Formal tree languages and their algorithmic learnability

Anna Kasprzik

FB IV, University of Trier, Germany

December 12, 2011

Abstract

This thesis centers on formal tree languages and on their learnability by algorithmic methods in abstractions of several learning settings.

After a general introduction (Chapter 1), we present a survey of relevant definitions for the formal tree concept as well as special cases (strings) and refinements (multi-dimensional trees) thereof in Chapter 2. In Chapter 3 we discuss the theoretical foundations of algorithmic learning in a specific type of setting of particular interest in the area of Grammatical Inference where the task consists in deriving a correct formal description for an unknown target language from various information sources (queries and/or finite samples) in a polynomial number of steps. We develop a parameterized meta-algorithm that incorporates several prominent learning algorithms from the literature in order to highlight the basic routines which regardless of the nature of the information sources have to be run through by all those algorithms alike. In this framework, the intended target descriptions are *deterministic finite-state tree automata*. We discuss the limited transferability of this approach to another class of descriptions, *residual finite-state tree automata*, for which we propose several learning algorithms as well. The learnable class by these techniques corresponds to the class of *regular tree languages*.
In Chapter 4 we outline a recent range of attempts in Grammatical Inference to extend the learnable language classes beyond regularity and even beyond context-freeness by techniques based on syntactic observations which can be subsumed under the term *distributional learning*, and we describe learning algorithms in several settings for the tree case taking this approach.
We conclude with some general reflections on the notion of learning from structural information.

Acknowledgements

First and foremost, I would like to thank my supervisor Henning Fernau (for whom the German "Doktorvater" is a much more adequate term), and to express my admiration for his ability to bring out the best in his Ph.D. students by never ever "pulling title" on them but being more like a friendly colleague and collegial friend.
During the past four years he has substantially contributed to my physical and mental well-being by providing me with huge amounts of the most wonderful apples from his garden and various useful principles such as: "Things take exactly as many of your resources as you give them."

I will not attempt the impossible task of thanking my companion in life for the past ten years, Daniel Hüttl, in any other way than stating that this thesis is as much his as it is mine.
I believe I can thank my parents most adequately by stating that I have *not* written this thesis for them but for myself.

A warm thanks goes to Frank Drewes, Johanna Björklund, and Suna Bensch in Sweden for all the encouragement I have received from them during the past year, and for two most congenial research exchanges.
I also thank Sandra Zilles for feedback and encouragement, and for opening up a window into a possible future.
I thank Alexander Clark and Ryo Yoshinaka for an illuminating communication and cooperation, and I thank Ryo Yoshinaka and Thomas Zeugmann for two very instructive stays in Sapporo, Japan.
I thank Colin de la Higuera for an inspiring stay in St Etienne, France.

I conclude by thanking the editor of my horoscope on a certain webpage. Temporary superstition has helped a lot.

Contents

1 Introduction **9**

2 Inventory: The object axis **15**
2.1 Strings 15
2.1.1 Classical strings 15
2.1.2 Multi-words 16
2.2 Trees 17
2.2.1 Classical trees 17
2.2.2 Multi-dimensional trees 25
2.3 Outlook 37

3 Learning regular tree languages in different settings **39**
3.1 Learning in a bounded number of steps 39
3.2 Theoretical preliminaries 44
3.2.1 Definitions and concepts 44
3.2.2 Discussing definitions and their consequences 47
3.3 The meta-algorithm GENDET 54
3.3.1 Description of GENDET 54
3.3.2 Behaviour of GENDET 64
3.3.3 Complexity of GENDET 72
3.3.4 Outlook 75
3.4 Residual finite-state tree automata 78
3.4.1 Preliminaries 79
3.4.2 Observation tables and RFTA 82
3.4.3 Theoretical groundwork 85
3.4.4 The algorithm RESI 96
3.4.5 The algorithm RRPNI 105
3.4.6 The algorithm MATRES 111
3.5 Discussion 117

4 Distributional learning **121**
4.1 Finitely characterizable distributions 122
4.2 The learner's strategy 129
4.2.1 Primal learning algorithms 131
4.2.2 A dual learning algorithm 140
4.2.3 One-shot learning settings 143
4.3 Outlook 156

5 Concluding reflections **159**

Index **169**

Bibliography **173**

Appendix **183**
A.1 $MQ = 1$: 4 ways to use a counterexample 183
A.2 About $\langle 1, 0, X_+, \emptyset \rangle$ and $\langle 1, 0, \emptyset, X_- \rangle$ 185
A.3 Proofs for $\mathcal{R}_L$ 187

Chapter 1

Introduction

This work is about generalization within the area of *Grammatical Inference (GI)*. It is an attempt to recapitulate and generalize a range of the principles, concepts, and notions on which this area is founded and to express the result of our research in formal terms. On the other hand, it is an important concern of this work to help on the common understanding by establishing as many intuitive connections and explanations as possible. This may also involve formal language theoretic questions and reflections that are prior to GI.

The area of GI has evolved as a special branch of general learning theory which is concerned with the study of learning processes. There is a multitude of learning processes in nature, most notably the acquisition of a (human) language. However, natural processes tend to be rather "noisy" and difficult to grasp, and hence it seems an easier start to study learning in more formal contexts such as the construction of robots or the extraction of information from huge databases as provided for example by the world wide web.

For the purpose of such a study, we obviously need an abstraction of both the learner and of the surrounding learning situation. In the area considered here, this is achieved by conceiving a learner as an algorithm that infers a description for an unknown formal language from given information.

Underlying the study of algorithmic language learnability is the fundamental question of how to convey information and which structure to choose for the purpose. However, we do not wish to target the process of encoding and decoding information directly – for I will not venture an opinion on the question what pure, unencoded information is – but we will rather aim for an extensive examination of certain tools and procedures developed by other researchers involved with algorithmic learning to handle the objects, descriptions, and models of their choice. We would like to illuminate those devices from various angles to explore their relationship with the abstract concepts they are taken to represent, also considering questions such as "Are they

'minimal' in some sense, or is there redundancy?", and "What happens if I use them for a purpose other than the intended one?". This might also be of use for the communication between different branches of general learning theory with their respective traditions of formalization.

In formal language theory, one instantiation of the notion of encoding and decoding can be seen in the operations of generating and parsing. Grammars generate objects in the sense that they provide us with the rules for their construction, automata parse objects in the sense that they allow us to reach a conclusion about an object and attribute a value to it. In the present work, Chapter 3 is based on the automaton concept whereas in Chapter 4 we refer to results that mainly use grammars as the description of choice.
Other instances of encoding and decoding operations can be perceived in a learning process where a learner is given only selected partial information, i.e., where taking the role of the teacher trivially we are not allowed to simply provide a complete description of the target language and consequently have to decide *which* bits and pieces of information are specific enough in order to make the learner identify the target correctly.

In an algorithmic learning process, numerous aspects suggest themselves for closer study. We briefly and non-exhaustively sketch some of them below:

(a) The *hypothesis dimension*: How can we (pre- or descriptively) classify the hypotheses issued by the learner, are some hypotheses disallowed a priori due to formal criteria, for example because they describe languages from a non-admissible class, or because the chosen description does not meet certain formal specifications? In the branch of learning theory often referred to as *inductive inference*, this gives rise to the notion of a *'hypothesis space'*, see for example [11, 69].

(b) The *learning target*: It seems that it is helpful when there is a *canonical description* of the target since it facilitates the task of verifying whether learner and teacher agree or not. To a certain extent, the complexity of this verification can be shifted from the teacher to the learner by the requirements of how many properties of such a canonical description the learner's hypothesis should fulfil before it is submitted to the teacher. In this work, once those conditions are fixed we will mostly eclipse the complexity of answering the learner's questions suffered by the teacher.

(c) The *circumstances* of the learning process: Which sources of information are available, and how reliable are the given data? In this work we will exclude noisy settings and probabilistic approaches (such as *PAC learning*, proposed in [111]) and concentrate on "clean" ones where the given information can be trusted and the hypotheses are exact.

(d) The learning *protocol*: How frequently does the learner present its hypotheses – does it react to each new given datum (a prominent learning model based on that assumption is *identification in the limit* suggested by Gold [59]), or does it return one final answer at some point (*"one-shot" learning*, see [87, 88])? Further lines of study could explore the various conceivable ways of interaction between learner and teacher, as for example: Do they exchange information (data, queries, answers, hypotheses) in turns – or can it be more efficient if one of them is in charge of the protocol and decides who should make the next move?[1]

(e) The internal strategy of the learner to *organize and represent* the information obtained – a prominent option is the concept of an *observation table*, which is also the chosen device and object of study in this work (for other options and references see Subsection 3.3.4 and [12, 49]).

(f) Criteria to *judge* a learner: Finally, learners can be judged with respect to the rather trivial requirement of *correctness*, and to their *efficiency* where ideally we would like measures such as the number of steps taken and the number of times that a learner asks for new information to be bounded polynomially, but we might also consider the amount of space taken by the internal representation of the data received so far and the amount of redundancy in that representation. Further criteria, which are especially of interest in the area of inductive inference, include the *number of mind changes*, *consistency* of the current hypothesis *with the given data*, and *convergence to the target* (see for example [71]).

As will be motivated next, this work is mainly concerned with the exploration of variations and issues related to items (b), (c), and (e) in this list.

Motivating the structure of this thesis

In an algorithmic learning process as sketched above, there are at least three axes along which one can explore the possibilities of generalization: The type of the objects in the target language, the type and number of the available information sources, and the class of the target language, which is intrinsically linked to the structural properties of suitable descriptions for that language. Each of the three following chapters is mainly concerned with one of them.

Evidently, these axes are not completely independent of each other, and one can pinpoint various interrelations. Formally defined objects as discussed

[1] The author is grateful for several discussions exploring this thread of thoughts with Colin de la Higuera during a two-week research stay in St Etienne, France.

in Chapter 2 can simply constitute the members of a language but can also be used as components of a description for it such as a grammar or an automaton or a regular expression, or as a means to convey other kinds of structural information. For instance, a string can encode the sequence of rules applied in a derivation of an object which may be of an entirely different type, giving rise to so-called *control languages*, see for example [52].
There is an even wider range of ways to use a tree – as a *derivation tree* but also in order to document a learning process (in so-called *discrimination* or *observation trees*, see [12, 49]), and generally to represent any kind of hierarchical structure with certain basic properties. In fact, the main object of interest in this work will be the tree. As a practical motivation, applications for tree representations can be found in many different areas including the mark-up language XML in computer science, syntactical issues associated with natural language processing in linguistics, and examples from various other fields such as for instance economics where one could name workflow charts or the hierarchical connections in a company. On the theoretical level, although they do not allow the representation of every single interrelation, trees can be considered as a relatively reasonable abstraction with not too complex properties. In Subsection 2.2.2 we will also describe a refinement of the tree notion, so-called *multi-dimensional trees*, and introduce a term-like notation for them which was developed by the author in previous work (see [77, 74]) and which enables us to adapt algorithms for trees to this more general tree-like kind of structures directly.

We will reach the realm of Grammatical Inference in Chapter 3. Within that realm we assume a basic one-shot learning model where a learner has access to different information sources and is required to return a single final hypothesis after a finite number of steps. There has been some discussion in the literature about the question which information sources are the most realistic or most natural or most desirable ones, depending on the respective motivations for studying them, but we will only very briefly and tentatively try to motivate our settings psychologically. Rather, as our main interest lies on formal aspects we restrict ourselves to the classical and most thoroughly studied ones and examine a range of issues concerning different conceivable ways of combining them. After presenting and discussing the necessary preliminaries in Section 3.2, we develop a meta-algorithm which is intended to generalize over several established learning algorithms for regular tree languages that use deterministic finite-state tree automata as their description of choice in Section 3.3. In Section 3.4 we switch to a related kind of description for regular languages, *residual finite-state automata*. After the discussion of the necessary preliminary definitions and theoretical results in Subsections 3.4.1–3.4.3, we adapt existing algorithms for the inference of deterministic

finite-state string automata to residual finite-state automata and to the tree case in Subsections 3.4.4–3.4.6, and we discuss the complications of such an adaptation. In particular, as stated in the concluding Section 3.5, we will find that in contrast to the deterministic case not all algorithms for the studied settings can be adapted to trees in a straightforward way alike.

The main part of this thesis is concerned with regular languages since they are based on the most elementary (non-trivial) principle of derivation favourable to algorithmic learnability. However, we will also give an outline of a range of recent studies on the inference of descriptions beyond regularity, which basically consist in extending the essential theoretical observations and mechanisms for the regular case widely enough to make them applicable to grammar formalisms generating context-free and even some context-sensitive languages, in Chapter 4. Research in that direction explores the boundaries of algorithmic learnability within and across the language classes organized in the Chomsky hierarchy. A deeper understanding of those boundaries is of interest as it may yield further (purely) formal language theoretic results but also because there is a close connection to attempts in formal linguistics to discover existing complexities and to develop appropriate characterizations for natural (human) language classes with respect to that hierarchy.
In principle, an extension of the learnable classes is achieved by increasing the amount of structural information revealed to a learner with respect to the target, and we conclude with some general reflections on the notion of learning from structural information in Chapter 5.

Chapter 2

Inventory: The object axis

Note that the following sections are not merely the preliminaries for chapters to come but a collection of object types that are of consequence in the area of Grammatical Inference. We also discuss some definitions in more detail.

2.1 Strings

In this work we will need (classical) strings mainly as a basic reference structure for the definition of various other formal concepts.

2.1.1 Classical strings

In formal language theory, the smallest unit needed to build bigger, composite structures is the symbol. One of the most basic objects that can be built from symbols is the string, i.e., a sequence of symbols from some given alphabet Σ where the elements of Σ are not further specified.

Definition 1 *The string represented by the empty sequence is denoted by* ε.
In strings, the usual sequence brackets $\langle , \rangle$ *are generally omitted.*
The concatenation of two strings v, w *is denoted by juxtaposition as* vw
or as $v \cdot w$ *using the symbol for general sequence concatenation.*
The length of a string w *is denoted by* $|w|$.
The set of all strings over an alphabet Σ *is denoted by* Σ^*, *and* $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$.
Any set $L \subseteq \Sigma^*$ *is a string language over* Σ.

For $w, u \in \Sigma^*$, u *is a* prefix *of* w *if there is* $v \in \Sigma^*$ *such that* $uv = w$,
u *is a* suffix *of* w *if there is* $v \in \Sigma^*$ *such that* $vu = w$,
u *is an* extension *of* w *if there is* $v \in \Sigma^*$ *such that* $u = wv$, *and*
u *is a* substring *of* w *if there are* $v, x \in \Sigma^*$ *such that* $vux = w$.

Remark Since this (traditional) notation does not distinguish between a single symbol (an element of Σ) and the corresponding string of length 1 the construction of a string from the symbols of the alphabet can be seen as a special case of concatenation. We note it because for other types of objects this may differ such that we have to define the recursive construction of an object on the one hand and an operation combining objects on the other. $\diamondsuit$

A central notion for this work is the decomposition of a structure into a substructure and the remaining part(s), its *context*. The structural properties of such a substructure and matching context trivially depend on the type of the underlying object as well as on the admissible ways of decomposing it. We will give definitions of different substructures and the corresponding contexts for each kind of object separately but we would also like to emphasize the parallels between the results for different kinds of objects which are due to the fact that all those results are based on the same common principle, viz., that every object in a certain set or language can be decomposed into parts and that these parts can be reassembled to form new objects of the same kind (also see the concluding reflections in Chapter 5).

Definition 2 *Let $\Box \notin \Sigma$ be a special symbol. A* (string) context *is a string $l\Box r$ over $\Sigma \cup \{\Box\}$ with $l, r \in \Sigma^*$. We define the combination of a context $l\Box r$ and a string $w \in \Sigma^*$ as $l\Box r[\![w]\!] := lwr$. For $w \in \Sigma^*$ and a language $L \subseteq \Sigma^*$, $l\Box r$ is a context for w (with respect to L) iff $lwr \in L$.*

Obviously, the commonly adopted version of the Myhill-Nerode theorem for strings (see [67], and we will restate it for trees in Subsection 2.2.1) is based on a special case of this notion where a string v is decomposed into a prefix w (serving as substructure) and its context $\varepsilon\Box r$ such that $\varepsilon\Box r[\![w]\!] = v$, where r is the corresponding suffix of v with $wr = v$. Since the traditional notation identifies $\varepsilon\Box r[\![w]\!]$ with wr (i.e., it ignores the difference in the natures of $\varepsilon\Box r$ and r), the roles of w and r can be swapped in the sense that v is decomposed into a suffix w' and its context $l\Box\varepsilon$ where l is the matching prefix of v with $lw' = v$, and we obtain the symmetric version of the Myhill-Nerode theorem.

For more results in formal string language theory, see for example [67].

2.1.2 Multi-words

A next step of generalization along the object axis is the extension to discontinuous strings, so-called *multi-words*, which can be denoted as sequences of continuous strings. We fix an alphabet Σ of unspecified symbols as before.

Definition 3 *For any $m \in \mathbb{N}$, an* m-word *is a sequence $\langle w_1, \ldots, w_m \rangle$ with $w_1, \ldots, w_m \in \Sigma^*$.*[1] *The set of all m-words over Σ is denoted by $(\Sigma^*)^{\langle m \rangle}$. We write $(\Sigma^*)^{\langle * \rangle} := \{w \mid \exists m \in \mathbb{N} : w \in (\Sigma^*)^{\langle m \rangle}\}$ and $(\Sigma^*)^{\langle + \rangle} := (\Sigma^*)^{\langle * \rangle} \setminus \{\langle\rangle\}$. Any element of $(\Sigma^*)^{\langle * \rangle}$ is also called a* multi-word.

Multi-words are objects that are also generated by the nonterminal symbols of a *multiple context-free grammar (MCFG)*, see [103, 117].

Since a multi-word can be conceived as another kind of substructure of a continuous string we can accordingly extend the notion of a context as well:

Definition 4 *For $m \in \mathbb{N} \setminus \{0\}$, an* m-context *is a string over $\Sigma \cup \{\Box\}$ in which the symbol $\Box$ occurs exactly m times. The set of all m-contexts is denoted by $\mathbb{C}_\Sigma^{\langle m \rangle}$.*

For an m-word $w = \langle w_1, \ldots, w_m \rangle \in (\Sigma^*)^{\langle m \rangle}$ and an m-context $e = e_0 \Box e_1 \Box \ldots \Box e_m$ with $e_0, \ldots, e_m \in \Sigma^*$, we define the combination of w and e as $e[\![w]\!] := e_0 w_1 e_1 \ldots w_m e_m$, i.e., the string obtained by replacing the ith occurrence of $\Box$ in e by w_i for each i with $1 \leq i \leq m$.

Let $L \subseteq \Sigma^*$. We say that $e \in \mathbb{C}_\Sigma^{\langle m \rangle}$ is a context for $w \in (\Sigma^*)^{\langle m \rangle}$ iff $e[\![w]\!] \in L$.

Remark We note that the decomposition of a string into a multi-word and its context yields two objects of an intrinsically different nature (as opposed to a prefix and a suffix, which are both strings by definition). ♢

2.2 Trees

Another classical type of object in formal language theory is the tree. A lot of important algorithms over strings have soon been adapted to trees, and in this thesis we will mainly develop our theoretical results with respect to trees.

2.2.1 Classical trees

We consider the standard kind of tree which is rooted, ordered, and labeled. There are various representations for trees issuing from and yielding different formal perspectives. Traditionally, a tree can be seen as a set of nodes ordered by two binary relations, called *dominance* and *precedence*. One way

[1] A remark on notation: As is usual, we assume that $0 \in \mathbb{N}$. Throughout this work we will have object enumerations which we often denote using an index $i = 1, \ldots, n$ for $n \in \mathbb{N}$. Obviously, despite appearances these can also consist of one element only for $n = 1$ or be empty for $n = 0$ (and note that in that case statements containing the universal quantifier will be considered trivially true). For example, the only 0-word is of course $\langle\rangle$.

to make the properties of these ordering relations precise is by basing the representation of a tree on a set of addresses for its nodes, a *tree domain*:

Definition 5 *A set* $\mathsf{T} \subseteq \mathbb{N}^*$ *is a* tree domain *if*

- *for all* $v \in \mathsf{T}$*, if* u *is a prefix of* v *then we have* $u \in \mathsf{T}$ *as well, and*
- *for all* $v \in \mathsf{T}$ *and* $i \in \mathbb{N}$*, if* $vi \in \mathsf{T}$ *then* $vj \in \mathsf{T}$ *for all* $j \in \mathbb{N}$ *with* $0 \leq j \leq i$.

The node with address ε *is called the* root*, nodes with an address* $u \in \mathsf{T}$ *for which there is no* $i \in \mathbb{N}$ *such that* $ui \in \mathsf{T}$ *are called* leaves*, the nodes with an address* $vi \in \mathsf{T}$ *for* $v \in \mathsf{T}$ *and* $i \in \mathbb{N}$ *are called the* children *of the node with address* v*, and the node with address* v *is called their* mother*.*
There is a path *between two nodes with addresses* $u, v \in \mathsf{T}$ *if there is* $w \in \mathbb{N}^*$ *such that* $v = uw$*, and the* length *of that path is* $|w|$*.*
Nodes can be identified with addresses. Let the set of all tree domains be $\mathcal{T}$*.*

As mentioned above, the nodes of a tree can be labeled with symbols.
Thus, a tree can be represented as a pair $\langle \mathsf{T}, \tau \rangle$ where $\mathsf{T} \in \mathcal{T}$ is a tree domain and $\tau : \mathsf{T} \longrightarrow \Sigma$ is a total labeling function assigning a label from some (non-empty, if T is non-empty) alphabet Σ to each address in T.

Definition 6 *Let* $t = \langle \mathsf{T}, \tau \rangle$ *for some finite* $\mathsf{T} \in \mathcal{T}$*,* $\tau : \mathsf{T} \longrightarrow \Sigma$*, and alphabet* Σ*. The* depth *dpt*(t) *of* t *is the maximal length featured in* T*, i.e.,* $dpt(t) := |w|$ *for some* $w \in \mathsf{T}$ *such that for all* $w' \in \mathsf{T}$ *we have* $|w'| \leq |w|$*.*

As for strings, we can concentrate on substructures in a tree, most notably:

Definition 7 *Let* $t = \langle \mathsf{T}, \tau \rangle$ *be a tree and* $x \in \mathsf{T}$*. The* subtree of t rooted at x *is* $t/x := \langle \mathsf{T}', \tau' \rangle$ *with* $\mathsf{T}' = \{y \mid xy \in \mathsf{T}\}$ *and* $\tau'(x') = \tau(xx')$ *for* $x' \in \mathsf{T}'$*. The* set of all subtrees of t *is defined as* $Subt(t) := \{t/z \mid z \in \mathsf{T}\}$*.*

Another convenient (linear) representation for trees preferred by most algorithmical applications is obtained by conceiving the label of a node as a function symbol and the subtrees rooted at its children as arguments, which yields a representation of trees based on terms in the algebraic sense.
However, since functions have a fixed number of arguments, in order to build representations of trees as terms the elements of the labeling alphabet Σ have to be further specified, and the set of admissible trees over Σ is restricted.

Definition 8 *A* ranked alphabet *is a pair* $\langle \Sigma, \sigma \rangle$ *where* Σ *is a set of symbols and* $\sigma : \Sigma \longrightarrow \mathbb{N}$ *is a total function assigning a* rank *to each element of* Σ*. We will abbreviate* $\langle \Sigma, \sigma \rangle$ *to* Σ*, and we denote by* $\Sigma_n := \{f \in \Sigma \mid \sigma(f) = n\}$ *the set of symbols from* Σ *that are associated with rank* n*.*

Remark According to this (traditional) definition every symbol has a single rank, but it is just as possible to admit several ranks for one symbol. ♢

As references for formal term-based tree language theory, also see [37, 57, 58].

Definition 9
The set $\mathbb{T}_\Sigma$ of all trees over a ranked alphabet Σ *is defined as the smallest set such that* $\Sigma_0 \subseteq \mathbb{T}_\Sigma$ *and* $t = f(t_1, \ldots, t_n) \in \mathbb{T}_\Sigma$ *for all* $n \geq 1$, $f \in \Sigma_n$, *and* $t_1, \ldots, t_n \in \mathbb{T}_\Sigma$. *Any set* $L \subseteq \mathbb{T}_\Sigma$ *is called a* tree language.

Let $t = f(t_1, \ldots, t_n) \in \mathbb{T}_\Sigma$ *for* $n \geq 0$. *We define the* set $Subt(t)$ of all subtrees of t *as the smallest set such that* $t \in Subt(t)$ *and, if* $n \geq 1$, $Subt(t_i) \subseteq Subt(t)$ *for all* $i \in \{1, \ldots, n\}$. *We call* $t_1, \ldots, t_n$ *the* direct subtrees *of* t.
We define $Subt(T) := \bigcup\{Subt(t) \mid t \in T\}$ *for a set* $T \subseteq \mathbb{T}_\Sigma$.[2]

For our purposes it is admissible to identify a tree with its representation. Note that $\mathbb{T}_\Sigma$ is empty if Σ_0 is empty.

Remark Trees represent a generalization of strings if strings in turn are conceived as a special kind of trees. A possible translation of a string w over some set of symbols Σ' into a tree would construct a non-branching tree of depth $|w|$, fix the first symbol of w as a leaf label, the last symbol as the root label, and assign the symbols in between to the remaining nodes in the obvious way. We choose this orientation because, as will also become clearer from the next subsection, most adaptations of formal concepts for strings to trees are defined such that in many aspects subtrees rather correspond to prefixes than to suffixes. However, if our translation is to yield a tree in term notation then we have to settle the question of rank as well, which is achieved most easily by prefixing a special "start-of-string" symbol $\diamond$ to the string that is to be translated and defining a ranked alphabet Σ with $\Sigma_0 = \{\diamond\}$ and $\Sigma_1 = \Sigma'$. This theoretical inclusion relation between strings and trees is also the reason why in general negative (learnability) results for string languages carry over to trees, and positive results for trees apply to strings as well. ♢

We will only consider trees in term notation from here on, and we fix a ranked alphabet Σ for the rest of this subsection.

Tree contexts

Just like a string, a tree can be decomposed into various substructures: We fix a substructure and conceive the remaining parts of the tree as its context.

[2] There are several ways to use the symbol $\bigcup$ for the extended set union in the literature. We use it such that $x \in \bigcup \mathbb{A}$ for some set of sets $\mathbb{A}$ iff there is a set $A \in \mathbb{A}$ such that $x \in A$.

If we choose subtrees as defined in Definition 9 as the substructure of interest then a tree context would be defined as follows:

Definition 10 *Let $\Box$ be a special symbol of rank 0 not contained in Σ. A tree $c \in \mathbb{T}_{\Sigma\cup\{\Box\}}$ in which $\Box$ occurs exactly once is a* context*, and the set of all contexts over Σ is denoted by $\mathbb{C}_\Sigma$. For $c \in \mathbb{C}_\Sigma$ and $s \in \mathbb{T}_\Sigma \cup \mathbb{C}_\Sigma$, $c[\![s]\!]$ denotes the tree obtained by substituting s for $\Box$ in c. The depth $cdp(c)$ of a context c is the length of the path from the root to the leaf labeled by $\Box$.*

Let $t \in \mathbb{T}_\Sigma \cup \mathbb{C}_\Sigma$ and $T \subseteq \mathbb{T}_\Sigma \cup \mathbb{C}_\Sigma$. We define
$Cont(t) := \{c \in \mathbb{C}_\Sigma \mid \exists t' \in Subt(t) : c[\![t']\!] = t\}$ *and*
$Cont(T) := \{c \in \mathbb{C}_\Sigma \mid \exists t' \in Subt(T) : c[\![t']\!] \in T\}$.

