}_{\mathfrak{A}}$ is computable.

Proof:

(a) If $\mathfrak{D}_{\mathfrak{A}} = \omega$ then $\mathfrak{L}^+_{\mathfrak{A}} = \{\mathfrak{A}\}$.
On the other hand if $\mathfrak{D}_{\mathfrak{A}} \neq \omega$, then for any number $\lceil e, x_0 \rceil \notin \mathfrak{D}_{\mathfrak{A}}$, we have that the structures $\mathfrak{D}_{\mathfrak{A}_0} = \mathfrak{D}_{\mathfrak{A}}$ and $\mathfrak{D}_{\mathfrak{A}_1} = \mathfrak{D}_{\mathfrak{A}} \cup \{\lceil e, x_0 \rceil\}$ are in $\mathfrak{L}^+_{\mathfrak{A}}$ and $\mathfrak{A}_0 \subset \mathfrak{A}_1$. So by lemma 2.3.2 $\mathfrak{L}^+_{\mathfrak{A}} \notin TxtFin$.

(b) This follows from lemma 2.3.6 and theorem 2.3.4.

□

Definition 2.3.7. *A set $\mathbb{A} \subseteq \omega$ is autoreducible if there is a Turing functional Ψ such that $\mathbb{A}(x) = \Psi^{\mathbb{A}\setminus\{x\}}(x)$ for all numbers x.*

Definition 2.3.8. *A structure $\mathfrak{A}$ is autoreducible if there is a Turing functional Ψ such that $\chi_{\mathfrak{D}_\mathfrak{A}}(\lceil e,x\rceil) = \Psi^{\mathfrak{D}_\mathfrak{A}\setminus\{\lceil e,x\rceil\}}(\lceil e,x\rceil)$, for all $\lceil e,x\rceil \in \mathfrak{D}_\mathfrak{A}$.*

A join operation between two atomic diagrams $\mathfrak{D}_\mathfrak{A}, \mathfrak{D}_\mathfrak{B}$ of two structures $\mathfrak{A}, \mathfrak{B}$ with language L is defined as $\mathfrak{D}_\mathfrak{A} \oplus \mathfrak{D}_\mathfrak{B} = \{2x : x \in \mathfrak{D}_\mathfrak{A}\} \oplus \{2x+1 : x \in \mathfrak{D}_\mathfrak{B}\}$.

In other words a structure is autoreducible if it can be reduced to itself by a Turing machine, which doesn't ask its own input to the oracle. In computable theory autoreducibility is used to separate the polynomial-time hierarchy from polynomial space. The way separation is made is by showing that all Turing-complete sets for certain levels of the exponential-time hierarchy are autoreducible, but there exist some Turing-complete sets for doubly exponenital space that is not, see [Buh+00].
Any computable structure is autoreducible, because it can be computed without using an oracle.
For an arbitrary c.e. structure $\mathfrak{A}$, which for simplicity think as having one relation, the structure
$\mathfrak{D}_\mathfrak{A} \oplus \mathfrak{D}_\mathfrak{A} = \{2x : x \in \mathfrak{D}_\mathfrak{A}\} \cup \{2x+1 : x \in \mathfrak{D}_\mathfrak{A}\}$ is autoreducible. In order to compute $2x$ from $\mathfrak{D}_\mathfrak{A} \setminus \{2x\}$ ask whether $2x+1$ is in the oracle set.

Theorem 2.3.7. *(Theorem 6.18, [AS16]) The following are equivalent:*

1. *$\mathfrak{L}^+_\mathfrak{A}$ is InfEx-learnable.*
2. *$\mathfrak{D}_\mathfrak{A}$ is autoreducible.*

Proof: See Theorem 6.18, [AS16]. □

Definition 2.3.9. *(Finite variants) Let $\mathfrak{A}, \mathfrak{B}$ be two computable structures with language L. $\mathfrak{A}$ and $\mathfrak{B}$ are said to be finite variants of each other, if the set $\{\mathfrak{D}_\mathfrak{A} \setminus \mathfrak{D}_\mathfrak{B}\} \cup \{\mathfrak{D}_\mathfrak{B} \setminus \mathfrak{D}_\mathfrak{A}\}$ is finite.*

When two structures, say $\mathfrak{A}$ and $\mathfrak{B}$ over the same language are finite variants of each other, we will denote that by $\mathfrak{A} =^* \mathfrak{B}$.

Recall that $\mathfrak{L}^*_\mathfrak{A} = \{\hat{\mathfrak{A}} =^* \mathfrak{A}\}$, where $\mathfrak{A}$ is an arbitrary c.e. structure. The class $\mathfrak{L}^*_\mathfrak{A}$ contains all finite variants of the structure $\mathfrak{A}$.

Theorem 2.3.8. *(Theorem 6.23, [AS16]) For any c.e. structure $\mathfrak{A}$, we have that $\mathfrak{L}^*_\mathfrak{A} \in InfBC$.*

Proof: Let $\mathfrak{A} \in \mathfrak{L}^*_\mathfrak{A}$ be a c.e. structure. Let I be an informant for a structure $\mathfrak{A}$. A succesful InfBC-learner M, which learns the class $\mathfrak{L}^*_\mathfrak{A}$, is based on the following idea: The learner M will suppose that the structure given to it, say

$\hat{\mathfrak{A}}$, differs from the structure $\mathfrak{A}$ only on an initial segment revealed to it so far. In other words the output of M($\lceil\hat{\sigma}\rceil$); $\hat{\sigma} \sqsubseteq I$, and $ln(\sigma) = n$, where I is an informant for the structure $\hat{\mathfrak{A}}$, is an index for the set

$$\{\lceil e,x,i\rceil : \lceil e,x,i\rceil \in content(\sigma)\} \cup \{\lceil e,x,i\rceil : \lceil e,x,i\rceil > n \wedge \mathfrak{A} \models \phi_e(x) \vee \mathfrak{A} \not\models \phi_e(x)\}.$$

Moreover, given $\hat{\mathfrak{A}} \in \mathfrak{L}^*_{\mathfrak{A}}$, we have that $\hat{\mathfrak{A}}$ is a finite variant of $\mathfrak{A}$ and hence $M(\lceil\hat{\sigma}\rceil)$ is an index for $\hat{\mathfrak{A}}$ for sufficiently large n, n $\in \omega$. □

Recall that $\mathbf{FIN}_L$ denotes the class containing all finite substructures of structures with language L.

Theorem 2.3.9. *(Theorem 6.12, [AS16]) Let $\mathfrak{A}$ be an infinite computable structure with language L. Then $\boldsymbol{FIN}_L \cup \{\mathfrak{A}\} \in InfEx$.*

Proof: A successful InfEx-learner M for the class $\mathfrak{L} = \mathbf{FIN}_L \cup \{\mathfrak{A}\}$ works in the following way. Let I be an informant for a structure $\mathfrak{A}_j \in \mathfrak{L}$. Suppose I is such that numbers in it are arranged in ascending order. Furhtermore, let f be a computable function, which maps a finite segment $\sigma \sqsubseteq I$ to the canonical index for a structure $\mathfrak{A}_i \in \mathfrak{L}$. At each step s, we check whether $content(I_s) = \mathfrak{D}_{\mathfrak{A}_j} \upharpoonright s$. If it is, we output an index for $\mathfrak{A}_j$. Otherwise we get the canonical index for a structure $\mathfrak{A}_k$, where $content(I_s) = \mathfrak{D}_{\mathfrak{A}_k}$, i.e. $M(\lceil I_s\rceil) = f(I_s)$. □

Comparison between [Fin, Ex, BC]

Proposition 2.3.1. *$TxtFin \subsetneq TxtEx \subsetneq TxtBC$.*

Proof:

- TxtFin $\subsetneq$ TxtEx

  - $TxtFin \subseteq$ TxtEx.
    Suppose $\mathfrak{L}$ is TxtFin-learnable class of computable structures. We define a TxtEx-learner M for the class $\mathfrak{L}$. Suppose M' is a TxtFin-learner for the class $\mathfrak{L}$. What M does is to simulate M' on each input which receives. The learner M' outputs an index for a structure, once enough information is provided to it. As M' is TxtFin-learner, M' will output an index once, and never change its mind. By assumption the class is TxtFin-learnable, and M' is TxtFin-learner. Hence M will output only one index from the moment at which M' outputs an index. But M satisfies the requirements of TxtEx-learnability, i.e. M *doesn't* change its mind from the first moment on. Hence $\mathfrak{L} \in TxtEx$.
  - $TxtEx \nsubseteq TxtFin$.
    Let $\mathfrak{L} = \{\mathfrak{A}, \mathfrak{B}\}$, where $\mathfrak{A} \subset \mathfrak{B}$. Then by lemma 2.3.2 $\mathfrak{L} \notin TxtFin$. But $\mathfrak{L} \in TxtEx$. Suppose M is a TxtEx-learner for the class $\mathfrak{L}$. Let M be provided with information about $\mathfrak{B}$, but arranged in such a way that at first receives information for $\mathfrak{A}$. Then from thm. 2.3.2,

M at first will output an index for $\mathfrak{A}$, and once when information for $\mathfrak{B}$, not contained in $\mathfrak{D}_{\mathfrak{A}}$, is provided to M, then M will switch its output to an index for $\mathfrak{B}$. As M changes it's mind only once, and M correctly identifies each structure in the class, then M will correctly learn the class $\mathfrak{L}$. Hence $\mathfrak{L} \in TxtEx$.

- TxtEx $\subsetneq TxtBC$
  It follows from lemma 2.3.4 and lemma 2.3.6.

□

Proposition 2.3.2. *(Lemma 6.3., [AS16]) InfFin $\subsetneq$ InfEx $\subsetneq InfBC$.*

Proof:

- InfFin $\subsetneq InfEx$
  - InfFin $\subseteq InfEx$
    Let $\mathfrak{L}$ be an InfFin-learnable c.e. class of computable structures over a language L. We define a new learner M which InfEx-learns the class $\mathfrak{L}$. Suppose M' is an InfFin-learner for the class $\mathfrak{L}$. Once enough information about a structure from $\mathfrak{L}$ is provided to M', then M' will output an index, and won't change its guess from that point on. Hence once M starts to output a guess, it will not change its mind, too. By assumption the class $\mathfrak{L}$ is InfFin-learnable class, and M' is InfFin-learner for the class. Hence M' will be correct on each input, hence so M. As M doesn't change its mind from the point when made a guess, then M satisfies the requirements for InfEx-learnability.Then M will correclty identify $\mathfrak{L}$. Hence $\mathfrak{L} \in InfEx$.
  - $InfEx \nsubseteq InfFin$
    We can use the same argument from proposition 2.3.1, from the case $TxtFin \subsetneq TxtEx$, but only changing the way of information presentation.
- InfEx $\subsetneq$ InfBC.
  - InfEx $\subseteq$ InfBC
    This follows by definition of Explanatory and Behaviourly correct learnability.
  - InfBC $\nsubseteq$ InfEx
    We prove this direction using the class $\mathfrak{L}^+_{\mathfrak{A}}$.
    We have that $\mathfrak{L}^+_{\mathfrak{A}} \in InfEx$, if $\mathfrak{A}$ is autoreducible, which follows from thm. 2.3.7.
    But $\mathfrak{L}^+_{\mathfrak{A}} \in InfBC$, if $\mathfrak{A}$ is any c.e. structure, which follows from thm.

2.3.8, as $\mathfrak{L}^{+}_{\mathfrak{A}} \subseteq \mathfrak{L}^{*}_{\mathfrak{A}}$. As is shown in Appendix 7, from [AS16], there are c.e. sets, which are not autoreducible. Hence this direction is proved.

□

Lemma 2.3.8. *(Lemma 6.4., [AS16]) For $X \in \{Fin, Ex, BC\}$ we have that $TxtX \subsetneq InfX$.*

Proof:
($\rightarrow$) For each X $\in \{Fin, Ex, BC\}$, TxtX $\subseteq$ InfX.
Suppose that $\mathfrak{L}$ is a c.e. class of computable structures which is TxtX-learnable. Suppose that M is a TxtX-learner for the class $\mathfrak{L}$. Based on the behaviour of M, we define an InfEx-learner $\hat{M}$ for the class $\mathfrak{L}$. What $\hat{M}$ does, is to simulate M only on positive information from its input and output whatever M does. As the class $\mathfrak{L}$ is TxtX-learnable, the definition of $\hat{M}$ gives us that $\mathfrak{L}$ is InfX-learnable too. Hence $\mathfrak{L} \in InfX$.
($\leftarrow$) For each X $\in \{Fin, Ex, BC\}$, InfX $\nsubseteq$ TxtX.
We will prove this direction for each case.

- InfFin $\nsubseteq$ TxtFin
  Let $\mathfrak{L}$ be a class of computable structures. Suppose there are two atomic diagrams $\mathfrak{A}$, $\mathfrak{B}$, s.t. $\mathfrak{D}^{+}_{\mathfrak{A}} \subseteq \mathfrak{D}^{+}_{\mathfrak{B}}$, but $\mathfrak{D}_{\mathfrak{A}} \cap \mathfrak{D}_{\mathfrak{A}} \neq \emptyset$, i.e. both diagrams contains the same positive information (with some exceptions), wile their full diagram differs in their negative information.
  Then from lemma 2.3.2 we know that the class is not TxtFin-learnable. But the class is InfFin, as an InfFin-learner can use that information in order to find the correct structure.
- InfEx $\nsubseteq$ TxtEx
  It follows from 6.22 from [AS16], that $\mathfrak{L}^{-}_{\mathfrak{A}} \in TxtEx$ iff $\mathfrak{D}_{\mathfrak{A}}$ is finite, and $\mathfrak{L}^{-}_{\mathfrak{A}} \in InfEx$ if $\mathfrak{D}_{\mathfrak{A}}$ is autoreducible.
- InfBC $\nsubseteq$ TxtBC
  $\mathfrak{L}^{-}_{\mathfrak{A}} \in TxtBC$ if $\mathfrak{A}$ is finite. Assume otherwise i.e. that $\mathfrak{A}$ is infinite. Than for any finite substructure $\mathfrak{F} \subset \mathfrak{A}$, there is a finite variant $\hat{\mathfrak{A}}$ of $\mathfrak{A}$, such that $\mathfrak{F} \subset \hat{\mathfrak{A}} \subset \mathfrak{A}$. By definition the class $\mathfrak{L}^{-}_{\mathfrak{A}}$ contains all finite variants, in which a number is removed from the atomic diagram of $\mathfrak{A}$, so $\hat{\mathfrak{A}} \in \mathfrak{L}_{\mathfrak{A}}$. By corollary 2.3.3 we get that $\mathfrak{L}^{-}_{\mathfrak{A}} \notin TxtBC$.
  $\mathfrak{L}^{-}_{\mathfrak{A}} \in InfBC$ for an arbitrary c.e. structure $\mathfrak{A}$.
  As $\mathfrak{L}^{-}_{\mathfrak{A}} \subseteq \mathfrak{L}^{*}_{\mathfrak{A}}$ and from 2.3.8 we get that $\mathfrak{L}^{-}_{\mathfrak{A}} \in InfBC$.

□

Recall that we are working with atomic diagrams of structures, so our negative information in case of atomic diagrams provides us with information which relations in a structure don't hold over given arguments. Based on that observation, the next lemma is true in our case, while doesn't hold for languages.

Lemma 2.3.9. *InfFin # TxtEx*

Proof:

- $InfFin \nsubseteq$ TxtEx
  Let $\mathfrak{L}$ be an InfFin-learnable class of computable structures with language L. Furthermore, suppose that there are two structures from the class, which positive atomic diagram is equivalent, but they differ only on its negative information. On a text t then for one of those structures, a TxtEx-learner M can not make a difference which atomic diagram is provided to it. Hence M will fail to identify correctly both structures in the class. Hence $\mathfrak{L} \notin TxtEx$.

- TxtEx $\nsubseteq InfFin$
  Let $\mathfrak{L} = \{\mathfrak{A}, \mathfrak{B}\}$ be a c.e. class of computable structures. Suppose $\mathfrak{A} \subset \mathfrak{B}$.
  We have that $\mathfrak{L} \notin InfFin$, from lemma 2.3.3.
  But $\mathfrak{L} \in TxtEx$, from thm. 2.3.2.

□

Corollary 2.3.4. *(Corollary 6.13, [AS16]) InfEx $\nsubseteq$ TxtBC.*

Proof: This follows from thm. 2.3.5 and thm. 2.3.9. □

2.3.2 [Monotonic, Strong-monotonic, Weak-monotonic]

Learnability types, which we are going to present here, are concerned with the behaviour of a learner and the relation between conjectures made from it. Here we will be interested from learnability types in which a learner is expected to improve its conjectures over time. The stronger the monotonicity constraint is, the less anomalies the learnability type allows. If a monotonic learner is expected to improve its conjecture in increasing manner, but no anomalies are permitted, the learnability type is called strongly monotonic. When a monotonic learner is allowed to made conjectures to structures which differ from the expected one, but improve its guess over time, the learnability type is called monotonic. When a monotonic learner changes its mind only when an inconsistency with its current guess appears, the learnability type is calld weakly-monotonic learnability.

Recall that for a text t (or informant I) by $t_x(I_x)$ we refer to $t \upharpoonright x(I \upharpoonright x)$, for some $x \in \omega$. When we use $x + k$ as index, we will mean adding to a text (Informant) k more elements from an atomic diagram of a structure. For a structure $\mathfrak{A}$, when we refer to $\mathfrak{A}_x$, we will mean an index, found at stage x $\in \omega$, for the structure $\mathfrak{A}$.

Definition 2.3.10. *(Definition 3, [LZK96]) A learner M is said to identify a c.e. family of structures $\mathfrak{L}$ from text(or informant)*

- *strong-monotonically*
- *monotonically*
- *weak-monotonically*

iff M TxtEx(InfEx) identifies $\mathfrak{L}$, and for any text t(or informant I) of a structure $\mathfrak{A} \in \mathfrak{L}$ as well as for any two consecutive hypothesis j_x, j_{x+k}; x, $k \in \omega$, which M has produced, when fed t_x and t_{x+k} (I_x and I_{x+k}), the following conditions are satisfied:

- *(Strong monotonically) $\mathfrak{D}_{\mathfrak{A}_x} \subseteq \mathfrak{D}_{\mathfrak{A}_{x+k}}$*
- *(Monotonically) $\mathfrak{D}_{\mathfrak{A}_x} \cap \mathfrak{D}_{\mathfrak{A}} \subseteq \mathfrak{D}_{\mathfrak{A}_{x+k}} \cap \mathfrak{D}_{\mathfrak{A}}$, where we will suppose that $\mathfrak{A}$ corresponds to the structure expected to be learnt*
- *(Weak monotonically) if $content(t_{x+k}) \subseteq \mathfrak{D}_{\mathfrak{A}_x}$, then $\mathfrak{D}_{\mathfrak{A}_x} \subseteq \mathfrak{D}_{\mathfrak{A}_{x+k}}$ ($content^+(I_{x+k}) \subseteq \mathfrak{D}_{\mathfrak{A}_x}$ then $\mathfrak{D}_{\mathfrak{A}_x} \subseteq \mathfrak{D}_{\mathfrak{A}_{x+k}}$)*

Necessary Properties

Here we are going to give couple of extra conditions of learnability which we will need later when we compare different monotonic learnability criteria.

Theorem 2.3.10. *(Theorem 1, [LZ92b]) Let $\mathfrak{L}$ be a c.e. class of computable structures with language L. Then $\mathfrak{L} \in WMonTxt$ if and only if there are a space of hypothesis $\hat{\mathfrak{G}}_{\mathfrak{L}} = (\hat{\mathfrak{A}}_i)_{i\in\omega}$, where each $\hat{\mathfrak{A}}_i \in \mathfrak{L}$, and a c.e. family $(\hat{T}_i)_{i\in\omega}$ of finite non-empty sets such that*

1. $\boldsymbol{HS}_{\mathfrak{L}} = \hat{\mathfrak{G}}_{\mathfrak{L}}$.
2. *For all $i \in \omega$, $\hat{T}_i \subseteq \mathfrak{D}_{\hat{\mathfrak{A}}_i}$*
3. *For all i, $k \in \omega$, if $\hat{T}_i \subseteq \mathfrak{D}_{\hat{\mathfrak{A}}_k}$, then $\mathfrak{D}_{\hat{\mathfrak{A}}_k} \not\subset \mathfrak{D}_{\hat{\mathfrak{A}}_i}$.*

Proof: See Theorem 1, [LZ92b]. □

Theorem 2.3.11. *(Theorem 2, [LZ92b]) Let $\mathfrak{L}$ be a c.e. class of computable structures with language L. Then $\mathfrak{L} \in SMonTxt$, if and only if there is a space of hypothesis $\hat{\mathfrak{G}}_{\mathfrak{L}} = (\hat{\mathfrak{A}}_j)_{j\in\omega}$, where each $\mathfrak{A}_j \in \mathfrak{L}$, and a c.e. family $(\hat{T}_j)_{j\in\omega}$ of finite non-empty sets such that*

1. $\boldsymbol{HS}_{\mathfrak{L}} = \hat{\mathfrak{G}}_{\mathfrak{L}}$.
2. *$\hat{T}_j \subseteq \mathfrak{D}_{\hat{\mathfrak{A}}_j}$ for all $j \in \omega$*
3. *For all j, $z \in \omega$, if $\hat{T}_j \subseteq \mathfrak{D}_{\hat{\mathfrak{A}}_z}$, then $\mathfrak{D}_{\hat{\mathfrak{A}}_j} \subseteq \mathfrak{D}_{\hat{\mathfrak{A}}_z}$.*

Proof: See Theorem 2, [LZ92b]. □

Theorem 2.3.12. *(Theorem 4, [LZ92b]) Let $\mathfrak{L}$ be a c.e. class of computable structures with language L. Then $\mathfrak{L} \in MonTxt$, if and only if there is a space of hypothesis $\hat{\mathfrak{G}}_{\mathfrak{L}} = (\hat{\mathfrak{A}}_j)_{j\in\omega}$, where each $\mathfrak{A}_j \in \mathfrak{L}$, and a c.e. family $(\hat{T}_j)_{j\in\omega}$ of finite non-empty sets such that*

1. $\boldsymbol{HS}_{\mathfrak{L}} = \hat{\mathfrak{G}}_{\mathfrak{L}}$
2. *For all $\mathfrak{A} \in \mathfrak{L}$ and all $i \in \omega$*
   - $\hat{T}_i \subseteq \mathfrak{D}_{\mathfrak{A}_i}$
   - *if $\hat{T}_i \subseteq \mathfrak{D}_{\mathfrak{A}}$, then $\mathfrak{D}_{\mathfrak{A}} \not\subset \mathfrak{D}_{\mathfrak{A}_i}$*
   - *For all $\mathfrak{A} \in \mathfrak{L}$ and all i, $j \in \omega$,*
     - *if $i < j$, then $\hat{T}_i \subset \hat{T}_j$*
     - *if $i < j$, $\hat{T}_j \subseteq \mathfrak{D}_{\mathfrak{A}}$, then $\mathfrak{D}_{\mathfrak{A}_i} \cap \mathfrak{D}_{\mathfrak{A}} \subseteq \mathfrak{D}_{\mathfrak{A}_j} \cap \mathfrak{D}_{\mathfrak{A}}$.*
   - *For all $\mathfrak{A} \in \mathfrak{L}$, there is no infinite sequence $(k_j)_{j\in\omega}$ such that for all $j \in \omega$, $k_j < k_{j+1}$ and $\bigcup_j \hat{T}_{k_j} = \mathfrak{D}_{\mathfrak{A}}$.*

Proof: See Theorem 4, [LZ92b]. □

Comparison between types

Recall that $\mathbb{O}$ is the set of odd numbers, and $\mathbb{E}$ is the set of even numbers. We suppose again that our structures are with one unary relation.

We define the following classes of structures:

- $\mathfrak{L}^{-}_{\mathfrak{A}} = \{\mathfrak{A}_i\}_{i\in\omega}$ be a class of structures, where
  $\mathfrak{D}_{\mathfrak{A}_1} = \{\lceil e, o\rceil \mid o \in \mathbb{O}\}$,
  $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}_1} \setminus \{\lceil e, 2x+1\rceil\}$, for all i > 1 and $i = \lceil e, x\rceil \in \omega$.
- $\mathfrak{L}_{\langle .,.\rangle} = \{\mathfrak{A}_i\}_{i\in\omega}, n \in \omega$, where
  $\mathfrak{D}_{\mathfrak{A}_i} = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \setminus \{\bar{o}\} \cup \{\bar{e}\})\}$,
  where $\bar{o} = \{o_{i_1}, o_{i_2}, ..., o_{i_n}\}$, and $\bar{e} = \{e_{i_1}, e_{i_2}, ..., e_{i_n}\}$, s.t. $i_1 < i_2 ..., < i_n$, $\bar{o} \in \mathbb{O}^n, \bar{e} \in \mathbb{E}^n, i_k, k, n \in \omega$, and by $\bar{e}$ we can find effectively the index of a structure $\mathfrak{A}_i$.
- $\mathfrak{L}_{restr} = \{\mathfrak{A}_i\}_{i\in\omega}$, where
  $\mathfrak{D}_1 = \{\lceil e, o\rceil \mid o \in \mathbb{O}\}$
  $\mathfrak{D}_i = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \setminus \{2i+1\} \cup \mathbb{E} \upharpoonright (2i+1)), i \in \omega, i > 1\}$.

Next theorem shows a connection between different monotonic-like learnability types. Strong-monotonic learnability will be denoted by SMon, monotonic learnability by Mon and weak-monotonic learnability by WMon.

Theorem 2.3.13. *(Theorem 1, [LZK96]) $SMonTxt \subset MonTxt \subset WMonTxt$*

Proof:

- SMonTxt $\subsetneq$ MonTxt
  ($\rightarrow$) SMonTxt $\subseteq MonTxt$
  Suppose $\mathfrak{L}$ is SMonTxt-learnable c.e. class of computable structures. Suppose M is a SMonTxt-learner for the class $\mathfrak{L}$. We can define a MonTxt-learner M' for the class $\mathfrak{L}$ using the learner M. As by assumption the class $\mathfrak{L}$ is SMonTxt-learnable and M is SMonTxt-learner for the class, then there is a finite sequence of indecies s.t. $(*)\mathfrak{A}_{i_1} \subseteq \mathfrak{A}_{i_2} \subseteq ...\mathfrak{A}_{i_n}$, where $i_n \in \omega$ is the expected answer. But for any structure $\mathfrak{A}_j \in \mathfrak{L}$, and by definition of MonTxt-learnability we will have that $(**)\mathfrak{A}_{i_1} \cap \mathfrak{A}_j \subseteq \mathfrak{A}_{i_2} \cap \mathfrak{A}_j \subseteq ...\mathfrak{A}_{i_n} = \mathfrak{A}_j$. Hence on any input for a structure $\mathfrak{A}_j \in \mathfrak{L}$ and from $(*)$ M will correctly identify $\mathfrak{A}_j$, so M'. From $(**)$ we will have that M' satisfies the requirements for Mon-learnability. Hence M' is MonTxt-learner for the class $\mathfrak{L}$, which gives us that $\mathfrak{L} \in MonTxt$. As the class was an arbitrary SMonTxt-learnable class, this proves this direction.
  ($\leftarrow$)$MonTxt \nsubseteq SMonTxt$
  We prove this direction by showing that the class $\mathfrak{L}_{\langle .,. \rangle} \in MonTxt$, but the same class $\mathfrak{L}_{\langle .,. \rangle} \notin SMonTxt$.
  Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}_{\langle .,. \rangle}$. We define a MonTxt-learner M for the class $\mathfrak{L}_{\langle .,. \rangle}$ based on the following idea:
  Suppose at some stage M receives a number $e_k \in \mathbb{E}$. Then M will change its mind for a structure $\mathfrak{A}$, s.t. the maximum even number in the atomic diaram is e_k(at position k). Suppose at later stage M receives another even number $e_n \in \mathbb{E}$, s.t. $n > k$. Then again M will change its mind for a structure, s.t. the maximum even number is e_n, and contains e_k. As M will receive a finite number of even numbers, then M will change its mind finitely many times. Let's see now that M is a monotonic learner. Suppose M receives information for a structure $\mathfrak{A}_k$, s.t. $k = \lceil \lceil i, e_{j_1} \rceil, ... \lceil i, e_{j_n} \rceil \rceil$, where $\{j_n\}_{n \in \omega} \subset \omega$, $\{e_j\}_{j \in \{j_n\}} \in \mathbb{E}$ and $i \in \omega$. Suppose at some stage M receives its first even number, say e_m, s.t. $e_m < e_{j_n}$. Then M will change its mind for the structure $\mathfrak{A}_m$. Suppose at later stage M receives a number e_n, s.t. $e_m < e_n < e_{j_n}$, then M will change again its mind to the structure $\mathfrak{A}_n$. Continuing in this manner we will get a finite sequence of even numbers, until we reach e_{j_n}. It can be seen that $\mathfrak{D}_{\mathfrak{A}_m} \cap \mathfrak{D}_{\mathfrak{A}_{j_n}} \subseteq \mathfrak{D}_{\mathfrak{A}_n} \cap \mathfrak{D}_{\mathfrak{A}_{j_n}} \subseteq ... \subseteq \mathfrak{D}_{\mathfrak{A}_x} = \mathfrak{D}_{\mathfrak{A}_{j_n}}$. Hence M is indeed a monotonic learner, and M identifies the class $\mathfrak{L}_{\langle .,. \rangle}$. Hence $\mathfrak{L}_{\langle .,. \rangle} \in MonTxt$.
  Now we show that $\mathfrak{L}_{\langle .,. \rangle} \notin SMonTxt$.
  Let t be a text for a structure $\mathfrak{A}_j \in \mathfrak{L}_{\langle .,. \rangle}$, for j $\in \omega$. Suppose that M is a SMonTxt-learner for the class $\mathfrak{L}_{\langle .,. \rangle}$. Let's follow the reaction of M on text for a structure $\mathfrak{A} \in \mathfrak{L}$. Suppose the first even number $e_n \in \mathbb{E}$, for some n $\in \omega$, appears in the text. Then the learner M will output a canonical index for the structure $\mathfrak{A}_n$, in order to not miss that case, or M will fail to identify the structure and hence the class. Suppose at some later stage the learner M receives new even number, $e_t \in \mathbb{E}$, for some t $\in \omega$, s.t. $e_t > e_n$. Then the learner M is supposed to change its mind to an index for a structure $\lceil \langle i, e_n \rangle, \langle i, e_t \rangle \rceil$, but by definition of SMonTxt-

learnability is forbidden of doing so, because the structure $\mathfrak{A}_n \nsubseteq \mathfrak{A}_t$, as both structures are defined for odd numbers not defined in the other structure. Hence the learner M fails to SMonTxt-learn the class $\mathfrak{L}_{\langle .,. \rangle}$. Hence $\mathfrak{L}_{\langle .,. \rangle} \notin SMonTxt$.