For a tree $f(t_1, \ldots, t_n) \in \mathbb{T}_\Sigma$ and $j \in \{1, \ldots, n\}$ we define $f(t_1, \ldots, t_n)_j^\Box$ as a context $f(t'_1, \ldots, t'_n) \in \mathbb{C}_\Sigma$ with $t'_j = \Box$ and $t'_i = t_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$.[3]

Remark Intuitively, if t is an ordinary tree over Σ then the set $Cont(t)$ can be obtained by exploring all possibilities of replacing any subtree of t by $\Box$ – however, if t is a context then we only include the results of replacing subtrees that also contain the occurrence of the symbol $\Box$ because otherwise some elements of $Cont(t)$ would not meet the definition of a context. ♢

Observe that the decomposition of a tree into a subtree and its context yields two continuous objects and is comparable to the decomposition of a string into a prefix and a suffix (see Subsection 2.1.1). However, this is not the only conceivable decomposition of a tree. For example, one that imitates the decomposition of a string into a substring and a context enclosing it on both sides is expressed by the following set of definitions, which were developed in cooperation with Ryo Yoshinaka as preliminaries for [122]:

Definition 11 *Let $\Box \notin \Sigma$ be a special symbol of rank 0 as before. For $k \in \mathbb{N}$, a* k-stub *is a tree $s \in \mathbb{T}_{\Sigma\cup\{\Box\}}$ in which exactly k leaves are labeled by $\Box$. We will also denote the ith occurrence of $\Box$ from left to right for $1 \leq i \leq k$ by $\Box_i$, and let $s' = f(\Box_1, \ldots, \Box_k)$ denote a k-stub with $dpt(s') = 1$ and $f \in \Sigma_k$. The set of all k-stubs over Σ is denoted by $\mathbb{S}_\Sigma^k$, and we write*
$\mathbb{S}_\Sigma := \{S \subseteq \mathbb{T}_{\Sigma\cup\{\Box\}} \mid \exists k \in \mathbb{N} : S = \mathbb{S}_\Sigma^k\}$.

Remark A 0-stub is an ordinary tree from $\mathbb{T}_\Sigma$. A 1-stub meets the definition of a tree context as given by Definition 10. ♢

Let us fix a k-stub in a tree as the substructure of interest. Accordingly, the corresponding context (which in this case we will call an "environment") is a structure characterized by the following definition:

[3] Obviously, $f(t_1, \ldots, t_n)_j^\Box[\![t_j]\!] = f(t_1, \ldots, t_n)$.

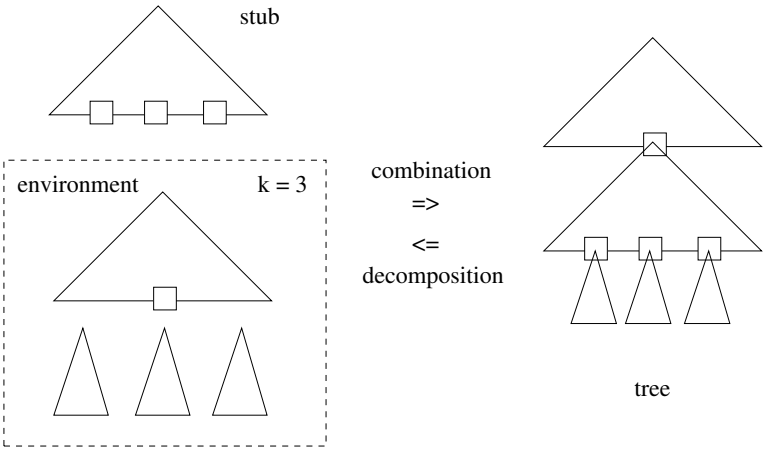

Figure 2.1: Object (tree), substructure, and context

Definition 12 *For $k \in \mathbb{N}$, we define a* k-environment *as a pair $\langle s, \langle t_1, \ldots, t_k \rangle\rangle$ in which $s \in \mathbb{S}^1_\Sigma$ is a 1-stub and $t_1, \ldots, t_k \in \mathbb{T}_\Sigma$. The set of all k-environments is denoted by $\mathbb{E}^k_\Sigma$, and we write $\mathbb{E}_\Sigma := \{E \subseteq \mathbb{S}^1_\Sigma \times (\mathbb{T}_\Sigma)^k \mid \exists k \in \mathbb{N} : E = \mathbb{E}^k_\Sigma\}$.*

The combining operation can be defined for stubs and environments as well:

Definition 13 *Let $s \in \mathbb{S}^k_\Sigma$ for some $k \in \mathbb{N}$, and $e = \langle s_1, \langle t_1, \ldots, t_k \rangle\rangle \in \mathbb{E}^k_\Sigma$. By $e[\![s]\!]$ we denote the tree that is obtained by replacing $\Box$ in s_1 by s and the occurrences of $\Box$ in s by $t_1, \ldots, t_k$ from left to right in that order.*[4]

However, note that with these definitions the sets of combinable substructures and contexts are parametrized by the measure k. Figure 2.1 illustrates the interrelations between trees, k-stubs and k-environments for $k = 3$.

Remark The empty string ε fulfils the role of representing the rather abstract concept of a string of length 0 as well as the role of a neutral element as the second component in a combination. In the case of trees the latter is fulfilled by $\Box$. For the former, while we can conceive a tree without nodes, including it into the set $\mathbb{T}_\Sigma$ poses all kinds of problems. We will rather adopt the concept of an "undefined tree" for some applications in chapters to come, and we will represent it by $\Box$. Note that $\Box$ is not an element of $\mathbb{T}_\Sigma$ by definition. $\diamondsuit$

[4] In [122] the occurrences of $\Box$ and their instantiation are handled via variable substitution instead of overriding the distinction between a symbol and its occurrence as done here – we choose the latter option for readability in order not to distract from the essential.

Tree automata

The traditional (term-based) tree language theory ([37, 57]) also establishes the concept of finite-state automata for trees. We consider bottom-up tree automata where the automaton assigns a state to each leaf and then works its way up by assigning states to the respective mother nodes according to the states assigned to the children and the admissible transition rules.

Definition 14 *Let Σ be finite.*
A finite-state tree automaton (FTA) *is a tuple $\mathcal{A} = \langle \Sigma, Q, F, \delta \rangle$ where*

- *Q is the finite set of states,*
- *$F \subseteq Q$ is the set of accepting states, and*
- *the transition relation δ is a set of mappings of the form $\langle f, q_1 \cdots q_n \rangle \mapsto q$ for $n \geq 0$, $f \in \Sigma_n$, and $q_1, \ldots, q_n, q \in Q$ where $q_1 \cdots q_n$ denotes the sequence of the states $q_1, \ldots, q_n$ in the same order which for $n = 0$ we will write as $\langle\rangle$. We will also use δ to denote a function such that $\delta(\langle f, q_1 \cdots q_n \rangle) = \{q \in Q \mid \exists \langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta\}$.*

From δ we can derive a function $\delta^* : \mathbb{T}_\Sigma \longrightarrow 2^Q$ such that $\delta^*(f(s_1, \ldots, s_n)) =$

$$\{q \in Q \mid \exists \langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta : \forall i \in \{1, \ldots, n\} : q_i \in \delta^*(s_i)\},^5$$

and a function $\delta^+ : Q \times \mathbb{C}_\Sigma \longrightarrow 2^Q$ such that $\delta^+(q, \Box) = \{q\}$ and for $e \neq \Box$,

$$\begin{aligned}\delta^+(q, e) = \{q' \in Q \mid \exists e', e'' \in \mathbb{C}_\Sigma : e = e'[\![e'']\!] \wedge \exists q'' \in Q : q' \in \delta^+(q'', e') \wedge \\ \exists n \geq 1 : \exists f(s_1, \ldots, s_n) \in \mathbb{T}_\Sigma : e'' = f(s_1, \ldots, s_n)^{\Box}_i \wedge \\ \exists \langle f, q_1 \cdots q_n \rangle \mapsto q'' \in \delta : q_i = q \wedge \forall j \in \{1, \ldots, n\} \setminus \{i\} : q_j \in \delta^*(s_j)\}.\end{aligned}$$

Intuitively, $\delta^*(t)$ is the set of all states that the tree t ends up in and $\delta^+(q, e)$ is the set of all states that can be reached from the state q by the context e.
For $q \in Q$, let $\mathcal{L}_q := \{s \in \mathbb{T}_\Sigma \mid q \in \delta^*(s)\}$ and
$$\mathcal{C}_q := \{e \in \mathbb{C}_\Sigma \mid \delta^+(q, e) \cap F \neq \emptyset\}.$$
Intuitively, $\mathcal{L}_q$ is the set of all trees that can end up in q and $\mathcal{C}_q$ is the set of all contexts that can lead from q into an accepting state.

We write $\mathcal{A}(s) = 1$ for $s \in \mathbb{T}_\Sigma$ if $\delta^*(s) \cap F \neq \emptyset$,
$\mathcal{A}(s) = 0$ if $\delta^*(s) \cap Q \neq \emptyset$ but $\delta^*(s) \cap F = \emptyset$, and
$\mathcal{A}(s) = *$ if $\delta^*(s) = \emptyset$.

The language accepted by $\mathcal{A}$ is defined as $\mathcal{L}(\mathcal{A}) := \{s \in \mathbb{T}_\Sigma \mid \mathcal{A}(s) = 1\}$.
Any tree language accepted by an FTA is called recognizable *or* regular.

[5] This is well-defined since we allow $n = 0$ as a base case, see Footnote 1.

If for all $n \geq 0$ *with* $\Sigma_n \neq \emptyset$ *and for all* $f \in \Sigma_n$ *and all* $q_1, \ldots, q_n \in Q$ *there is* $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta$ *then* $\mathcal{A}$ *is* total.
If for no $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta$ *there is* $\langle f, q_1 \cdots q_n \rangle \mapsto q' \in \delta$ *with* $q' \neq q$ *then* $\mathcal{A}$ *is* deterministic (a DFTA).
In that case we may abbreviate $\delta(\langle f, q_1 \cdots q_n \rangle) = \{q\}$ *to* $\delta(\langle f, q_1 \cdots q_n \rangle) = q$,
$\delta^*(s) = \{q\}$ *to* $\delta^*(s) = q$, *and*
$\delta^+(s, e) = \{q\}$ *to* $\delta^+(s, e) = q$.
Otherwise, if $\delta(\langle f, q_1 \cdots q_n \rangle) = \emptyset$ *or* $\delta^*(s) = \emptyset$ *or* $\delta^+(s, e) = \emptyset$ *then we say that* $\delta(\langle f, q_1 \cdots q_n \rangle)$, $\delta^*(s)$, *or* $\delta^+(s, e)$ *are undefined, respectively.*

Remarks Observe that transitions of the form $\langle a, \langle\rangle \rangle \to q$ with $a \in \Sigma_0$ that assign a state to a leaf trivially do not have to take into account any other previously computed states. However, we do not admit spontaneous transitions (i.e., the equivalents of ε-transitions in the string case) since δ is only defined for symbols from Σ, and δ^* is not defined for $\Box$ either. Also note that as soon as any of the sets Σ, Q, F, or δ is empty the language recognized by the automaton must be empty as well.

The definition of an FTA can be modified in several ways, with various consequences: For example, acceptance can be defined such that *all* possible parses must reach an accepting state (i.e., $\mathcal{A}(s) = 1$ iff $\delta^*(s) \subseteq F$), thus reducing the language that is recognized. Moreover, an FTA can be defined with explicit rejecting states, and all possible definitions of weak and strong acceptance or rejection including accepting, rejecting, and neutral states can be compared accordingly. Rejecting states may be of some use in learning applications in order to document as soon as possible that a certain state is not going to be accepting but there are alternative strategies that can do without. ♢

For any tree language $L \subseteq \mathbb{T}_\Sigma$, the equivalence relation $\equiv_L$ is defined such that $t \equiv_L t'$ for $t, t' \in \mathbb{T}_\Sigma$ iff $e[\![t]\!] \in L \Leftrightarrow e[\![t']\!] \in L$ for all contexts $e \in \mathbb{C}_\Sigma$. The *index* I_L of L is the cardinality of the set $\mathcal{E}_L := \{[t]_L \mid t \in \mathbb{T}_\Sigma\}$ where $[t]_L$ denotes the equivalence class containing t. The Myhill-Nerode theorem (see for example [67] for strings, and [37] for trees) states that I_L is finite iff L is recognizable by a finite-state automaton. For every regular tree language $L \subseteq \mathbb{T}_\Sigma$ there is a unique FTA $\mathcal{A}_L = \langle \Sigma, Q_L, F_L, \delta_L \rangle$ with I_L states where

- $Q_L := \mathcal{E}_L$,
- $F_L := \{x \in Q_L \mid x \subseteq L\}$, and
- $\delta_L := \{\langle f, x_1 \cdots x_n \rangle \mapsto x \mid x_1, \ldots, x_n, x \in Q_L \wedge f \in \Sigma_n \wedge$
$\exists t_1, \ldots, t_n \in \mathbb{T}_\Sigma : \forall i \in \{1, \ldots, n\} :$
$t_i \in x_i \wedge f(t_1, \ldots, t_n) \in x\}$.

Each state $x \in Q_L$ recognizes the equivalence class under $\equiv_L$ it is associated with, i.e., we have $\mathcal{L}_x = x$. As a consequence, $\mathcal{A}_L$ recognizes L and is the only state-minimal total DFTA recognizing L up to isomorphism (see [37]).
If we have $\mathbb{T}_\Sigma \setminus Subt(L) \neq \emptyset$ then there is a *failure state* q in that DFTA with $\mathcal{C}_q = \emptyset$ and hence there exists a non-total DFTA for L with one less state which is obtained by stripping the total one of the failure state q. We denote any of those two canonical DFTA for L by $\mathcal{A}_L$ in general but in cases where it matters, we will denote the total one by $\mathcal{A}_L^\bullet$ and the not necessarily total one without a failure state by $\mathcal{A}_L^\circ$.

Remark For $\mathbb{T}_\Sigma \setminus Subt(L) = \emptyset$ the automata $\mathcal{A}_L^\bullet$ and $\mathcal{A}_L^\circ$ coincide. ♢

Tree grammars

In parallel to the formal string grammars which determine the Chomsky hierarchy, one can also define formal grammars that generate tree languages. For the definitions of regular and context-free tree grammars, see [58].

In Chapter 4 we will refer to the special case of *simple context-free tree grammars*,[6] which were treated in a cooperation with Ryo Yoshinaka in [122] and which we define as follows:

Definition 15
Let $r \in \mathbb{N}$. An r-simple context-free tree grammar (r-SCFTG) *is a 4-tuple $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ where*

- *Σ is a ranked alphabet of* terminal symbols *(or* terminals*) such that $\Sigma_m = \emptyset$ for all $m \in \mathbb{N}$ with $r < m$,*
- *N is a ranked alphabet of* nonterminal symbols *(or* nonterminals*) such that $N_m = \emptyset$ for all $m \in \mathbb{N}$ with $r < m$,*
- *$I \subseteq N_0$ is a set of* initial symbols,[7] *and*
- *P is a finite set of production rules of the form $A \to s$ such that $A \in N_m$ is a nonterminal and $s \in \mathbb{S}^m_{\Sigma \cup N}$ is an m-stub for some $m \in \mathbb{N}$.*

Let P_m denote the set $\{A \to s \in P \mid A \in N_m\}$ for $m \in \mathbb{N}$.

For $m' \in \mathbb{N}$ and $u, v \in \mathbb{S}^{m'}_{\Sigma \cup N}$ we say that v can be derived (in one step) *from u in the grammar $\mathcal{G}$ and write $u \Rightarrow_\mathcal{G} v$ if there is $m \in \mathbb{N}$ and $A \to s \in P_m$ and $e \in \mathbb{E}^m_{\Sigma \cup N \cup \{\square\}}$ such that $e[\![A(\square_1, \ldots, \square_m)]\!] = u$ and $e[\![s]\!] = v$.*

[6] SCFTGs have also been called *linear context-free tree grammars* in the literature.
[7] Unlike in the established definition of context-free tree grammars we allow multiple initial symbols which however does not change the expressive power of the formalism.

Let the relation $\Rightarrow_{\mathcal{G}}^{*}$ *be the reflexive transitive closure of* $\Rightarrow_{\mathcal{G}}$.

The stub language *generated by a nonterminal* $A \in N_m$ *for some* $m \in \mathbb{N}$ *is the set* $\mathcal{L}(\mathcal{G}, A) = \{s \in \mathbb{S}_{\Sigma}^{m} \mid A(\Box_1, \ldots, \Box_m) \Rightarrow_{\mathcal{G}}^{*} s\}$. *We simply write* $\mathcal{L}_A$ *where* $\mathcal{G}$ *is understood. The* tree language generated by $\mathcal{G}$ *is* $\mathcal{L}(\mathcal{G}) = \bigcup\{\mathcal{L}_A \mid A \in I\}$.

Remark In accordance with the observation above that strings can be seen as a special case of trees, context-free string grammars can be interpreted as and rewritten into 1-SCFTGs. ♢

The yield operation

Traditionally, the *yield* of a tree is defined as the string that is obtained when reading off and concatenating the symbols labeling its leaves from left to right. In more formal terms, we can define a function *yd* from the set of trees over the ranked alphabet $\langle \Sigma, \sigma \rangle$ into the set of strings over Σ such that:

$$yd(t) = \begin{cases} t & \text{if } t = a \text{ for some } a \in \Sigma_0, \text{ and} \\ yd(t_1) \cdot \ldots \cdot yd(t_n) & \text{if } t = f(t_1, \ldots, t_n) \text{ for } n \geq 1,\ f \in \Sigma_n, \\ & \qquad \text{and } t_1, \ldots, t_n \in \mathbb{T}_{\Sigma}. \end{cases}$$

Moreover, this function can be extended to sets of trees such that we obtain $yd(L) := \{yd(t) \mid t \in L\}$ for any set $L \subseteq \mathbb{T}_{\Sigma}$.

There is a close connection between regular tree languages and context-free string languages: Every context-free string language equals the yield of a regular tree language, and the yield of any regular tree language is a context-free string language (see for example [37]), which implies that the class of the yields of regular tree languages and the class of context-free string languages coincide (a classical reference for this correspondence is [109]).

2.2.2 Multi-dimensional trees

A refinement of the tree notion is given by so-called *multi-dimensional trees*. Multi-dimensional trees are tree-like structures which can be represented using a parametrized number of dimensions in space.

Multi-dimensional trees can serve as a special kind of derivation trees for all kinds of tree generating formalisms, which makes them particularly interesting not only for pure formal tree language theory but also for mathematical linguistics. They were studied by Rogers [100, 101] in an attempt to fit the prominent linguistic formalism of *Tree Adjoining Grammars (TAG)*, which

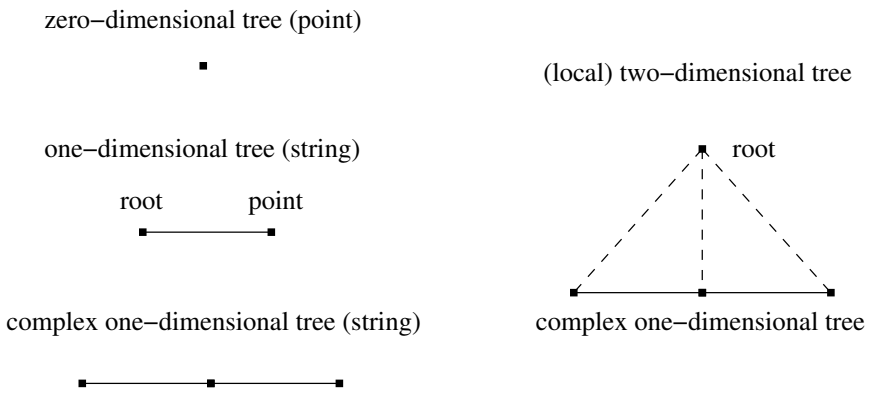

Figure 2.2: Points, strings, (classical) trees as a progression over dimensions

was developed by Joshi [72] in the 1980's in order to deal with certain non-context-free phenomena in natural languages, into a regular framework.
In his work, Rogers mainly considers three-dimensional trees but also mentions some linguistic phenomena where four dimensions allow a more satisfactory analysis (see [100], Section 9.2).

Rogers bases his representation on the notion of multi-dimensional tree domains and thus contents himself with an unpartitioned labeling alphabet. However, besides the conception which discriminates between those objects by a dimensional parameter there is also a related perspective which rather concentrates on the general properties that characterize classical and multi-dimensional trees alike and conceives them as tree-like structures where the children of a node are not only ordered linearly but arranged into another tree-like structure (albeit of a smaller number of dimensions). We propose a term-based representation for multi-dimensional trees (published in [74, 77]) centering on the second aspect. As a consequence, one can easily show that term-based algorithmic applications can directly be extended to classes of these kinds of tree structures as well.

Thus, when building up the hierarchy of multi-dimensional trees as a progression, the base is given by zero-dimensional trees or points, one-dimensional trees correspond to strings, and classical trees are two-dimensional, where the fact that the children of a node are arranged in a linear order is recaptured in the perspective that they form a (one-dimensional) string (see Figure 2.2).[8] This progression can be continued by defining three-dimensional trees of depth 1 (so-called *local trees*) as consisting of a root and a two-dimensional child structure. Complex three-dimensional trees can then be built by identi-

[8]Figures 2.2, 2.3, 2.5, and 2.6 are inspired by similar ones in [101].

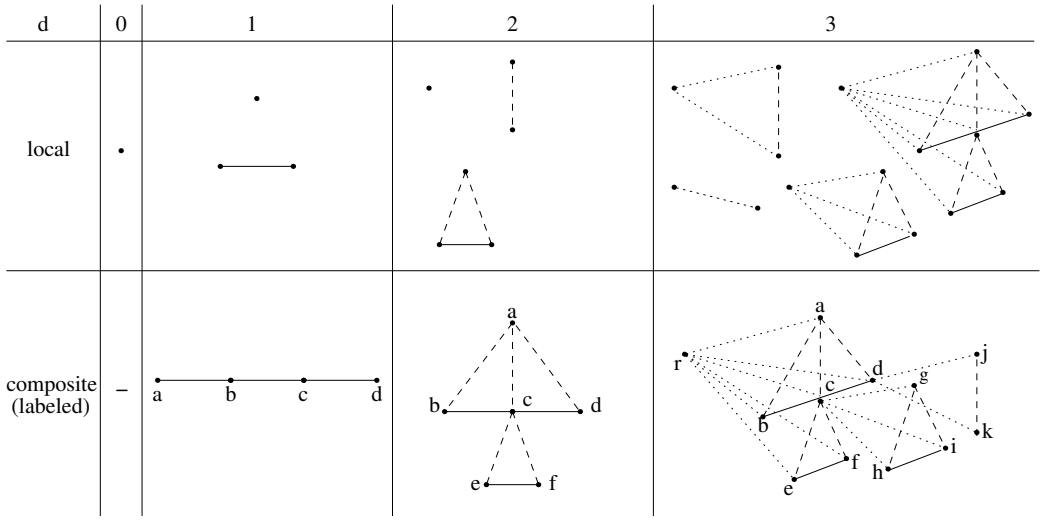

Figure 2.3: Including a third dimension

fying the root of one local tree with a node in the child structure of another. This is shown in Figure 2.3. The hierarchy thus sketched evidently extends to classes of objects over a fixed but arbitrary number of dimensions – but beyond three dimensions an intuitive graphical representation is rather hard to find, and we restrict our examples to the three-dimensional case.

A multi-dimensional tree is still a set of nodes. However, dominance and precedence are now generalized into a single hereditary relation which can be made precise by defining multi-dimensional tree domains as follows:

Definition 16 *Let* $\flat$ *be a special counting symbol. Define a hierarchical set based on* $\flat$ *such that* ${}^{0}\flat := \{\flat\}$ *and* ${}^{d}\flat := ({}^{d-1}\flat)^*$ *is the set of all sequences over elements of* ${}^{d-1}\flat$*. We fix that henceforth the concatenation* $y' \cdot y''$ *of two sequences* $y' \in {}^{d'}\flat$*,* $y'' \in {}^{d''}\flat$ *with* $d', d'' \in \mathbb{N} \setminus \{0\}$ *be only defined for* $d' = d''$*.*

We denote the set containing the unique 0-dimensional tree domain ${}^{0}\flat$ *by* $\mathcal{T}^0$ *and for* $d \geq 1$ *we define the set* $\mathcal{T}^d$ *of all* d-dimensional tree domains *as the set of all finite non-empty sets of addresses* $\mathsf{T} \subseteq {}^{d}\flat$ *such that*

- *for all* $x, x' \in {}^{d}\flat$*, if* $x \cdot x' \in \mathsf{T}$ *then we have* $x \in \mathsf{T}$ *as well, and*
- *for all* $y' \in {}^{d}\flat$*, if the set* $\mathsf{T}^{-1}_{y'} := \{y'' \in {}^{d-1}\flat \mid y' \cdot \langle y'' \rangle \in \mathsf{T}\}$ *is non-empty then it forms a* $(d-1)$*-dimensional tree domain, i.e.,* $\mathsf{T}^{-1}_{y'} \in \mathcal{T}^{d-1}$*.*

The node with address $\langle\rangle$ *is the* root*, and nodes with an address* $x \in \mathsf{T}$ *for which* T^{-1}_{x} *is empty are* leaves*. For any* $y' \in \mathsf{T}$*, the set* $\mathsf{T}^{-1}_{y'}$ *is called the* child

structure *of the node with address* y'*, the elements of* $\mathsf{T}_{y'}^{-1}$ *are the* children *of that node, and the node itself is their* mother.
There is a path *between two nodes with addresses* $x, y \in \mathsf{T}$ *if there is* $z \in {}^{d}\flat$ *with* $y = xz$*, and the* length *of that path is the length* $|z|$ *of the sequence* z.
Nodes can be identified with addresses.
From here on, we assume $d \geq 0$ *to be fixed if not specified explicitly.*

The nodes of a d-dimensional tree can be labeled with symbols as well. Thus, a d-dimensional tree can be represented as a pair $\langle \mathsf{T}, \tau \rangle$ with $\mathsf{T} \in \mathcal{T}^d$ and a total labeling function $\tau : \mathsf{T} \longrightarrow \Sigma$ for some unspecified alphabet Σ.

The definition of depth requires only minimal modifications with respect to the classical tree notion, and the same is true for the concept of a subtree:

Definition 17 *Let* $t = \langle \mathsf{T}, \tau \rangle$ *for some* $\mathsf{T} \in \mathcal{T}^d$, $\tau : \mathsf{T} \longrightarrow \Sigma$*, and alphabet* Σ.
The depth *of* t *is defined as* $dpt(t) := |w|$ *for some sequence* $w \in \mathsf{T}$ *such that for all* $w' \in \mathsf{T}$ *we have* $|w'| \leq |w|$*. Trees of depth at most 1 are called* local.

Definition 18 *Let* $t = \langle \mathsf{T}, \tau \rangle$ *for some* $\mathsf{T} \in \mathcal{T}^d$, $\tau : \mathsf{T} \longrightarrow \Sigma$*, and alphabet* Σ.
Let $x \in \mathsf{T}$*. The* subtree of t rooted at x *is* $t/x := \langle \mathsf{T}', \tau' \rangle$ *with*
$\mathsf{T}' = \{y \in {}^{d}\flat \mid x \cdot y \in \mathsf{T}\}$ *and* $\tau'(x') = \tau(x \cdot x')$ *for* $x' \in \mathsf{T}'$.
The set of all subtrees of t *is defined as* $Subt(t) := \{t/z \mid z \in \mathsf{T}\}$.

As a preparation for a term-based definition of multi-dimensional trees, we will now define the notion of a d-dimensional tree labeling alphabet Σ^d as a set of symbols in which each symbol is associated with some $(d-1)$-dimensional tree domain or with the empty set $\emptyset$.

Definition 19 *For* $d \geq 1$*, a* d-dimensional tree labeling alphabet *is a pair* $\Sigma^d := \langle \Sigma, \sigma \rangle$ *where* Σ *is a set of symbols and* $\sigma : \Sigma \longrightarrow \mathcal{T}^{d-1} \cup \{\emptyset\}$ *is a total function assigning an admissible child structure to each labeling symbol in* Σ.
A 0-dimensional tree labeling alphabet is a pair $\Sigma^0 := \langle \Sigma, \sigma \rangle$ *where* Σ *is a set of symbols and* $\sigma : \Sigma \longrightarrow \{\emptyset\}$*. We denote by* $\Sigma_t^d := \{f \in \Sigma \mid \sigma(f) = t\}$ *the set of all symbols from* Σ *that are associated with the structure* $t \in \mathcal{T}^{d-1} \cup \{\emptyset\}$ *in* Σ^d*. The* rank *of a symbol* $f \in \Sigma$ *is the cardinality of the set* $\sigma(f)$.
By $\sigma(\Sigma)$ *we denote the set* $\{\sigma(f) \mid f \in \Sigma\}$.