- MonTxt $\subsetneq WMonTxt$
  ($\rightarrow$) MonTxt $\subseteq WMonTxt$
  Let $\mathfrak{L}$ be a MonTxt-learnable class. We define a WMonTxt-learner M for the class $\mathfrak{L}$. As the class is MonTxt-learnable, then from thm. 2.3.10 we know that there is a family c.e. family $(T_j)_{j\in\omega}$.
  $M(\lceil t_x \rceil)$ = "Search for the least i, s.t.
  $T_i \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$
  if such i is found, output i and request next input,
  otherwise output '?' and request next input."
  Let's prove that M indeed identifes $\mathfrak{L}$ and that M is WMonTxt-learner.
  Claim 1: The learner M correctly identifies the class $\mathfrak{L}$.
  Suppose otherwise, i.e. that M has output a wrong guess. Suppose i is the final answer of M, and j is the expeted guess, $i \neq j$. This means that the learner M has failed with the test $T_j \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. In other words, we will have that $T_j \nsubseteq content(t_x)$ or $content(t_x) \nsubseteq \mathfrak{D}_{\mathfrak{A}_j}$. In both cases, from the definition of M, then M will find a least index j, which passes the test, and will output it as a guess. By assumption the class is MonTxt-learnable and the family $(T_j)_{j\in\omega}$ satisfies the requirements from thm. 2.3.12. Then if j is the expected guess and i is the last given output, we will have that $i = j$. A contradiciton. Hence M works correctly, and it identifies the class $\mathfrak{L}$.
  Claim 2: The learner M is WMonTxt-learner.
  Suppose otherwise, i.e. M is not a WMonTxt-learner for the class $\mathfrak{L}$. Then either M doesn't correctly identify the class $\mathfrak{L}$, which we proved is not possible from the previous case, or M changes it mind infinitely often. If M changes its mind and never stops, then the class $\mathfrak{L}$ is not MonTxt-identifiable, which contradicts our assumption. Hence M is WMonTxt-learnable.
  ($\leftarrow$)$WMonTxt \nsubseteq MonTxt$
  We will prove this direction by showing that $\mathfrak{L}_{restr} \notin MonTxt$, but $\mathfrak{L} \in WMonTxt$.
  - $\mathfrak{L}_{restr} \notin MonTxt$.
    Assume by contradiction that $\mathfrak{L}_{restr} \in MonTxt$. Suppose that M is a MonTxt-learner for $\mathfrak{L}_{restr}$. Let t be a text for a structure $\mathfrak{A}_i \in \mathfrak{L}_{restr}$. Suppose at first the learner M receives only numbers from $\mathbb{O}$, then M will give as guess the structure $\mathfrak{A}_1$(if M doesn't output index for $\mathfrak{A}_1$, then when M receives it as input, M will fail to learn it as the class is infinite). Suppose an $e_j \in \mathbb{E}$ appears, s.t. $j < i$. Then the learner M is supposed to change its guess for the structure $\mathfrak{A}_j$ (at first the learner M doesn't know which structure receives, so it doesn't know the maximum even number). When M outputs it as guess, then M

will fail to be monotonic, because $\mathfrak{D}_{\mathfrak{A}_1} \cap \mathfrak{D}_{\mathfrak{A}_i} \not\subseteq \mathfrak{D}_{\mathfrak{A}_j} \cap \mathfrak{D}_{\mathfrak{A}_i}$. Hence $\mathfrak{L}_{restr} \notin MonTxt$.

– $\mathfrak{L}_{restr} \in WMonTxt$
A WMonTxt-learner M for the class $\mathfrak{L}_{restr}$ is defined in the following way. Let $\mathfrak{A}_i \in \mathfrak{L}$ and let t be a text for $\mathfrak{A}_i$. At first M outputs a canonical index for the set $\mathbb{O}$, i.e. an index for the structure $\mathfrak{A}_1$. In case an even number $e_z \in \mathbb{E}$ appears in t, the learner M changes its mind to an index for a structure $\mathfrak{A}_z$. If a new number $e_n > e_z$, where $e_n \in \mathbb{E}$ appears, then M changes its mind to the structure containing the new element. The learner M changes its mind every time when a new greatest even number appears, which is not contained in its current guess. As each set of even numbers is finite, M will change its mind finitely many times. Hence $\mathfrak{L}_{restr} \in WMonTxt$.

□

Theorem 2.3.14. *For $X \in \{SMon, Mon, WMon\}$, we have that $XTxt \subsetneq XInf$.*

Proof:

- XTxt$\subseteq$ XInf.
Let $\mathfrak{L} \in XTxt$ be a c.e. class of computable structures. Let M be a XTxt-learner for the class $\mathfrak{L}$. We define a XInf-learner M' for the class $\mathfrak{L}$ by using the behavour of M. Suppose I is an informant for $\mathfrak{A} \in \mathfrak{L}$. Suppose $\sigma \in \mathbf{SEQ}$, s.t. $\sigma \sqsubseteq I$. What M' does is to provide to M all positive examples from σ, ignoring all negative ones. As M is a learner for the class $\mathfrak{L}$, then M will learn each structure $\mathfrak{A} \in \mathfrak{L}$ from a text in any order. Hence M' will correctly identify each structure $\mathfrak{A} \in \mathfrak{L}$. Hence XTxt$\subseteq$ XInf.

- XInf $\not\subseteq$ XTxt.

  – SMonInf $\not\subseteq$ SMonTxt.
  Let $\mathfrak{L}'$ be a c.e. class of computable structures defined in the following way: $\mathfrak{D}_i = \{\lceil e, n \rceil \mid n \in \omega\} \setminus \{\lceil e, x \rceil\}$, where $i = \lceil e, x \rceil$. Then $\mathfrak{L}' \in SMonInf$, but $\mathfrak{L}' \notin SMonTxt$.
  (1) $\mathfrak{L}' \in SMonInf$.
  A SMonInf-learner M for the class $\mathfrak{L}'$ will wait for the first negative example to appear, say x_i and will output and index for $\mathfrak{A}_{x_i}$. As there is only one negative example, M will not change its mind from that point on. Hence $\mathfrak{L}' \in SMonInf$.
  (2) $\mathfrak{L}' \notin$ SMonTxt.
  Suppose otherwise, i.e. $\mathfrak{L}' \in SMonTxt$. Suppose M' is a SMonTxt-learner for the class $\mathfrak{L}'$. Let t be a text for a structure $\mathfrak{A}_i \in \mathfrak{L}'$. As t is a text for a structure $\mathfrak{A}_i$, then there is a stage, say $n_0 \in \omega$, when M will correctly identify the structure $\mathfrak{A}_i$. Suppose n_0 is the stage at which M doesn't receive the ith number, say $\lceil e, m \rceil$. Based on this behaviour we construct a new text t', where we rearrange t in such

a way to fool M. We define t' based on the following idea:
$\forall n < n_0(t'(n) = t(n)) \wedge (t'(n_0) = t(n_0 + k)) \wedge (t'(n_0 + k) = t(n_0)) \wedge$
$\forall n > n_0(t'(n) = t(n))$, for some k $\in \omega$.
The idea is to define t' in such a way that to fool M' in thinking that it has received information for another structure, and to wrongly identify that structure. Once M outputs an index for it, M will fail from that point on to be a strong monotonic learner, as the requirement $\mathfrak{D}_{\mathfrak{A}_i} \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ is not satisfied. Hence M' fails to strongly identify the class $\mathfrak{L}$. Hence $\mathfrak{L} \notin SMonTxt$.

– MonInf $\nsubseteq$ MonTxt.
We prove this direction by showing that $\mathfrak{L}^-_{\mathfrak{A}} \in MonInf$ and $\mathfrak{L}^-_{\mathfrak{A}} \notin MonTxt$.
(1) $\mathfrak{L}^-_{\mathfrak{A}} \in MonInf$.
A MonInf-learner M for the class $\mathfrak{L}^-_{\mathfrak{A}}$ at first will output an index for $\mathfrak{A}_1$. At later moment, when a negative example appears, say n_i, M will change its mind for a structure with index $i = \lceil e, n_i \rceil$. As $\mathfrak{D}_{\mathfrak{A}_1} \cap \mathfrak{D}_{\mathfrak{A}_i} \subseteq \mathfrak{D}_{\mathfrak{A}_i}$, the learner M will correctly identify each structure in the class. Hence $\mathfrak{L}^-_{\mathfrak{A}} \in MonInf$.
(2) $\mathfrak{L}^-_{\mathfrak{A}} \notin MonTxt$.
Suppose t is a text for the structure $\mathfrak{A}_1$. We will construct a modified text t_{fool} of t, with which we will make a MonTxt-learner M to wrongly output indecies $j_1, j_2 \neq i$, by which we will make M to fail to be a monotonic learner. Suppose t is arranged in ascending order. Suppose $n_0 \in \omega$ is the stage at which on input t M identifies the structure $\mathfrak{A}_1$. At first we give M an input $t(2), ..., t(n)$ until M wrongly guesses that the structure provided to it is $\mathfrak{A}_i$, s.t. $i = \lceil e, 1 \rceil$. Then we provide to M an input $t(2), ..., t(n), t(1), t(r+1), ...t(r+n-1)$ until M again makes a wrong output for the structure $\mathfrak{A}_j$ where $j = \lceil e, r \rceil$. Based on this idea we can define t_{fool} in the followowing way"
$t_{fool} = t(2), ..., t(n), t(1), t(r), ...t(r+n_1), t(r), t(r+1)....$ On this text M at first will output a guess for the structure $\mathfrak{A}_i$, once when M finds the missing elements, M will output an index for $\mathfrak{A}_j$. But M will fail to be monotonic, because $\mathfrak{D}_{\mathfrak{A}_i} \cap \mathfrak{D}_{\mathfrak{A}_1} \nsubseteq \mathfrak{D}_{\mathfrak{A}_j} \cap \mathfrak{D}_{\mathfrak{A}_1}$. Hence M can't identify the class $\mathfrak{L}^-_{\mathfrak{A}}$. Hence $\mathfrak{L}^-_{\mathfrak{A}} \notin MonTxt$.

– WMonInf $\nsubseteq$ WMonTxt.
To prove this direction we will use the class $\mathfrak{L}^-_{\mathfrak{A}}$. We have that $\mathfrak{L}^-_{\mathfrak{A}} \in WMonInf$, as an WMonInf-learner at first will output an index for $\mathfrak{A}_1$, and once when it receives a negative example will switch to the correct index.
But $\mathfrak{L}^-_{\mathfrak{A}} \notin WMonTxt$. Suppose otherwise, i.e. that $\mathfrak{L}^-_{\mathfrak{A}} \in WMonTxt$. Then from thm. 2.3.10 we have a c.e. family $(T_j)_{j \in \omega}$ of finite non-empty sets, satisfying the requirements from the theorem. Suppose then $T_0 \subseteq \mathfrak{D}_{\mathfrak{A}_1}$. Let $m = max\{n \mid n = k, \text{ where } \lceil e, k \rceil \in T_0\}$. Then from the definition of the class and requirements in the theorem we have that $T_0 \subseteq \mathfrak{D}_{\mathfrak{A}_m}$, but $(*)\mathfrak{D}_{\mathfrak{A}_m} \subseteq \mathfrak{D}_{\mathfrak{A}_1}$. From $(*)$ we get a contra-

diction with the 3th case in the theorem. Hence the class can not be WMonTxt-identified. Hence $\mathfrak{L}^{-}_{\mathfrak{A}} \notin WMonTxt$.

□

Theorem 2.3.15. *For $X \in \{Txt, Inf\}$, we have that*

- $XFin \subsetneq SMonX$
- $WMonX = XEx$

Proof: We prove this theorem only for text, where for informant can be proved in analogous way.

- $TxtFin \subsetneq SMonTxt$
  $(\leftarrow) TxtFin \subseteq SMonTxt$
  Suppose $\mathfrak{L}$ is a TxtFin-learnable class of structures with language L. Suppose M is TxtFin-learner for the class $\mathfrak{L}$. Once M receives enough evidence for a structure $\mathfrak{A} \in \mathfrak{L}$, the learner M will output an index for $\mathfrak{A}$ and will stop computation. We define a SMonTxt-learner M' for the class $\mathfrak{L}$ based on the behaviour of M. On input $\sigma \in \mathbf{SEQ}$, if M produces an output, an index for a structure, then M' produces the same output. As M learns each structure in the class $\mathfrak{L}$, then on each input $\sigma \in \mathbf{SEQ}$, M will give the same correct answer. Then M' can either be defined as ignoring the input once when M has produced an output, or it can simulate M on each input given the previous observation.
  M' = "On input $\sigma \in \mathbf{SEQ}$, s.t. $\sigma \sqsubseteq t$,
  output $\leftarrow M(\lceil \sigma \rceil)$
  if output $\neq '?'$
  give output as answer and request next input,
  else request next input"
  As M outputs a guess only once, M' will satisfy the requirements for SMonTxt-learnability. It follows that $\mathfrak{L} \in SMonTxt$. Hence this direction is proved.
  $(\rightarrow) SMonTxt \nsubseteq TxtFin$
  Let $\mathfrak{L} = \{\mathfrak{A}, \mathfrak{B}\}$ be a c.e. class of computable structures, s.t. $\mathfrak{A} \subset \mathfrak{B}$.
  Then we have that $\mathfrak{L} \in SMonTxt$, which follows from thm. 2.3.11.
  But $\mathfrak{L} \notin TxtFin$, which follows from lemma 2.3.2.
- WMonTxt = TxtEx It follows from thm. 2.3.10 and thm. 2.3.2.

□

By # we will denote that two learnable classes are different, i.e. for a class A, and a class B, we have that $A \nsubseteq B$ and $B \nsubseteq A$.

Theorem 2.3.16. *(Theorem 8, [LZK96])*

1. $MonInf \# TxtEx$

2. $MonInf\#WMonTxt$

3. $SMonInf\#MonTxt$

4. $InfFin\#SMonTxt$

Proof:

1. $MonInf\#TxtEx$
   $(\rightarrow) MonInf \nsubseteq TxtEx$
   We will prove this direction by showing that $\mathfrak{L}_k^- \in MonInf$, but $\mathfrak{L}_k^- \notin TxtEx$.

   - $\mathfrak{L}_{\mathfrak{A}}^- \in MonInf$.
     A MonInf-learner M works in the following way: At first M outputs a cannonical index j_1 for $\mathfrak{A}_1$. If at some step of the learnability process, some number k $\in \omega$ appears, which doesn't make the relation of a structure true, M changes its mind to a cannonical index for the structure $\mathfrak{A}_k$. Because $\mathfrak{A}_1 \cap \mathfrak{A}_k = \mathfrak{A}_k$, for each k > 1, this shows that the learner M works monotonically.
   - $\mathfrak{L}_{\mathfrak{A}}^- \notin TxtEx$.
     Suppose otherwise, i.e. that $\mathfrak{L}_{\mathfrak{A}}^- \in TxtEx$. Then from thm. 2.3.2, we have a c.e. family of computable non-empty sets $(T_j)_{j\in\omega}$ satisfying the requirements in theorem. Here we need to show that there is no such set in the family for the structure $\mathfrak{A}_1$, with which we will prove this direction. Suppose otherwise, i.e. that there is $T_0 \subseteq \mathfrak{D}_{\mathfrak{A}_1}$. Let $z = max\{k \mid k = n, \text{where } \lceil e, n \rceil \in T_0\}$. Then from the definition of the class, we get that $T_0 \subseteq \mathfrak{D}_{\mathfrak{A}_z}$. But again from the definition of $\mathfrak{L}$ we have that $\mathfrak{D}_{\mathfrak{A}_z} \subseteq \mathfrak{D}_{\mathfrak{A}_1}$, which contradicts the 2d case from the theorem. We reach a contradiction. Hence the class can not be TxtEx-identified, and $\mathfrak{L}_{\mathfrak{A}}^- \notin TxtEx$.

   $(\leftarrow) TxtEx \nsubseteq MonInf$
   We prove this direction by showing that $\mathfrak{L}_{restr} \in TxtEx$, but $\mathfrak{L}_{restr} \notin MonInf$.

   - $\mathfrak{L}_{restr} \in TxtEx$.
     A TxtEx-learner M when given an initial segment σ from the set $\mathbb{O}$ starts to output an index for the structure $\mathfrak{A}_1$. Once after a number $e_z \in \mathbb{E}$ appears, the learner M will switch to an index for the structure $\mathfrak{A}_z$. After that the learner M will change its guess when a new $e_n \in \mathbb{E}$ appears, when $e_n > e_z$. As $\mathbb{E}$ is finite, the learner M will reach a moment when correctly will output only one guess from that moment on. Hence $\mathfrak{L}_{restr} \in TxtEx$.
   - $\mathfrak{L}_{restr} \notin MonInf$.
     Assume by contradiction that M is MonInf-learner for the class $\mathfrak{L}_{restr}$. Suppose I is an informant for a structure $\mathfrak{A}_i \in \mathfrak{L}_{restr}$. Suppose at first M receives only positive examples from the set $\mathbb{O}$. Then M will

output an index for the structure $\mathfrak{A}_1$, in order to not miss a chance to guess the structure $\mathfrak{A}_1$. Suppose M receives a number $e_z \in \mathbb{E}$. Now if e_z is a positive example, and $z \leq i$, M is supposed to change its mind to the structure $\mathfrak{A}_z$. If M doesn't, then M will fail to learn the structure $\mathfrak{A}_i$, as we don't know the maximum even number for the structure. But if $z < i$, then the learner M will fail to be monotonic, because we will have that $\mathbb{O} \cap \mathfrak{A}_i \nsubseteq \mathfrak{A}_z \cap \mathfrak{A}_i$. Hence $\mathfrak{L}_{restr} \notin MonInf$.