Remark Observe how the notion of rank (i.e., the number of nodes in the admissible child structure) still subsumes the one for classical (two-dimensional) trees from the previous subsection. Also note that, as for classical term-based trees and ranked alphabets, the notion of a d-dimensional tree labeling alphabet further restricts the set of conceivable labeled d-dimensional trees. ♢

The set $\mathbb{T}_{\Sigma^d}$ of all *d-dimensional trees over* Σ^d for a given measure $d \geq 0$ and d-dimensional tree labeling alphabet Σ^d can then be defined as follows:

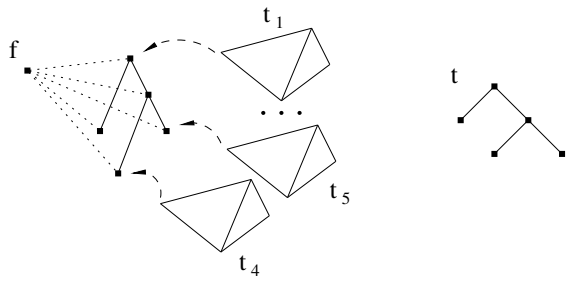

Figure 2.4: $f \in \Sigma_t^3$ (t_2 and t_3 are left out due to lack of space)

Definition 20 *Let $\Sigma^d := \langle \Sigma, \sigma \rangle$ be a d-dimensional tree labeling alphabet. For $d = 0$, we define $\mathbb{T}_{\Sigma^0} := \Sigma$. For $d \geq 1$, we define $\mathbb{T}_{\Sigma^d}$ as the smallest set such that $\Sigma \subseteq \mathbb{T}_{\Sigma^d}$ and $s = f(s_1, \ldots, s_n)_t \in \mathbb{T}_{\Sigma^d}$ for all symbols $f \in \Sigma_t^d$ with $t \in \mathcal{T}^{d-1} \cup \{\emptyset\}$ where $n = |t|$ and $s_1, \ldots, s_n \in \mathbb{T}_{\Sigma^d}$ are rooted in that order at the nodes of t assuming that the addresses in t are first ordered by length and then numerically by the length of their respective components.*

Remark For $d = 2$, this linearization corresponds to what is widely known as a breadth-first traversal. The length-numerical ordering is an ad-hoc settlement – any other linearization for the nodes of t would do as well. ♢

*Any set $L \subseteq \mathbb{T}_{\Sigma^d}$ is a d-*dimensional tree language.

As for classical trees, we define the set $Subt(s)$ as the smallest set such that $s \in Subt(s)$ and, if $n \geq 1$, $Subt(s_i) \subseteq Subt(s)$ for all $i \in \{1, \ldots, n\}$.
We call $s_1, \ldots, s_n$ the direct subtrees *of s, and*
we define $Subt(T) := \bigcup \{Subt(s) \mid s \in T\}$ for a set $T \subseteq \mathbb{T}_{\Sigma^d}$.

Figure 2.4 exemplifies the construction of a 3-dimensional tree using five other 3-dimensional trees $t_1, \ldots, t_5$ as direct subtrees and a symbol $f \in \Sigma_t^3$ from some 3-dimensional tree labeling alphabet Σ^3 for the root while observing the admissible child structure t for f depicted on the right.

Remark The multi-dimensional perspective yields a different generalization of strings than the one sketched in Subsection 2.2.1 above: Strings are no longer conceived as two-dimensional trees in which "accidentally" no node has more than one child but as one-dimensional trees in which the child structure of every node is a (zero-dimensional) point or the empty set by definition. However, in a conceivable formal translation of a string over some alphabet Σ' in classical notation into a one-dimensional tree we would still arrange the symbols of the string with the last symbol labeling the overall root of the

resulting tree due to the obvious parallels between prefixes and d-dimensional subtrees, and we would define a corresponding 1-dimensional labeling alphabet Σ^1 with $\Sigma^1_\emptyset := \{\diamond\}$ for the start-of-string symbol $\diamond$ and $\Sigma^1_{\{b\}} := \Sigma'$. ♢

Obviously, d-dimensional trees can be decomposed into substructures as well. If we fix the notion of a subtree from Definition 20 as the substructure of interest then we obtain a definition of d-dimensional tree contexts as follows:

Definition 21 *Let $\Box \notin \Sigma^d$ be a special symbol associated with $\emptyset$.*
A tree $c \in \mathbb{T}_{\Sigma^d \cup \{\Box\}}$ in which $\Box$ occurs exactly once is a context*, and the set of all contexts over Σ^d is denoted by $\mathbb{C}_{\Sigma^d}$. For $c \in \mathbb{C}_{\Sigma^d}$ and $s \in \mathbb{T}_{\Sigma^d} \cup \mathbb{C}_{\Sigma^d}$, $c[\![s]\!]$ denotes the tree obtained by substituting s for $\Box$ in c. The depth $cdp(c)$ is defined as the length of the path from the root to the leaf labeled by $\Box$.*

Let $t \in \mathbb{T}_{\Sigma^d} \cup \mathbb{C}_{\Sigma^d}$ and $T \subseteq \mathbb{T}_{\Sigma^d} \cup \mathbb{C}_{\Sigma^d}$. We define
$Cont(t) := \{c \in \mathbb{C}_{\Sigma^d} \mid \exists t' \in Subt(t) : c[\![t']\!] = t\}$ and
$Cont(T) := \{c \in \mathbb{C}_{\Sigma^d} \mid \exists t' \in Subt(T) : c[\![t']\!] \in T\}$.

Remark It is not difficult to define similar notions based on other d-dimensional substructures along the lines of Definitions 11–13 either. ♢

Due to the term-based notation developed above we are now able to give a definition for multi-dimensional finite-state tree automata which runs fairly parallel to the one for classical trees.

Definition 22 *A* d-dimensional finite-state tree automaton (d-FTA) *is a tuple $\mathcal{A} = \langle \Sigma^d, Q, F, \delta \rangle$ where*

- *Σ^d is a finite d-dimensional tree labeling alphabet,*
- *Q is the finite set of states,*
- *$F \subseteq Q$ is the set of accepting states, and*
- *the transition relation δ is a set of mappings of the form $\langle f, t(q_1, \ldots, q_n) \rangle \mapsto q$ where $t \in \mathcal{T}^{d-1} \cup \{\emptyset\}$, $f \in \Sigma^d_t$, $q \in Q$, and if $n \geq 1$ then $t(q_1, \ldots, q_n)$ denotes a $(d-1)$-dimensional tree over t whose nodes are labeled in length-numerical order by $q_1, \ldots, q_n \in Q$.*
  For $n = 0$ we write $t(q_1, \ldots, q_n)$ as $\Box$. By δ we also denote a function such that $\delta(\langle f, t(q_1, \ldots, q_n) \rangle) = \{q \in Q \mid \exists \langle f, t(q_1, \ldots, q_n) \rangle \mapsto q \in \delta\}$.

From δ we can derive a function $\delta^* : \mathbb{T}_{\Sigma^d} \longrightarrow 2^Q$ with $\delta^*(f(s_1, \ldots, s_n)_t) =$

$$\{q \in Q \mid \exists \langle f, t(q_1, \ldots, q_n) \rangle \mapsto q \in \delta : \forall i \in \{1, \ldots, n\} : q_i \in \delta^*(s_i)\}.$$

The language accepted by $\mathcal{A}$ is defined as $\mathcal{L}(\mathcal{A}) := \{s \in \mathbb{T}_{\Sigma^d} \mid \delta^*(s) \cap F \neq \emptyset\}$.
Any d-dimensional tree language accepted by a d-FTA is called recognizable

or regular. *The remaining notions and notations from Definition 14 can be adapted to multi-dimensionality as well.*

Remark Note that the criteria for effecting a certain transition includes a check of the admissible child structure – just as a classical finite-state tree automaton requires the input states of a transition to observe a certain linear order (i.e., to be assigned to specific direct subtrees of the input tree). ♢

Remark A 0-dimensional FTA is rather trivial since it simply checks the symbol labeling the input point and accepts or rejects it accordingly. ♢

For any d-dimensional tree language $L \subseteq \mathbb{T}_{\Sigma^d}$, we can define the equivalence relation $\equiv_L$ such that $s \equiv_L s'$ for $s, s' \in \mathbb{T}_{\Sigma^d}$ iff $e[\![s]\!] \in L \Leftrightarrow e[\![s']\!] \in L$ for all contexts $e \in \mathbb{C}_{\Sigma^d}$ (assuming that we choose to define a context as in Definition 21). Given the definition of a d-FTA above, it is rather straightforward to see that the Myhill-Nerode theorem for classical trees can be extended to multi-dimensional trees along with all its consequences, in particular the fact that for each d-dimensional tree language recognizable by some d-FTA there exists a unique deterministic d-FTA which is minimal with respect to the number of states.

The yield of multi-dimensional trees

As stated in Subsection 2.2.1, the yield of a classical tree is defined as the string that is obtained when reading off and concatenating the symbols labeling its leaves from left to right. From the perspective conceiving strings as 1-dimensional trees, it is a projection of the tree onto the next lower level, i.e., the number of dimensions available for its representation is reduced by one. In that sense, d-dimensional trees for $d \geq 3$ have several yields, one for each dimension that is taken away, down to the 1-dimensional string yield (which is the smallest "interesting" yield, since the next projection merely results in a point labeled by some symbol). A string yield is obtained by taking the yield $(d-1)$ times. However, the yield operation is precisely the occasion where a multi-dimensional tree reveals its additional structural information in comparison to a classical tree. This implies that when taking the yield of a d-dimensional tree with $d \geq 3$, some thought has to go into the question of how to interweave the child structures of the various local trees it is composed of to form a coherent $(d-1)$-dimensional tree, since there are often several possibilities (see Figure 2.5). Rogers solves this by introducing special nodes, so-called *heads*, and defines them such that in the child structure of every local d'-dimensional tree of depth 1 for $d' \leq d$ there is a unique path of heads, called the *primary spine* of that child structure, leading from the root to a leaf. This leaf is called the *foot* of the child structure and marks the splicing

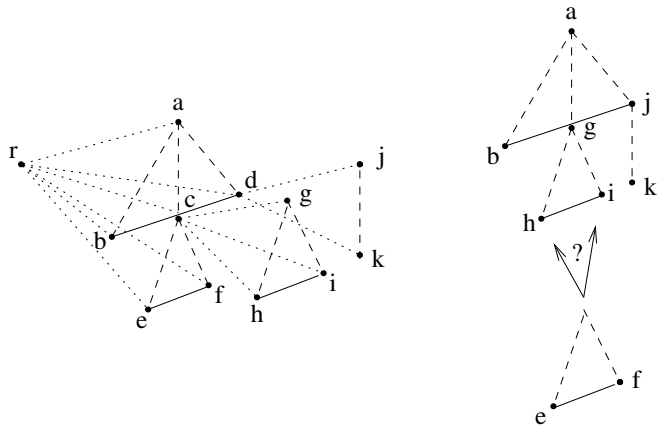

Figure 2.5: Ambiguity in the yield

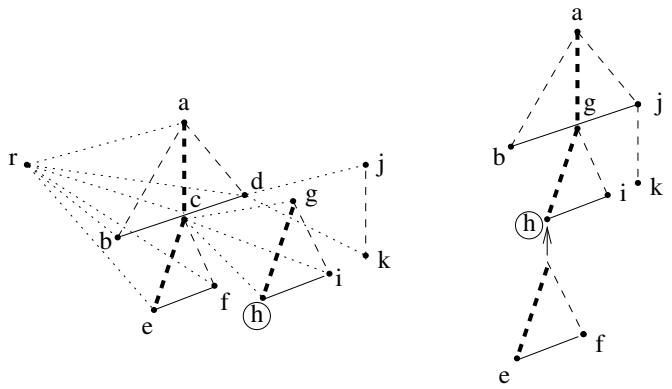

Figure 2.6: Ambiguity in the yield resolved by primary spines

point, i.e., the point where the yield of the subtree rooted at the mother of the foot node should be connected to the remaining part of the overall yield. In Figure 2.6 the ambiguity from Figure 2.5 has been resolved by marking the primary spines and ensuring their continuity.

Since the full formal definition of such a yield operation is quite intricate and since the main focus of this work does not lie on the relationship between multi-dimensional trees and their respective yields, we only give an intuitive sketch how such a yield operation could be defined using our term notation. The full domain-based approach has been elaborated by Rogers in [101].

In order to transfer Rogers' solution to our term notation we would define the notion of a *head-pointing* d-dimensional tree labeling alphabet as follows:

Definition 23 [9] *A* head-pointing *d-dimensional tree labeling alphabet* Σ_h^d *is a triple* $\langle \Sigma, \sigma, h \rangle$ *such that* $\Sigma^d = \langle \Sigma, \sigma \rangle$ *is an ordinary d-dimensional tree labeling alphabet and* $h \subseteq (\mathcal{T}^{d-1} \cup \{\emptyset\}) \times ({}^{d-1}\natural)^d$ *is a relation specifying the admissible head distributions by assigning to each admissible child structure* $t \in \sigma(\Sigma) \setminus \{\emptyset\}$ *sequences of the form* $H = \langle H_1, \ldots, H_d \rangle$ *where* $H_1, \ldots, H_d$ *are sets of addresses in t such that each* H_i *for* $1 \leq i \leq d-1$ *contains exactly one node in the child structure of each i-dimensional local tree of depth 1 in t and* H_d *contains exactly one node in t. For* $t = \emptyset$ *the only sequence assigned to t by h is the sequence* $H_0^d = \langle H_1, \ldots, H_d \rangle$ *with* $H_i = \emptyset$ *for* $1 \leq i \leq d$. *Observe that for* $t \neq \emptyset$ *the set* H_{d-1} *contains a subset of addresses in t that unambiguously defines a continuous path from the root of t to a single leaf. This is the* primary spine *of t.*

We denote by $\Sigma_{t,H}^d$ *the set of all symbols from* Σ_t^d *associated with the child structure t for which the specification of heads is encoded by the sequence H, i.e.,* $\langle t, H \rangle \in h$. *A tree over* Σ_h^d *is an expression of the form* $a_{\emptyset,H_0}$ *for* $a \in \Sigma_\emptyset^d$ *or of the form* $f(s_1, \ldots, s_n)_{t,H}$ *for* $t \in \mathcal{T}^{d-1}$, $f \in \Sigma_{t,H}^d$ *and* $s_1, \ldots, s_n \in \mathbb{T}_{\Sigma_h^d}$.

We would then define a yield function yd_{d-1} that takes a d-dimensional tree $s \in \mathbb{T}_{\Sigma_h^d}$ as input and returns a $(d-1)$-dimensional tree over an adapted head-pointing $(d-1)$-dimensional tree labeling alphabet $\Sigma_{h'}^{d-1}$ such that:

$$yd_{d-1}(s) = \begin{cases} a_{\emptyset,H_0^{d-1}} & \text{if } s = a_{\emptyset,H_0} \\ & \quad \text{for some } a \in \Sigma_\emptyset^d, \\ ydh(t, H, \langle yd_{d-1}(s_1), \ldots, yd_{d-1}(s_n) \rangle) & \text{if } s = f(s_1, \ldots, s_n)_{t,H} \\ & \quad \text{for some } t \in \mathcal{T}^{d-1}, \\ & \quad f \in \Sigma_{t,H}^d, \text{ and} \\ & \quad s_1, \ldots, s_n \in \mathbb{T}_{\Sigma_h^d}. \end{cases}$$

The function *ydh* would arrange the $(d-1)$-dimensional yields of $s_1, \ldots, s_n$ into the structure determined by t and then splice them together such that the primary spine of t, which is now interrupted by the yields of $s_1, \ldots, s_n$, forms once again a continuous spine – see Figure 2.7 for a more detailed illustration of how the $(d-1)$-dimensional yield of the tree in Figure 2.6 is determined by the structure of t plus the given head distribution.
Moreover, the function *ydh* would also have to specify the head distribution $H' = \langle H'_1, \ldots, H'_{d-1} \rangle$ in the resulting $(d-1)$-dimensional tree. This includes transferring the head distribution among the nodes of t as specified by H to the roots of the trees $yd_{d-1}(s_1), \ldots, yd_{d-1}(s_n)$ that were inserted at those

[9] This definition is inspired by a comparable one for classical trees in [55].

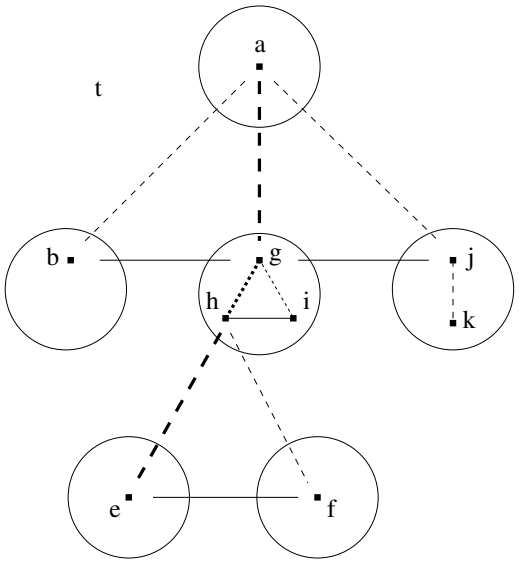

Figure 2.7: The function *ydh*

nodes on the one hand and keeping the distribution of heads in the inserted trees while adjusting the addresses of those heads to the surrounding structure on the other. Note that this entails that the tree domain underlying the $(d-1)$-dimensional yield of s has one continuous spine as well, determined by the set H'_{d-1}, which guarantees that we can recursively take the yield of any d-dimensional tree containing s as a subtree.
Finally, we would have to ensure that the yield of s can be given in term notation by defining a new $(d-1)$-dimensional tree labeling alphabet $\Sigma_{h'}^{d-1} = \langle \Sigma, \sigma', h' \rangle$ assigning to each symbol in Σ all the child structures and all the head distributions it can appear with in the $(d-1)$-dimensional yield of any tree $s' \in \mathbb{T}_{\Sigma_h^d}$. Note that as long as σ and h are finite σ' and h' are sure to be finite as well which is due to the fact that the admissible child structures in $\sigma(\Sigma)$ completely determine the form of all local d'-dimensional tree structures in a d-dimensional tree over Σ^d for any number $d' \leq d$ of dimensions.

The function yd_{d-1} can be extended to sets of d-dimensional trees such that $yd_{d-1}(L) := \{s' \in \Sigma_{h'}^{d-1} \mid \exists s \in L : s' = yd_{d-1}(s)\}$ for any set $L \subseteq \mathbb{T}_{\Sigma_h^d}$. The string language classes that can be associated with recognizable multi-dimensional tree languages via repeated applications of the yield operation form a well-nested infinite hierarchy ordered by the number of dimensions which is properly contained in the context-sensitive class and located in the family of the so-called *mildly context-sensitive* language classes, with the classes of (rather trivial) finite sets of single symbols, of regular languages,

of context-free languages (string yields of 2-dimensional tree languages), and of the string languages that are associated with recognizable 3-dimensional tree languages as the first four steps (see [100, 101]).

Remark A further line of research might be obtained by defining extended versions of the yield operation where the function *ydh* is allowed to permute and/or copy the yields of the subtrees $s_1, \ldots, s_n$ and then studying the effects on the resulting string language classes. ♢

Example

Example 1 contains the definition of a 3-FTA $\mathcal{A}^3_{ww}$ over a tree labeling alphabet Σ^3 recognizing a 3-dimensional tree language whose set of string yields $yd_1(L(\mathcal{A}^3_{ww}))$ is the well-known copy language $L_{ww} = \{ww | w \in \{a, b\}^+\}$.

Example 1 *Define* $\mathcal{A}^3_{ww} = \langle \Sigma^3, \{q_a, q_b, q_g, q_y, q_z, q_f, q_x\}, \{q_f\}, \delta \rangle$ *as follows.*

- $\Sigma^3 = \Sigma^3_\emptyset \cup \Sigma^3_{t_1} \cup \Sigma^3_{t_2}$ *with* $\Sigma^3_\emptyset = \{a, b, f, g, h\}$ *and* $\Sigma^3_{t_1} = \Sigma^3_{t_2} = \{f\}$ *for two tree domains* $t_1, t_2 \in \mathcal{T}^2$ *shown on the left-hand side in Figure 2.8. Note that in this example we admit several tree domains for* f *since it results in a more compact representation of the transition function* δ*.*

- *The (non-total, deterministic) transition function* δ *is defined such that*

$$\begin{array}{rclrcl}
\delta(\langle a, \emptyset \rangle) & = & q_a & \delta(\langle f, t_1(q_g, q_a, q_z, q_a) \rangle) & = & q_f \\
\delta(\langle b, \emptyset \rangle) & = & q_b & \delta(\langle f, t_1(q_g, q_b, q_z, q_b) \rangle) & = & q_f \\
\delta(\langle f, \emptyset \rangle) & = & q_z & \delta(\langle f, t_2(q_g, q_a, q_z, q_y, q_a) \rangle) & = & q_z \\
\delta(\langle g, \emptyset \rangle) & = & q_g & \delta(\langle f, t_2(q_g, q_b, q_z, q_y, q_b) \rangle) & = & q_z \\
\delta(\langle h, \emptyset \rangle) & = & q_y. & & &
\end{array}$$

Figure 2.8 shows the two non-empty tree domains $t_1, t_2 \in \mathcal{T}^2$ *admissible for* f *on the left, three trees* $t_a, t_b, t_c \in L(\mathcal{A}^3_{ww})$ *that are accepted by* $\mathcal{A}^3_{ww}$ *in the middle, and the 2-dimensional yield for* t_c*, whose 1-dimensional yield is the string* $abab \in L_{ww}$*, on the right.*

Wrapping up

Since our term notation for multi-dimensional trees conveys their tree structure, all algorithms for classical term-based trees referred to or developed in this work can be directly rewritten for multi-dimensional trees, with the admissible tree structures as a refinement introduced by the given alphabet. For previous work by the author on that subject, also see [74, 77, 80].

Due to the same reason, since the definitions of regular and context-free grammars for classical trees are based on the term notion as well (see [58])

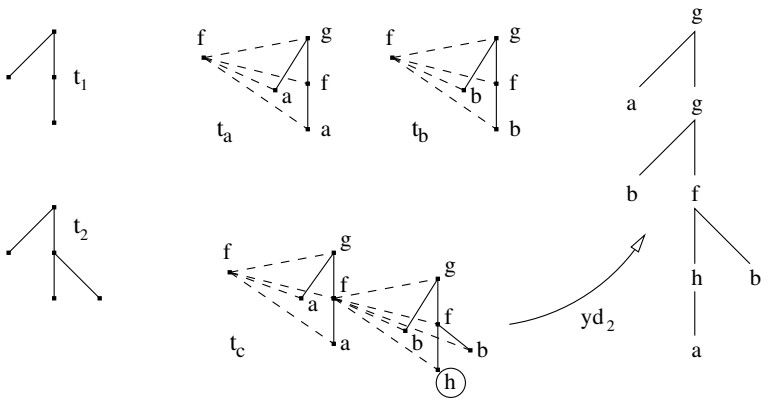

Figure 2.8: Example 1

this also entails that with the notation given above these formalisms can be extended such that we obtain definitions of regular and context-free multi-dimensional tree grammars in a straightforward way.

Rogers' main motivation for studying multi-dimensional trees was the representation of derivation in certain linguistic tree-generating formalisms. From a formal language theoretic perspective, this amounts to adding another level (or rather: arbitrarily many levels) of control. Accordingly, this means that when we define the yield of a multi-dimensional tree we would like to take into account the structural information that is given or relevant in the underlying formalism that we want to model. TAG is a linguistically motivated tree-generating grammar formalism in which the (two-dimensional) tree components of a grammar are labeled by symbols from an unranked alphabet and the splicing points are specified via so-called foot nodes as well. Rogers uses three dimensions in order to be able to rewrite the derivation process in a TAG, which involves breaking a tree into a discontinuous structure and interlacing it with another tree, as a regular process which simply concatenates local three-dimensional trees without decomposing them (see Figure 2.9). Taking the two-dimensional yield of the three-dimensional derivation trees thus obtained while respecting the head information as specified by the foot nodes allows us to reconstruct the trees that are generated by the TAG, and taking the yield once again we obtain the string language associated with it. Rogers has shown in [100] that the class of tree languages generated by a TAG corresponds exactly to the class of two-dimensional yields of recognizable three-dimensional tree languages. Intuitively, the tree components of a TAG unambiguously define a restricted set of admissible two-dimensional child structures which are then encoded into local three-dimensional trees

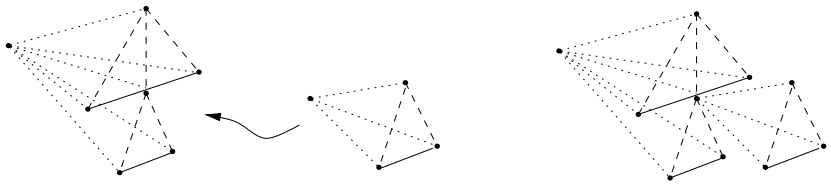

Figure 2.9: Derivation in TAG represented by local 3-dimensional trees

that can be combined in a regular manner as suggested by Figure 2.9. We have provided a most natural term-based handle on that restriction by our definition of a d-dimensional tree labeling alphabet.

In her M.A. thesis [73], the author has given direct translations between the set of three-dimensional trees that represent derivation trees of a given TAG and the set of two-dimensional trees that are obtained via another method of regularization for a TAG based on the algebraic operation of lifting.

2.3 Outlook

This concludes our collection of object structures that have been and still are of particular consequence within the realm of Grammatical Inference. Of course this enumeration merely covers a certain section of the object axis. Other options for further study include

- *hedges*, i.e., sequences of trees, which were introduced by Courcelle [39],
- *pictures*, i.e., matrices of symbols, see [44],

  Remark Note that pictures allow another generalization where strings can be seen as vectors of symbols, i.e., matrices of width/height 1. $\diamondsuit$
- and *graphs*, see [50] for graph grammars.

Moreover, in general all objects considered so far are assumed to be finite, and we could study various kinds of extensions to infinity, see for example [91] for so-called ω-regular infinite strings and [13] for finite-state automata with infinite transition graphs.

For each of the considered kinds of objects we would have to define at least the notion of a substructure and its context, an unambiguous concatenation operation for those structures, suitable descriptions of formalisms recognizing them and possibly among those a uniquely determined canonical one.

Chapter 3

Learning regular tree languages in different settings

3.1 Learning in a bounded number of steps

As mentioned in the Introduction, the area of Grammatical Inference (GI) is concerned with learning algorithms, i.e., algorithms that infer a description (for example, a grammar or an automaton) for an unknown formal language from given information in finitely many steps.

One of the first specifications of such an algorithmic learning process was given by Gold in [59] where he defines the concept of *identification in the limit*: The learner receives a continuous stream of positive examples, and has to react to each datum with an updated hypothesis representing a possible description of the conjectured target language. A learner is said to *learn* a certain language *in the limit* if after a finite number of given data and computation steps each description proposed by the learner is correct with respect to the target and moreover is identical to the previous one.[1] A learner learning a language *class* in the limit learns each member of the class in the limit.

The sources of information in a learning process can take various forms. Besides the continuous data stream mentioned above, the most studied ones are probably the following. Let L be the target language under consideration.

- *Equivalence queries (EQs):* The learner has access to a teacher, also called *oracle*, who is able to answer queries inquiring the correctness of a description $\mathcal{A}$ for the target, i.e., queries of the form '$L = \mathcal{L}(\mathcal{A})$?', and the teacher returns a *counterexample* $c \in (L \setminus \mathcal{L}(\mathcal{A})) \cup (\mathcal{L}(\mathcal{A}) \setminus L)$ in case

[1] There are other learning models in which the learner is allowed to change its hypothesis description after the first successful identification as long as it still correctly represents the target language. Such a learner is said to be *behaviourally correct*, see for example [71].

of non-equivalence. EQs are very powerful in the sense that if an EQ is answered in the positive then obviously this implies immediate success for the learner but also in the sense that they require a very powerful teacher since an entire description has to be judged with respect to L, and the definition given here even assumes that the teacher accepts any correct description as a solution rather than requiring isomorphy to a concrete reference. Moreover, the task of deciding where to direct the search for a suitable hypothesis is left entirely to the learner as well.

- *Membership queries (MQs)*: The learner has access to an oracle who is able to answer whether an object w is a member of L, i.e., queries of the form '$w \in L$?'. MQs are less demanding for the teacher in terms of complexity and represent a very flexible and useful tool for the learner to test its own hypothesis.

- *Finite samples of L, which can be positive (subsets of L) or negative (subsets of the complement of L)*: No teacher is required, or rather, the only task of the teacher is to procure such a sample and give it to the learner before the learning process is entered. However, in order to ensure identification, especially if no query-based information source is given, those sets have to fulfil certain significant properties with respect to L. The learner has to rely on the given sample containing all necessary information in some form. We will concentrate on cases where this information notably depends on the structure of the target, and more specifically, on the structure of a canonical description of L.

  Remark We could also conceive learning situations where instead of basing the given information on certain structural properties of L a collaborative teacher may merely hide some unambiguous code for the target in the data that is passed on, and it is always possible to define a corresponding learner which just "happens" to search for a code of just that type and can thus claim to identify all members of the language class in question correctly. However, in this work we will mostly eclipse such a scenario by our assumptions. On the other hand, we do not allow for "evil informants" or wrong samples either, i.e., we assume that all information given to the learner is correct. ♢

Scenarios involving queries have also been called *on-line* because interaction with the teacher is possible throughout the learning process. Scenarios not involving queries are accordingly called *off-line*. Throughout this work we will concentrate on combinations of the information sources given above. There are other options, such as for example *correction queries* where the teacher

anwers a membership query for a non-member by returning a correct element that would have been "closest" to the queried one, under various definitions of the notion of 'closest' – for references, see Subsection 3.3.4.

During the past decades a considerable range of learning algorithms based on various settings have been developed and presented in the literature. One of the language classes studied most thoroughly with respect to its algorithmic learnability so far is the class of regular languages. It has been shown (see [6, 8, 59]) that the regular class cannot be efficiently learned from one kind of query or sample alone. Three well-studied combinations of two such sources favourable to regular inference join MQs and EQs (exploited by Angluin's seminal algorithm LSTAR; see [5, 102, 46]), MQs and a finite positive sample ([3, 14]), and finite positive and negative samples (RPNI; [94, 41, 93]).

Remark We give references featuring string and tree formalisms without distinction since most of the positive algorithmic learnability results for strings cited above have soon been adapted to trees. Also recall that as strings can be conceived as a special case of trees (see Subsection 2.2.1) in general negative learnability results for strings imply negative results for trees. ♢

Our learning model of choice assumes that a learner should take a finite number of computation steps and then present a solution. This conception has also been termed *"one-shot" learning* – see [87, 88]. In addition to being finite, we want the number of steps taken by the learner to be bounded by some significant measure with respect to the target, ideally a polynomial bound, although that measure should obviously not be known to the learner. In the references given in the previous paragraph the successful identification of a regular language L is possible in a polynomially bounded number of steps and/or queries depending on the size of a correct canonical description for L and of the data received throughout the process.

Remark Observe that we refer to the overall number of steps and not just the number of steps taken by the learner in order to update its hypothesis between two inputs, which is often considered the main criterion of interest in the model of identification in the limit – also see [87, 88] for a comparison of one-shot learning and Gold-style learning.
The use of the term 'one-shot' in connection with EQs has been criticized. However, in our model the learner still decides precisely when it will return its official answer regardless of whether the last EQ was answered in the positive (but of course this provides a very good termination criterion). ♢

The various information sources listed above have been considered more or less suitable for the description of a natural learning process such as first language acquisition (also see the book on "Linguistic nativism and the

poverty of the stimulus" by Clark and Lappin [35]). Tentatively, membership queries could be related to a real-world situation where a child utters a sentence and may judge from the reactions if the utterance is correct, and correction queries could be related to a situation where the reaction to an incorrect sentence is more informative than mere incomprehension. Equivalence queries might correspond to a more general situation where a child realizes that it is understood most of the time and can thus claim to "speak" a language. However, note that in the real world there is rarely such a sharply outlined notion as "having learned successfully and being aware of it" – such a success must be defined in a purely asymptotic way as a lack of negative evidence for a long time and/or a large number of cases. Furthermore, any (correct) linguistic material in the child's environment represents a positive sample but finding natural examples for purely negative samples is hard – one might think of nonsensical poems or language games from which a child might observe that the application of correct morphological or syntactical rules may still result in expressions without an accepted interpretation.
We assume that any real-world learner will soon form a hypothesis consisting of more complex structures beyond the merely sequential input. Since trees are a classic and reasonably expressive means to represent linguistic structure we will concentrate on tree formalisms as the formalization of choice and give all our results in this chapter with respect to trees.

All algorithms described or cited in this chapter that infer a finite-state automaton for a regular language L from certain data are based on the retrieval of the correct set of classes under the equivalence relation $\equiv_L$ using the substructure-context relation in some way or other. Many of them recur to the concept of a so-called *observation table*, which is a versatile and relatively abstract means to perform and document the inference process at the same time since it allows to represent the relationship between the available substructures and contexts with respect to a language in a fairly intuitive way without having to deal with the idiosyncrasies of more specific descriptions such as finite-state automata or grammars, which moreover can be easily derived from it. The concept can be traced back to Gold [60] where he constructs a "state characterization matrix" and has been widely established in the GI community since Angluin has adopted it for the description of her seminal algorithm LSTAR in [5]. The majority of the algorithms by which the results in the present chapter are inspired use an observation table, and since we are also interested in examining the tools of choice in algorithmic learning theory as such we have chosen to concentrate on the notion of an observation table in this chapter as well.

Contents of this chapter

In this chapter we would like to contribute to the completion of the picture that is framed by existing algorithms for well-established settings.

We begin by giving some more preliminary definitions that are used and needed for the inference of regular languages based on observation tables and on deterministic finite-state automata as the chosen language description in Subsection 3.2.1. Most of them were originally formulated for a more specific purpose and/or setting but we have tried to cast them into a more general form in order to make them applicable to the whole range of algorithms that appear in this work. We will comment on some of these modified notions and we reprove all necessary theorems involving them in Subsection 3.2.2.
In Section 3.3 we present a meta-algorithm which is intended as a generalization of existing and conceivable learning algorithms based on the retrieval of the correct set of equivalence classes under the Myhill-Nerode relation $\equiv_L$ as defined in Subsection 2.2.1 from a combination of the information sources introduced above using an observation table, and we discuss its behaviour, design, and complexity. We have tried to factorize our learner into a reasonably large number of subprocedures in order to achieve a clearer perspective on the basic routines that regardless of the nature of the available sources have to be run through in all settings alike. Preliminary work by the author in that direction can also be found in [74, 77, 75, 78, 80].
In Section 3.4 we attempt to reproduce the discussion from Section 3.3 for the case where the language description of choice is a so-called *residual finite-state automaton*, which is a special kind of non-deterministic automaton, and we argue that this shift from determinism to non-determinism introduces a considerable amount of additional intricacy when instead of the string case we consider the case of trees. We conclude with some more reflections on the relationship between the deterministic and the residual approach and sketch some ideas for future work in Section 3.5.

3.2 Theoretical preliminaries

3.2.1 Definitions and concepts

Let us fix a finite ranked alphabet Σ and a regular tree language $L \subseteq \mathbb{T}_\Sigma$ for the rest of this section. The type of learning algorithm we will consider tries to infer a state-minimal (D)FTA for L from given information. As mentioned above, it solves this task principally by means of an *observation table* in which it keeps track of the information it has obtained and processed so far. The rows of the table are labeled by trees, the columns are labeled by contexts.

Definition 24 *A triple $T = \langle S, E, obs\rangle$ containing two finite sets $S \subseteq \mathbb{T}_\Sigma$ and $E \subseteq \mathbb{C}_\Sigma$ with $\Box \in E$ is called an* observation table *if*

- *S is* subtree-closed*, i.e., if $f(s_1, \ldots, s_n) \in S$ implies $s_1, \ldots, s_n \in S$ for all $f(s_1, \ldots, s_n) \in \mathbb{T}_\Sigma$, and*
- *$obs : \mathbb{T}_\Sigma \times \mathbb{C}_\Sigma \longrightarrow \{0, 1, *\}$ is a total function with*

$$obs(s,e) = \begin{cases} 1 & \text{if } e[\![s]\!] \in L \text{ is confirmed,} \\ 0 & \text{if } e[\![s]\!] \notin L \text{ is confirmed,} \\ * & \text{if unknown.} \end{cases}$$

For trees $s, s' \in \mathbb{T}_\Sigma$ and a context $e \in \mathbb{C}_\Sigma$, we will say that

- *e is a* positive *context for s if $obs(s, e) = 1$,*
- *e is a* negative *context for s if $obs(s, e) = 0$, and that*
- *e is a* distinguishing *or* separating *context for s and s' if $obs(s, e) \neq obs(s', e)$.*

The row of $s \in \mathbb{T}_\Sigma$ *is $row(s) := \{\langle e, obs(s, e)\rangle \mid e \in E\}$.*
Also, let $row(X) := \{row(s) \mid s \in X\}$ for any set $X \subseteq \mathbb{T}_\Sigma$,
and $row(s)(e) := obs(s, e)$ for $s \in \mathbb{T}_\Sigma$ and $e \in E$.

We say that two rows $r_1, r_2 \in row(\mathbb{T}_\Sigma)$ are obviously different *and denote it by $r_1 <> r_2$ if there is a context $e \in E$ with $\{r_1(e), r_2(e)\} = \{0, 1\}$.*
We also say that two trees $s_1, s_2 \in \mathbb{T}_\Sigma$ are obviously different *and denote it by $s_1 <> s_2$ if $row(s_1) <> row(s_2)$.*
For $row(s_1) = row(s_2)$ we also write $s_1 \approx s_2$.

A row or table not containing any occurrences of the $*$-symbol is complete.

Remark We define the function *obs* for the entire domain of $\mathbb{T}_\Sigma \times \mathbb{C}_\Sigma$, and *row* for the entire domain of $\mathbb{T}_\Sigma$ instead of restricting them to $S \times E$ and S (as is more usual in the literature, see for example [5]), respectively. We do so in order to express that while any concrete table built by the learner is finite theoretically a learner can always establish the row for any given tree by filling in the cells to the best of its knowledge (and it will simplify the appearance of our proofs). However, we must ensure that the sets in the range of *row* are of finite and equal cardinality (namely, $|E|$) to make rows comparable in a finite number of steps. We also assume that all information actually available to the learner is correct, i.e., if ever we have $obs(s, e) = 1$ for some $s \in \mathbb{T}_\Sigma$ and $e \in \mathbb{C}_\Sigma$ then $e[\![s]\!]$ really is a member of the target language L, and if we have $obs(s, e) = 0$ then it really is not. $\diamondsuit$

According to criteria proper to each learner the set labeling the rows of the table will be divided into two sets RED and BLUE such that RED $\cup$ BLUE $= S$ and RED $\cap$ BLUE $= \emptyset$. Intuitively, RED will contain the elements the learner has already processed and set down to represent constituents of his current hypothesis whereas BLUE will contain elements the learner has already "in sight" and intends to consider next. During the learning process, elements are moved successively from BLUE to RED and BLUE may be filled up with elements from a third "supply" set WHITE.[2]

Let us fix an observation table $T = \langle S, E, obs \rangle$ with $S =$ RED $\cup$ BLUE for the rest of this section.

Definition 25 *The table T is* closed *if* $\forall s \in \text{BLUE} : \exists s' \in \text{RED} : \neg(s <> s')$.

Definition 26 *The table T is* weakly consistent *if,*
for all $f(s_1, \ldots, s_n), f(s'_1, \ldots, s'_n) \in S$,
$\neg(s_i <> s'_i)$ *for* $1 \leq i \leq n$ *implies* $\neg(f(s_1, \ldots, s_n) <> f(s'_1, \ldots, s'_n))$.

We have added 'weakly' because the $*$-symbol may mask differences that are not yet obvious to the learner. Definition 27 is intended to rule out the cases in which hidden differences might prove fatal:

Definition 27 *The table T is* strongly consistent *iff it is* weakly consistent *and, for all $s_1, s_2 \in S$ with $s_1 \neq s_2$, $\neg(s_1 <> s_2)$ implies that $row(s_1)$ and $row(s_2)$ are complete and therefore obviously identical.*

Remark This implies that any complete and weakly consistent table is also strongly consistent, and in that case we will simply call it consistent. $\diamondsuit$

[2] We choose these names as a reference to the description of state-merging algorithms in [41] where in turn they have been inspired by the blue-fringe algorithm described in [86].

From T we can construct an FTA $\mathcal{A}_T = \langle \Sigma, Q_T, F_T, \delta_T \rangle$ defined by

- $Q_T := row(\text{RED})$,
- $F_T := \{q \in Q_T \mid q(\Box) = 1\}$, and
- $\delta_T := \{\langle f, q_1 \cdots q_n) \rangle \mapsto q \mid q_1, \ldots, q_n, q \in Q_T \wedge$
$\exists s_1, \ldots, s_n, f(s_1, \ldots, s_n) \in S :$
$\forall i \in \{1, \ldots, n\} : \neg(q_i <> row(s_i)) \wedge$
$\neg(q <> row(f(s_1, \ldots, s_n)))\}$.

It follows directly from the definition of δ_T and of strong consistency that if T is strongly consistent then $\mathcal{A}_T$ is deterministic. As S is subtree-closed, if T is closed then there is no $q \in Q_T$ with $\mathcal{L}_q = \emptyset$, i.e., all states can be reached.[3]

In any DFTA $\mathcal{A}$, for each state q the set $\mathcal{L}_q$ of trees ending up in q is a subset of some equivalence class under the relation $\equiv_{\mathcal{L}(\mathcal{A})}$ whereas in a non-deterministic FTA the set $\mathcal{L}_q$ may be the union of subsets of several of those classes. Abstractly speaking, all learning algorithms figuring in Section 3.3 can be conceived to start out with a provisional set of equivalence classes and then to try and converge to the partition induced by $\equiv_L$ on $\mathbb{T}_\Sigma$ by splitting up or merging these classes according to the obtained information, which effectively translates into inferring a state-minimal DFTA $\mathcal{A}_L$ for L where for each state q the set $\mathcal{L}_q$ corresponds exactly to an equivalence class of L and no class is represented twice. The sets S and E were so named to indicate that the rows of S are candidates for *states* in a DFTA for L and that E contains *experiments* proving that two elements of S belong to distinct classes and should represent different states.

Another concept we will need is the *subtree automaton* for a finite set of trees. This definition is loosely based on [41] and [93].

Definition 28 *Define the* subtree automaton (STA) *for a finite set $X \subseteq \mathbb{T}_\Sigma$ as the deterministic automaton $STA(X) := \langle \Sigma, Q, F, \delta \rangle$ with*

- $Q = \{\{s\} \mid s \in Subt(X)\}$,
- $F = \{\{s\} \mid s \in X\}$, *and*
- $\delta = \{\langle f, \{s_1\} \cdots \{s_n\} \rangle \mapsto \{f(s_1, \ldots, s_n)\} \mid$
$s_1, \ldots, s_n, f(s_1, \ldots, s_n) \in Subt(X)\}$.

[3]Note: While theoretically $\mathcal{A}_T$ may have a failure state without being total this never happens with tables built by any concrete learning algorithm we consider in this work.

Observe that the states of $STA(X)$ are labeled by singleton sets of trees. The meta-algorithm we will present in Section 3.3 will unite some of these sets during the learning process according to the information that has been processed up to that point, so that at each step each state is labeled by the set of all trees ending in it the learner has found so far.

Finally, finite samples given to a learner can have certain useful properties. Definition 29 is taken in a modified form from [14]. Let $\mathcal{A}_L^\bullet = \langle \Sigma, Q_\bullet, F_\bullet, \delta_\bullet \rangle$ be the total and $\mathcal{A}_L^\circ = \langle \Sigma, Q_\circ, F_\circ, \delta_\circ \rangle$ the not necessarily total minimal DFTA for L (see Subsection 2.2.1).

Definition 29 *A finite set $X_+ \subseteq Subt(L)$ is* representative *for L if for every transition $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_\circ$ there is $f(s_1, \ldots, s_n) \in Subt(X_+)$ with $\delta_\circ^*(s_i) = q_i$ for $1 \leq i \leq n$.*

Intuitively, X_+ is representative for L if every transition of $\mathcal{A}_L^\circ$ is needed to assign states to all trees in X_+. Equivalently, for every tree $f(t_1, \ldots, t_n) \in Subt(L)$ there is $f(s_1, \ldots, s_n) \in Subt(X_+)$ such that $s_i \equiv_L t_i$ for $1 \leq i \leq n$.

Definition 30 *A finite set $X \subseteq \mathbb{T}_\Sigma$ is* separative *for L if for all $q_1, q_2 \in Q_\bullet$ with $q_1 \neq q_2$ there is $s \in X$, $s' \in \mathbb{T}_\Sigma$, and $e \in \mathbb{C}_\Sigma$ with $s = e[\![s']\!]$ such that*

- *either $\delta_\bullet^*(s') = q_1$ and $\delta_\bullet^+(q_1, e) \in F_\bullet$ and $\delta_\bullet^+(q_2, e) \notin F_\bullet$*
- *or $\delta_\bullet^*(s') = q_2$ and $\delta_\bullet^+(q_2, e) \in F_\bullet$ and $\delta_\bullet^+(q_1, e) \notin F_\bullet$.*

Intuitively, X is separative for L if for any pair of distinct states of $\mathcal{A}_L^\bullet$ it contains a tree s composed of a subtree ending in one of them and a context leading from there into an accepting state which proves that these states must represent different classes under $\equiv_L$ due to the fact that the context does not lead into an accepting state from the other.

3.2.2 Discussing definitions and their consequences

In the previous subsection we have given a definition of an observation table which is as general as possible in order to make it a suitable instrument to describe the actions of all the algorithms we will study in this chapter. To achieve more clarity, at the beginning of this subsection we would like to compare our definitions briefly to those in [46] which are direct adaptations of the ones in Angluin's seminal paper [5] to the tree case.[4]

[4]There is a more recent version of [46] (see [48]) which introduces various improvements but we cite this version in order to demonstrate the effects of an adaptation to trees that keeps as close as possible to Angluin's original paper [5].

Note that the tables in [46] are complete. In an incomplete table there may be rows which should be identical but differ in the cells for some contexts such that one contains the *-symbol and the other a value from $\{0, 1\}$. Those rows will appear as distinct states and hence the DFTA derived from a strongly consistent table including such a situation may not be minimal. On the other hand, in a complete table T we can translate $\neg(s_1 \approx s_2)$ for $s_1, s_2 \in S$ into $s_1 <> s_2$, and $\neg(s_1 <> s_2)$ into $s_1 \approx s_2$.

The setting in [46] requires a stricter condition on the partitioning of S, and BLUE is accordingly defined as the set containing all and only trees $f(s_1, \ldots, s_n) \in \mathbb{T}_\Sigma$ with $f \in \Sigma_n$ and $s_1, \ldots, s_n \in$ RED that are not in RED themselves, i.e., BLUE equals the set

$$\Sigma(\text{RED}) := \{f(s_1, \ldots, s_n) \mid f \in \Sigma_n \wedge s_1, \ldots, s_n \in \text{RED}\} \setminus \text{RED}.^5$$

As a consequence, the DFTA derived from a closed and consistent table in [46] is always total whereas with our relaxed conditions it may not be.

Also note the impact of closedness: Consider a complete and consistent table T with BLUE $= \Sigma($RED$)$. If T is not closed then $\mathcal{A}_T$ cannot be total as there is at least one one-symbol extension of RED labeling a BLUE row that is not a state in Q_T, and thus there is no transition for that symbol from the corresponding rows in RED.[6]

Furthermore, Angluin's definition of an observation table for strings in [5] features the additional condition that E should be *suffix-closed*, and the definition for trees in [46] accordingly features the condition of

- *generalization-closedness* for E, i.e., for all $e \in E$,
  the set $\{e' \in \mathbb{C}_\Sigma \mid \exists e'' \in \mathbb{C}_\Sigma : e'[\![e'']\!] = e\}$ is also a subset of E,
  and moreover requires

- *S-composure*, i.e., for all $e \in E$ and all $s \in \mathit{Subt}(e)$,
  if s is not a context then $s \in S$.

However, neither of those two properties is essential for the extraction of an automaton from a table – the only relevant function of E is to create rows that can be compared cell by cell. This can easily be seen by taking a table T where E is generalization-closed and S-composed and replacing each context in E by an equivalent one yielding the same column such that the resulting set E' is not generalization-closed or S-composed anymore. As neither S nor the set $\mathit{row}(S)$ has changed the automaton derived from the modified table will recognize exactly the same language as $\mathcal{A}_T$.

[5] Note that in order to keep the table finite this requires that Σ be finite as well.

[6] Please observe that the set $\Sigma($RED$)$ and the term 'one-symbol extension of RED' include all trees $s \notin$ RED consisting of a single node, i.e., with $s = a$ for some symbol $a \in \Sigma_0$.

The article [46] also contains a lemma adapted from [5] which (translated into our definition framework) states that

- if T is a complete closed and strongly consistent observation table with $\text{BLUE} = \Sigma(\text{RED})$ then $\mathcal{A}_T(e[\![s]\!]) = 1$ iff $e[\![s]\!] \in L$ for all $s \in S$ and $e \in E$, i.e., $\mathcal{A}_T$ correctly predicts the content of all cells in T.

However, clearly this is only ensured because in their framework E is both generalization-closed and S-composed – in general there may be contexts in E that cannot be parsed correctly by the automaton $\mathcal{A}_T$ because the necessary transitions and/or accepting states are not represented in S.

Therefore, if we assume our relaxed and generalized conditions for a table T and the sets BLUE and E as given in Subsection 3.2.1 we must formulate and prove several new lemmata and theorems preserving exactly those results that are essential for the success of an algorithmic learner of the kind we will study in Section 3.3. To that end, let us first establish some more definitions and assumptions. We fix a complete, closed, and consistent observation table $T = \langle S, E, obs\rangle$ with $S = \text{RED} \cup \text{BLUE}$ for the rest of this section. Note that as T is complete the consistency of T is automatically strong and $\mathcal{A}_T$ is a DFTA. Moreover, we will assume $obs(s, e) \in \{0, 1\}$ for all $s \in \mathbb{T}_\Sigma$ and $e \in E$, i.e., the information whether $e[\![s]\!]$ is in L or not is available.
Define $\mathcal{L}_T := \{e[\![s]\!] \mid s \in S \wedge e \in E\}$.

Definition 31 *We say that S is* T-representative *if for all $s \in Subt(\mathcal{L}_T \cap L)$ with $s = f(s_1, \ldots, s_n)$ there are elements $s'_1, \ldots, s'_n, f(s'_1, \ldots, s'_n) \in S$ such that $s_i \approx s'_i$ for $1 \leq i \leq n$ and $f(s_1, \ldots, s_n) \approx f(s'_1, \ldots, s'_n)$.*

Definition 32 *We say that $\mathcal{A}_T$ is* T-consistent *if we have*
$$\mathcal{A}_T(e[\![s]\!]) = 1 \Leftrightarrow obs(s, e) = 1 \text{ for all } s \in S \text{ and } e \in E.$$

We will show that S being T-representative suffices to make $\mathcal{A}_T$ T-consistent.

Remark Note that if E is generalization-closed and S-composed and if moreover we have $\text{BLUE} = \Sigma(\text{RED})$ then S is T-representative. However, in general the reverse implication does not hold as $\Sigma(\text{RED})$ may contain transitions that are not needed to parse the contexts in E and as stated above, none of those properties is a necessary condition to make the FTA $\mathcal{A}_T$ T-consistent. ♢

The following lemma is based on a corresponding one in [5] but adapted to trees and to our framework of conditions for T.

Lemma 1 *We have $\delta_T^*(s) = row(s)$ for every $s \in S$.*

Proof. By induction over the depth of s. The claim clearly holds for $s \in \Sigma_0$ by the fact that T is closed so that $row(s) \in Q_T$, and by the definition of δ_T.[7] Now let $s = f(s_1, \ldots, s_n)$ and assume the claim to hold for $s_1, \ldots, s_n$. As S is subtree-closed we have $s_1, \ldots, s_n \in S$. As T is closed we have $row(s_i) \in Q_T$ for $1 \leq i \leq n$, and $\delta_T^*(s_i) = row(s_i)$ by the induction assumption. Then

$$\begin{aligned} \delta_T^*(f(s_1, \ldots, s_n)) &= \delta_T(\langle f, \delta_T^*(s_1) \cdots \delta_T^*(s_n)\rangle) \\ &= \delta_T(\langle f, row(s_1) \cdots row(s_n)\rangle) \\ &= row(f(s_1, \ldots, s_n)) \end{aligned}$$

by the definition of δ_T, and thus the claim is proven. ■

Lemma 2 *If S is T-representative then we have $\delta_T^*(s) = row(s)$ for all $s \in Subt(\mathcal{L}_T \cap L)$.*

Proof. If $s \in S$ then we refer to Lemma 1. As S is T-representative we obviously have $s \in S$ for all $s \in Subt(\mathcal{L}_T \cap L) \cap \Sigma_0$. Otherwise, let $s = f(s_1, \ldots, s_n)$ and assume the claim to hold for $s_1, \ldots, s_n$. As S is T-representative and as T is closed there are $s_1', \ldots, s_n', f(s_1', \ldots, s_n') \in S$ with $row(s_i) = row(s_i') \in Q_T$ for $1 \leq i \leq n$ and $row(s) = row(f(s_1', \ldots, s_n')) \in Q_T$. Then

$$\begin{aligned} \delta_T^*(f(s_1, \ldots, s_n)) &= \delta_T(\langle f, \delta_T^*(s_1) \cdots \delta_T^*(s_n)\rangle) \\ &= \delta_T(\langle f, row(s_1) \cdots row(s_n)\rangle) && \text{(induction base)} \\ &= \delta_T(\langle f, row(s_1') \cdots row(s_n')\rangle) \\ &= \delta_T(\langle f, \delta_T^*(s_1') \cdots \delta_T^*(s_n')\rangle) && \text{(by Lemma 1)} \\ &= \delta_T^*(f(s_1', \ldots, s_n')) \\ &= row(f(s_1', \ldots, s_n')) && \text{(by Lemma 1)} \\ &= row(f(s_1, \ldots, s_n)), \end{aligned}$$

and thus the claim is proven for all $s \in Subt(\mathcal{L}_T \cap L)$. ■

Lemma 3 *If S is T-representative then $\mathcal{A}_T$ is T-*consistent*.*

Proof. Let $s \in Subt(\mathcal{L}_T)$.
For $s \in Subt(L)$ this is a direct consequence of Lemma 2.
We show $\mathcal{A}_T(s) \neq 1$ for $s \notin Subt(L)$ by induction over the depth of s.
Let $s = a$ for some $a \in \Sigma_0$. If $s \in S$ then $\delta_T^*(s) = row(s)$ by Lemma 1 and as $s \notin Subt(L)$ we have $obs(s, \Box) = 0$ and $row(s) \notin F_T$. Otherwise $\delta_T(\langle a, \langle\rangle\rangle)$ and hence $\delta_T^*(s)$ are undefined and $\mathcal{A}_T$ does not accept s either.
Now let $s = f(s_1, \ldots, s_n)$ and assume the claim to hold for $s_1, \ldots, s_n$. If there

[7] Recall that due to completeness we can translate $\neg(s_1 <> s_2)$ into $s_1 \approx s_2$ for all $s_1, s_2 \in Subt(\mathcal{L}_T)$. This also applies to the definition of δ_T.

is $j \in \{1, \ldots, n\}$ with $\delta_T^*(s_j) = \emptyset$ then we obtain $\delta_T^*(s) = \emptyset$ and $\mathcal{A}_T(s) = *$ immediately by the definition of δ_T^*. Assume $\delta_T^*(s_i) = row(s_i)$ for $1 \leq i \leq n$. If there is $f(s'_1, \ldots, s'_n) \in S$ with $row(s'_i) = row(s_i)$ for $1 \leq i \leq n$ then we have $\delta_T^*(s) = row(f(s'_1, \ldots, s'_n)) = row(s)$ by the definition of δ_T. Obviously, we also have $obs(s, \Box) \neq 1$ and $row(s) \notin F_T$ due to the fact that $s \notin Subt(L)$. If there is no such $f(s'_1, \ldots, s'_n) \in S$ then the necessary transition is missing and we get $\mathcal{A}_T(s) = *$. Either way, since there is no row for s containing a 1 in any cell $\mathcal{A}_T$ cannot assign a state in F_T to s, and the claim is shown. ■

Recall that we have fixed T to be complete, closed, and consistent, and that I_L denotes the number of equivalence classes under the relation $\equiv_L$.