2. $MonInf \# WMonTxt$

   - MonInf $\nsubseteq$ WMonTxt
     From corollary 2.3.15 we have that $WMonTxt = TxtEx$. From thm. 2.3.16, 1. we proved that $\mathfrak{L}_k^- \notin TxtEx$, but $\mathfrak{L}_k^- \in MonInf$. Hence this direction is proved.
   - WMonTxt $\nsubseteq MonInf$
     We showed that $\mathfrak{L}_{restr} \notin MonInf$ in the case of $MonInf \# TxtEx$. We have $\mathfrak{L}_{restr} \in WMonTxt$ from thm. 2.3.13.

3. $SMonInf \# MonTxt$
   $(\rightarrow) SMonInf \nsubseteq MonTxt$
   Let $\mathfrak{L}$ be a c.e. class of computable structures defined in the following way:
   Let $\mathfrak{D}_{\mathfrak{A}_1} = \{\lceil e, n \rceil \mid n \in \omega\}$.
   Let for $j > 1$:
   $\mathfrak{D}_{\mathfrak{A}_j} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \upharpoonright j \cup \{e_j\})\}$, where $e_j \in \mathbb{E}$.
   $\mathfrak{D}_{\mathfrak{A}_{j+1}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \upharpoonright (j+1) \cup \{e_j, e_{j+1}\})\}$, where $e_{j+1} \in \mathbb{E}$.

   - $\mathfrak{L} \in SMonInf$.
     A SMonInf-learner M is based on the following observation: In order to identify the structure $\mathfrak{A}_1$, M needs to wait for two non-consecutive or three positive even numbers $e_n \in \mathbb{E}$. If at first M finds one positive even number, say $e_k \in \mathbb{E}$, then M will change its mind for a structure $\mathfrak{A}_k$. If at later stage $e_z \in \mathbb{E}$ appears M can do the following: if $z = k+1$, then M will change its guess to a structure $\mathfrak{A}_z$, if $z > k+1$, M will change its guess to the structure $\mathfrak{A}_1$. From the definition of $\mathfrak{L}$ and the behaviour of M we get that $\mathfrak{A}_k \subseteq \mathfrak{A}_z \subseteq \mathfrak{A}_1$. But the behaviour of M is correct for any other structure, as M just needs to check the condition on which it is. Hence $\mathfrak{L}$ can be correctly SMonInf-identified. Hence $\mathfrak{L} \in SMonInf$.
   - $\mathfrak{L} \notin MonTxt$.
     Suppose otherwise, i.e. that $\mathfrak{L} \in MonTxt$. Then from thm. **??** we have that there is a c.e. family $(T_j)_{j \in \omega}$. The idea here will be to prove that there is no set T_i from the family, for the structure $\mathfrak{A}_1$. Suppose otherwise, i.e. that there is $T_0 \subseteq \mathfrak{D}_{\mathfrak{A}_1}$. Let $m = max\{n \mid n = k, \text{ where} \lceil e, k \rceil \in T_0\}$. By the definition of $\mathfrak{L}$ we have $T_0 \subseteq \mathfrak{D}_{\mathfrak{A}_k}$. But we reach a contradiction with thm. **??**, case 2, because $\mathfrak{D}_{\mathfrak{A}_k} \subseteq \mathfrak{D}_{\mathfrak{A}_1}$.

Hence there is no such $T_j \in (T_i)_{i\in\omega}$ for the structure $\mathfrak{A}_1$. Hence $\mathfrak{L}$ can not be MonTxt-identified. Hence $\mathfrak{L} \notin MonTxt$.

$(\leftarrow) MonTxt \not\subseteq SMonInf$
We have that $\mathfrak{L}_{\langle .,. \rangle} \in MonTxt$ from thm. 2.3.10.
We now show that $\mathfrak{L}_{\langle .,. \rangle} \notin SMonInf$. Suppose otherwise, i.e. that there is a SMonInf-learner M for the class $\mathfrak{L}_{\langle .,. \rangle}$. Suppose M receives a number $e_i \in \mathbb{E}$ and e_i is a positive example, then M will output an index for the structure $\mathfrak{A}_i$. Suppose at later stage another positive example $e_k \in \mathbb{E}$ appears, s.t. $e_k > e_i$. Then M is supposed to change its guess to the structure $\mathfrak{A}_k$, but is forbidden of doing so as by assumptuion M is SMonInf-learner, and changing its mind will break the requirement of SMonInf-learnability. Hence $\mathfrak{L} \notin$ SMonInf.

4. $InfFin\#SMonTxt$
$(\rightarrow) InfFin \not\subseteq SMonTxt$.
Let $\mathfrak{L}$ be a finite class of structures with language L.
Let $\mathfrak{A}_{\lceil i,1 \rceil}$ be s.t. $\mathfrak{D}_{\mathfrak{A}_{\lceil i,1 \rceil}} = \{\lceil e, n \rceil \mid n \in (\omega \setminus \{\lceil i, 1 \rceil\})\}$, and let for each $j > 1$, and $i < j$, $\mathfrak{A}_{\lceil i,j \rceil}$ be s.t. $\mathfrak{D}_{\mathfrak{A}_{\lceil i,j \rceil}} = \{\lceil e, n \rceil \mid n \in \{\omega \setminus \{\lceil e, i \rceil, \lceil e, i+1 \rceil ..., \lceil e, j-1 \rceil, \lceil e, j \rceil\}\}\}$.
We show then that $\mathfrak{L} \in InfFin$, but $\mathfrak{L} \notin SMonTxt$.

- $\mathfrak{L} \in InfFin$.
  An InfFin-learner M is based on the following idea. Let I be an informant for a structure $\mathfrak{A} \in \mathfrak{L}$. The learner M needs to wait a finite amount of time before making a correct guess(as by assumption the class is finite, see lemma ... and the negative information for each structure is finite). So $\mathfrak{L} \in InfFin$.
- $\mathfrak{L} \notin SMonTxt$.
  Suppose t is a text for the structure $\mathfrak{A}_{\lceil i,j \rceil}$. Let $i \neq 1$ and $j \neq 6$. We will construct a modified text t' of t, with which we will fool a SMonTxt-learner. Suppose we feed M with t $t(7), t(8), ..., t(z)$, until M outputs a wrong guess $j_{\lceil e,1 \rceil, \lceil e,2 \rceil, ... \lceil e,6 \rceil}$.
  We set $t' = t(7), t(8), ..., t(z), t(1), t(2), ..., t(6), ...$ We are done. On input t' M will wrongly output an index for a structure $\mathfrak{A}_{\lceil 1,6 \rceil} \neq \mathfrak{A}_{\lceil i,j \rceil}$, and will fail to be strongly monotonic. Hence $\mathfrak{L} \notin SMonTxt$.

$(\leftarrow) SMonTxt \not\subseteq InfFin$
Let $\mathfrak{L}$ be an infinite c.e. class of computable strutures, where for each $i, j \in \omega$, s.t. $i < j$, we have that $\mathfrak{D}_{\mathfrak{A}_{\lceil i,j \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction i \cup \mathbb{E} \restriction j)\}$. The class is SMonTxt-learnable. Suppose M is a SMonTxt-learner for the class $\mathfrak{L}$. We define M based on the following idea: Suppose M has received an odd number $o_n \in \mathbb{O}$, and even number $e_j \in \mathbb{E}$. Then M will output an index for a structure $\lceil i, j \rceil$, where o_n and e_j are the maximum elements. Suppose at later stage a new odd $o_k \in \mathbb{O}$, or even number $e_m \in \mathbb{E}$ appears, s.t. $o_n < o_k$, or $e_j < e_m$. Then if we have the case $o_n < o_k$, M will search for a structure $\lceil i', j' \rceil$, which maximum odd number is o_k, and is defined

for the least $e_{n'} \in \mathbb{E}$(as we have that by definition of the class $\mathfrak{L}$, $i' < j'$). The other case is analogous. Hence M wil change its mind in increasing number, when it will reach a moment at which will correctly identify the structure under consideration. Hence $\mathfrak{L} \in SMonTxt$.
But $\mathfrak{L} \notin InfFin$. This follows from lemma ... and lemma 2.3.7.

□

2.3.3 [Set-driven, Rearrangment independent]

Here we will look at two types of learnability, when the order of data doesn't matter. When we are not interested from the arrangment of data, the learnability type is called set-driven learning. When we add as constraint only the length of the available data, but the arrangment of data still doesn't matter the learnability type is called rearrangment independent.

Definition 2.3.11. *(Set-driven, [LZ04]) A learner M is said to be set-driven iff its output depends only on the range of its input; that is, iff* $M(\lceil t_x \rceil) = M(\lceil t_y^{'} \rceil)$ *(or for informant* $M(\lceil I_x \rceil) = M(\lceil I_y \rceil)$*), for all* $x, y \in \omega$*, all texts t,* $t^{'}$ *(or informants I, I′), provided that* $content(t_x) = content(t_y^{'})$ *(or* $content(I_x) = content(I_y)$*).*

Definition 2.3.12. *(Rearrangment independent, [LZ04]) A learner M is said to be rearrangment independent iff its output depends on the range and on the length of its input; that is iff* $M(\lceil t_x \rceil) = M(\lceil t_x^{'} \rceil)$ *(or for informants* $(M(\lceil I_x \rceil) = M(\lceil I_x^{'} \rceil))$ *for all* $x \in \omega$*, all texts t,* $t^{'}$ *(or informants I, I′) provided* $content(t_x) = content(t_x^{'})$ *(or* $content(I_x) = content(I_x^{'})$*).*

Set-driven learnability will be denoted by Sd. Rearrangment-independent learning will be denoted by r.

Comparison between types

While in the case of grammars it was shown that most types, concerned with hypothesis space, coinside (see [LZ04]), it's an open problem what is the relationship, when we observe classes of c.e. structures. First we will define three sets, which we will use trough the subsection.
Let

- $\mathbb{D}_1 = \{n \mid n \in \omega \wedge n \mod 3 = 2\}$,
- $\mathbb{D}_2 = \{n \mid n \in \omega \wedge n \mod 3 = 1\}$,
- $\mathbb{D}_3 = \{n \mid n \in \omega \wedge n \mod 3 = 0\}$.

We define the halting problem by $\mathbb{K} = \{\lceil k, k \rceil \mid \phi_k(k) \downarrow\}$, where $\{\phi_k\}_{k \in \omega}$ is an effective enumeration of all partial computable functions. By $\xi_k(n)$, for $n \in \omega$,

we will denote a function, which measures the number of steps for which a partial function with index k and input n converges (if it does), or we will have that $\xi_k(n) = \infty$ if that function doesn't converge.
All theorems here will be proved for the case of text, where for the case of informant can be proved in analogous way.

Theorem 2.3.17. *For $X \in \{Txt, Inf\}$, we have that SdXFin = XFin.*

Proof: Let $\mathfrak{L}$ be a c.e. TxtFin-learnable class of computable structures with language L.
We will use the c.e. class $(T_j)_{j\in\omega}$ from thm. 2.3.1. We know that such class exists, as by assumption the class is TxtFin-learnable.
Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$. We define a SdTxtFin-learner M for the class $\mathfrak{L}$ in the following way:
$M(\lceil t_x \rceil)$ = "if $ln(t_x) = 0$ or $ln(t_x) > 0$,
and there is no output for t_{x-1}
then execute instruction (A1)
(A1): Search for the least index j,
for which $content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$.
check whether or not $T_j \subseteq content(t_x)$
In case such j is found, output j and stop.
Otherwise output nothing and request next input. "

In order to prove that $\mathfrak{L} \in SdTxtFin$ we will prove the following two claims:

Claim 1: The learner M identifies finitely $\mathfrak{L}$.
Let $\mathfrak{A}_i \in \mathfrak{L}$ and let t be a text for $\mathfrak{A}_i$. Our purpose will be to show that the learner M stops sometimes, with output j, and that $\mathfrak{A}_i = \mathfrak{A}_j$. Suppose M has stopped with output $k \in \omega$. Then by definition of M and the c.e. family $(T_j)_{j\in\omega}$, there must be an $x \in \omega$, for which learner M has verified that $T_k \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_k}$. The only way which will prevent M stopping is if there exists $k' < k$, for which $\mathfrak{D}_{\mathfrak{A}_k} \subset \mathfrak{D}_{\mathfrak{A}_{k'}}$ and $T_{k'} \nsubseteq \mathfrak{D}_{\mathfrak{A}_k}$. But in this case, from the definition of M, we will have that M has found that $T_{k'} \subseteq content(t_y) \subseteq \mathfrak{D}_{\mathfrak{A}_{k'}}$. But the only way M to output a hypothesis in this case will be only when $\mathfrak{D}_{\mathfrak{A}_k} = \mathfrak{D}_{\mathfrak{A}_{k'}}$. Hence M will stop sometimes.
Suppose M when fed with t_x stops and outputs a hypothesis, say i. Hence from the definition of M, M has checked that $T_i \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$. Hence $T_j \subseteq \mathfrak{D}_{\mathfrak{A}_k}$. From second clause of thm. 2.3.1 we get that $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}_k}$. Hence Claim 1 is proved.

Claim 2: The learner M is set-driven.
Let $\mathfrak{A}, \mathfrak{A}' \in \mathfrak{L}$. Let t, t' be texts for $\mathfrak{A}$ and $\mathfrak{A}'$ respectively. Let x, y $\in \omega$ be two numbers, s.t. $t_x = t'_y$.
We need to show that $M(\lceil t_x \rceil) = M(\lceil t'_y \rceil)$. Suppose at stage x on text t, M has outputted an index i, and at stage y on t', M has outputted j. From the last it follows that $content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$, and $content(t'_y) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. By assumption we have that $t_x = t'_y$, hence $i = j$.

Suppose M has stopped on t_{x-1}. Then M has output a hypothesis, say j. We need to show then that $M(\lceil t'_y \rceil) = j$. The learner M is a finite learner for the class $\mathfrak{L}$. Then when M has output an index j, then by definition M has verified that $T_j \subseteq content(t_z) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$, for some $z < x$. By assumption $t_x = t'_y$, and hence $t_z \subseteq t'_y$. We need to observe two cases:
Case 1: M on input t'_{y-1} doesn't stop.
By definition then M will execute stage y on input t'_y. We have that j is the least index for which $T_j \subseteq content(t_z) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ and j is the correct index for the given structure, hence we can conclude that $content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. By assumption $t_x = t_y$, and $T_j \subseteq content(t_z) \subseteq content(t'_y)$, it follows that $T_j \subseteq content(t'_y) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. Then on input t'_y, M has output a hypothesis, say $i \in \omega$ and to stop. Suppose that $i \neq j$. Then from $T_j \subseteq content(t'_y) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$, we obtain that $i < j$. On the other hand $t_z \subseteq t'_y$ and $content(t'_y) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$ imply $content(t_z) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$. Therefore if $T_i \nsubseteq content(t_r)$, for all $r < x$, then M doesn't stop on input t_{x-1}. But by assumption M has stopped, which means it has verified that $T_i \subseteq content(t_z)$, since $z < x$. Hence M has to infer the structure $\mathfrak{A}_i$, but $i \neq j$, a contradiction. Hence M on stage y outputs an index j and stops.
Case 2: M on input t'_{y-1} stops.
By definition M has to infer a hypothesis, say i and to stop. Hence M has verified that $T_i \subseteq content(t'_r) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$. We need to show that $i = j$. Suppose otherwise, that $i \neq j$. As j is a least index which satisfies $T_j \subseteq content(t_z) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ and because the structure $\mathfrak{A}_j$ is under consideration and by assumptuon $t_x = t'_y$, we can conclude that $content(t'_r) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. Hence $i \neq j$ implies $i < j$. As $content(t_x) = content(t'_y) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$, it follows that $content(t_z) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$. But $M(\lceil t_z \rceil) = j$, and $i \neq j$. A contradiction. Hence $j = i$. □

Theorem 2.3.18. *(Theorem 2, [LZ96]) For $X \in \{Txt, Inf\}$, we have that SdXEx $\subsetneq$ XEx.*

Proof: Let $\mathfrak{L}$ be a c.e. class of computable structures with language L, defined in the following way:
$\mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \mathbb{E})\}$. Let $m = 2$.
For each k $\in \omega$ and all $j \in \omega^+$ we distinguish the following cases:
Case 1: $\neg\xi_k(k) \leq j$
Then we set $\mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}} = \mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}}$.
Case 2: $\xi_k(k) \leq j$
$\mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m\})\}$.