Theorem 1 *If $\mathcal{A}_T$ is T-consistent then $\mathcal{A}_T$ has $I_{\mathcal{L}(\mathcal{A}_T)} - 1$ or $I_{\mathcal{L}(\mathcal{A}_T)}$ states.*

Note that the DFTA $\mathcal{A}_T$ contains at most one failure state: Any $s \in \text{RED}$ such that there is $e \in E$ with $obs(s, e) = 1$ represents a state q with $\mathcal{C}_q \neq \emptyset$ because e must be in $\mathcal{C}_q$ due to the fact that $\mathcal{A}_T$ is T-consistent. Since T is complete there is at most one row without any 1s. If there is a state q_0 with $\mathcal{C}_{q_0} = \emptyset$ then let $\mathcal{A}_T^\circ$ be the DFTA obtained by stripping $\mathcal{A}_T$ of q_0, i.e.,
$\mathcal{A}_T^\circ := \langle \Sigma, Q_T^\circ, F_T, \delta_T^\circ \rangle$ with $Q_T^\circ = Q_T \setminus \{q_0\}$ and
$\delta_T^\circ := \{\langle f, q_1 \cdots q_n) \rangle \mapsto q \mid q_1, \ldots, q_n, q \in Q_T \setminus \{q_0\} \wedge$
$\exists s_1, \ldots, s_n, f(s_1, \ldots, s_n) \in S :$
$\forall i \in \{1, \ldots, n\} : \neg(q_i <> row(s_i)) \wedge \neg(q <> row(f(s_1, \ldots, s_n)))\}$,
and let $\mathcal{A}_T^\circ := \mathcal{A}_T$ otherwise. Note that as $\mathcal{A}_T^\circ$ recognizes the same language as $\mathcal{A}_T$, if $\mathcal{A}_T$ is T-consistent then $\mathcal{A}_T^\circ$ is as well.

We prove Theorem 1 by the following lemma. The proof is partly based on a parallel one in [5] but adapted to trees and to our conditions for T.

Lemma 4 *The DFTA $\mathcal{A}_T^\circ$ is isomorphic to the minimal DFTA $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$.*

Proof. Let $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ = \langle \Sigma, Q, F, \delta \rangle$. For each $q \in Q$, define
$$row(q) := \{\langle e, o \rangle \mid e \in E \wedge o \in \{0, 1\} \wedge (o = 1 \Leftrightarrow \delta^+(q, e) \in F)\}.$$
Since $\mathcal{A}_T^\circ$ is T-consistent and $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ recognizes the same language as $\mathcal{A}_T^\circ$ $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ is T-consistent as well. Then for each $s \in S$ and for $e \in E$, we have $\delta^*(e[\![s]\!]) = \delta^+(\delta^*(s), e) \in F$ iff $obs(s, e) = 1$, and hence $row(\delta^*(s)) = row(s)$. As a consequence, the minimal(!) DFTA $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ must have at least as many states as there are distinct rows in T, i.e., at least as many states as $\mathcal{A}_T^\circ$. Obviously $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ and $\mathcal{A}_T^\circ$ have the same number of states.
Thus, for each $s \in S$ there is a unique $q \in Q$ with $row(q) = row(s)$, namely $\delta^*(s)$. Define a mapping ϕ by $\phi(row(s)) = \delta^*(s)$. This mapping is a bijection.

It obviously maps $\delta_T^*(s) = row(s)$ to $\delta^*(s)$ for all $s \in S$. We must verify that it also preserves the transition function and carries F_T to F.
For all $s_1, \ldots, s_n, f(s_1, \ldots, s_n) \in S$, we have

$$\begin{aligned} \phi(\delta_T(\langle f, row(s_1) \cdots row(s_n)\rangle)) &= \phi(row(f(s_1, \ldots, s_n))) \\ &= \delta^*(f(s_1, \ldots, s_n)). \end{aligned}$$

We also have

$$\begin{aligned} \delta(\langle f, \phi(row(s_1)) \cdots \phi(row(s_n))\rangle) &= \delta(\langle f, \delta^*(s_1) \cdots \delta^*(s_n)\rangle) \\ &= \delta^*(f(s_1, \ldots, s_n)). \end{aligned}$$

And hence,

$$\phi(\delta_T(\langle f, row(s_1) \cdots row(s_n)\rangle)) = \delta(\langle f, \phi(row(s_1)) \cdots \phi(row(s_n))\rangle),$$

i.e, all transitions in δ_T are preserved. Moreover, since $\mathcal{A}_T^\circ$ and $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ have the same number of states, are both DFTA, and recognize the same language $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ cannot have more transitions than $\mathcal{A}_T^\circ$.
To complete this isomorphism proof we must show that ϕ maps F_T to F: Consider $row(s) \in F_T$ and $\phi(row(s)) = q \in Q$. Then $\mathcal{A}_T^\circ(s) = 1$, and q must be an accepting state in F as well because $\delta^*(s) = q$ and $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ accepts s. Conversely, let $\phi(row(s')) = q' \in F$ for $s' \in S$. As $row(q') = row(s')$ we have $obs(s', \Box) = 1$ and therefore $row(s') \in F_T$ by the definition of F_T. ■

This also concludes the proof of Theorem 1: We have shown that the DFTA $\mathcal{A}_T$ is either isomorphic to $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$ or has one more state q_0 such that $\mathcal{C}_{q_0} = \emptyset$, and q_0 is reachable due to the fact that S is subtree-closed.

Remark Theorem 1 even holds in a slightly modified form if the table T is closed and strongly consistent but not complete – in that case $\mathcal{A}_T$ may have several failure states instead of one, and the DFTA obtained by stripping $\mathcal{A}_T$ of *all* those failure states is isomorphic to $\mathcal{A}_{\mathcal{L}(\mathcal{A}_T)}^\circ$. ♢

Theorem 2 *Let S be representative for L. Then $\mathcal{L}(\mathcal{A}_T) = L$.*

Proof. We prove Theorem 2 in two steps: First we show $\mathcal{A}_T(s) = 1$ iff $s \in L$ for all $s \in Subt(L)$ by induction over the depth of s.
Let $s \in \Sigma_0$. As S is representative for L there is $s' \in S$ with $s \equiv_L s'$ and hence $s \approx s'$ so that we have $row(s) \in row(S)$ and, as T is closed, even $row(s) \in Q_T$. The claim follows directly by the definition of δ_T.
Now let $s = f(s_1, \ldots, s_n)$ and assume the claim to hold for $s_1, \ldots, s_n$. Again, as S is representative for L and T is closed there are $s_1', \ldots, s_n' \in S$ and $f(s_1', \ldots, s_n') \in S$ with $row(s_i) = row(s_i') \in Q_T$ for $1 \leq i \leq n$ and $row(s) = row(f(s_1', \ldots, s_n')) \in Q_T$. Then

$$\begin{aligned}
\delta_T^*(f(s_1,\ldots,s_n)) &= \delta_T(\langle f, \delta_T^*(s_1)\cdots\delta_T^*(s_n)\rangle) \\
&= \delta_T(\langle f, row(s_1)\cdots row(s_n)\rangle) \quad \text{(induction base)} \\
&= \delta_T(\langle f, row(s_1')\cdots row(s_n')\rangle) \\
&= \delta_T(\langle f, \delta_T^*(s_1')\cdots\delta_T^*(s_n')\rangle) \quad \text{(by Lemma 1)} \\
&= \delta_T^*(f(s_1',\ldots,s_n')) \\
&= row(f(s_1',\ldots,s_n')) \quad \text{(by Lemma 1)} \\
&= row(f(s_1,\ldots,s_n)),
\end{aligned}$$

and thus $\mathcal{A}_T(s) = 1$ iff $s \in L$ since we have $row(s) \in F_T$ iff $obs(s, \Box) = 1$ and $obs(s, \Box) = 1$ iff $s \in L$.

For $s \notin Subt(L)$ we have $\mathcal{A}_T(s) \neq 1$ by the same chain of argument as in the proof of Lemma 3. This concludes the proof of Theorem 2. ■

Remark Considering the proofs of Lemma 3 and of Theorem 2 it is easy to see that S may even contain elements that are not subtrees of L, i.e., requiring a *subset* of S to be representative for L is enough. Intuitively, this is due to the fact that the row of any $s \notin Subt(L)$ must consist of 0s only. ♢

Corollary 1 (of Theorem 1 and 2)
If T is complete, closed, and strongly consistent and if there is a subset of S that is representative for L then $\mathcal{A}_T$ recognizes L and represents the state-minimal DFTA $\mathcal{A}_L^\circ$ for L or has one more state which is a failure state.

Proof. The fact that a subset of S is representative for L implies that $\mathcal{A}_T$ is T-consistent by Theorem 2 and then we can apply Theorem 1. ■

3.3 The meta-algorithm GENDET

In this section we present a meta-algorithm which is intended to generalize over several existing algorithms for settings including equivalence queries, membership queries, positive and negative finite samples of the regular target tree language L as information sources and thus covers the whole range of combinations allowing the inference of a DFTA from those sources with polynomial complexity. We have made special efforts to design GENDET as modular as possible with the main objective to find out and describe the essential subprocedures of such a learning process. A further objective was the potential reusability of certain subroutines in the design of other (meta-)algorithms (see for example Section 3.4). GENDET uses the concept of an observation table in order to retrieve the correct set of equivalence classes defined by L as well as to document the retrieval process at the same time.

3.3.1 Description of GENDET

Let L be the regular target tree language over some finite ranked alphabet Σ'. We represent the input of our meta-algorithm as a 4-tuple $IP = \langle EQ, MQ, X_+, X_- \rangle$ with two Boolean values indicating if the learner has access to a teacher answering equivalence queries (EQs) and/or membership queries (MQs), respectively, and a positive and a negative finite sample of L (i.e., $X_+ \subseteq L$ and $X_- \subseteq \mathbb{T}_{\Sigma'} \setminus L$) in that order. We assume the components of IP to be visible as global variables throughout all procedures, as well as all other variables that are not explicitly passed on, such as the Boolean termination criterion τ and the set C of all counterexamples obtained so far. Let $T = \langle S, E, obs \rangle$ with $S = \text{RED} \cup \text{BLUE}$ and $\mathcal{O} = \langle \Sigma, Q_{\mathcal{O}}, F_{\mathcal{O}}, \delta_{\mathcal{O}} \rangle$ always be defined by the current values of their respective components, and $obs(s, e) := *$ if the value has not been set explicitly by any procedure. We also assume that the learner is given the *smallest* alphabet Σ such that $L \subseteq \mathbb{T}_\Sigma$ (but note that alternatively Σ could be deduced step by step from the given data). Moreover, we want L to contain at least one tree of depth at least 1 which seems justifiable since finite languages are trivial to learn.

We will present GENDET step by step. The main body is simple:

Input: $IP = \langle EQ, MQ, X_+, X_- \rangle$, a finite ranked alphabet Σ.
Output: A DFTA.

```
1     INIT; CLOSURE;
2     τ := 0;
3     while τ = 0 do
          NEXTDIST; CLOSURE;
4     return A_T.
```

The table T is initialized via the procedure INIT and closed via the procedure CLOSURE. While the termination criterion τ is not met we call the procedure NEXTDIST in order to check if we can still find states in our current hypothesis automaton that should be split up and then we reestablish closedness. At the end GENDET returns the automaton derived from T.

procedure INIT

```
5     O := MQORACLE;
6     P := POOL;
7     if P ≠ ∅ then RED := {min_≼(P)}; else RED := ∅;
8     if MQ = 1 then BLUE := Σ(RED); else BLUE := P ∩ Σ(RED);
9     E := {□};
10    WHITE := P \ (RED ∪ BLUE);
11    C := ∅;
12    UPDATE.
```

The procedure INIT initializes the membership oracle $\mathcal{O}$ (using the procedure MQORACLE), and the components of T. It resorts to the procedure POOL to establish the set P of trees from which to draw the initial members of RED, BLUE, and WHITE. Throughout the learning process, the set RED consists of those candidates that were fixed to represent a state in the final automaton whereas BLUE contains candidates representing states into which there exists a transition from a (possibly empty) combination of states that are represented in RED. If we have access to a perfect membership oracle then we consider all transitions that can be constructed based on the alphabet Σ and the elements of RED, otherwise we consider only those transitions that are actually represented in the given data (line 8). WHITE constitutes the pool of candidates from which BLUE will be filled up.
Let $\preceq$ be a total order relation which compares trees first by their depth and then according to some other arbitrary but fixed criterion. For $X \subseteq \mathbb{T}_\Sigma$, we define $min_\preceq(X) := s \in X$ with $\forall s' \in X : s \preceq s'$ if $X \neq \emptyset$, and $min_\preceq(\emptyset) := \square$, where we use $\square$ ("the undefined tree") as a failure symbol. Thus, RED will

be initialized with a singleton set containing the minimal tree with respect to $\preceq$ in P if P is non-empty, and with the empty set otherwise.
The value of C is set to $\emptyset$ to indicate that no counterexample has yet been retrieved. The cells of the initial table are filled by the procedure UPDATE.

---

procedure `MQORACLE`
13 **if** $MQ = 1$ **then return** $\mathcal{O}_L$;
14 **else return** $STA(X_+)$.

The procedure MQORACLE returns the best membership oracle the learner can hope for at the time. For $MQ = 1$ this is trivial: We instantiate it with a total DFTA $\mathcal{O}_L$ recognizing L which is a blackbox to the learner. Otherwise the oracle is initialized by the subtree automaton for X_+ (which for $X_+ = \emptyset$ is the empty automaton $\langle \Sigma, \emptyset, \emptyset, \emptyset \rangle$). This imperfect oracle will be developed further during the process every time the learner gains new insight.

---

procedure `POOL`
15 **if** $IP = \langle 0, 1, \emptyset, X_- \rangle$ **then**
$X_+ := \{s \in \mathbb{T}_\Sigma \mid dpt(s) \leq 2^{|Cont(X_-)|} - 1 \ \wedge \ \mathcal{O}(s) = 1\}$;
16 **if** $EQ = 0 \wedge X_+ \neq \emptyset$ **then return** $Subt(X_+)$;
17 **else if** $MQ = 1$ **return** $\{s \in \mathbb{T}_\Sigma \mid dpt(s) \leq 1\}$;
18 **else return** $\emptyset$.

The procedure POOL builds a suitable set of candidates using the currently available information. If $EQ = 0$ and we have a positive sample $X_+ \neq \emptyset$ then POOL returns $Subt(X_+)$ (line 16), otherwise the pool is initialized with the set of all trees up to depth 1 for $MQ = 1$ and with $\emptyset$ for $MQ = 0$. Note that any set returned by POOL is subtree-closed and hence for POOL $\neq \emptyset$ the set S contains at least one element s with $s = a \in \Sigma_0$ at any time which implies that the automaton $\mathcal{A}_T$ has at least one reachable state.
Line 15: If we are given a membership oracle and only a negative sample X_- then we depend on X_- being separative for L because in that case we can extract information about the maximal number of states in $\mathcal{A}_L$ from X_- and build a positive representative sample for L via MQs: The absolute minimal number of contexts needed to distinguish I_L equivalence classes is $\lceil log_2 I_L \rceil$. We can thus compute an upper bound $n := 2^{|Cont(X_-)|}$ for I_L and use the set of all members of L up to depth $n-1$ as a potentially representative positive sample X_+ for L since $I_L - 1$ is the maximal depth of any depth-minimal tree leading into some state of $\mathcal{A}_L$.[8] Observe that this construction may generate exponentially many candidates with respect to n.

[8]Consider: $\mathcal{A}_L$ can be built incrementally, starting out with the set of states that are accessible by a single symbol from Σ_0 and then successively adjoining new states by adding

We will now present and discuss the procedures used in the main loop.

```
procedure CLOSURE
18    while T is not closed do
19        find s ∈ BLUE such that ∀s' ∈ RED : s <> s';
20        RED := RED ∪ {s};
21        BLUE := BLUE \ {s};
22        UPDATE.
```

The procedure CLOSURE is straightforward, it successively finds all BLUE elements preventing the closedness of T, moves them to RED, and calls UPDATE to fill up BLUE and to fill in the table. Note that as CLOSURE is the only procedure moving elements to RED and as it only moves them if they are obviously different from every element in RED, the RED elements are all pairwise obviously different, and no equivalence class of L can have more than a single(!) official representative in RED. As a further consequence, if T is closed and complete and we have BLUE $\subseteq \Sigma$(RED) then T is also strongly consistent and $\mathcal{A}_T$ is a DFTA.

```
procedure NEXTDIST
23    ⟨s, e⟩ := FINDNEXT;
24    if s = □ then τ := 1;
25    else MAKEOD(⟨s, e⟩); UPDATE.
```

The task of the procedure NEXTDIST is to ensure that as long as the learning process is still in progress there is at least one BLUE element which will be moved to RED by CLOSURE within the same execution of the main loop, i.e., that the number of states in $\mathcal{A}_T$ is increased by at least 1. NEXTDIST relies on T being closed and calls the procedure FINDNEXT to retrieve the next piece of information that should be used to refine the learner's hypothesis. FINDNEXT (see below) returns a pair of a tree $s \in \{\Box\} \cup$BLUE and a context e. If s is not the undefined tree $\Box$ then we know that s should be fixed as a distinct state. The pair $\langle s, e\rangle$ is passed on to the procedure MAKEOD which modifies the table (using the context e if $e \neq \Box$ and other resources such as the negative sample X_- otherwise) such that CLOSURE will move s to RED. If FINDNEXT has returned a pair with $s = \Box$ (line 24) then this is taken as a signal that no more relevant candidates can be found and that the learning process is not to be continued. The termination criterion τ is set to 1 such that the algorithm will exit the loop after the (effectless) call of CLOSURE.

transitions that can be constructed from the existing states and the symbols in Σ. Each newly created state can only increase the maximal depth of any minimal access tree to some state by 1, and thus that depth must be bounded by the number of states in $\mathcal{A}_L$.

```
procedure FINDNEXT
26    if MQ = 1 ∧ (EQ = 1 ∨ X+ ≠ ∅) then
27        if COMPATIBLE(A_T, O, C)
28            if X+ ≠ ∅ ∧ ∃s0 ∈ S ∪ WHITE : ∃e0 ∈ E ∪ Cont(X+) :
                                ¬(A_T(e0⟦s0⟧) = 1 ⇔ O(e0⟦s0⟧) = 1)
                                                then C := C ∪ {e0⟦s0⟧};
29            else if EQ = 1 ∧ EQ(A_T) ≠ □ then C := C ∪ {EQ(A_T)};
30        return MINIMIZE(PREVENT(A_T, O, C));
31    else if MQ = 0 then MERGENEXT;
32            return ⟨min_⪯({s ∈ BLUE | q_s ∩ RED = ∅}), □⟩;
33    return ⟨□, □⟩.
```

```
procedure COMPATIBLE(A, A', X) [X ⊆ T_Σ]
34    if ∀x ∈ X : A(x) = 1 ⇔ A'(x) = 1 then return true;
35    else return false.
```

```
procedure PREVENT(A, A', X) [X ⊆ T_Σ]
36    if ∃x ∈ X : ¬(A(x) = 1 ⇔ A'(x) = 1) then return x;
37    else return □.
```

Since FINDNEXT is the most intricate procedure of the algorithm its various parts need some careful explanation. In the following, we separately discuss the cases with $MQ = 1$ and with $MQ = 0$.

$MQ = 1$: If the learner has access to a membership oracle then it can make use of a counterexample. If currently C does not contain a counterexample (which is tested via the procedure COMPATIBLE in line 27) then we try to find another one (lines 28–29).

Remark We maintain C in order to extract the maximal amount of information from the data available and thus to keep down the average number of steps taken since a counterexample may yield several distinctions between states, and some of them may not even be possible to detect when the counterexample is first retrieved. Literary note: A similar idea was studied for the case of learning regular tree languages from MQs and EQs in [47]. ♢

Line 28 ($X_+ \neq \emptyset$): First of all, note that $\mathcal{A}_T$ might not predict all cells of the table correctly (see Subsection 3.2.2) and thus the learner can find a counterexample in T itself without asking further queries by searching for s_0 in S and for e_0 in E. If $\mathcal{A}_T$ is T-consistent then line 28 is an attempt to retrieve a counterexample in an extension $T_\circ = \langle S_\circ, E_\circ, obs \rangle$ of T augmented with all candidates available from the set WHITE and with all contexts that

can be constructed from the given sample X_+ such that

- $S_\circ = \text{RED}_\circ \cup \text{BLUE}_\circ$ with $\text{RED}_\circ = S \cup \text{WHITE}$ and $\text{BLUE}_\circ = \emptyset$,
- $E_\circ = E \cup \mathit{Cont}(X_+)$, and
- $\mathit{row}_\circ(s) := \{\langle e, \mathcal{O}(e[\![s]\!])\rangle \mid e \in E_\circ\}$ for $s \in \mathbb{T}_\Sigma$.

We have $\mathit{Subt}(X_+) \subseteq \text{RED}_\circ = S_\circ$, and we can state the following lemma:

Lemma 5 *Let X_+ be representative for L.*
As long as $\mathcal{L}(\mathcal{A}_T) \neq L$ we can derive a counterexample for $\mathcal{A}_T$ from $T_\circ$.

Proof.
Observe that T is closed, complete, and also strongly consistent as we have $\text{BLUE} = \Sigma(\text{RED})$. Since we assume $\mathcal{A}_T$ to be T-consistent $\mathcal{A}_T$ is a state-minimal DFTA for $\mathcal{L}(\mathcal{A}_T)$ by Theorem 1. As the learner has access to a perfect membership oracle we can assume $T_\circ$ to be complete as well, and $T_\circ$ is trivially closed since $\text{BLUE}_\circ = \emptyset$. Then either $T_\circ$ is strongly consistent or it is not.

If $T_\circ$ is strongly consistent: As X_+ is representative for L and as we have $\mathit{Subt}(X_+) \subseteq S_\circ$ the table $T_\circ$ encodes the minimal DFTA $\mathcal{A}_L$ by Corollary 1 in Subsection 3.2.2 above. Then $\mathcal{A}_T$ recognizes a subset of $\mathcal{L}(\mathcal{A}_{T_\circ}) = L$ which can be seen as follows: Define a partial mapping ϕ by $\phi(\mathit{row}_\circ(s')) = \mathit{row}(s')$ if $\mathit{row}(s') \in \mathit{row}(\text{RED})$. This mapping clearly preserves the property of being an accepting state since we have $\Box \in E \subseteq E_\circ$, and since $S \subseteq S_\circ$ the chains of transitions by which those elements of $\mathit{row}(S_\circ)$ for which ϕ is defined can be reached are either preserved as well or they are disrupted at some point in $\mathcal{A}_T$.[9] Clearly $\mathcal{A}_T$ cannot have the same number of states as $\mathcal{A}_{T_\circ}$ since all elements of S that are obviously different in T must also be obviously different in $T_\circ$, and $\mathcal{A}_T$ is total as we have $\text{BLUE} = \Sigma(\text{RED})$ – thus, if $\mathcal{A}_T$ had the same number of states as $\mathcal{A}_{T_\circ}$ then it would recognize the same language L. Therefore, there is an element $s_0 \in S_\circ$ with $\mathit{row}_\circ(s_0) \neq \mathit{row}_\circ(s)$ for all $s \in \text{RED}$. Since $\mathcal{A}_T$ is total we have $\delta^*_T(s_0) \neq *$ which implies that there must be a context $e_0 \in E_\circ$ distinguishing s_0 from the single(!) tree $s' \in \text{RED}$ with $\delta^*_T(s_0) = \mathit{row}(s')$ and since $\mathcal{A}_T$ is also deterministic either $e_0[\![s_0]\!]$ or $e_0[\![s']\!]$ must be a counterexample for $\mathcal{A}_T$.

If $T_\circ$ is not consistent then there are trees $s_1, \ldots, s_n, s'_1, \ldots, s'_n, s, s' \in S_\circ$ with $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ for some symbol $f \in \Sigma_n$ such that $\mathit{row}_\circ(s_i) = \mathit{row}_\circ(s'_i)$ for $1 \leq i \leq n$ but $\mathit{row}_\circ(s) \neq \mathit{row}_\circ(s')$. Hence, there must be a context $e \in E_\circ$ distinguishing s from s'.

[9] Intuitively speaking, $\mathcal{A}_{T_\circ}$ is obtained from $\mathcal{A}_T$ merely by splitting states and/or adding transitions, and none of those actions can decrease the recognized language. Alternatively, note that S can only contain a representative set for a subset of L and apply Corollary 1.

As above, if there is $t \in \{s_1, \ldots, s_n, s'_1, \ldots, s'_n\}$ such that there is no $t' \in \text{RED}$ with $row(t') = row(t)$ then there must be a context $e_0 \in E_\circ$ distinguishing t from the single $s \in \text{RED}$ with $\delta^*_T(t) = row(s)$ and either $e_0[\![t]\!]$ or $e_0[\![s]\!]$ must be a counterexample for $\mathcal{A}_T$.
Thus, assume that there are trees $t_1, \ldots, t_n, t'_1, \ldots, t'_n \in \text{RED}$ with $row(t_i) = row(s_i)$ and $row(t'_i) = row(s'_i)$ for $1 \leq i \leq n$. As $row_\circ(s_i) = row_\circ(s'_i)$ implies $row(s_i) = row(s'_i)$ we also have $row(t_i) = row(t'_i)$ for $1 \leq i \leq n$. We have $\delta^*_T(s_i) = row(s_i)$ and $\delta^*_T(s'_i) = row(s'_i)$ due to the fact that we maintain $\text{BLUE} = \Sigma(\text{RED})$ and an argument that runs exactly parallel to the proof of Lemma 2, and therefore $\delta^*_T(s_i) = \delta^*_T(s'_i)$ for $1 \leq i \leq n$.
Since $\mathcal{A}_T$ is deterministic we have $\delta^*_T(s) = \delta^*_T(s')$ and $\delta^*_T(e[\![s]\!]) = \delta^*_T(e[\![s']\!])$, and consequently either $e[\![s]\!]$ or $e[\![s']\!]$ must be a counterexample for $\mathcal{A}_T$. ■

Remark If X_+ is not representative for L and $T_\circ$ is inconsistent then it contains a counterexample for $\mathcal{A}_T$ by the same arguments as in the proof above. However, if $T_\circ$ is consistent then the existence of a counterexample for $\mathcal{A}_T$ in $T_\circ$ is not predictable. ♢

Remark In a concrete implementation we would try to search for a counterexample in a favourable order, i.e., we would first extend S to $S \cup \text{WHITE}$, then E to $E \cup Cont(X_+)$ and then combine the two, and we would also make the learner look for the smallest elements fulfilling the required conditions in lines 36, 39, and 40 in order to save computing resources. ♢

If the previous efforts have not resulted in a counterexample but we have $EQ = 1$ then the learner can still resort to an equivalence query (in line 29). In case of a negative answer the learner obtains a new counterexample from the teacher – let $\text{EQ}(\mathcal{A}_T)$ be a blackbox routine that returns a real counterexample for $\mathcal{A}_T$ if $\mathcal{L}(\mathcal{A}_T) \neq L$ and the undefined tree $\square$ otherwise.
Finally, if all attempts to find a counterexample have failed then we consider the learning process – successfully or not – concluded. This information will be passed on up via the procedures PREVENT and MINIMIZE (see below).

Line 30: Now assume that there still exists an actual counterexample c for $\mathcal{A}_T$ in C (which can be retrieved using the procedure PREVENT). Intuitively speaking, at least one subtree of c represents a new state, and we must make sure that this information will be integrated into the learner's hypothesis. In particular, we must ensure that there is an equivalent candidate in BLUE which will be processed accordingly in the subsequent loop executions. The task of extracting that information is handled by the procedure MINIMIZE.