Claim 1: $\mathfrak{L} \notin SdTxtEx$.
We prove this Claim by showing that if $\mathfrak{L} \in SdTxtEx$, then we can construct an algorithm for solving the halting problem.

Lemma 1: Suppose M is a SdTxtEx-learner for the class $\mathfrak{L}$. Then we can use M to solve the halting problem.
Proof: We define an algorithm **Alg** using M, for solving the halting problem,

in the following way:
Alg: "On input k execute (A1).
(A1) Simulate M on input t_x, s.t.
$content(t_x) = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m\})\}$.
If M requests an input without outputting a hypothesis,
output $\phi_k(k)\uparrow$, and stop.
Otherwise, let $z = M(\lceil t_x \rceil)$.
Execute instruction (A2).
(A2) Test whether or not $\{\lceil e, n\rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m, 4\})\} \subseteq \mathfrak{D}_{\mathfrak{A}_z}$,
In case it is, output $\phi_k(k)\uparrow$, and stop.
Else output $\phi_k(k)\downarrow$ and stop."

By assumption, the learner M is expected to learn the class $\mathfrak{L}$. By definition M is a set-driven learner, so on input $content(t_x) = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m\})\}$, M is supposed to make a correct guess for the structure $\mathfrak{A}_{\lceil k,j\rceil}$. In other words, M on t, must output an index $\lceil k, j\rceil$, s.t. $\mathfrak{D}_{\mathfrak{A}_{\lceil k,j\rceil}} = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m\})\}$. Therefore, **Alg** terminates on every input k.
Let's show that **Alg** is correct. Suppose otherwise, i.e. **Alg** has output $\phi_k(k)\uparrow$, but was expected to output $\phi_k(k)\downarrow$. Then $\neg\xi_k(k) \leq j$. But then we will have that $\mathfrak{D}_{\mathfrak{A}_{\lceil k,j\rceil}} = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \mathbb{E})\}$. But by asumption M is a set-driven learner for the class $\mathfrak{L}$, so the behaviour of **Alg** is correct.
Suppose now that **Alg** has output $\phi_k(k)\downarrow$. Then we know that $j \neq 0$, and $\mathfrak{D}_{\mathfrak{A}_{\lceil k,j\rceil}} = \{\lceil e, n\rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m\})\}$. From the definition of $\mathfrak{L}$ we then get that $\xi_k(k) \leq j$. Hence $\phi_k(k)$ is indeed defined.
But we have that the halting problem is unsolvable. Then from the definition of $\mathfrak{L}$ and the behaviour of M, M will wrongly guess an index for a structure. Hence M is not a SdTxtEx-learner for $\mathfrak{L}$. From this it follows that the class is not SdTxtEx-learnable. This proves Lemma 1.

Claim 2: $\mathfrak{L} \in TxtEx$.
We define a TxtEx-learner M' for the class $\mathfrak{L}$. Suppose t is a text for a structure $\mathfrak{A} \in \mathfrak{L}$.
$M(\lceil t_x \rceil) =$ "
Determine the maximum k, s.t.
$o_k = max\{o \mid o = n, \text{ where} \lceil e, n\rceil \in content(t_x)\}$,
if there is not such k, output nothing and request next input
otherwise test whether or not $\xi_k(k) \leq x$.
In case it is, go to (1).
Otherwise, output index $\lceil k, 0\rceil$ and request next input.
(1) Check whether or not $content(t_x)$ contains a number
two numbers $\lceil e, n_1\rceil$ s.t. $n_1 \in \mathbb{E}$, s.t.
$n_1 \neq m$.
In case it is output $\lceil k, 0\rceil$ and request next input.
Otherwise go to (2).
(2) check whether $m \in content(t_x)$
In case it is, output $\lceil k, x\rceil$, and request next input.

Otherwise output nothing, and request next input. □

Theorem 2.3.19. *For $X \in \{Txt, Inf\}$, we have that $SdWMonX \subsetneq WMonX$.*

Proof: We have by thm. 2.3.15 that $WMonX = TxtEx$. Then we can use the same argument as in thm. 2.3.18 in order to prove this theorem. □

Theorem 2.3.20. *For $X \in \{Inf, Txt\}$, we have that $SdSMonX \subsetneq SMonX$.*

Proof: Let $\mathfrak{L}$ be a c.e. class of computable structures with language L. For all $k, i \in \omega$, we set
$\mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \mathbb{E})\}$, where $\mathfrak{A}_{\lceil k,i \rceil} \in \mathfrak{L}$. For each k $\in \omega$ and all $j \in \omega^+$ we distinguish the following cases:
Case 1: $\neg \xi_k(k) \leq j$
Then we set $\mathfrak{D}_{\mathfrak{A}_{\lceil k,i \rceil}} = \mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}}$.
Case 2: $\xi_k(k) \leq j$
$\mathfrak{D}_{\mathfrak{A}_{\lceil k,i \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \upharpoonright k \cup \{m\}\}$, where $m = 4$. Case 1: $\mathfrak{L} \notin SdSMonTxt$. We prove this Claim, by showing that if $\mathfrak{L} \in SdSMonTxt$, then we can construct an algorithm for solving the halting problem.
We define an algorithm **Alg** in the following way:
Alg: "On input k execute (B1) and (B2).
(B1): For $z \in \omega$,
generate successively the canonical text t of $\mathfrak{A}_{\lceil k,0 \rceil}$
until M on input t_z
outputs for the first time a hypothesis j
s.t. $content(t_z) \cup \{\mathbb{O} \upharpoonright k \cup \{m\}\} \subseteq \mathfrak{D}_{\mathfrak{A}_j}$.
(B2): Test whether $\xi_k(k) \leq z + 1$.
In case it is, output $\phi_k(k) \downarrow$, and stop.
Otherwise output $\phi_k(k) \uparrow$, and stop.

The learner M is suppose to output an index for a structure $\mathfrak{A}_j \in \mathfrak{L}$. on input t_z outputs an index j, which satisfies the test $(*) content(t_z) \cup \{\mathbb{O} \upharpoonright k \cup \{m\}\} \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. Furthermore the test (*), can be effectively performed, since the membership in $\mathfrak{D}_{\mathfrak{A}_j}$ is uniformly decidable. By assumption (B2) is effectively executable. Hence **Alg** is computable.
Let's see that $\phi_k(k) \uparrow$, if $\xi_k(k) \leq z + 1$. Suppose otherwise, i.e. there exists a $y > z+1$, s.t. $\xi_k(k) = y$. By definition of $\mathfrak{L}$, we have that t_z is an initial segment of a text $\hat{t}$ for a structure $\mathfrak{A}_i \in \mathfrak{L}$, s.t. $i \neq \lceil k, 0 \rceil$. The learner M is set-drivent and strongly monotonic, which means that M is supposed to output an index for the structure $\mathfrak{A}_i$, satisfying the requirements of strong monotonicity. But M doesn't identify $\mathfrak{A}_i$ from its text $\hat{t}$, as M outputs an index $M(\lceil t_{z+r} \rceil) = j$, where $j = \lceil k, 0 \rceil; \forall r \in \omega$. Hence M fails to identify $\mathfrak{L}$.

Claim 2: $\mathfrak{L} \in SMonTxt$.
We define the following MonTxt-learner for the class $\mathfrak{L}$, with which the claim will be proved and hence the theorem. Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$.
$M(\lceil t_x \rceil) =$ "

find the maximum k, s.t. $o_k = max\{o \mid o = n \wedge \lceil e, n \rceil \in content(t_x) \wedge n \in \mathbb{O}\}$
Test whether or not $\xi_k(k) \leq x$.
In case it is, go to (1).
Otherwise, output nothing and request next input.
(1) Test whether or not there is $n \neq m$,
s.t. $\lceil e, n \rceil \in content(\sigma)$
In case it is output $\lceil k, 0 \rceil$ and request next input. Otherwise go to (2).
(2) Check if $\lceil e, m \rceil \in content(t_x)$.
in case it is, output $\lceil k, x \rceil$ and request next input.
Otherwise output nothing, and request next input. □

When comparing strong monotonic learnbility with rearrangment independent learnability, we see that both have the same learnability power.

Theorem 2.3.21. *(Theorem 8, [LZ04]) For $X \in \{Txt, Inf\}$, we have that $rSMonX = SMonX$.*

Proof: Let $\mathfrak{L} \in SMonTxt$ be a c.e. class of computable structures. By applying thm. 2.3.11 to structures, we get a c.e. family $(T_j)_{j \in \omega}$ of finite non-empty texts satisfying the requirements from the theorem.
Using the family $(T_j)_{j \in \omega}$ we define a rSMonTxt machine M, which learns the class $\mathfrak{L}$. Let $\mathfrak{A} \in \mathfrak{L}$, and let t be a text for $\mathfrak{A}$, let x $\in \omega$.
M($\lceil t_x \rceil$) = "Search for the least $j \leq x$ for which $T_j \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$.
If such j is found,
output j and request next input.
Otherwise, output '?' and request next input. "

The machine M is defined as rMonTxt-learner(see def. 2.3.12).

Next step is to show that M is SMonTxt-learner for the class $\mathfrak{L}$.
Claim 1: The learner M learns the structure $\mathfrak{A}$ on text t.
Let j $\in \omega$ be the least, s.t. $\mathfrak{D}_{\mathfrak{A}_j} = \mathfrak{D}_{\mathfrak{A}}$. Then there is a least $n \in \omega$, for which $T_j \subseteq content(t_n)$. By condition we have that for all $k < j$, $\mathfrak{D}_{\mathfrak{A}_k} \subset \mathfrak{D}_{\mathfrak{A}}$ (otherwise we will have $\mathfrak{D}_{\mathfrak{A}_k} = \mathfrak{D}_{\mathfrak{A}_j} = \mathfrak{D}_{\mathfrak{A}}$ from the clause (2) from thm. 2.3.11). Then there exists a stage $y \in \omega$ for which $content(t_y) \nsubseteq \mathfrak{D}_{\mathfrak{A}_k}$ for all $k < j$, for which $\mathfrak{D}_{\mathfrak{A}_k} \subseteq \mathfrak{D}_{\mathfrak{A}}$. Hence M($\lceil t_{y+z} \rceil$) = j, for all $z \in \omega$.

Claim 2: The learner M is a strong monotonic learner.
Let $M(\lceil t_x \rceil) = j$ and $M(\lceil t_{x+r} \rceil) = k$ for some $x, j, k \in \omega, r \in \omega^+$. Then M first has found that $T_j \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$, and later has changed its mind to a new index k, s.t. $T_k \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_k}$. But the family $(T_j)_{j \in \omega}$ satisfies the requirements from thm. 2.3.11. Hence we will have that $T_j \subseteq \mathfrak{D}_{\mathfrak{A}_k}$, hence $\mathfrak{D}_{\mathfrak{A}_j} \subseteq \mathfrak{D}_{\mathfrak{A}_k}$. From the last we get that M is indeed a strong monotonic learner. □

An obvious observation is:

Corollary 2.3.5. *For $X \in \{Txt, Inf\}$, $XFin \subsetneq rSMonX$.*

For a set $\mathbb{D} \subseteq \omega$, by $\mathbb{D}[m]$ we will refer to the m-th element of the set $\mathbb{D}$, for $m \in \omega$.
Recall that for a segment $\sigma \in \mathbf{SEQ}$, where $ln(\sigma)$ we have that $\sigma^- = \sigma \restriction (n-1)$, if $n > 1$, and $\sigma^- = \sigma$, otherwise.
While rearrangment independence does not restrict strong monotonic learnability, it restricts monotonic learnaility type. Based on Theorem 10 from [LZ93b] we can prove that.

Theorem 2.3.22. *For $X \in \{Txt, Inf\}$, $rMonX \subsetneq MonX$.*

Proof:

$(\leftarrow) rMonTxt \subseteq MonTxt$
Let $\mathfrak{L}$ be a c.e. class of computable structures, s.t. $\mathfrak{L} \in rMonTxt$. We define a learner M which MonTxt-identifies $\mathfrak{L}$. Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$. Based on thm. 2.3.12, we use the family $(T_j)_{j\in\omega}$, which we know it exists as the class is rMonTxt-learnable.

$M(\lceil t_x \rceil)$ = "Search for the least index i,
s.t. $T_i \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$.
If such i is found,
output i and request next input.
Otherwise output '?' and request next input."
We prove that M MonTxt-identifies the class $\mathfrak{L}$.

Case 1: M learns each structure in $\mathfrak{L}$.
Suppose otherwise, i.e. that there is a structure $\mathfrak{A} \in \mathfrak{L}$, which M fails to identify. Suppose $\mathfrak{A}_i$ is that structure. Then by definition M has failed with the test $T_i \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$. I.e. M has found an index $j < i$, s.t. $T_j \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. Then by definition of $(T_j)_{j\in\omega}$, we will have that $T_j \cap \mathfrak{D}_{\mathfrak{A}_j} \subseteq T_i \cap \mathfrak{D}_{\mathfrak{A}_j}$, and $\mathfrak{D}_{\mathfrak{A}_i} \cap \mathfrak{D}_{\mathfrak{A}_j} \subseteq \mathfrak{D}_{\mathfrak{A}_j}$. But in order M to learn the structure $\mathfrak{A}_i$, then at some moment M has received elements not contained in $\mathfrak{A}_j$ and by definition M has found an index which passes the test, which is different from j. I.e. the test $T_j \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ has failed at some moment and M will identify the next index which will pass the test based on the information available to M. Based on that observation and the fact that $(T_j)_{j\in\omega}$ is defined based on thm. 2.3.12, we get that M is correct.

Case 2: M is monotonic.
Suppose otherwise, i.e. there are indecies j_1, j_2, $j_1 < j_2$, s.t. $\mathfrak{D}_{\mathfrak{A}_{j_1}} \cap \mathfrak{D}_{\mathfrak{A}} \nsubseteq \mathfrak{D}_{\mathfrak{A}_{j_2}} \cap \mathfrak{D}_{\mathfrak{A}}$. Then M has failed with the test $T_{j_1} \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_{j_1}}$. But then we have that $T_{j_1} \subseteq T_{j_2}$ and $T_{j_2} \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_{j_2}}$, i.e. $content(t_x)$ contains an element not contained in previous stages. Hence M correctly changes its mind. As each T_j in the family $(T_j)_{j\in\omega}$ satisfies the requirements from thm. 2.3.12, M is monotonic. Hence $\mathfrak{L} \in MonTxt$.

$(\rightarrow) MonTxt \nsubseteq rMonTxt$

To prove this direction we will use the sets $\mathbb{D}_1, \mathbb{D}_2$ and $\mathbb{D}_3$.
We define a class of c.e. structures $\mathfrak{L}$ based on their atomic diagrams in the following way:

$$\mathfrak{D}_{\mathfrak{A}_{4k+z}} = \begin{cases} \{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_2[k]\} \cup \mathbb{A}_k)\}, & \text{, if z} = 0 \\ \{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_3[k]\} \cup \mathbb{B}_k)\}, & \text{, if z} = 1 \\ \{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_2[k]\} \cup \{\mathbb{D}_3[k]) \cup \mathbb{A}_k)\}, & \text{, if z} = 2 \\ \{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_2[k]\} \cup \{\mathbb{D}_3[k]) \cup \mathbb{B}_k)\}, & \text{, if z} = 3 \end{cases}$$

where we define the sets $\mathbb{A}_k$ and $\mathbb{B}_k$ with their respective characteristic functions.

$$\chi_{\mathbb{A}_k}(n) = \begin{cases} 1, & n \in (\mathbb{D}_2 \upharpoonright k \cup \{\mathbb{D}_1[m]\}) \wedge \xi_k(k) = m \\ 0, & otherwise \end{cases}$$

$$\chi_{\mathbb{B}_k}(n) = \begin{cases} 1, & n \in (\mathbb{D}_3 \upharpoonright k \cup \{\mathbb{D}_1[m]\}) \wedge \xi_k(k) = m \\ 0, & otherwise \end{cases}$$

Now we prove that:
(1) $\mathfrak{L} \in MonTxt$
(2) $\mathfrak{L} \notin rMonTxt$

Claim 1: $\mathfrak{L} \in MonTxt$.
We define a monotonic learner M, which for each text t for a structure $\mathfrak{A} \in \mathfrak{L}$, will give as output an index for $\mathfrak{A}$. Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$.
M = "On input $\sigma \in$ **SEQ**, s.t. $\sigma \sqsubseteq t$,
if $\sigma = \emptyset$, or $M(\lceil \sigma^- \rceil) = '?'$,
then execute (1)
else execute (2)

(1) Check whether or not for some k $\in \omega$:,
$(a1)\{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_2[k]\})\} \subseteq content(\sigma)$ or
$(a2)\{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_3[k]\})\} \subseteq content(\sigma)$.
If (a1) is true, output $4k$ and request next input,
else output $4k + 1$ and request next input.

(2) Let j = $M(\lceil \sigma^- \rceil)$.
(a1) Check whether or not $content(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$.
If (a1) is true, output j and request next input,
otherwise go to (3).

(3) If j = $4k$ or j = $4k + 1$, output $j + 2$ and request next input,
else if j = $4k + 2$ output $4k + 3$ and request next input,
otherwise output $4k + 2$ and request next input."

From the definition of the learner M, it can be seen that for a custom k $\in \omega$, for an abitrary text t for a structure with index 4k or $4k + 1$, M works monotonically. It remains to show that for an arbitrary text t for

a structure with index $4k+2$ or $4k+3$, M remains monotonic. Let t be an arbitrary text for a structure with index $4k+2$ or $4k+3$. We need to observe two cases:
Case 1: $\phi_k(k)\uparrow$
Then we get that $\mathfrak{D}_{\mathfrak{A}_{4k+2}} = \mathfrak{D}_{\mathfrak{A}_{4k+3}}$. As we have that t is a text for a structure with index $4k+2$, then there is a moment n $\in \omega$, and $\sigma \in \mathbf{SEQ}$, s.t. $\sigma = t_n$ and $content(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}_{4k+2}}$. This gives us that for all $r \in \omega^+$, $M(\lceil t_{n+r} \rceil) = j$, where j is an index for a structure $\mathfrak{A}_{4k+2}$. Furthermore the learner M has changed its mind at most once. Hence M is monotonic.
Case 2: $\phi_k(k)\downarrow$
We have that $\mathfrak{D}_{\mathfrak{A}_{4k+2}}$ and $\mathfrak{D}_{\mathfrak{A}_{4k+3}}$ are finite sets, and can be seen that the learner M will give a correct guess. There are subceses to observe.
Subcase 2.1.: t is a text for $\mathfrak{D}_{\mathfrak{A}_{4k+2}}$.
If the learner M first outputs $4k$, then M will need only one mind change to correctly output $4k+2$. Hence M is monotonic.
On the other hand if M first gives as guess $4k+1$, then M produces the hypotheses $4k+1, 4k+3, 4k+2$. By definition of this type of monotonic learnability, we get that $\mathfrak{D}_{\mathfrak{A}_{4k+1}} \cap \mathfrak{D}_{\mathfrak{A}_{4k+2}} \subseteq \mathfrak{D}_{\mathfrak{A}_{4k+3}} \cap \mathfrak{D}_{\mathfrak{A}_{4k+2}} \subseteq \mathfrak{D}_{\mathfrak{A}_{4k+2}}$. This shows that M is monotonic.
Subcase 2.2.: t is a text for $\mathfrak{D}_{\mathfrak{A}_{4k+3}}$.
If the learner M first produces output $4k+1$, then again M will need one mind change to output a correct guess $4k+3$.
Otherwise M will produce as guess $4k, 4k+2, 4k+3$. In both cases we get that M works monotonically.

Claim 2: $\mathfrak{L} \notin rMonTxt$.
Suppose that there is a $rMonTxt - learner$ M which witnesses that $\mathfrak{L} \in rMonTxt$.

Claim 3: Given any algorithm for the learner M which witnesses that $\mathfrak{L} \in rMonTxt$ one can define an algorithm solving the halting problem.
Let's define such algorithm.
Alg = "On input k execute (A1)
until one of both (a1) or (a2) is successful.
After that execute (A2).
(A1) : for x $\in \omega$ execute (a1) and (a2) simultaneously, until one of them becomes successful.
(a1) Test whether $\xi_k(k) \leq x$.
(a2) Simulate M on inputs t_n and $\hat{t}_n$, where
$\{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \mathbb{D}_2[k])\} \subseteq content(t_n)$,
and $\{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \mathbb{D}_3[k])\} \subseteq content(\hat{t}_n)$.
If M on both inputs outputs two different hypothesis,
say j and $\hat{j}$, then test whether
$\{\lceil e, x \rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \mathbb{D}_2[k])\} \subseteq \mathfrak{D}_{\mathfrak{A}_j}$,
$\lceil e, \mathbb{D}_3[k] \rceil \notin \mathfrak{D}_{\mathfrak{A}_j}$,

$\{\lceil e, x\rceil \mid x \in (\mathbb{D}_1 \upharpoonright k \cup \mathbb{D}_3[k])\} \subseteq \mathfrak{D}_{\mathfrak{A}_j}$,
$\lceil e, \mathbb{D}_2[k]\rceil \notin \mathfrak{D}_{\mathfrak{A}_j}$.
(A2): If (a1) happens first, output $\phi_k(k)\downarrow$ and stop.
Otherwise simultaneously execute (b1) and (b2) until one of them becomes successful.
(b1) Test whether $\xi_k(k) \leq x + y$.
(b2) Test whether M generates a consisent hypothesis n,
i.e. $M(\lceil t_{x+y}\rceil) = n$ and $content(t_{x+y}) \subseteq \mathfrak{D}_{\mathfrak{A}_n}$.
If (b1) happens first, then output $\phi_k(k)\downarrow$ and stop.
Otherwise output $\phi_k(k)\uparrow$ and stop.

Both (a1) and (b1) can be effectively performed. Furthermore membership is uniformly decidable which from which it follows that (a2) and (b2) can be effectively performed, too. From this it follows that **Alg** is computable.
Let's see that **Alg** terminates for all k $\in \omega$. Assume otherwise, i.e. that there exists k $\in \omega$, s.t. (A1) doesn't terminate. Then $\phi_k(k)$ diverges. Because (a2) will never terminate successfully, then M will fail to identify one of the sets $\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_2[k]\}$ or $\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_3[k]\}$. We reach a contradiction. The same argument applied to (A2) gives us a contradiction. Hence **Alg** is defined for all $k \in \omega$.

Let's see that **Alg** works correctly.
When the algorithm stops with $\phi_k(k)\downarrow$, then $\phi_k(k)$ is defined. Assume otherwise, that $\phi_k(k)\uparrow$, but $\phi_k(k)$ is defined. From the definition of **Alg** it follows that there exists x $\in \omega$, s.t. $M(\lceil t_x\rceil) = n$ and $M(\lceil \hat{t}_x\rceil) = \hat{n}$, where t_x is a text for $\mathfrak{A}_{4k}$ and $\hat{t}_x$ is a text for $\mathfrak{A}_{4k+1}$. Furthermore there exists y $\in \omega$, s.t. $M(\lceil t_{x+y}\rceil) = n$ and $M(\lceil \hat{t}_{x+y}\rceil) = n$ (which we get as result, because M is rearrangment independent).

Because we have that $\mathbb{D}_1 \upharpoonright k \cup \{\mathbb{D}_2[k]\} \cup \{\mathbb{D}_3[k]\} \subseteq \mathfrak{D}_{\mathfrak{A}_n}$, where $\mathfrak{A}_n \in \mathfrak{L}$, we distinguish two cases.
Case 1: $\mathfrak{A}_k = \mathfrak{A}_{4k+2}$. The text $\hat{t}_{x+y}$ is an initial segment of a text for $\mathfrak{A}_{4k+3}$, too. In some subsequent step of its computation, M has already generated hypothesis m and n. As $\phi_k(k)$ is defined, then $\mathfrak{D}_{\mathfrak{A}_{4k+1}} \cap \mathfrak{D}_{\mathfrak{A}_{4k+3}} \nsubseteq \mathfrak{D}_{\mathfrak{A}_{4k+2}} \cap \mathfrak{D}_{\mathfrak{A}_{4k+3}}$. Hence M is not a monotonic learner.
Case 2: $\mathfrak{A}_k = \mathfrak{A}_{4k+3}$. We use the same argument as before, and get that M is not a monotonic learner.
With this we prove the correctness of our algorithm. Furthermore we have that the halting problem is undecidable. Hence the lemma is proved.

□

From thm. 2.3.22 we can conclude that.

Corollary 2.3.6. *For $X \in \{Txt, Inf\}$, $SdMonX \subsetneq MonX$.*

Theorem 2.3.23. *(Theorem 3, [LZ04]) There exists a class of computable structures $\mathfrak{L}$, s.t. for $X \in \{Txt, Inf\}$:*

- $\mathfrak{L} \in rSMonX$
- $\mathfrak{L} \notin SdXEx$

Proof: Let $\mathfrak{L}$ be a c.e. class of computable structures with language L. For each k $\in \omega$, s.t. $\mathfrak{A}_k \in \mathfrak{L}$, we set $\mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \mathbb{E}),$ where $n < k\}$.
For each k $\in \omega$ and all $j \in \omega^+$ we distinguish the following cases:
Case 1: $\neg\xi_k(k) \leq j$
We set $\mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}} = \mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}}$.
Case 2: $\xi_k(k) \leq j$
We set $\mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \{m\}\}$, where $m = 6$.

Claim 1: $\mathfrak{L} \in rSMonTxt$.
We define a rSMonTxt-learner M for the class $\mathfrak{L}$. Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$.
$M(\lceil t_x \rceil)$ = "find the maximum k, s.t. $o_k = max\{o \mid o = n,$ where $\lceil e, n \rceil \in content(t_x) \wedge n \in \mathbb{O}$.
if k is not found, output nothing and request next input.
check wether $m = e_n$, and $\lceil e, e_n \rceil \in content(t_x)$
and e_n is the only even number.
In case such e_n is found, execute (A1).
Otherwise, check if there are $n_1, n_2 \in \mathbb{E}$, s.t.
$n_1 = e_{n_1}$, and $n_2 = e_{n_2}$, where
$\lceil e, e_{n_1} \rceil \in content(t_x)$ and $\lceil e, e_{n_2} \rceil \in content(t_x)$ In case it is, output $\lceil k, 0 \rceil$.
Otherwise, output nothing and request next input.
(A1) Test whether or not $\neg\xi_k(k) \leq x$.
In case it is output nothing and request next input.
Otherwise output $\lceil k, \xi_k(k) \rceil$ and request next input.

It can be seen that the learner M is rearrangment independent. To prove that $\mathfrak{L} \in rSMonTxt$, we look at the following cases.
Case 1: $\phi_k(k) \uparrow$.
From the definition of the class $\mathfrak{L}$ we get that $\mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}} = \mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}}$ for all $j \in \omega$. As t is a text for a structure $\mathfrak{A} \in \mathfrak{L}$, then there is a stage x, s.t. $content(t_x) \nsubseteq \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \{m\})\}$. The learner M, after having seen the text t_x, it will always output $\lceil k, 0 \rceil$, which is a correct hypothesis. This shows that in this case M works strong-monotonically.

Case 2: $\phi_k(k) \downarrow$.
We have two possibilities:
$\mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \{m\})\}$. Then the learner M will always execute instruction $(A1)$. Furthermore there is a stage $n_0 \in \omega$, s.t. $\xi_k(k) \leq x$ for all

$x \geq n_0$. Then the learner M after having seen t_{n_0} will always output $\lceil k, \chi_k(k) \rceil$, and will be correct.
$\mathfrak{D}_{\mathfrak{A}_{\lceil k,j \rceil}} = \mathfrak{D}_{\mathfrak{A}_{\lceil k,0 \rceil}}$. We have two subcases. Like in Case 1, M can find a stage at which to output $\lceil k, 0 \rceil$, which means that M has found at some stage $n_0 \in \omega$, that $content(t_{n_0}) \nsubseteq \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \{m\})\}$, and will be correct. The second subcase is if M at first finds an initial segment of t, s.t. $content(t_x) \subseteq \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \{m\})\}$, then M will output $\lceil k, \chi_k(k) \rceil$. Then at some later stage M will find more even numbers and will correctly switch to an index for $\lceil k, 0 \rceil$. It can be seen that this mind change doesn't break the strong-monotonicity requirement. Hence M works correctly and is strong-monotonic.

Claim 2: $\mathfrak{L} \notin SdTxtEx$.
We prove this Claim, by showing that if $\mathfrak{L} \in SdTxtEx$, then we can construct an algorithm for solving the halting problem.
Lemma: Let M be a SdTxtEx-learner for the class $\mathfrak{L}$. Then M can be used to solve the halting problem.
Proof: We define an algorithm **Alg** in the following way:
Alg: "On input k execute instruction (a1).
(a1): Simulate M on input $content(\sigma) = \{\lceil e, n \rceil \mid n \in \{\mathbb{O} \restriction k \cup \{m\}\}$.
If M doesn't output a hypothesis, output $\phi_k(k) \uparrow$, and stop.
Otherwise, let $z = M(\lceil \sigma \rceil)$. Execute (a2).
(a2) Test whether or not there is $m_2 \in \mathbb{E}$, s.t. $m \neq m_2$, where $content(\sigma) \cup \{\lceil e, m_2 \rceil\} \subseteq \mathfrak{D}_{\mathfrak{A}_z}$.
In case it is, output $\phi_k(k) \uparrow$.
Else output $\phi_k(k) \downarrow$ and stop."

Let's see that **Alg** behaves correctly. Suppose **Alg** outputs $\phi_k(k) \uparrow$, but we have that $\phi_k(k) \downarrow$. Then as M is set-driven, M has to output a correct hypothesis, but it doesn't, as **Alg** terminates with a wrong behaviour. With this contradiciton, we get that **Alg** behaves correctly.
Let's assume that **Alg** terminates with $\phi_k(k) \downarrow$. Then the test in (a2) will give us $\mathfrak{D}_{\mathfrak{A}_z} = \{\lceil e, n \rceil \mid n \in (\mathbb{O} \restriction k \cup \{m\})\}$. Hence $\phi_k(k) \downarrow$.