---

procedure MINIMIZE(c)

38 **if** $c = \square$ **then return** $\langle \square, \square \rangle$;

39 **if** $\exists s \in \texttt{BLUE} : \exists e \in \mathbb{C}_\Sigma : e[\![s]\!] = c \ \wedge \ \neg \exists s' \in \texttt{RED} : \neg(s <> s') \ \wedge$
$\mathcal{O}(c) = \mathcal{O}(e[\![s']\!]) \neq \mathcal{A}_T(e[\![s']\!])$ **then return** $\langle s, e \rangle$;

40 **else find** $s \in \texttt{BLUE}$, $e \in \mathbb{C}_\Sigma$, **and** $s' \in \texttt{RED}$ **such that**
$e[\![s]\!] = c \ \wedge \ \neg(s <> s') \ \wedge \ \mathcal{O}(c) = \mathcal{O}(e[\![s']\!]) \neq \mathcal{A}_T(e[\![s']\!])$;

41 $C := C \cup \{e[\![s']\!]\}$; **return** MINIMIZE($e[\![s']\!]$).

MINIMIZE recursively exchanges subtrees of the given counterexample which are also elements of BLUE by RED trees with the same row such that the property of being a counterexample for $\mathcal{A}_T$ (positive or negative, respectively) is preserved (line 40), each time adding the result to C. Intuitively, this corresponds to an attempt of $\mathcal{A}_T$ to parse the original counterexample c.[10]
The process is continued until there is a subtree $s \in$ BLUE for which there is no such tree in RED, which implies that s should be obviously different from all RED elements and thus represent a distinct state of the solution. Due to the condition in line 39 we also know that the matching context e fulfils the property of distinguishing s from all $s' \in$ RED with $\neg(s <> s')$, and thus an addition of e to E (effected via the procedure MAKEOD, called in line 25, described below) will cause s to be moved to RED by CLOSURE.

Remark Note that any given counterexample c' has at least one subtree in BLUE which can be seen by considering that at least all trees $s' \in \mathbb{T}_\Sigma$ with $s' = a$ for some $a \in \Sigma_0$ are either in RED or in Σ(RED), and for $MQ = 1$ we have BLUE $= \Sigma$(RED) at any time. Also note that c' cannot be in BLUE itself because otherwise $\mathcal{A}_T$ would parse c' correctly by Lemma 1. $\diamondsuit$

Line 37 covers the case where there is no real counterexample available such that MINIMIZE cannot extract an appropriate candidate from its input and returns the pair $\langle \square, \square \rangle$ as a "failure signal" instead.

For $MQ = 0$ we continue merging states in the oracle $\mathcal{O}$ that we have constructed from $STA(X_+)$ so far until we find information preventing a merge. For each $s \in Subt(X_+)$ we denote the unique(!) state $q \in Q_\mathcal{O}$ with $s \in q$ by q_s. The procedure MERGENEXT (called in line 31, given below) iteratively retrieves the next BLUE tree b with respect to the total order $\preceq$ such that b represents a state that has not yet been merged with some state containing a RED tree in its label (note that order matters – see Subsection 3.3.2). We check the mergeability of a pair of states by giving $\mathcal{O}$ with the two states

[10] The strategy of MINIMZE is inspired by a similar one for MQs and EQs in [46] which in turn is based on the technique of *contradiction backtracing* developed by Shapiro [106].

merged, the total all-rejecting automaton $\mathcal{A}^{\bullet}_{\emptyset}$, and the negative sample X_- as input components to the procedure COMPATIBLE, a strategy which provides us with a test if after the merge all elements of X_- are still correctly rejected by $\mathcal{O}$. If such a tree b is found and there is a state with a RED tree s' in its label such that q_b and $q_{s'}$ can be merged then the merge is done by the procedure RECMERGE which calls MERGE and then recursively "repairs" any non-determinism introduced by the merge.[11] Then BLUE is filled up with the available one-symbol extensions of RED $\cup \{b\}$ from WHITE and WHITE is updated as well. Note that b stays in BLUE but will not be considered again since q_b now contains an element from RED.
After the call of MERGENEXT in FINDNEXT either all BLUE trees correspond to states that result from a merge with a RED state and we have $min_{\preceq}(\{s \in \text{BLUE} \mid q_s \cap \text{RED} = \emptyset\}) = \Box$ or the next tree b' with $q_{b'} \cap \text{RED} = \emptyset$ represents a non-mergeable state. This tree should be a distinct state of the solution and is returned along with the context $\Box$ but note that for $MQ = 0$ this second component will not be put to use (see procedure MAKEOD).

---

procedure MERGENEXT

42 **while** $\exists b \in \text{BLUE} : b = min_{\preceq}(\{s \in \text{BLUE} \mid q_s \cap \text{RED} = \emptyset\}) \ \wedge$
$\exists s' \in \text{RED} :$ COMPATIBLE(RECMERGE$(q_{s'}, q_b, \mathcal{O}), \mathcal{A}^{\bullet}_{\emptyset}, X_-)$ **do**

43 $\mathcal{O}$:= RECMERGE$(q_{s'}, q_b, \mathcal{O})$;

44 BLUE := BLUE $\cup (\Sigma(\text{RED} \cup \{b\}) \cap \text{WHITE})$; UPDATE.

---

procedure RECMERGE$(p, q, \mathcal{A})$ $[\mathcal{A} = \langle \Sigma, Q, F, \delta \rangle,\ p, q \in Q]$

45 $\mathcal{A}$:= MERGE$(p, q, \mathcal{A})$;

46 **for** $n \geq 0$ with $\Sigma_n \neq \emptyset$ **do**

47 **for** $f \in \Sigma_n$ **do**

48 $D := \{q_0 \in Q \mid \exists \langle f, q_1 \cdots q_n \rangle \mapsto q_0 \in \delta \ \wedge$
$\exists i \in \{1, \ldots, n\} : q_i = (p \cup q)\}$;

49 **if** $|D| > 1$ **then** find $p' \neq q' \in D$; $\mathcal{A}$:= RECMERGE$(p', q', \mathcal{A})$;

50 **return** $\mathcal{A}$.

---

procedure MERGE$(p, q, \mathcal{A})$ $[\mathcal{A} = \langle \Sigma, Q, F, \delta \rangle,\ p, q \in Q]$

51 $\wp := p \cup q$;

52 $Q := (Q \setminus \{p, q\}) \ \cup \ \{\wp\}$;

53 **if** $p \in F \ \vee \ q \in F$ **then** $F := (F \setminus \{p, q\}) \ \cup \ \{\wp\}$;

54 $\delta := \{\langle f, q'_1 \cdots q'_n \rangle \mapsto q'_0 \mid \exists \langle f, q_1 \cdots q_n \rangle \mapsto q_0 \in \delta \ \wedge \ \forall i \in \{0, \ldots, n\} :$
$(q_i \in \{p, q\} \ \Rightarrow \ q'_i = \wp) \ \wedge \ (q_i \notin \{p, q\} \Rightarrow q'_i = q_i)\}$;

55 **return** $\mathcal{A}$.

---

[11] The procedures RECMERGE and MERGE are inspired by an early version of [41].

In all cases that were not covered by the distinctions in lines 26–32 we have to state that we cannot reliably find another candidate to promote and return the failure signal $\langle \square, \square \rangle$ (line 33). This ends the description of FINDNEXT.

---

procedure MAKEOD($\langle s, e \rangle$)
56 **for** $s' \in$ RED **do**
57 **if** $\neg(s <> s')$ **then**
58 **if** $e \neq \square$ **then** $E := E \cup \{e\}$;
59 **else** $c :=$ PREVENT(RECMERGE($q_{s'}, q_s, \mathcal{O}$), $\mathcal{A}^{\bullet}_{\emptyset}, X_{-}$);
60 $E' := \{e' \in Cont(c) \mid \exists x \in Subt(c) \cap (\mathcal{L}_{q_{s'}} \cup \mathcal{L}_{q_s}) : e'[\![x]\!] = c\}$;
61 $E := E \cup E'$; $obs(s', \square) := 0$; $obs(s, \square) := 1$.

The procedure MAKEOD is called if FINDNEXT has returned a pair $\langle s, e \rangle$ with $s \in$ BLUE because then we know that CLOSURE should move s to RED to represent a distinct state in the final automaton. For $e \neq \square$ the pair $\langle s, e \rangle$ must be an output of MINIMIZE. In that case, as the elements in RED are pairwise obviously different and all rows of S are complete there is only one RED tree s' that is *not* obviously different from s, and due to the condition in line 39 the given e is a context distinguishing s and s'. As a consequence, we only have to add a single experiment to E and let the next call of UPDATE (in line 25) fill in the cells of the newly created column. Note that there are various ways to exploit a counterexample (conceivable or actually used in the literature), several of which are discussed and shown to be equivalent to our method here in Appendix A.1.
If $e = \square$ then we have $MQ = 0$. In that case, since we have not set any value explicitly all cells in $row(s)$ are filled by $*$ and we have to make s obviously different from every RED element s' "by hand": We retrieve a counterexample $c \in X_{-}$ forbidding the merge of $q_{s'}$ and q_s via the procedure PREVENT. Such a counterexample must exist as in all other cases with $MQ = 0$ FINDNEXT would have returned the pair $\langle \square, \square \rangle$. Moreover, c must include a context that distinguishes s from s' but since we do not know which of the conceivable decompositions of c will yield that context we retrieve all subtrees of c that when parsed by $\mathcal{O}$ end up in $q_{s'}$ or q_s and add the resulting contexts to E, thus ensuring that the table now contains a context separating the rows of s and s'. Then we arbitrarily pick the context $\square$ and fill the two associated cells for s and s' with differing values from $\{0, 1\}$ in order to make CLOSURE move s to RED in its next call. Note that these values do not have to be correct with respect to L as after the next call of CLOSURE they will never be of consequence again – however, we must make sure that throughout the call of MAKEOD all RED elements receive the same value (our choice: 0) to save distinctions that were established by previous executions of the for loop from

being obliterated. Thus, although strictly speaking *obs* may temporarily fail to assign correct values this is not fatal as the cells of T will be overwritten completely and in case of a successful identification of L correctly by $\mathcal{O}$ in the very last call of UPDATE (effected by CLOSURE in line 3) before the algorithm terminates. Since E contains a context separating the rows of s and s', if we have $\mathcal{L}(\mathcal{O}) = L$ then this distinction will be preserved.

---

procedure UPDATE
62 **if** $MQ = 1$ **then** BLUE := BLUE $\cup\, \Sigma($RED$)$;
63 **else** BLUE := BLUE $\cup\, (\Sigma($RED$) \cap$ WHITE$)$;
64 WHITE := WHITE $\setminus$ BLUE;
65 **if** $MQ = 1 \ \vee \ \tau = 1$ **then**
66 **for** $s \in S$ **do**
67 **for** $e \in E$ **do**
68 **if** $\mathcal{O}(e[\![s]\!]) = 1$ **then** $obs(s, e) := 1$; **else** $obs(s, e) := 0$.

Since the sets RED and BLUE may have been modified, the procedure UPDATE fills up BLUE with all (for $MQ = 0$: available) one-symbol extensions of RED and deletes all elements which are now contained in BLUE from the pool. Then it goes on to fill in the cells of the resulting table by consulting the membership oracle $\mathcal{O}$ on condition that $\mathcal{O}$ can be assumed to be perfect. For $MQ = 1$ this is true at any time and for $MQ = 0$ it is true when the learning process has been terminated – provided that it was successful.

3.3.2 Behaviour of GENDET

We will now show that our meta-algorithm returns a correct solution for constellations where the identification of L from the given information sources is guaranteed in a polynomial number of steps with respect to certain measures of the input. We will also briefly discuss the behaviour of GENDET for cases in which (polynomial) inference is not ensured.

In the following explanations, we will call an information source *non-empty*

- for MQs if $MQ = 1$ and for EQs if $EQ = 1$, and
- for a sample if the given set is non-empty.

We will call an information source *non-void*

- for queries if $MQ = 1$ or $EQ = 1$, respectively,
- for a positive sample if it is representative, and
  for a negative sample if it is separative.

Intuitively, an information source is non-empty if it provides the learner with any answers or material at all while it is non-void if it provides the maximal amount of correct and useful information possible for that kind of source. Note that trivially the emptiness of a given source can be checked immediately whereas in the case of samples voidness is a property the learner cannot ascertain (for example, an empty sample X_+ is non-void for $L = \emptyset$, an empty sample X_- is non-void for $L = \emptyset$ or $L = \mathbb{T}_\Sigma$, and any non-empty sample may still fail to fulfil the condition of being representative or separative for L).

Remark We could conceive void query sources by admitting teachers who give incorrect or inconclusive answers but since we disallow that possibility (which is in accordance with the fact that we neither allow the positive X_+ to contain trees from $\mathbb{T}_\Sigma \setminus L$ nor the negative X_- to contain elements of L) in our case non-empty and non-void query-based sources coincide. ♢

Theorem 3 *GENDET terminates for any admissible input and returns an FTA. For an input including at least two non-void information sources except for $\langle 1, 0, X_+, X_- \rangle$ with X_+ or X_- void the output is a state-minimal DFTA recognizing L; for $\langle 0, 0, X_+, X_- \rangle$ this only holds if the pair $\langle X_+, X_- \rangle$ fulfils additional conditions (A1) and (A2) as stated below.*

We show this by discussing each constellation individually (in more or less detail). Cases with only brief explanations can be easily verified by executing the algorithm step by step while observing the relevant case distinctions.
In cases including just one non-empty information source, (polynomial) identification is not ensured (not even for strings, see [6, 8] – recall that generally negative learning results for strings imply negative results for trees).

- $\langle 0, 0, \emptyset, \emptyset \rangle$, $\langle 1, 0, \emptyset, \emptyset \rangle$, $\langle 0, 0, \emptyset, X_- \rangle$: GENDET returns $\langle \Sigma, \emptyset, \emptyset, \emptyset \rangle$ after one loop execution – POOL returns $\emptyset$, the initial table is empty and hence trivially closed, and since BLUE $= \emptyset$ the test in line 42 fails which causes FINDNEXT to return the termination signal $\langle \Box, \Box \rangle$.

- $\langle 0, 1, \emptyset, \emptyset \rangle$: POOL returns all trees of depth at most 1, and GENDET terminates as soon as the table is closed because FINDNEXT returns $\langle \Box, \Box \rangle$ due to its case distinctions. The final table has only one column and hence the (total) output automaton can have at most two states.

- $\langle 0, 0, X_+, \emptyset \rangle$: GENDET returns the minimal DFTA for the language $\mathbb{T}_{\Sigma'}$ with $\Sigma' = \{f \in \Sigma \mid \exists s_1, \ldots, s_n \in \mathbb{T}_{\Sigma'} : f(s_1, \ldots, s_n) \in X_+\}$ (without a failure state). The initial table is trivially closed since it does not contain any values from $\{0, 1\}$, the oracle $\mathcal{O}$ is initialized with $STA(X_+)$, and since there is no negative evidence MERGENEXT

merges all states of $\mathcal{O}$ in one go which causes FINDNEXT to return the termination signal $\langle \Box, \Box \rangle$ due to line 32.

The architecture of our meta-algorithm is essentially determined by and adapted to the following three constellations. We prove that if the employed information sources are non-void then the automaton $\mathcal{A}_T$ derived from the final table T is a state-minimal DFTA for L.

- $\langle 1, 1, \emptyset, \emptyset \rangle$: After the termination of GENDET T is closed due to the last call of CLOSURE in line 3, complete due to $MQ = 1$, and strongly consistent due to the fact that all RED elements are pairwise obviously different and that BLUE $\subseteq \Sigma(\text{RED})$. By the last EQ we know that $\mathcal{A}_T$ fulfils $\mathcal{L}(\mathcal{A}_T) = L$, which implies that $\mathcal{A}_T$ is T-consistent and hence $\mathcal{A}_T$ must be isomorphic to $\mathcal{A}_L^\bullet$ by Theorem 1 in Subsection 3.2.2 and the fact that BLUE $= \Sigma(\text{RED})$. ■

  Remark Due to the way the candidates are constructed each $s \in$ RED is a minimal representative for $[s]_L$ with respect to depth. ♢

- $\langle 0, 1, X_+, \emptyset \rangle$: After the termination of GENDET T is closed, complete, and strongly consistent for the same reasons as in the previous case. We show that S contains a representative set for L: In line 28 we refer to an extended table $T_\circ$ with $Subt(X_+) \subseteq S_\circ$. When GENDET terminates $T_\circ$ cannot contain a counterexample for $\mathcal{A}_T$ anymore since this is the termination criterion. Moreover, $T_\circ$ must be strongly consistent because otherwise $T_\circ$ would still contain a counterexample by the same argument as in the proof of Lemma 5. Hence, at that point the equality relation between rows of $T_\circ$ and the equivalence relation $\equiv_L$ between the elements labeling those rows must coincide.
  All equivalence classes that are represented in $S_\circ$ are also represented in RED. This can be seen as follows: If there were an element $s \in S_\circ$ for which there is no element $s_0 \in$ RED with $s_0 \equiv_L s$ then this would contradict the fact that $T_\circ$ cannot contain a counterexample for $\mathcal{A}_T$ because there would be $s' \in$ RED with $row_\circ(s) \neq row_\circ(s')$ but $\delta_T^*(s) = \delta_T^*(s')$ and hence there would also be a context $e' \in E_\circ$ distinguishing s' from s and $e'[\![s]\!]$ or $e'[\![s']\!]$ would be a counterexample for $\mathcal{A}_T$. Obviously, as we have $S \subseteq S_\circ$ the table T cannot contain *more* rows than $T_\circ$, and as we have $E \subseteq E_\circ$ the tables T and $T_\circ$ must make the same distinctions. Consequently, since X_+ is representative for L and BLUE $= \Sigma(\text{RED})$ due to line 62 in UPDATE the set S contains a representative set for L and the DFTA $\mathcal{A}_T$ is isomorphic to $\mathcal{A}_L^\bullet$ by Corollary 1. ■

Remark Like in the previous setting each tree $s \in \text{RED}$ is a minimal representative for $[s]_L$ with respect to depth. $\diamondsuit$

Remark Note that (as can be deduced from Footnote 9) if X_+ is void then the output is a DFTA that recognizes a subset of L. $\diamondsuit$

- $\langle 0, 0, X_+, X_- \rangle$: In this setting, in order to ensure that the learner identifies L correctly X_+ and X_- must meet the conditions A1 and A2 below. Define $mSubt(X_+) := \{s \in Subt(X_+) \mid \forall s' \in Subt(X_+) \cap [s]_L : s \preceq s'\}$. If X_+ is representative for L then $mSubt(X_+)$ contains exactly one element $s \in \chi$ for each equivalence class χ of L with $\chi \subseteq Subt(L)$ and s is minimal with respect to $\preceq$.

  (A1) $\Sigma(mSubt(X_+)) \cap Subt(L) \subseteq Subt(X_+)$ and
  $mSubt(X_+) \cap L \subseteq X_+$,

  (A2) $\forall s_1 \in mSubt(X_+) : \forall s_2 \in mSubt(X_+) \cup \Sigma(mSubt(X_+)) :$
  $\neg(s_1 \equiv_L s_2) \Rightarrow \exists e \in \mathbb{C}_\Sigma :$
  $e[\![s_1]\!] \in X_+ \wedge e[\![s_2]\!] \in X_- \vee e[\![s_1]\!] \in X_- \wedge e[\![s_2]\!] \in X_+.$

  The necessity of those conditions can be motivated as follows: The elements of $Subt(X_+)$ are processed in the order determined by $\preceq$. As we want to prevent states of $\mathcal{O}$ (which is initialized with the subtree automaton $STA(X_+)$) representing subsets of distinct equivalence classes of L from being merged we need the relevant information with respect to their in- and outgoing transitions as soon as the first tree from a certain equivalence class is about to be processed. Since we do not have access to a perfect membership oracle in this setting the only information we can rely on is the membership of a tree in the given positive sample (of which we know that it contains only members of L) or in the given negative sample (of which we know that it contains only non-members of L). More precisely, if for some trees $s_1, s_2 \in \mathbb{T}_\Sigma$ there is a context $e \in \mathbb{C}_\Sigma$ such that both $e[\![s_1]\!]$ and $e[\![s_2]\!]$ actually occur in the given data but not in the same sample then this context e simulates a kind of "skeletal" column conveying the only information that the corresponding cells for s_1 and s_2 must differ, and we exploit this fact in the procedure MAKEOD.
  Let $s \in Subt(X_+) \setminus \text{RED}$. We show that when s is processed, for any $t \in \text{RED}$ with $\neg(s \equiv_L t)$ the state q_s will not be merged with q_t. We presuppose $\text{RED} \subseteq mSubt(X_+)$ – this is clearly true for the initial table and it is a corollary of the following that this property stays preserved throughout the learning process. We distinguish two cases:

– Let $s \in mSubt(X_+) \cup \Sigma(mSubt(X_+))$. As $t \in mSubt(X_+)$ there is a context $e \in \mathbb{C}_\Sigma$ with either $e[\![t]\!] \in X_+$ and $e[\![s]\!] \in X_-$ or $e[\![t]\!] \in X_-$ and $e[\![s]\!] \in X_+$ by Condition A2. Assume $e[\![t]\!] \in X_+$. This implies $e[\![t]\!] \in L$ and we have $\mathcal{O}(e[\![t]\!]) = 1$ due to the way $\mathcal{O}$ is constructed. A merge of q_t and q_s would result in $\delta^*_\mathcal{O}(e[\![s]\!]) = \delta^*_\mathcal{O}(e[\![t]\!]) \in F_\mathcal{O}$ but the merge is prevented by the fact that $e[\![s]\!] \in X_-$. The argument runs analogously for the symmetric case with $e[\![s]\!] \in X_+$.
For $s \in mSubt(X_+)$ there can be no $t' \in$ RED with $t' \equiv_L s$ and thus s is promoted to RED via MAKEOD and CLOSURE. Otherwise there is a single $s' \in$ RED with $s' \equiv_L s$ and q_s is merged with $q_{s'}$ because there can be no information preventing the merge in X_-.

– If $s = f(s_1, \ldots, s_n) \notin mSubt(X_+) \cup \Sigma(mSubt(X_+))$ then all trees in $mSubt(X_+) \cup \Sigma(mSubt(X_+))$ must already have been processed. This implies that there are $s'_1, \ldots, s'_n, s' \in$ RED with $s_i \equiv_L s'_i$ for $1 \leq i \leq n$ and $s' \equiv_L s$, and that we have $\delta^*_\mathcal{O}(f(s'_1, \ldots, s'_n)) = q_{s'}$. As a consequence, a merge of q_t and q_s yields $\delta^*_\mathcal{O}(t) = \delta^*_\mathcal{O}(s) = \delta^*_\mathcal{O}(s')$ but since for t and s' there is a separating context $e \in \mathbb{C}_\Sigma$ with either $e[\![t]\!] \in X_-$ or $e[\![s']\!] \in X_-$ either $e[\![t]\!]$ or $e[\![s']\!]$ is now wrongly accepted by $\mathcal{O}$ and hence the procedure COMPATIBLE will object. On the other hand, q_s is sure to be merged with $q_{s'}$ because there can be no information preventing the merge in X_-.

Thus, when all candidates have been processed, each state in $Q_\mathcal{O}$ recognizes a subset of an equivalence class of L distinct from all others, and all states accepting a subset of L are contained in $F_\mathcal{O}$. Consequently, as X_+ is representative for L when the termination criterion τ is met the oracle $\mathcal{O}$ is isomorphic to $\mathcal{A}^\circ_L$ and the table is filled in completely and correctly by UPDATE (line 65). At that point we have $S = Subt(X_+)$, the table is closed due to the last call of CLOSURE, and it is strongly consistent since E contains a separating context for each pair $s_1, s_2 \in S$ with $\neg(s_1 \equiv_L s_2)$ by line 60. The derived automaton $\mathcal{A}_T$ is T-consistent by Theorem 2 and isomorphic to $\mathcal{A}^\circ_L$ by Corollary 1. ■

Remark We have based our conditions (A1) and (A2) on [93] where the authors define

$sSubt(L) := \{s \in Subt(L) \mid \forall s' \in \mathbb{T}_\Sigma : s \equiv_L s' \Rightarrow s \preceq s'\}$, and
$kSubt(L) := \{f(s_1, \ldots, s_n) \in Subt(L) \mid s_1, \ldots, s_n \in sSubt(L)\}$,

and require the given samples to fulfil

(B1) $\forall s \in kSubt(L) : \exists e \in \mathbb{C}_\Sigma : e[\![s]\!] \in X_+ \wedge (s \in L \Rightarrow e = \Box)$,
(B2) $\forall s_1 \in kSubt(L) : \forall s_2 \in sSubt(L) : \neg(s_1 \equiv_L s_2) \Rightarrow \exists e \in \mathbb{C}_\Sigma :$
$e[\![s_1]\!] \in X_+ \wedge e[\![s_2]\!] \in X_- \vee e[\![s_1]\!] \in X_- \wedge e[\![s_2]\!] \in X_+$.

If X_+ is representative for L then $sSubt(L) = mSubt(X_+)$ but we prefer a definition that refers to the sample and to the learner's strategy, i.e., that anticipates the situation when the first member of a certain equivalence class is processed. Also note that in contrast to (A1) the second part of (B1) ranges over the entire set $\Sigma(mSubt(X_+))$ which is not fully minimal as a condition. ♢

Remark If X_+ is representative and X_- is void then the output is a DFTA for a superset of L because illegitimate merges create illegal possibilities to reach accepting states. ♢

- $\langle 0, 1, \emptyset, X_- \rangle$: See $\langle 0, 1, X_+, \emptyset \rangle$. GENDET uses the given X_- to build a positive sample X_+ (in line 15) which however may not be polynomial in size with respect to the total sum of nodes in X_-.

Inputs of the form $\langle 1, 0, X_+, \emptyset \rangle$ or $\langle 1, 0, \emptyset, X_- \rangle$ are eclipsed from the meta-algorithm GENDET by the relevant case distinctions – the procedure POOL returns the empty set which eventually causes FINDNEXT to return the termination signal in the first loop execution. This alternative was chosen due to the fact that in those cases inference performing a polynomial number of steps or queries is not ensured. Also see the discussion in Appendix A.2.

For input constellations containing more than two non-empty information sources, the design of GENDET is such that the learner chooses one of the options above and ignores the remaining sources. In particular, any input $\langle 1, 1, X_+, X_- \rangle$ with $X_+ \neq \emptyset$ triggers the solution for MQs and EQs, an input $\langle 0, 1, X_+, X_- \rangle$ with $X_+ \neq \emptyset$ triggers the solution for MQs and a positive sample, and an input $\langle 1, 0, X_+, X_- \rangle$ is treated like $\langle 0, 0, X_+, X_- \rangle$.

Termination: Trivially the first call of CLOSURE terminates because with one column the table cannot contain more than two distinct rows. The previous explanations show that in each execution of the main loop except the last one NEXTDIST adds contexts that distinguish at least one BLUE element from all RED ones which causes CLOSURE to move that element to RED. NEXTDIST sets the termination criterion to 1 as soon as FINDNEXT cannot find another real distinction. In all cases with $MQ = 1$, since the membership oracle is perfect it is not possible to have more than I_L distinct rows in the table. We start out with one RED row and hence GENDET terminates after at most $1+I_L-1 = I_L$ executions of the main loop. In all other cases the candidates in the pool will all be either merged or promoted to RED successively, and the learner terminates when there are no more unprocessed candidates left in BLUE due to line 32 in FINDNEXT.

This concludes the **Proof of Theorem 3**.

Discussing the design of GENDET

There is more than one point in the design of GENDET where we might have chosen a different architectural solution. Let us briefly motivate our decisions.

For those settings not covered by some well-known existing algorithm, we have aimed to make the reactions of our meta-learner to the given input justifiable from an intuitive point of view. For example, if only positive data or no positive data is given then we retreat to maximal overgeneralization and undergeneralization, respectively. With only MQs the learner attempts to ascertain at least if the target language is non-empty or unequal to $\mathbb{T}_\Sigma$. There may be other options to treat more than two given sources, for example for $\langle 1, 1, X_+, X_- \rangle$ with $X_+ \neq \emptyset$ we could use X_+ to establish the pool and only start generating more candidates if after processing the elements of P an EQ is still answered in the negative, which in some cases might improve complexity due to the fact that we avoid the rather expensive procedure of filling up BLUE with all possible one-symbol extensions, and might thus yield a non-total output. However, we have chosen not to rely on the given samples being non-void – the choice for $\langle 1, 1, X_+, X_- \rangle$ with $X_+ \neq \emptyset$ can be motivated by the fact that we do not know if X_+ is indeed representative for L, and the choice for $\langle 0, 1, X_+, X_- \rangle$ with $X_+ \neq \emptyset$ by the fact that we do not know if X_- is separative for L – whereas if we have access to an oracle then we assume it to be perfect. Also note that for $\langle 0, 1, X_+, X_- \rangle$ we might miss a chance to succeed if X_+ is void but X_- is not – however, we prefer positive to negative data since negative information seems rather rare and unnatural. Another design principle of this algorithm is the fact that it is deterministic and does not backtrack. Of course a learner with access to an equivalence or a membership oracle only could adopt a non-deterministic strategy and suggest possible DFTA or trees in any order but as polynomial success is not guaranteed this is not an acceptable approach for the meta-algorithm developed here and is therefore inhibited by corresponding case distinctions.

Although various parts of our meta-algorithm were substantially inspired by the algorithms given in [5, 46] (learning regular string/tree languages from MQs and EQs), [14] (MQs and a positive sample), and [41, 93] (a positive and a negative sample) we would like to motivate that GENDET is more than a comprehensive collection of case distinctions by observing that we have modified, reordered, and generalized their components into a range of subprocedures as required by our purpose, viz., to cast the respective actions of those algorithms into a uniform framework. Some concrete examples:

- The consistency check, which is the essential procedure to make the algorithm in [14] (learning from MQs and a positive sample) terminate, has been eliminated and generalized into the search for a counterex-

ample in an extended table, which implies that in all non-void settings with $MQ = 1$, as soon as we can derive a DFTA from the table the next step is an equivalence test based on the available information sources.

Remark Note that this requires maintaining the set of *all* conceivable one-symbol extensions of RED in BLUE in order to ensure the validity of Lemma 5 – otherwise it is possible that the extended table $T_\circ$ contains only inconsistencies from which we cannot construct a counterexample for $\mathcal{A}_T$ due to missing transitions. $\diamondsuit$

- We apply the strategy of MINIMIZE, which was developed in a similar form in [46] for MQs and EQs to atone for the size of unnecessarily large counterexamples received from the teacher, to the setting of learning from MQs and a positive sample as well, with the consequence that the representatives in the set RED are minimal with respect to depth even if the given positive sample does not fulfil that condition.

- The procedure COMPATIBLE was inspired by the description of RPNI in [41] but we have generalized its specification in order to check the set C of counterexamples obtained so far against the current hypothesis in FINDNEXT for $MQ = 1$. As a consequence, we had to use COMPATIBLE in a novel way (namely, by feeding it the all-rejecting automaton as a perfect oracle for the negative sample X_-) when testing the mergeability of states in settings where we have $MQ = 0$.

Observe that as our meta-learner incorporates algorithms for several different settings with their various idiosyncrasies, unlike those and other algorithms that were developed for one setting only we cannot exploit the whole range of conceivable strategies to reduce theoretical or empirical complexity. The design of GENDET may cause some objectionable artefacts which can nevertheless be justified given our goal of a unified perspective:

- When we learn from MQs and a positive sample we process the elements of $Subt(X_+)$ incrementally but keep referring to a table containing all of them, and we build a counterexample using a candidate that already represents a distinct state of the solution but in general derive another candidate from it to be moved to RED. However, we have chosen to keep the clean incremental approach as taken in [46] which entails that only one-symbol extensions of RED are considered for promotion to RED.[12]

[12] A (genuinely) incremental algorithm learning string languages from MQs and a stream of labeled data is given in [97]. Whether their approach is adaptable to the tree case while retaining polynomial complexity remains to be studied. Probably a good start for reading would be [21] where the authors describe an efficient algorithm that allows the addition or removal of (unranked) trees to a DFTA, which is clearly an essential ingredient.

- When we learn from a positive and a negative sample we document our progress in a table which we do not use to construct a hypothesis until the oracle is perfect. However, staying in line with the other algorithms under consideration the observation table is our chosen standard format which the learner should use to return its solution.

Thus, based on the algorithms as developed in [46, 14, 93] we obtain a meta-learner for regular tree languages that uses an observation table and starts out with a single tentative equivalence class under the Myhill-Nerode relation which is then split up according to the available information. This approach has also been called *specializing*.

3.3.3 Complexity of GENDET

Let us briefly analyze the complexity of GENDET with respect to the number of steps taken and/or queries asked for those settings in which (polynomial) identification of the target language by GENDET is ensured. We assume that the results of MQs are stored in order to avoid querying any tree twice, which makes the most sense with regard to the MQ-consuming calls of the subprocedures in FINDNEXT and of UPDATE. Generally, we assume that storage space is not an issue and rather aim to minimize the number of queries and other computation steps taken during the process. However, we will not focus on the complexity of actions such as parsing (which can be shown to be polynomial, see [93]) and comparing (which is linear), and we also disregard the complexity of the available oracles by treating them as blackboxes.

Let n_+ be the maximal tree size featured in X_+, i.e., $\forall t \in X_+ : |t| \leq n_+$, let $m_+ := \sum_{t \in X_+} |t|$ and $m_- := \sum_{t \in X_-} |t|$ be the size of all trees in X_+ and X_- added up, respectively, and let ρ be the maximal rank in Σ such that $\Sigma_\rho \neq \emptyset$. Let ζ be the maximal tree size featured in the set C of counterexamples.

- EQs and MQs:
  The number of EQs is bounded by I_L as in a worst case NEXTDIST (and thus FINDNEXT) may have to be called I_L times and every counterexample may reveal just one more distinct state.
  The number of MQs is bounded as follows:
  - Since in each call of NEXTDIST only a single context is added to E the table T itself contains a number of cells that is bounded by $I_L(I_L + |\Sigma| {I_L}^\rho) = {I_L}^2 + |\Sigma| {I_L}^{\rho+1}$ to be filled via MQs, and

  - MINIMIZE may have to check every subtree of a counterexample for substitutability by a RED element which requires an additional number of MQs bounded by $\zeta {I_L}^2$.

- MQs and a representative X_+:
  The cardinalities of $Subt(X_+)$ and $Cont(X_+)$ are bounded by m_+.

  - The extended table $T_\circ$ has a number of rows bounded by $m_+ + |\Sigma| {m_+}^\rho$ since we may have added elements to S in order to represent all possible transitions, and a number of columns bounded by $I_L + m_+$. Since I_L is bounded by m_+ as well $T_\circ$ contains a number of cells bounded by $2{m_+}^2 + 2|\Sigma| {m_+}^{\rho+1}$ to be filled.
  - The size of the biggest tree in S is at most $n_+\rho + 1$ and the size of the biggest context in $Cont(X_+)$ is at most n_+. In each call of FINDNEXT the size of the biggest context in E can increase by at most $n_+\rho + 1$ and thus the size of the biggest counterexample that is constructed is bounded by $I_L(n_+\rho + 1) + n_+$ or $m_+n_+\rho + m_+ + n_+$. Hence, the number of additional MQs caused by calls of MINIMIZE is bounded by ${m_+}^3(n_+\rho + 1) + {m_+}^2 n_+$.

- Representative X_+ and separative X_-: Since for the construction of the oracle we perform the same steps as RPNI [93] GENDET is at least as complex as RPNI. If no two trees in X_+ have a common subtree then the automaton $STA(X_+)$ has m_+ states and m_+ transitions. Each state of $STA(X_+)$ is tested for mergeability with an established RED state (at most ${m_+}^2$ pairings), and each time the result is checked against X_-. Testing if the DFTA we have developed from $STA(X_+)$ so far (wrongly) accepts a tree in X_- takes a number of steps bounded by $m_+m_-\rho$ and due to the tree structure of the STA the computation of the merges in question is of polynomial complexity as well (see [93]), and thus the construction of our oracle is of a complexity bounded by $O({m_+}^3 m_-\rho)$. In addition, UPDATE uses the oracle once to fill a table of a size which is bounded by m_+m_- since its rows are labeled with candidates from $Subt(X_+)$ and the columns with contexts from $Cont(X_-)$.

  Remark We could have considered collecting the elements preventing a merge as found by COMPATIBLE and passing them on to MAKEOD directly but this would have required additional case distinctions. $\diamondsuit$

- MQs and a separative sample: Unfortunately building a positive sample X_+ from X_- as done by the procedure POOL can cost a number of MQs which is not polynomial with respect to m_-. Generating all strings up

to a certain length k is already of exponential complexity with respect to k, and for trees and a given depth, which in our case is bounded by 2^{m-}, this cost is increased by some measure depending on ρ.

Remark An absolute lower bound for the size of T is $(I_L + x)\lceil I_L/2 \rceil$ with $x = 0$ for cases where we use a positive sample and $x = |\Sigma| {I_L}^{\rho}$ otherwise. $\diamondsuit$

The theoretical complexity based on the assumption of a worst case has been discussed above. However, also in order to make some amends for the costs of generalization, we have made some choices in the design of GENDET which in favourable cases may serve to reduce the size of the official table as defined by the current values of the sets S and E, and may thus spare us calls to the oracle which would have been needed to fill in the additional cells. This implies that the final table may have much less entries than the overall number of queries that have been asked. Note the following:

- We assume that the given alphabet Σ is the smallest alphabet such that $L \subseteq \mathbb{T}_\Sigma$, which reduces the cost of filling up BLUE with all one-symbol extensions of RED.

- We initialize RED with a single element from Σ_0 instead of taking the entire set as would have been suggested by a direct adaptation of the algorithm LSTAR [5]. Observe that by definition at that point all other elements of Σ_0 are elements of BLUE.

- We store all counterexamples in C since a single counterexample may serve to deduce more than one distinction – also see [47].

- At any time, RED contains at most one representative for each equivalence class of L, and for $MQ = 1$ the set E contains at most I_L contexts for their mutual distinction (also compare Appendix A.1).

- We have aimed to reduce the number of recursive calls to MINIMIZE by preferably instantiating the subtree s with a non-replaceable candidate from BLUE instead of choosing that candidate freely as done in [46].

- For $MQ = 1$ and $X_+ \neq \emptyset$ we keep adding all one-symbol extensions of RED to BLUE in order to represent all conceivable transitions. This may increase the number of candidates in the pool but it also minimizes the size of elements in RED and, provided that we force the learner to search the extended table for a suitable subtree and context in a certain order, of the counterexamples that are constructed as well.

Remark Note that there is a certain trade-off between instantiating the set S with a representative sample of L in order to ensure that all necessary transitions are represented in the table as done in [14] and our simulation of this fact by maintaining the set of *all* one-symbol extensions of RED in BLUE – the latter may lead to a certain redundancy and to a multitude of rows representing the failure state in BLUE but then on the other hand any given positive sample may be unnecessarily big as well. ♢

As mentioned above, the use of the procedure MINIMIZE can significantly reduce the number of cells in the table when compared to the other methods discussed in Appendix A.1. However, as stated in [12], the number of MQs to derive a distinguishing context from a counterexample c can be optimized from $O(|c|)$ to $O(log|c|)$, a bound which is achieved by algorithms given in [99] and [82]. We recommend [12, 45, 49] for an extended discussion of strategies to minimize the query complexity of algorithms that learn from membership and equivalence queries.

Remark One might ask why there is no (inevitable) exponential blow-up when learning tree instead of string languages, i.e., why we can keep at least some complexity measures at a polynomial bay. First of all, note that in the complexity discussion above the *size* of a tree refers to the number of nodes in it, and not to its depth. Moreover, during the learning process we have a fixed inventory of potential subtrees – the elements of RED – and the only candidates considered in addition are their one-symbol extensions. However, as pointed out in [46], if the maximal rank ρ of the alphabet is seen as a part of the input then the procedure of filling up BLUE is indeed of exponential complexity with respect to ρ – see [48] and [14] for remedies in the settings of MQs and EQs and of MQs and a positive sample, respectively. ♢

3.3.4 Outlook

We have aimed to factorize our meta-algorithm into reasonably many sub-procedures in order to obtain a uniform perspective on different settings and corresponding existing algorithms with their various idiosyncrasies by fitting them into a common mould with special emphasis on the basic routines that regardless of the setting are run through alike. This may help to formulate even clearer explanations for the interchangeability of information sources in algorithmical learning processes (also see [96, 70] and the survey in [98]). Moreover, GENDET can be used as a starting point for the instantiation of algorithms in hitherto unstudied settings, see for example Section 3.4.

This line of research can be extended in various directions. One of them is the object axis – as stated in Subsection 2.2.2, an adaptation of GENDET

to multi-dimensional trees is straightforward but we could explore further possibilities such as hedges, pictures, graphs, and infinite strings or trees as listed in Section 2.3 above. As a first step, we would have to define an unambiguous concatenation operation for each of those object types, which might not be completely trivial to accomplish. Also note that infinite structures as such do not necessarily lead to non-learnability – for example, [91] contains learnability results for so-called ω-regular sets of infinite strings, and [13] for so-called deterministic one-counter automata, that is, a kind of augmented finite-state automata which can be represented by transition graphs that are infinite while still featuring certain regularities.

Then there exist a range of other kinds of information sources throughout the literature that can be of use in the computation of distinctions between the equivalence classes of the target, including

- *subset* and *superset queries*, i.e., queries of the form 'Does my current hypothesis generate a subset/superset of the target language?', see [70],

- *correction queries* [110], where the teacher anwers an MQ by returning an object within minimal edit distance from the queried one, for various definitions of the notions of 'edit distance' and 'minimal',

- *label queries* [9], where the teacher has labeled the states of the target automaton in some way and informs the learner about the label of the state that is reached when parsing an input proposed by the learner,

- *active exploration* and *homing sequences* [99], where the learner can move stepwise along the edges of the transition graph of the target in which the states have been shaded in some way and is provided with a certain sequence of movements which uniquely determines any state by the sequence of shadings that is encountered when executing it,

- *distinguishing functions*, where the learner is provided with a certain function f over the objects in question and if f and the (deterministic) target automaton are such that f returns the same value for all objects ending up in the same state and different values for distinct states that are either both accepting or reach the same state by the same transition then f serves to resolve backward non-determinism and thus, like label queries and homing sequences, also allows some more insight into the internal structure of the target automaton, see [53].

We could try to modify our meta-learner such that the information sources suggested above can be made available via the input parameters as well.

The learning process of GENDET is executed and documented using an incrementally constructed observation table. An observation table shows the syntactic interrelations between all candidates and contexts that are taken into closer consideration by the learner. See [12] for a survey of other suitable representations for the case of strings where the authors propose the abstract concept of an *observation pack* and name observation tables and *discrimination trees* as two instantiations, and [49] for (weighted) tree automata where the authors define an *abstract observation table* and also instantiate it with an *observation tree*. A meta-learner based on one of those more process-oriented tree-like structures would probably benefit in terms of efficiency.

Finally, there is a line of research which extends the language class that is learned beyond regularity and even beyond context-freeness where instead of learning the target language itself we learn regular sets of control structures that encode certain descriptions or derivation processes for the language in question. A simple project would be the inference of mildly context-sensitive string languages by letting GENDET learn regular sets of multi-dimensional trees (as done in [74] for MQs and EQs) but there are other strategies and learnability results that we could try to reproduce in the framework of our meta-algorithm – see [52] for the control-based inference of *matrix grammars* featuring entire sets of context-free rules that have to be applied in parallel, or [51] for languages recognizable by a pair of automata where one simulates a counter and the other serves as a control structure for that counter.
We could also study conditions that we may impose on the given input when GENDET is learning certain restricted regular classes such as the reversible, function-distinguishable or locally testable languages (see [4, 53, 66]).

3.4 Residual finite-state tree automata

The results in the previous sections intrinsically rely on the existence, and moreover, on the specific structural properties of a canonical representation for the regular target language. Most beneficial for Grammatical Inference is the property of any kind of canonical representation that for every member of the language class in question, it is unambiguously defined.

In the previous sections this role was fulfilled by a state-minimal deterministic finite-state automaton. However, there is a price to pay when choosing a deterministic automaton as a reference description: In a worst case it can have exponentially many more states than a state-minimal non-deterministic finite-state automaton for the same language, and as for many applications a small number of states is a desirable feature it seems worth considering if one can find a way to obtain such a non-deterministic automaton instead. In [43], Denis et al. introduce a special case of non-deterministic finite-state automata for strings (NFA), so-called *residual finite-state automata* (RFSA), where each state represents a residual language of the language recognized by the NFA. A residual language is the set of all contexts extending a certain element into the language in question. For any formal language there is a natural one-to-one correspondence between the residual languages and the equivalence classes it defines. Thus, contrary to NFA in general, RFSA also have the advantageous property that for every regular language there exists a unique state-minimal (transition-maximal or *saturated*) RFSA, which makes them an attractive choice for descriptions in the design of learning algorithms and their applications due to their succinctness since that RFSA can be exponentially smaller than the state-minimal DFA and is at most as big.

As stated above, the output of all algorithms described or mentioned so far in this work is (in case of success) a deterministic finite-state automaton. However, in [42] Denis et al. provide an algorithm learning regular string languages from given data that returns a saturated RFSA with certain properties such that for every regular target language it is unambiguously defined and for many languages has a smaller number of states than the corresponding state-minimal DFA, and eventually Bollig et al. [17] have presented an algorithm for the learning setting of MQs and EQs (based on Angluin's LSTAR [5] for the deterministic case) which in case of success returns an RFSA that is isomorphic to the canonical state-minimal RFSA mentioned above.

The notion of RFSA for strings can be equally extended to trees: *Residual finite-state tree automata* (RFTA) have been defined and studied in [20]. As in the case of strings, for every regular tree language L there exists a unique state-minimal RFTA $\mathcal{R}_L$, which moreover can be exponentially more succinct than the corresponding state-minimal DFTA $\mathcal{A}_L$.

Potential applications for learning via the residual approach are all those that particularly benefit from a low practical complexity (as shown by Bollig et al. [17]) on the one hand and a succinct description on the other. One application named in [17] is automatic verification. Other areas specifically related to trees include the extraction of information from semi-structured data that may for example be given in XML (also consult `http://mostrare.lille.inria.fr`), or various applications in computational linguistics since linguistic structure is often represented in tree form and one can imagine a range of situations where one might want to derive a succinct description for the language under consideration from huge amounts of data contained in treebanks or similar databases.

In this section we will attempt to reproduce the discussion from the previous sections under the assumption that the chosen canonical description is not a DFTA but an RFTA. The learnability results for residual finite-state automata so far were all given for the case of strings ([42, 17]) – we treat the tree case which involves a fair amount of additional intricacy when one considers the transitions in a possible target tree automaton due to the interactions between neighbouring subtrees. On the one hand, we want to demonstrate that we can assemble algorithms with components similar to the ones developed in Subsection 3.3.1 since it shows that due to the Myhill-Nerode theorem the inference of DFTA and RFTA is based on the same fundamental principle. On the other hand, such an adaptation is not completely straightforward and we would also like to discuss in some detail the differences and difficulties encountered when trying to formulate minimal conditions for successful identification. This results in several new algorithms and theorems that serve to complete the picture of regular tree language inference in settings involving combinations of the information sources fixed above. For previous work by the author on that subject, also see [76, 79, 81].

3.4.1 Preliminaries

We fix a finite ranked alphabet Σ for the next two subsections.
In this section we will denote the state-minimal total DFTA for a regular tree language $L \subseteq \mathbb{T}_\Sigma$ by $\mathcal{A}_L^\bullet = \langle \Sigma, Q_\bullet, F_\bullet, \delta_\bullet \rangle$ and the state-minimal DFTA for L without a failure state by $\mathcal{A}_L^\circ = \langle \Sigma, Q_\circ, F_\circ, \delta_\circ \rangle$.
The following three definitions are based on [20].

Definition 33 *The* (bottom-up) residual language $t^{-1}L$ *of a tree* $t \in \mathbb{T}_\Sigma$ *with respect to a language* $L \subseteq \mathbb{T}_\Sigma$ *is defined as the set* $\{e \in \mathbb{C}_\Sigma \mid e[\![t]\!] \in L\}$. *The set of all residual languages defined by* L *is denoted by* $\mathcal{C}_L$.

There is a natural correspondence between the equivalence classes and the residual languages of L determined by $s^{-1}L = t^{-1}L \Leftrightarrow s \equiv_L t$ for any $s, t \in \mathbb{T}_\Sigma$ which is due to the fact that the definitions of both concepts are based on the substructure-context relation. As a consequence, each of the I_L equivalence classes under $\equiv_L$ defines a unique residual language with respect to L, and any pair of equivalence classes of L can be distinguished by their differing – but not necessarily disjoint – sets of contexts. Obviously, since the index I_L is finite if and only if L is regular the same holds for $\mathcal{C}_L$. This precise correspondence also accounts for Lemmata 6 and 7, which were proven in [20]:

Lemma 6 *Let $t_1, \ldots, t_n, t'_1 \ldots, t'_n \in \mathbb{T}_\Sigma$ for some $n \geq 1$.*
If we have ${t_i}^{-1}L \subseteq {t'_i}^{-1}L$ for all $i \in \{1, \ldots, n\}$ then we also have
$$f(t_1, \ldots, t_n)^{-1}L \subseteq f(t'_1 \ldots, t'_n)^{-1}L \text{ for all } f \in \Sigma_n.$$

Lemma 7 *Let $t_1, \ldots, t_n \in \mathbb{T}_\Sigma$ for some $n \geq 1$, and*
for each $i \in \{1, \ldots, n\}$, let $X_i \subseteq \mathbb{T}_\Sigma$ be a set of trees such that
${t_i}^{-1}L = \bigcup\{t^{-1}L \in \mathcal{C}_L \mid t \in X_i\}$. Let $f \in \Sigma_n$. We have
$$f(t_1, \ldots, t_n)^{-1}L = \bigcup\{f(t'_1, \ldots, t'_n)^{-1}L \in \mathcal{C}_L \mid \forall i \in \{1, \ldots, n\} : t'_i \in X_i\}.$$

Associated with the notion of a residual language one can define a special kind of FTA as follows:

Definition 34 *A* (bottom-up) residual finite-state tree automaton (RFTA) *is an FTA $\mathcal{R} = \langle \Sigma, Q, F, \delta \rangle$ such that*
for all $q \in Q$ there is $t \in \mathbb{T}_\Sigma$ with $\mathcal{C}_q = t^{-1}\mathcal{L}(\mathcal{R})$.

The class of tree languages that are recognized by bottom-up RFTA corresponds exactly to the class of regular tree languages as a whole (see [20]).
Remark In [20] the authors also define the concept of a top-down RFTA, and show that the class of tree languages that are recognized by top-down RFTA is properly included in the regular class. In this work we will consider bottom-up RFTA only and therefore drop the adjunct 'bottom-up'. ♢

Definition 35 *Let $L \subseteq \mathbb{T}_\Sigma$.*
A residual language $\gamma \in \mathcal{C}_L$ is composed *iff $\gamma = \bigcup\{\gamma' \in \mathcal{C}_L \mid \gamma' \subsetneq \gamma\}$.*
Otherwise we say that it is prime.
The set of prime residual languages of L is denoted by $\mathcal{P}_L$.

Remark If the empty union is defined as the empty set then by the definition above the empty set must be classified as composed which also implies that prime residual languages are intrinsically non-empty. ♢

For a regular $L \subseteq \mathbb{T}_\Sigma$, one can define an FTA $\mathcal{R}_L = \langle \Sigma, Q_L, F_L, \delta_L \rangle$ by

- $Q_L := \mathcal{P}_L$,
- $F_L := \{y \in Q_L \mid \Box \in y\}$, and
- $\delta_L := \{\langle f, y_1 \cdots y_n \rangle \mapsto y \mid y_1, \ldots, y_n, y \in Q_L \wedge f \in \Sigma_n \wedge \exists t_1, \ldots, t_n \in \mathbb{T}_\Sigma : \forall i \in \{1, \ldots, n\} : y_i = {t_i}^{-1}L \wedge y \subseteq f(t_1, \ldots, t_n)^{-1}L\}$.

Observe how the transitions in δ_L are determined by the inclusion relations among the residual languages of L. The resulting FTA $\mathcal{R}_L$ recognizes L and meets the definition of an RFTA. Moreover, for each state $y \in Q_L = \mathcal{P}_L$ the set $\mathcal{C}_y$ equals y. The definition of δ_L also entails that $\mathcal{R}_L$ is *saturated*, i.e., any addition to the transitions of δ_L would result in an automaton that recognizes a superset of L. Thus, $\mathcal{R}_L$ is the unique state-minimal saturated RFTA recognizing L up to isomorphism (see [20] – we have reproduced the corresponding theorem and proofs in Appendix A.3) and is suitable as a canonical description for L. Note that there may be other RFTA recognizing L with the same number of states and less transitions – however, a state-minimal RFTA for L featuring the smallest possible number of transitions may not be unique and is therefore less attractive for an algorithmic learning model.

Remark The adjective 'prime' evokes the notion that prime residual languages are in some sense "elementary" or "indispensable". Intuitively, the prime residual languages of a language L are exactly those needed to ensure that an NFTA fulfils the two conditions of being an RFTA and recognizing L. ♢

Remark We observe that since prime residual languages are non-empty by definition Q_L cannot contain a failure state y_0 with $\mathcal{C}_{y_0} = \emptyset$ and thus in general $\mathcal{R}_L$ is not total. Hence, unlike in the case of DFTA (see Subsection 2.2.1) the term 'the state-minimal saturated RFTA for L' can only refer to a single RFTA up to isomorphism. However, consider the following alternative definition of composed residual languages used by Bollig et al. in [17]:
A residual language $\gamma \in \mathcal{C}_L$ is composed if there is a subset $Y \subseteq (\mathcal{C}_L \setminus \{\gamma\})$ such that $\gamma = \bigcup Y$, otherwise γ is prime.
By this definition the empty set is prime which would imply that the unique RFTA $\mathcal{R}_L$ as defined above can have a failure state y_0 with $\mathcal{C}_{y_0} = \emptyset$ which moreover fulfils $y_0 \in \delta_L^*(t)$ for any $t \in \mathbb{T}_\Sigma$ due to the definition of δ_L. As a consequence, as for DFTA we could define two different notions of a canonical RFTA for L, both saturated, such that one ($\mathcal{R}_L^\circ$; based on Definition 35)

is state-minimal and the other ($\mathcal{R}_L^\bullet$; based on the definition of composed residual languages in [17] given above) is total and thus may have one more state, which is a failure state. For our purposes, by keeping to Definition 35 we choose $\mathcal{R}_L^\circ$ as the canonical description $\mathcal{R}_L$ for L since we would like to concentrate on sets of contexts that extend at least one tree into L. $\diamondsuit$

3.4.2 Observation tables and RFTA

We need some more tools and notions to relate observation tables and RFTA.

Definition 36 *For two trees $s, s' \in \mathbb{T}_\Sigma$ with $s^{-1}L \not\subseteq s'^{-1}L$ and a context $e \in \mathbb{C}_\Sigma$, we call e an* excluding *context for s and s' if $e \in s^{-1}L$ but $e \notin s'^{-1}L$.*

For the following definitions (based on [17] where the string case is treated), we fix an observation table $T = \langle S, E, obs\rangle$ with $S = \text{RED} \cup \text{BLUE}$.

Definition 37 *We fix an order $<$ on $\{0, 1, *\}$ such that $0 < * < 1$, and we let $max_<(X) := x \in X$ such that $\forall x' \in X : x' < x \vee x' = x$ for $X \subseteq \{0, 1, *\}$.*
For $r_1, r_2 \in row(\mathbb{T}_\Sigma)$ and $R \subseteq row(\mathbb{T}_\Sigma)$, we define a join operation *such that*

$r_1 \sqcup r_2 := \{\langle e, x\rangle \mid e \in E \wedge x = max_<(\{r_1(e)\} \cup \{r_2(e)\})\}$, *and*
$\bigsqcup R := \{\langle e, x\rangle \mid e \in E \wedge x = max_<(\{r(e) \mid r \in R\})\}$.