□

Theorem 2.3.24. *(Theorem 9, [LZ04]) Fox $X \in \{Txt, Inf\}$, we have that $SdMonX \subsetneq rMonX$.*

Proof:

- $SdMonX \subseteq rMonX$.
  We use thm. 2.3.12 in order to prove this direction.
  Suppose $\mathfrak{L}$ is a SdMonX-learnable. We define a rMonX-learner M for the class $\mathfrak{L}$.
  $M(\lceil t_x \rceil)$ = "Search for the least $i \leq ln(t_x)$,
  s.t. $T_i \subseteq content(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$,

where $\mathfrak{A}_i \in \mathfrak{L}$.
If such i is found, output i and request next input.
Otherwise output '?' and request next input."
It can be seen that M is rearrangment independent.
Let's see that M correctly identifies $\mathfrak{L}$ and that M is indeed a monotonic learner.
Case 1: The learner M correctly identifies the class $\mathfrak{L}$.
Suppose that M has output an index j_1, and that M is wrong, i.e. it has to output an index $i < j_1$. But by definition of M, there has to be an element $n \notin \mathfrak{D}_{\mathfrak{A}_i}$, when M changes its mind. Hence M is correct.
Case 2: The learner M is monotonic.
Suppose otherwise, i.e. M has output indecies j_1, j_2, where $j_1 < j_2$, s.t. $\mathfrak{D}_{\mathfrak{A}_{j_1}} \cap \mathfrak{D}_{\mathfrak{A}} \nsubseteq \mathfrak{D}_{\mathfrak{A}_{j_2}} \cap \mathfrak{D}_{\mathfrak{A}}$.
But by assumption $(T_j)_{j\in\omega}$ satisfies the requirements in thm. 2.3.12. By definition M, on input σ, will output an index satisfying the test. As M changes its mind only on elements not contained in previous guesses M correctly will identify each structure in $\mathfrak{L}$. As each $(T_j)_{j\in\omega}$ satisfies the monotonicity requirements hence M does too. Hence $\mathfrak{L} \in rMonTxt$.

- $rMonX \nsubseteq SdMonX$.
From thm. 2.3.23 we have that $rMonX \nsubseteq SdXEx$. From thm. 2.3.13 and thm. 2.3.15, we can conclude that $SdMonX \subsetneq SdXEx$. Hence $rMonX \nsubseteq SdMonX$.

□

Theorem 2.3.25. *For $X \in \{Txt, Inf\}$, we have that $rXEx = XEx$.*

Proof:

- $rTxtEx \subseteq TxtEx$.
Let $\mathfrak{L}$ be a rTxtEx-learnable class. Suppose M is a rTxtEx-learner for the class $\mathfrak{L}$. We define a TxtEx-learner M' for the class $\mathfrak{L}$ based on the behaviour of M. What M' does is just to wait M to output an answer. As M correctly ideintifies each structure in the class $\mathfrak{L}$, the so M'. Hence $\mathfrak{L} \in TxtEx$.

- $TxtEx \subseteq rTxtEx$.
We use the thm. 2.3.2 in order to prove this direction.
Let $\mathfrak{L} \in TxtEx$. We define a rTxtEx-learner M for the class $\mathfrak{L}$ by which we will prove this direction.
Suppose t is a text for a structure $\mathfrak{A} \in \mathfrak{L}$, and $\sigma \sqsubseteq t$ then:
$M(\lceil\sigma\rceil)$ = "Search for the least index $i \leq ln(\sigma)$,
s.t. $T_i \subseteq content(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}_i}$.
If such i is found output i as answer and request next input,
otherwise output '?' and request next input.
By def. of rearrangment independent learnability, the learner M is clearly rearrangment independent (see def. 2.3.12).

Claim: The learner M TxtEx-learns the structure $\mathfrak{A}$ from a text t.
Proof: Let $i \in \omega$ be the least index for which we have that $\mathfrak{D}_{\mathfrak{A}_i} = \mathfrak{D}_{\mathfrak{A}}$. Then there is a least $n \in \omega$, for which $T_i \subseteq content(t_n)$. For all $j < i$, we will have that $\mathfrak{D}_{\mathfrak{A}_j} \subset \mathfrak{D}_{\mathfrak{A}_i}$ (or they will be equal, from the second clause of the thm. 2.3.2). The least part means there exists a stage z, for which $content(t_z) \nsubseteq \mathfrak{D}_{\mathfrak{A}_k}$, for all $k < i$, where $\mathfrak{D}_{\mathfrak{A}_j} \subseteq \mathfrak{D}_{\mathfrak{A}_i}$. Hence $M(\lceil t_{z+r} \rceil) = i$, for all $r \in \omega$.

□

Theorem 2.3.26. *For $X \in \{Txt, Inf\}$, we have that $rWMonX = WMonX$.*

Proof: We can use the same idea as in thm. 2.3.25. □

From thm. 2.3.17 and by definition of rearrangment independent (see def. 2.3.12), we can conclude that.

Corollary 2.3.7. *For $X \in \{Txt, Inf\}$, we have that $SdXFin = rXFin$.*

From thm. 2.3.18 and thm. 2.3.25 we can conclude that.

Corollary 2.3.8. *For $X \in \{Txt, Inf\}$, $SdXEx \subsetneq rXEx$.*

By using lemma 2.3.8 and thm. 2.3.14.

Corollary 2.3.9. *For $X \in \{Fin, Ex, SMon, Mon, WMon\}$ and $Y \in \{r, Sd\}$, we have that $YTxtX \subsetneq YInfX$.*

2.3.4 [Memory limitation]

Here we will look at a type of learnability concerned with restricting the information available to a learner. This type of learnability was influenced from learning from children, who (with high probability) can forget about a given sentence, once when processed.

Recall that for a segment $\sigma \in \mathbf{SEQ}$, σ^- is a segment in which the last element of σ is removed, if σ has at least one element. Then, for $n \in \omega$, we define σ^{-n} as a new segment $\sigma' \in \mathbf{SEQ}$ in which first n elements from σ are removed, provided that $ln(\sigma) \geq n$, or $\sigma^{-n} = \sigma$ otherwise.

Definition 2.3.13. *(n-memory limited learner, [OSW86]) Let $\mathfrak{L}$ be a class of structures with language L. Let M be a learner for the class $\mathfrak{L}$. Let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$ (or I be an informant for $\mathfrak{A} \in \mathfrak{L}$). For $n \in \omega$, M is said to be n-memory limited learner, if for all $\sigma, \tau \in \mathbf{SEQ}$, s.t. $\sigma, \tau \sqsubseteq t$ (or $\sigma, \tau \sqsubseteq I$), if $\sigma^{-n} = \tau^{-n}$ and $M(\lceil \sigma^- \rceil) = M(\lceil \tau^- \rceil)$, then $M(\lceil \sigma \rceil) = M(\lceil \tau \rceil)$.*

By n-MLim we will denote the classes of structures which are learnt by an n-memory limited machines, where n $\in \omega$.

Recall that, from lemma 2.3.5, if ϕ_e is a partial computable function with program with index e, then there is a one-to-one computable function f: $\omega^n \to \omega$, s.t. for any e $\in \omega$, and $\bar{x} \in \omega^{n-1}$, we have that $\phi_{f(e,\bar{x})} = \phi_e$.

Lemma 2.3.10. *(Lemma 4.4.1A, [OSW86])* $1 - MLim = n - MLim$, *for an arbitrary* $n \in \omega$.

Proof:
$(\to) 1 - MLim \subseteq n - MLim$. This direction is trivial.
$(\leftarrow) n - MLim \subseteq 1 - MLim$.
Suppose that $\mathfrak{L}$ is a class of structures with language L, which is $n - MLim$-learnable. We define M a 1-MLim learner for the class $\mathfrak{L}$.
To prove this direction we will make a modification to the input of M.
For a number $m \in \omega$, by m^k we will denote the repetion of m k-times in a given text, for some k $\in \omega$. Let $\sigma \in$ **SEQ**, s.t. $\sigma \sqsubseteq t$, where t is a text for a structure $\mathfrak{A} \in \mathfrak{L}$. We define a new segment $\hat{\sigma}$ which modifies σ, in such a way to guarantee the learnability of the class $\mathfrak{L}$ by M, and hence by a 1-MLim-learner.
We define $\hat{\sigma}$ in the following way $\hat{\sigma} = \sigma(0)^n \hat{} \sigma(1) \hat{} \sigma(0)^n \hat{} ... \hat{} \sigma(ln(\sigma)) \hat{} \sigma(0)^n$.
We define the learner M as follows:

$$\mathrm{M}(\lceil \sigma \rceil) = \begin{cases} i, & \phi_{f(i,\sigma)} = \phi_i, \text{ s.t. } content(\sigma) \subseteq \mathbb{W}_i \\ ?, & \text{otherwise} \end{cases}$$

Let's see why M learns the class $\mathfrak{L}$ and that indeed M is 1-MLim.
By the way we modify the input to M, we get that for any text t for a structure $\mathfrak{A} \in \mathfrak{L}$, the modified text $\hat{t}$ is also a text for $\mathfrak{A}$, as we don't contradict the conditions of $n - MLim$-learnability. With this we prove that M learns the class $\mathfrak{L}$.
We will show now that M is 1-MLim. Suppose that for two segments $\sigma, \tau \in$ **SEQ**, s.t. $\sigma, \tau \sqsubseteq t$, we have that $M(\lceil \sigma^- \rceil) = M(\lceil \tau^- \rceil)$ and $\sigma^{-1} = \tau^{-1}$.
As we modify both segments before providing them to M, we will get that $M(\lceil \hat{\sigma}^- \rceil) = M(\lceil \hat{\tau}^- \rceil)$ and $\hat{\sigma}^{-1} = \hat{\tau}^{-1}$. Then by definition of n-MLim learnability we get that indeed M 1-MLim identifies the class $\mathfrak{L}$. □

Proposition 2.3.3. *(Proposition 5.6.3A, [OSW86])* $MLimTxt \subsetneq MLimInf$.

Proof:

$(\to)$ MLimTxt $\subseteq MLimInf$.
Let $\mathfrak{L}$ be a MLimTxt-learnable class of structures. Let M be a MLimTxt-learner for the class $\mathfrak{L}$. We define an MLimInf-learner M', which provides input to M, only the positive information from an informant I, and outputs whatever M does. As M identifies each structure $\mathfrak{A} \in \mathfrak{L}$, the behaviour of M' will be correct. Hence $\mathfrak{L} \in MLimInf$.

$(\leftarrow) MLimInf \not\subseteq MLimTxt$. Let $\mathfrak{L}$ be a class of computable structures with language L, with one unary relation, defined in the following way:
$\mathfrak{D}_{\mathfrak{A}_1} = \{\lceil e, n \rceil \mid n \in \mathbb{O}\}$,

$$\mathfrak{D}_{\mathfrak{A}_j} = \begin{cases} \{\lceil e, n \rceil \mid n \in (\mathbb{O} \cup \{e_{j-1}\})\}, & \text{if } j\%2 = 0 \\ \{\lceil e, n \rceil \mid n \in (\mathbb{O} \setminus \{j\} \cup \{e_j\})\}, & \text{if } j\%2 \neq 0 \end{cases}$$

We will prove this direction for 1-MLimTxt.

– $\mathfrak{L} \in MLimInf$
A successful MLimInf-learner M for the class $\mathfrak{L}$ is based on the following idea: M needs to check whether an even number $e_j \in \mathbb{E}$ is provided to it, and whether there is a negative information about an element $o_j \in \mathbb{O}$ is given to it. At first if M gets only positive examples from the set $\mathbb{O}$ M will output an index for $\mathfrak{A}_1$, once an even number $e_j \in \mathbb{E}$ appears, or negative example $o_j \in \mathbb{O}$ appears M will switch either to $\mathfrak{A}_{2j}$, or $\mathfrak{A}_{2j-1}$. Hence M will identify each structure $\mathfrak{A}_i \in \mathfrak{L}$. Hence $\mathfrak{L} \in MLimInf$.

– $\mathfrak{L} \notin MLimTxt$
Suppose otherwise, i.e. that $\mathfrak{L} \in MLimTxt$. Let M be a 1-MLimTxt-learner for the class $\mathfrak{L}$. Suppose that σ is a locking sequence for $\mathfrak{D}_{\mathfrak{A}_1}$. Let $\sigma^{'} = \sigma\hat{}\lceil e, e_j\rceil$, and $\sigma^{''} = \sigma\hat{}\lceil e, o_j\rceil\hat{}\lceil e, e_j\rceil$, where $e_j \in \mathbb{E}$ and $o_j \in \mathbb{O}$, s.t. $o_j \notin content(\sigma)$. Then we have that $M(\lceil\sigma'\rceil) = M(\lceil\sigma''\rceil)$ as σ is a locking sequence for $\mathfrak{D}_{\mathfrak{A}_1}$, and M is 1-MLimTxt. We define two texts t^1, t^2 for $\mathfrak{D}_{\mathfrak{A}_{2j}}$ and $\mathfrak{D}_{\mathfrak{A}_{2j-1}}$ respectively. Let $t^1 = \sigma'\hat{}\lceil e, 1\rceil\hat{}\lceil e, 3\rceil\hat{}...\hat{}\lceil e, 2i-1\rceil$, and $t^2 = \sigma''\hat{}\lceil e, 1\rceil\hat{}\lceil e, 3\rceil\hat{}...\hat{}\lceil e, 2i-1\rceil$, for all $i \neq j$. Then M on both texts will output the same index as σ is a locking sequence for $\mathfrak{D}_{\mathfrak{A}_1}$. Thus M will fail to identify both structures $\mathfrak{A}_{2j}$ and $\mathfrak{A}_{2j-1}$. Hence M will fail to learn the class $\mathfrak{L}$. Hence $\mathfrak{L} \notin MLimTxt$.

□

Theorem 2.1. *For $X \in \{Inf, Txt\}$ and $Y \in$ { Fin, Ex, BC, SMon, Mon, WMon, SdFin, SdEx, SdSMon, SdMon, SdWMon, rSMon, rMon, rWMon }, we have that MLimX = XY.*

Proof:

- XY $\subseteq$ MLimX Let $\mathfrak{L}$ be a class of structures. Let M be an XY-learner for the class $\mathfrak{L}$. We define a new MLimX-learner M', where we modify the input of M' in order to get the desired result.
For simplicity let M' be a 1-MLim memory machine, and let t be a text for a structure $\mathfrak{A} \in \mathfrak{L}$ (the same technique can be used in the case of informant). For each segment $\sigma \sqsubseteq t$, we modify $\sigma = \langle\lceil e, n_1\rceil, \lceil e, n_2\rceil, ..., \lceil e, n_k\rceil\rangle$ in the following way:
$\sigma' = \langle\lceil e, n_1\rceil, \lceil e, n_1\rceil, \lceil e, n_2\rceil, \lceil e, n_2\rceil, ..., \lceil e, n_k\rceil, \lceil e, n_k\rceil\rangle$. In such a way M' will never forget an element and will correctly identify the structure $\mathfrak{A}$ under the desired criteria. Hence $\mathfrak{L} \in$ MLimX.

- MLimX $\subseteq$ XY As we remove elements(by def. of MLimX-learnability) and we correctly idenfity a class of structures, is not hard to see that defining a XY-learnability machine outputting the same guess as an MLimX-learner for the given class will correctly identify the same class, too.

□

2.3.5 [NUShapedBC]

Here we will consider a behaviour of a learner, which emerged from the behaviour of a child, learning language. In the case of english, it was found that children often learn correctly past tense forms, but for some reason abandons that correct behaviour and start to make mistakes. Then again they start to correctly determine the right form of forming verbs.
If we forbidden a child of such correct-incorrect-correct behaviour a type of learnability occurs known as Non-U-Shaped learnability. This type of learnability decreases the learnability power when considering it under Behaviourly correct learning.

Definition 2.3.14. *(NUShBC) Let $X \in \{Inf, Txt\}$. Let $\mathfrak{L}$ be a class of structures with language L. A XBC-learner M for $\mathfrak{L}$ is non-U-shaped (denoted by NUShTxtBC, or NUShInfBC), iff for every $\mathfrak{A} \in \mathfrak{L}$, there are no three indecies k, m, n such that $k < m < n$ and $W_{M(\lceil t_k \rceil)} = \mathfrak{D}_{\mathfrak{A}}$, $W_{M(\lceil t_m \rceil)} \neq \mathfrak{D}_{\mathfrak{A}}$ and $W_{M(\lceil t_n \rceil)} = \mathfrak{D}_{\mathfrak{A}}$.*

Comparison between types

Theorem 2.3.27. *(Theorem 2.7, [Car+05]) NUShTxtBC $\subset$ TxtBC*

Proof:

- NUShTxtBC $\subseteq$ TxtBC
  Follows by definition of TxtBC-learnability.

- TxtBC $\nsubseteq$ NUShTxtBC
  We use the idea given in Theorem 4, [FJO94].
  Suppose $\{M_i\}_{i\in\omega}$ is a c.e. set of all partial computable learners. Let $\mathfrak{L}$ be a c.e. class of c.e. structures with language L, defined in the following way:
  For all $j \in \omega$, we set $\mathfrak{D}_{\mathfrak{A}_j} = \{\lceil e, n \rceil \mid n \in \omega\}$.
  For all $j, k \in \omega$, let the atomic diagram of $\mathfrak{A}^{j+1}_{\lceil j,k \rceil}$ be defined as $\mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k \rceil}} = \{\lceil e, n \rceil \mid n \leq k\}$, and let the atomic diagram of $\mathfrak{A}^{j+2}_{\lceil j,k \rceil}$ be defined as $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k \rceil}} = \mathbb{W}_{M_j(\lceil \sigma_{j,k} \rceil)}$, where $\sigma_{j,k} = \{\lceil j, 0 \rceil, ...\lceil j, k \rceil\}$.
  If there is a least k $\in \omega$ and n $\in \omega$ for which
  $$(*)\mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k \rceil}} \subseteq \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k \rceil}} \restriction n \subseteq \mathfrak{D}_{\mathfrak{A}_j},$$
  then let $k_0 \in \omega$ be that k.
  If such k_0 exists let the class of structures $\mathfrak{L}_j = \{\mathfrak{A}^{j+1}_{\lceil j,k_0 \rceil}, \mathfrak{A}^{j+2}_{\lceil j,k_0 \rceil}\}$, otherwise let $\mathfrak{L}_j = \{\mathfrak{A}_j\}$. We set $\mathfrak{L} = \bigcup_{j\in\omega}\{\mathfrak{L}_j\}$.

  Claim 2.3.1. $\mathfrak{L} \in TxtBC$

  Proof: In order to prove this part, we will define a learner M, which TxtBC-learns the class $\mathfrak{L}$. We define a TxtBC-learner M in the following

way:
M = "On input $\sigma \in \mathbf{SEQ}$, s.t. $ln(\sigma) = n$
let j $\in \omega$ be such that $content(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$
if $n \leq 1$
then let $M(\lceil \sigma \rceil) = e$, where $\mathbb{W}_e = \mathfrak{D}_{\mathfrak{A}_j}$
else
let c = $\{k \leq n \mid \mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k \rceil}} \subset \mathbb{W}^n_{M_j(\lceil j,k \rceil)} \restriction ln(\sigma) \subseteq \mathfrak{D}_{\mathfrak{A}_j}\}$
if c = $\emptyset$
then let $M(\lceil \sigma \rceil) = e$, where $\mathbb{W}_e = \mathfrak{D}_{\mathfrak{A}_j}$
else
let $k_0 = min(c)$
if $content(\sigma) \subseteq \mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$
then let $\mathbb{W}_{M(\lceil \sigma \rceil)} = \mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$
else if $content(\sigma) \subseteq \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$
then let $\mathbb{W}_{M(\lceil \sigma \rceil)} = \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$
else let $\mathbb{W}_{M(\lceil \sigma \rceil)} = \mathfrak{D}_{\mathfrak{A}_j}$
endif
endif
endif
end"

We observe 3 cases in order to prove the claim. Suppose t is a text for $\mathfrak{A}_j$.

Case 1: If there is no such k $\in \omega$ for which (*) holds.
We have two cases: either there is a k' and an index $j' < j$ for which $(*)$ holds, i.e. $\mathfrak{D}^{j'+1}_{\mathfrak{A}_{\lceil j',k' \rceil}} \subseteq \mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}} \restriction n \subseteq \mathfrak{D}_{\mathfrak{A}_{j'}}$. Then from the definition of M, at first we will output an index for $\mathfrak{D}^{j'+1}_{\mathfrak{A}_{\lceil j',k' \rceil}}$, until we get an element not contained in $\mathfrak{D}^{j'+1}_{\mathfrak{A}_{\lceil j',k' \rceil}}$. Then M will check whether $content(t_x) \subseteq \mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}}$. In case this happens, M will output an index for $\mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}}$. Then again M can receive an element $n \in \mathfrak{D}_{\mathfrak{A}_{j'}} \setminus \mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}}$, and will fall in the third clause from its definition. But we have that $\mathfrak{D}_{\mathfrak{A}_{j'}} = \mathfrak{D}_{\mathfrak{A}_j}$, hence M will be correct as M is TxtBC-learner. If there is no such k', M can output an index for $\mathfrak{D}_{\mathfrak{A}_{j'}} = \mathfrak{D}_{\mathfrak{A}_j}$. From the definition of TxtBC, M again will be correct.

Case 2: There is a k $\in \omega$ for which (*) holds.
Let $k_0 \in \omega$ be the least such k. Then:
Case 2.1. : If $content(t_x) \subseteq \mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. Then M will output an index for $\mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. Then at later stage, M will get an element n $\in \mathfrak{D}_{\mathfrak{A}_j} \setminus \mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. If n is contained in $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$, M will output an index for it, otherwise M

will output an index for $\mathfrak{D}_{\mathfrak{A}_j}$. We can have either that $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} = \mathfrak{D}_{\mathfrak{A}_j}$, or $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} \cap \mathfrak{D}_{\mathfrak{A}_j} \neq \emptyset$. In the first case we will always pass the second test from the definition of M, and will be correct as M is TxtBC-learner. In the second case, there will be a moment at which we will receive an element $n \in \mathfrak{D}_{\mathfrak{A}_j} \setminus \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$ and will pass the third test from the definition of M. Hence the behaviour of M will be correct. Hence M will identify each structure $\mathfrak{A}_j \in \mathfrak{L}$. Hence $\mathfrak{L} \in TxtBC$.

Case 2.2.: If $content(t_x) \subseteq \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. Then M will output an index for $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$.
Then we have two possibilities. Either $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} = \mathfrak{D}_{\mathfrak{A}_j}$, or $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} \cap \mathfrak{D}_{\mathfrak{A}_j} \neq \emptyset$. In the first case, we will always output an index for $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$, and as M is TxtBC-learner, M will be correct. In the second case, at some later stage M will receive an element n $\in \mathfrak{D}_{\mathfrak{A}_j} \setminus \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$, as t is a text for $\mathfrak{D}_{\mathfrak{A}_j}$. In both cases M will correctly identify each structure $\mathfrak{D}_{\mathfrak{A}_j} \in \mathfrak{L}$.

From both cases it follows that M will correctly identify each structure $\mathfrak{A}_j \in \mathfrak{L}$. Hence $\mathfrak{L}$ is TxtBC-learnable. Hence $\mathfrak{L} \in TxtBC$. □

Claim 2.3.2. $\mathfrak{L} \notin NUShTxtBC$

Proof: Suppose otherwise, i.e. there is a learner M which NUShTxtBC-learns the class $\mathfrak{L}$.
Suppose t is a text for a structure $\mathfrak{A}_j \in \mathfrak{L}$. We look at the following cases.
Case 1: If there is no k $\in \omega$ for which (*) holds, then either there are k' and $j' < j$ for which we can find structures for which $\mathfrak{D}^{j'+1}_{\mathfrak{A}_{\lceil j',k' \rceil}} \subseteq \mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}} \upharpoonright ln(t_x) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ or there are not such k', j'. If there are, then M will output an index for $\mathfrak{D}^{j'+1}_{\mathfrak{A}_{\lceil j',k' \rceil}}$, at later stage if M receives an element $n \in \mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}} \setminus \mathfrak{D}^{j'+1}_{\mathfrak{A}_{\lceil j',k' \rceil}}$, M will switch to an index for $\mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}}$. Suppose $\mathfrak{D}^{j'+2}_{\mathfrak{A}_{\lceil j',k' \rceil}} \cap \mathfrak{D}_{\mathfrak{A}_{\lceil j',k' \rceil}} \neq \emptyset$, then M will fail to be NUShTxtBC-learner for $\mathfrak{D}_{\mathfrak{A}_{\lceil j',k' \rceil}}$ as M changed its mind to an incorrect guess. Now if there are not such k', j', then we can not guarantee that at some later point M will not change its mind to an incorrect guess. Hence this case is proved.
Case 2: If on the other hand there is such a k, then let $k_0 \in \omega$ be the least such k. Now we have that the structures $\mathfrak{A}^{j+1}_{\lceil j,k_0 \rceil}, \mathfrak{A}^{j+2}_{\lceil j,k_0 \rceil} \in \mathfrak{L}$.
Suppose at first M has received $\sigma_{j,k_0} \sqsubset t$, s.t. $content(\sigma_{j,k_0}) \subseteq \mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. Then M will output an index for $\mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. Suppose at later stage M receives an input $\sigma'_{j,k_0} \sqsubset t$, s.t. $\mathfrak{D}^{j+1}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} \subset content(\sigma'_{j,k_0}) \subseteq \mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} \upharpoonright ln(\sigma'_{j,k_0}) \subseteq \mathfrak{D}_{\mathfrak{A}_j}$ and that $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}} \cap \mathfrak{D}_{\mathfrak{A}_j} \neq \emptyset$. Then M will output an index for $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$. Hence M will wrongly output an index for $\mathfrak{D}^{j+2}_{\mathfrak{A}_{\lceil j,k_0 \rceil}}$,

until turns back again to $\mathfrak{D}_{\mathfrak{A}_j}$. Hence M will fail to NUShTxtBC-identify $\mathfrak{A}_j$. Hence $\mathfrak{L} \notin NUShTxtBC$.

□

□

References

[AS16] Klaus Ambos-Spies. *Learnability and Numberings of Families of Computably Enumerable Sets*. 2016.

[Ang80] Dana Angluin. "Inductive inference of formal languages from positive data". In: *Information and Control* 45.2 (1980), pp. 117 – 135. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(80)90285-5. URL: http://www.sciencedirect.com/science/article/pii/S0019995880902855.

[Ash+89] Chris Ash et al. "Generic copies of countable structures". In: *Annals of Pure and Applied Logic* 42.3 (1989), pp. 195–205. ISSN: 0168-0072. DOI: https://doi.org/10.1016/0168-0072(89)90015-8. URL: https://www.sciencedirect.com/science/article/pii/0168007289900158.

[BFM20] Nikolay Bazhenov, Ekaterina Fokina, and Luca San Mauro. "Learning families of algebraic structures from informant". In: *Information and Computation* 275 (2020), pp. 104–590. ISSN: 0890-5401. DOI: 10.1016/j.ic.2020.104590. URL: http://www.sciencedirect.com/science/article/pii/S089054012030078X.

[BB75] Lenore Blum and Manuel Blum. "Toward a mathematical theory of inductive inference". In: *Information and Control* 28.2 (1975), pp. 125–155. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(75)90261-2. URL: https://www.sciencedirect.com/science/article/pii/S0019995875902612.

[Buh+00] Harry Buhrman et al. "Separating Complexity Classes Using Autoreducibility". In: *SIAM Journal on Computing* 29.5 (2000), pp. 1497–1520. DOI: 10.1137/S0097539798334736. eprint: https://doi.org/10.1137/S0097539798334736. URL: https://doi.org/10.1137/S0097539798334736.

[Car+05] Lorenzo Carlucci et al. "U-shaped learning may be necessary 3". In: *Journal of Mathematical Psychology* 49 (Feb. 2005).

[Chi90] John Chisholm. "Effective model theory vs. recursive model theory". In: *The Journal of Symbolic Logic* 55.3 (1990), 1168–1191. DOI: 10.2307/2274481.

[FKSM19] Ekaterina Fokina, Timo Kötzing, and Luca San Mauro. "Limit Learning Equivalence Structures". In: *Proceedings of the 30th International Conference on Algorithmic Learning Theory*. Ed. by Aurélien Garivier and Satyen Kale. Vol. 98. Proceedings of Machine Learning Research. PMLR, 2019, pp. 383–403. URL: https://proceedings.mlr.press/v98/fokina19a.html.

[FJO94] Mark Fulk, Sanjay Jain, and Daniel N. Osherson. "Open problems in "systems that learn"". In: *Journal of Computer and System Sciences* 49.3 (1994). 30th IEEE Conference on Foundations of Computer Science, pp. 589–604. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(05)80072-8. URL: https://www.sciencedirect.com/science/article/pii/S0022000005800728.

[Gol67] E. Mark Gold. "Language identification in the limit". In: *Information and Control* 10.5 (1967), pp. 447–474. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(67)91165-5. URL: http://www.sciencedirect.com/science/article/pii/S0019995867911655.

[LZ93a] Steffen Lange and Thomas Zeugmann. "Language Learning in Dependence on the Space of Hypotheses". In: *Proceedings of the Sixth Annual Conference on Computational Learning Theory*. COLT '93. Santa Cruz, California, USA: Association for Computing Machinery, 1993, 127–136. ISBN: 0897916115. DOI: 10.1145/168304.168320. URL: https://doi.org/10.1145/168304.168320.

[LZ93b] Steffen Lange and Thomas Zeugmann. "On the Impact of Order Independence to the Learnability of Recursive Languages". In: 1993.

[LZ04] Steffen Lange and Thomas Zeugmann. "Set-Driven and Rearrangement-Independent Learning". In: *Theory of Computing Systems* 29 (Apr. 2004), pp. 599–634. DOI: 10.1007/BF01301967.

[LZ96] Steffen Lange and Thomas Zeugmann. "Set-Driven and Rearrangement-Independent Learning of Recursive Languages". In: *Mathematical systems theory* 29 (Dec. 1996), pp. 599–634. DOI: 10.1007/BF01301967.

[LZ92a] Steffen Lange and Thomas Zeugmann. "Types of Monotonic Language Learning and Their Characterization". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, 377–390. ISBN: 089791497X. DOI: 10.1145/130385.130427. URL: https://doi.org/10.1145/130385.130427.

[LZ92b] Steffen Lange and Thomas Zeugmann. "Types of Monotonic Language Learning and Their Characterization". In: *Annual Conference Computational Learning Theory*. July 1992. DOI: 10.1145/130385.130427.

[LZK96] Steffen Lange, Thomas Zeugmann, and Shyam Kapur. "Monotonic and dual monotonic language learning". In: *Theoretical Computer Science* 155.2 (1996), pp. 365–410. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(95)00284-7. URL: https://www.sciencedirect.com/science/article/pii/0304397595002847.

[Mon09] Antonio Montalbán. "Notes on the Jump of a Structure". In: *Mathematical Theory and Computational Practice*. Ed. by Klaus Ambos-Spies, Benedikt Löwe, and Wolfgang Merkle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 372–378. ISBN: 978-3-642-03073-4.

[Mon18] Antonio Montalbán. "Coding and Definability in Computable Structures". In: *Notre Dame Journal of Formal Logic* 59.3 (2018), pp. 285 –306. DOI: 10.1215/00294527-2017-0032. URL: https://doi.org/10.1215/00294527-2017-0032.

[Mon21] Antonio Montalbán. *Computable Structure Theory: Within the Arithmetic*. Perspectives in Logic. Cambridge University Press, 2021. DOI: 10.1017/9781108525749.

[OSW86] Daniel N. Osherson, Michael Stob, and Scott Weinstein. *Systems that learn : an introduction to learning theory for cognitive and computer scientists / Daniel N. Osherson, Michael Stob, Scott Weinstein*. English. MIT Press Cambridge, Mass, 1986, ix, 205 p. ; ISBN: 0262150301.

[Soa16] R.I. Soare. *Turing Computability: Theory and Applications*. Theory and Applications of Computability. Springer Berlin Heidelberg, 2016. ISBN: 9783642319334. URL: https://books.google.bg/books?id=4PZ6DAAAQBAJ.

[SS09] Alexandra Soskova and Ivan Soskov. "A Jump Inversion Theorem for the Degree Spectra". In: *Journal of Logic and Computation* 19.1 (2009), pp. 199–215. DOI: 10.1093/logcom/exn024.

[Vat11] Stefan Vatev. "Conservative Extensions of Abstract Structures". In: *Proceedings of the 7th Conference on Models of Computation in Context: Computability in Europe*. CiE'11. Sofia, Bulgaria: Springer-Verlag, 2011, 300–309. ISBN: 9783642218743.