For two rows $r, r' \in row(\mathbb{T}_\Sigma)$ we say that r is covered *by r', and denote it by $r \sqsubseteq r'$, if $r(e) \in \{1, *\}$ implies $r'(e) = 1$ for all $e \in E$.*
For two trees $s, s' \in \mathbb{T}_\Sigma$, if $row(s) \sqsubseteq row(s')$ then we also write $s \sqsubseteq s'$.

A row $r \in row(S)$ is said to be composed *if there are rows $r_1, \ldots, r_n \in row(S) \setminus \{r\}$ such that $r = \bigsqcup\{r_1, \ldots, r_n\}$. If r is not composed and there is at least one $e \in E$ with $r(e) = 1$ then r is said to be* prime.
We define $\mathcal{P}_X := \{r \in row(X) \mid r$ is prime$\}$ for a set $X \subseteq \mathbb{T}_\Sigma$, and $\mathcal{P}_s := row(\{s' \in S \mid s' \sqsubseteq s\}) \cap \mathcal{P}_{\text{RED}}$ for $s \in \mathbb{T}_\Sigma$.

Remark We require the existence of at least one positive context for prime rows in addition to the original definition by Bollig et al. [17] because in accordance with the assumption that prime residual languages are intrinsically non-empty we do not want to classify a potential failure row containing only 0s as prime. Also see the remark at the end of the previous subsection. $\diamondsuit$

Definition 38 *T is* R-closed *if it fulfils $\mathcal{P}_{\text{BLUE}} \subseteq \mathcal{P}_{\text{RED}}$.*

T is weakly R-consistent *if,*
for all $f(s_1, \ldots, s_n), f(s'_1, \ldots, s'_n) \in S$ and $1 \le i \le n$,
$s_i \sqsubseteq s'_i$ *implies* $f(s_1, \ldots, s_n) \sqsubseteq f(s'_1, \ldots, s'_n)$.

		$\Box$	$f(\Box, b)$	$f(a, \Box)$
RED	a	0	1	0
	b	1	1	1
	$f(a,a)$	1	0	0
BLUE	$f(a,b)$	1	1	0
	$f(b,b)$	1	0	0

Figure 3.1: An example for an observation table and relations in it

T is strongly R-consistent *if it is weakly R-consistent and, for all $s_1, s_2 \in S$ with $s_1 \neq s_2$, $s_1 \sqsubseteq s_2$ implies that $row(s_1)$ is complete.*

Remark It is easy to see that closedness implies R-closedness whereas (weak, strong) R-consistency implies (weak, strong) consistency. ♢

Let us consider a small example to clarify the definitions given above.

Example 2 In the small (complete) observation table given in Figure 3.1,

- $row(f(a,b))$ is the only one that is not prime because it can be composed from $row(a)$ and $row(f(a,a))$ or $row(f(b,b))$,
- $row(b)$ covers all others, $row(f(a,b))$ covers $row(a)$, $row(f(a,a))$ and $row(f(b,b))$, and $row(f(a,a))$ and $row(f(b,b))$ cover each other.

Moreover, this table is R-closed because the only prime row in $row(\text{BLUE})$, $row(f(b,b))$, is also an element of $row(\text{RED})$ due to the fact that there is $f(a,a) \in \text{RED}$ with $row(f(a,a)) = row(f(b,b))$ but it is not R-consistent because $row(b)$ covers $row(a)$ but $row(f(b,b))$ does not cover $row(f(a,b))$.

From T we can derive an FTA $\mathcal{R}_T = \langle \Sigma, Q_T, F_T, \delta_T \rangle$ defined by

- $Q_T := \mathcal{P}_{\text{RED}}$,
- $F_T := \{r \in Q_T \mid r(\Box) = 1\}$, and
- $\delta_T := \{\langle f, q_1 \cdots q_n \rangle \mapsto q \mid q_1, \ldots, q_n, q \in Q_T \wedge$
  $\exists s_1, \ldots, s_n, f(s_1, \ldots, s_n) \in S :$
  $\forall i \in \{1, \ldots, n\} : q_i = row(s_i) \wedge$
  $q \sqsubseteq row(f(s_1, \ldots, s_n))\}$.

Note that if T is R-consistent then the following holds: Consider $s_1, \ldots, s_n$, $s'_1, \ldots, s'_n \in \mathcal{P}_{\text{RED}}$ with $s_i \approx s'_i$ for $1 \leq i \leq n$. Clearly, we have $s_i \sqsubseteq s'_i$ and $s'_i \sqsubseteq s_i$. For any $f \in \Sigma_n$, if the trees $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ are also in S then due to the R-consistency of T we have $s \sqsubseteq s'$ and $s' \sqsubseteq s$ and hence $s \approx s'$ as well such that $row(s)$ and $row(s')$ cover the same (prime) rows, i.e., states in Q_T. Conversely, if T is not R-consistent then different instantiations of $s_1, \ldots, s_n$ as representatives of $q_1, \ldots, q_n$ within the definition of δ_T may contribute different subsets of $\{q \in Q_T \mid \langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_T\}$.

Remark The derivation of the FTA $\mathcal{R}_T$ from T is an adaptation of a similar construction in [17] (for strings) to the tree case – however, whereas the authors of [17] depend on R-consistency to ensure that their transition function be well-defined at all since they use a single arbitrary representative for the state from which a transition is going out we have avoided this particular problem by collecting them all via an existential quantifier. $\diamondsuit$

Obviously, R-closedness is needed to establish all prime rows of T as states in Q_T. In conjunction with R-closedness, R-consistency is a necessary (albeit not sufficient) condition if we want to ensure that the derived automaton $\mathcal{R}_T$ is an RFTA that recognizes the target language L. A first intuitive explanation runs as follows: As can be inferred from the next subsection, if the learning process succeeds then the covering relation $\sqsubseteq$ between the rows of T and the subset relation between the residual languages of L defined by the row labels coincide. An R-inconsistency reveals that some rows in T are still wrongly covered by others. As a consequence, due to the use of $\sqsubseteq$ in the definition of δ_T there may be trees $t \in \mathbb{T}_\Sigma$ for which $\delta^*_T(t)$ contains states q such that $\mathcal{C}_q$ does not constitute any residual language actually included in $t^{-1}L$, i.e., there may be contexts $e \in \mathcal{C}_q$ such that $\mathcal{R}_T(e[\![t]\!]) = 1$ but $e[\![t]\!] \notin L$. Thus, R-consistency serves to prevent overgeneralization.

For some more discussion of the relationship between observation tables and residual finite-state automata for strings by the author considering the close connection between the state-minimal residual automaton for a given regular string language and an arbitrary residual automaton for its reversal shown in [43], also see [76, 79].

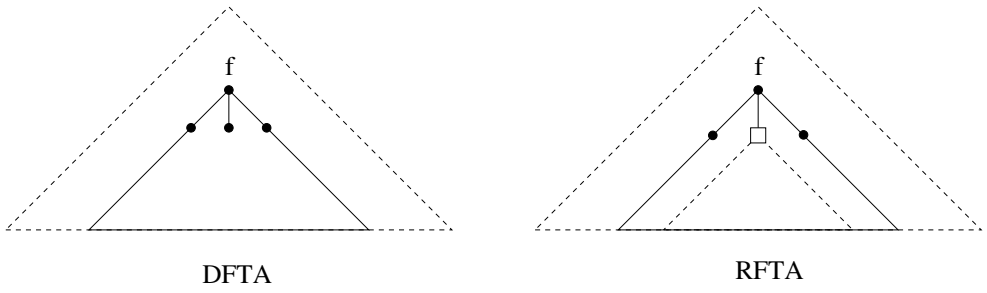

Figure 3.2: Transitions in DFTA and RFTA

3.4.3 Theoretical groundwork

In this subsection we establish a range of lemmata and theorems exploring the relationship between observation tables and RFTA in a similar way as undertaken for the case of DFTA in Subsection 3.2.2. However, the residual case is generally more intricate which intuitively can be motivated by the following circumstances: (a) Instead of basing the automaton derived from a table on the identity between rows we now base it on the covering relation between rows, which is linked to the fact that elements of distinct equivalence classes can share some contexts (but not all of them), (b) finding a single pair of cells with obviously differing values is enough to exclude the identity of two rows but may not be enough to exclude that one row covers the other – the two cells must also be "poled" the right way, (c) in a worst case we need at least one more context to establish that a row is prime than to establish that it is obviously different from all others, (d) the fact that a row can only be identical to one "other" row but can cover several others corresponds to (admissible!) non-determinism on the automaton level which also implies that from a certain state the conceivable transition sequences to reach an accepting state multiply in proportion to the number of states, and (e) when compared to strings the tree case represents a particular complication: Whereas in order to understand the transitions in a DFTA we have to concentrate on the constellation formed by the root label and the direct subtrees of a tree and could consider any context for that tree as a blackbox of some sort, for RFTA we rather have to concentrate on the constellation formed by the root label and direct subtrees of the tree rooted at the mother of $\Box$ in a certain context and in turn consider any plugged-in tree as a blackbox (i.e., consider the symbol $\Box$ as the representative of a state that is yet to specify), which makes it harder to keep all relevant components in view (also see Figure 3.2).

We fix a finite ranked alphabet Σ, a regular tree language $L \subseteq \mathbb{T}_\Sigma$, and a complete, R-closed, and R-consistent table $T = \langle S, E, obs \rangle$ with $S = \text{RED} \cup \text{BLUE}$ for the rest of this subsection. Note that since T is complete the R-consistency of T is automatically strong. As in Subsection 3.2.2, we assume that we have $obs(s, e) \in \{0, 1\}$ for all $s \in \mathbb{T}_\Sigma$ and all $e \in E$.

Theorem 4 *Assume T to fulfil the following three conditions.*

(1) E is S-composed,

(2) T is saturated*, i.e.,* $\text{BLUE} = \Sigma(\text{RED})$*, and*

(3) $\mathcal{R}_T$ is T-consistent.

Then $\mathcal{R}_T$ is a state-minimal saturated RFTA.

This will be proven by a range of lemmata where the basic ideas are based on comparable ones for strings as given in [17] but since the tree case adds a considerable amount of intricacy both the preconditions (listed in Theorem 4) and the proofs of those lemmata had to be reorganized completely.

Lemma 8 *We have $q \sqsubseteq row(s)$ for all $s \in S$ and all $q \in \delta_T^*(s)$.*

Proof. By induction over the depth of s. If $s = a$ for some $a \in \Sigma_0$ then the claim follows directly from the definition of δ_T. Let $s = f(s_1, \ldots, s_n)$. Then $\delta_T^*(s) = \{q \in Q_T \mid \exists \langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_T : \forall i \in \{1, \ldots, n\} : q_i \in \delta_T^*(s_i)\}$. Let $q \in \delta_T^*(s)$ and assume the claim to hold for $s_1, \ldots, s_n$. By the definition of δ_T there are trees $s'_1, \ldots, s'_n \in S$ and $q_i \in \delta_T^*(s_i)$ such that $q_i = row(s'_i)$ for $1 \leq i \leq n$ and $q \sqsubseteq row(f(s'_1, \ldots, s'_n))$. By the induction assumption we have $q_i \sqsubseteq row(s_i)$. This yields $s'_i \sqsubseteq s_i$, and also $f(s'_1, \ldots, s'_n) \sqsubseteq f(s_1, \ldots, s_n)$ due to the R-consistency of T and hence $q \sqsubseteq row(f(s_1, \ldots, s_n)) = row(s)$. ■

For Lemmata 9–10 assume that T fulfils conditions (1)–(3) as stipulated in Theorem 4. Intuitively, Lemma 9 states the following: Provided that we have reached a certain state of Q_T then $\mathcal{R}_T$ correctly classifies the contexts and their subcontexts in E. Recall that we had defined $q(e) := obs(s, e)$ for $q \in row(\mathbb{T}_\Sigma)$, $e \in E$, and any $s \in \mathbb{T}_\Sigma$ with $row(s) = q$ (see Subsection 3.2.1).

Lemma 9 *For all $q \in Q_T$ and all $e \in Cont(E)$ we have $q(e) = 1 \Leftrightarrow e \in \mathcal{C}_q$.*

Proof. Let $q \in Q_T$. We prove this by induction over the depth of e.
For $e = \Box$ we have $q(\Box) = 1 \Leftrightarrow q \in F_T \Leftrightarrow \Box \in \mathcal{C}_q$, and the claim is shown.

Let $e = e'[\![e'']\!]$ with $e', e'' \in \mathbb{C}_\Sigma$ and $e'' = f(s_1, \ldots, s_n)$ such that $s_j = \Box$ for some $j \in \{1, \ldots, n\}$, and assume the claim to hold for e'.

Let $s \in S$ with $row(s) = q$. Note that we have $(\{s_1, \ldots, s_n\} \setminus \{s_j\}) \subseteq S$ due to condition (1) and $e''[\![s]\!] \in S$ due to condition (2).

"$\Rightarrow$": Let $q(e) = 1$. Since $\mathcal{R}_T$ is T-consistent there has to be a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q' \in \delta_T$ with states $q_1, \ldots, q_n, q' \in Q_T$ such that $q_i \in \delta_T^*(s_i)$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ and $q_j \in \delta_T^*(s)$ and $e' \in \mathcal{C}_{q'}$. Let $s'_1, \ldots, s'_n \in \text{RED}$ with $row(s'_i) = q_i$ for $1 \leq i \leq n$. We have $f(s'_1, \ldots, s'_n) \in S$ by condition (2), $s'_i \sqsubseteq s_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ and $s'_j \sqsubseteq s$ due to Lemma 8, and $q' \sqsubseteq row(f(s'_1, \ldots, s'_n))$ due to the definition of δ_T. By condition (2) there is also an element $f(t'_1, \ldots, t'_n) \in S$ with $t'_i = s'_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ and $t'_j = s$ and we have $row(f(s'_1, \ldots, s'_n)) \sqsubseteq row(f(t'_1, \ldots, t'_n))$ since T is R-consistent. This entails $q' \sqsubseteq row(f(t'_1, \ldots, t'_n))$ which in turn implies that there is a transition $\langle f, q'_1 \cdots q'_n \rangle \mapsto q' \in \delta_T$ with $q'_i = q_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ and $q'_j = q$ as well, and we can conclude that $e'[\![e'']\!] = e$ is in $\mathcal{C}_q$.

"$\Leftarrow$": Let $q(e) = 0$. Choose states $q_1, \ldots, q_n \in Q_T$ with $q_i \in \delta_T^*(s_i)$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ and $q_j = q$, and let $s'_1, \ldots, s'_n \in \text{RED}$ such that $row(s'_i) = q_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ and $s'_j = s$. We have $f(s'_1, \ldots, s'_n) \in S$ by condition (2) and $s'_i \sqsubseteq s_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ due to Lemma 8. Moreover, we have $row(f(s'_1, \ldots, s'_n)) \sqsubseteq row(e''[\![s]\!])$ due to the R-consistency of T.
If there is no row $r \in \mathcal{P}_{\text{RED}}$ with $r \sqsubseteq row(f(s'_1, \ldots, s'_n))$ then clearly there is no transition $\langle f, q_1 \cdots q_n \rangle \mapsto q' \in \delta_T$ for any $q' \in Q_T$. If there is $r \in \mathcal{P}_{\text{RED}}$ with $r \sqsubseteq row(f(s'_1, \ldots, s'_n))$ then there exists a transition $\langle f, q_1 \cdots q_n \rangle \mapsto r \in \delta_T$. Obviously, $row(s)(e) = 0$ implies $row(e''[\![s]\!])(e') = 0$, which in turn entails $row(f(s'_1, \ldots, s'_n))(e') = 0$ and $r(e') = 0$. We obtain $e' \notin \mathcal{C}_r$ by the induction assumption. Since $q_1, \ldots, q_n$ were chosen arbitrarily except for $q_j = q$ we can deduce $q'(e') = 0$ and $e' \notin \mathcal{C}_{q'}$ for all states q' that are reachable from q via the symbol f, and consequently e cannot be an element of $\mathcal{C}_q$. ■

Remark Note the impact of the fact that in contrast to a string context a tree context may have subtrees that are elements of $\mathbb{T}_\Sigma$ and have to be taken into account as well: Whereas in the string case saturation of S suffices to prove both directions of Lemma 9 (see [17]), in the tree case the "$\Rightarrow$"-direction particularly depends on $\mathcal{R}_T$ assigning suitable states to those subtrees when trying to classify the context e. The easiest way to achieve this is by requiring S-composure of E (in order to make Lemma 8 applicable) and T-consistency of $\mathcal{R}_T$ (in order to ensure the existence of the necessary transitions). ♢

Lemma 10 is of importance because it entails that the finite(!) set E contains enough contexts to distinguish all (prime) residual languages of $\mathcal{L}(\mathcal{R}_T)$.

Lemma 10 *For all $q, q' \in Q_T$ we have $q \sqsubseteq q' \Leftrightarrow \mathcal{C}_q \subseteq \mathcal{C}_{q'}$.*

Proof: Let $q, q \in Q_T$ and $s, s' \in S$ with $row(s) \sqsubseteq q$ and $row(s') \sqsubseteq q'$.

"$\Rightarrow$": Let $q \sqsubseteq q'$. Choose $e \in \mathcal{C}_q$. For $e = \Box$ we have $q(\Box) = 1$ which implies $q'(\Box) = 1$ due to $q \sqsubseteq q'$ and thus $q' \in F_T$ and $\Box \in \mathcal{C}_{q'}$.

Now let $e = e'[\![e'']\!]$ with $e', e'' \in \mathbb{C}_\Sigma$ and $e'' = f(s_1, \ldots, s_n)$ and $s_j = \Box$ for some $j \in \{1, \ldots, n\}$. Note that $e''[\![s]\!], e''[\![s']\!] \in S$ by condition (2).
We have $e'[\![e'']\!] \in \mathcal{C}_q$ and thus there is a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q'' \in \delta_T$ with $q_1, \ldots, q_n, q'' \in Q_T$ such that $q_i \in \delta_T^*(s_i)$ for $i \in \{1, \ldots, n\} \backslash \{j\}$ and $q_j = q$ and $e' \in \mathcal{C}_{q''}$. Let $s'_1, \ldots, s'_n \in \text{RED}$ with $row(s'_i) = q_i$ for $i \in \{1, \ldots, n\}$. We have $f(s'_1, \ldots, s'_n) \in S$ by condition (2), $s'_i \sqsubseteq s_i$ for all $i \in \{1, \ldots, n\} \backslash \{j\}$ and $s'_j \sqsubseteq s$ due to Lemma 8, and $q'' \sqsubseteq row(f(s'_1, \ldots, s'_n))$ due to the definition of δ_T. By condition (2) there is $f(t'_1, \ldots, t'_n) \in S$ with $t'_i = s'_i$ for $i \in \{1, \ldots, n\} \backslash \{j\}$ and $t'_j = s'$. Moreover, we have $row(f(s'_1, \ldots, s'_n)) \sqsubseteq row(f(t'_1, \ldots, t'_n))$ due to the fact that T is R-consistent and $q \sqsubseteq q'$. This entails $q'' \sqsubseteq row(f(t'_1, \ldots, t'_n))$ which in turn implies the existence of a transition $\langle f, q'_1 \cdots q'_n \rangle \mapsto q'' \in \delta_T$ with $q'_i = q_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ and $q'_j = q'$, and we can conclude that $e'[\![e'']\!] = e$ is in $\mathcal{C}_{q'}$ as well.

The inclusion $\mathcal{C}_q \subseteq \mathcal{C}_{q'}$ follows from the fact that e was chosen arbitrarily.

"$\Leftarrow$": Let $\neg(q \sqsubseteq q')$. By the definition of $\sqsubseteq$ there must be a context $e \in E$ with $q(e) = 1$ but $q'(e) = 0$. We obtain $e \in \mathcal{C}_q$ and $e \notin \mathcal{C}_{q'}$ by Lemma 9, which immediately proves $\mathcal{C}_q \nsubseteq \mathcal{C}_{q'}$. ■

Lemma 11 *For all $s \in S$ with $row(s) \in Q_T$ we have $row(s) \in \delta_T^*(s)$.*

Proof. Suppose $row(s) \notin \delta_T^*(s)$. We get a contradiction as follows.
We have $q \sqsubseteq row(s)$ and $\mathcal{C}_q \subseteq \mathcal{C}_{row(s)}$ for all $q \in \delta_T^*(s)$ by Lemmata 8 and 10. As $row(s)$ is prime there is $e \in E$ such that $row(s)(e) = 1$ but $row(s')(e) = 0$ for all $s' \in \mathcal{P}_{\text{RED}}$ with $s' \sqsubseteq s$ and $s' <> s$. As we have assumed that $row(s)$ is not in $\delta_T^*(s)$ itself this implies that e cannot be in $\mathcal{C}_{q'}$ for any $q' \in \delta_T^*(s)$ by Lemma 9, and hence $\mathcal{R}_T$ would not accept $e[\![s]\!]$. However, this contradicts condition (3) requiring $\mathcal{R}_T$ to be T-consistent, and the claim is shown. ■

The **Proof of Theorem 4** can now be concluded as follows.

We abbreviate $\mathcal{L}(\mathcal{R}_T)$ to L_R. First we show that $\mathcal{R}_T$ is an RFTA by proving the stronger claim $\mathcal{C}_{row(s)} = s^{-1}L_R$ for all $s \in S$ with $row(s) \in Q_T$:

"$\subseteq$": We have $row(s) \in \delta_T^*(s)$ by Lemma 11, which implies $\mathcal{C}_{row(s)} \subseteq s^{-1}L_R$.

"$\supseteq$": We have $q \sqsubseteq row(s)$ and $\mathcal{C}_q \subseteq \mathcal{C}_{row(s)}$ for all $q \in \delta_T^*(s)$ by Lemmata 8 and 10, which yields $s^{-1}L \subseteq \mathcal{C}_{row(s)}$, and thus $\mathcal{C}_{row(s)} = s^{-1}L_R$.

It remains to show that $s^{-1}L_R$ is prime to ensure the state-minimality of $\mathcal{R}_T$: We have $\mathcal{C}_{row(s')} \subseteq \mathcal{C}_{row(s)}$ for all $s' \in \mathcal{P}_{\text{RED}}$ with $s' \sqsubseteq s$ by Lemma 10 and $s^{-1}L_R \supsetneq \bigcup\{s'^{-1}L_R \subseteq \mathbb{C}_\Sigma \mid s' \in \mathcal{P}_{\text{RED}} \setminus \{s\} : s' \sqsubseteq s\}$ due to the fact that

there is a context $e \in E$ such that $row(s)(e) = 1$ but $row(s')(e) = 0$ for all $s' \in \mathcal{P}_{\text{RED}}$ with $s' \sqsubseteq s$ and $s' <> s$.

Finally, $\mathcal{R}_T$ is saturated by condition (2) and hence isomorphic to $\mathcal{R}_{L_R}$. ■

We have shown that the properties listed as preconditions of Theorem 4 together suffice to make $\mathcal{R}_T$ a state-minimal saturated RFTA for $\mathcal{L}(\mathcal{R}_T)$. However, we are able to formulate an alternative set of conditions not only ensuring that $\mathcal{R}_T$ is a state-minimal saturated RFTA but also that it recognizes exactly L. Basically, we can trade in the saturation of T and S-composure of E for the much stronger precondition of S being representative for the prime residual languages of L and for all transitions between them, and the T-consistency of $\mathcal{R}_T$ for the condition that E provides enough information to correctly identify a prime residual language of L if represented by some element of S and to prevent incorrect covering relations between rows in T. In reminiscence of Definitions 29 and 30 we define the following properties that a given set of data can fulfil with respect to the residual languages of L:

Definition 39 *A finite set $X_+ \subseteq Subt(L)$ is* R-representative *for L if*

- *for every state $q \in Q_L = \mathcal{P}_L$ of the canonical RFTA $\mathcal{R}_L$ there is $s \in Subt(X_+)$ with $s^{-1}L = q$, and*
- *for every transition $\langle f, q_1 \cdots q_n \rangle \mapsto q' \in \delta_L$ there is $f(s_1, \ldots, s_n) \in Subt(X_+)$ with ${s_i}^{-1}L = q_i$ for $1 \leq i \leq n$.*

Remark Clearly, any representative set for L is R-representative for L which is due to the fact that each prime residual language of L can be mapped to some equivalence class under $\equiv_L$. ♢

Definition 40 *A finite set $X \subseteq \mathbb{T}_\Sigma$ is* exclusive *for L if for all $t, t' \in \mathbb{T}_\Sigma$ with $t^{-1}L \nsubseteq t'^{-1}L$ there is $e \in Cont(X)$ such that $e \in t^{-1}L \setminus t'^{-1}L$.*
We also say that a finite set $C \subseteq \mathbb{C}_\Sigma$ is exclusive *for L if for all $t, t' \in \mathbb{T}_\Sigma$ with $t^{-1}L \nsubseteq t'^{-1}L$ there is $e \in C$ such that $e \in t^{-1}L \setminus t'^{-1}L$.*

Intuitively, a set is exclusive for L if for each pair of residual languages γ, γ' of L with $\gamma \nsubseteq \gamma'$ it features a context by which the inclusion can be disproven (but note that in order to actually disprove the inclusion we need additional information about the membership status of certain trees with respect to L).

Remark Obviously, any exclusive set $X \subseteq \mathbb{T}_\Sigma$ for L is separative for L since $t^{-1}L \nsubseteq t'^{-1}L$ implies $\neg(t \equiv_L t')$ for $t, t' \in \mathbb{T}_\Sigma$ and any excluding context e with $e \in t^{-1}L \setminus t'^{-1}L$ can be used to disprove the equivalence of t and t'. ♢

Definition 41 *A finite set $X \subseteq \mathbb{T}_\Sigma$ ($C \subseteq \mathbb{C}_\Sigma$) is* R-exclusive *for L if for all $t, t' \in \mathbb{T}_\Sigma$ with $t^{-1}L \in \mathcal{P}_L$ and either $t'^{-1}L \in \mathcal{P}_L$ or $t' = f(t_1, \ldots, t_n) \in Subt(L)$ with ${t_i}^{-1}L \in \mathcal{P}_L$ for $1 \leq i \leq n$ and $t^{-1}L \nsubseteq t'^{-1}L$ there is $e \in Cont(X)$ ($e \in C$) such that $e \in t^{-1}L \setminus t'^{-1}L$.*

Intuitively, an R-exclusive set for L will prevent us from introducing incorrect transitions into our transition function. If the set E is R-exclusive for L and the prime rows in the table correspond to prime residual languages of L then the FTA $\mathcal{R}_T$ derived from the table is isomorphic to a subautomaton of $\mathcal{R}_L$. In order to ensure the latter, we need the additional definition of:

Definition 42 *A finite set $X \subseteq \mathbb{T}_\Sigma$ is* p-exclusive *for L if for each $\gamma \in \mathcal{P}_L$ there is $e \in Cont(X)$ such that $e \in \gamma$ but $e \notin \bigcup\{\gamma' \in \mathcal{C}_L \mid \gamma' \subsetneq \gamma\}$.*
We also say that a finite set $C \subseteq \mathbb{C}_\Sigma$ is p-exclusive *for L if for each $\gamma \in \mathcal{P}_L$ there is $e \in C$ such that $e \in \gamma$ but $e \notin \bigcup\{\gamma' \in \mathcal{C}_L \mid \gamma' \subsetneq \gamma\}$.*

Intuitively, a set is p-exclusive for L if for every prime residual language γ of L it features a context e by which it is possible to prove that γ is indeed prime. Note that this implies $e \in Cont(L)$.

Let S be R-representative and E be both exclusive and p-exclusive for L for the remaining part of this subsection.

Theorem 5 *$\mathcal{R}_T$ is a state-minimal saturated RFTA and fulfils $L_R = L$.*

Theorem 5 will be proven via the following range of lemmata and corollaries.

Definition 40 entails that if $s_1 \sqsubseteq s_2$ for $s_1, s_2 \in S$ then we can conclude ${s_1}^{-1}L \subseteq {s_2}^{-1}L$. Hence, the inclusion relations among rows in T exactly reflect those among the corresponding residual languages of L. Moreover:

Lemma 12 *For all $s \in S$ we have $s^{-1}L \in \mathcal{P}_L \Leftrightarrow row(s) \in \mathcal{P}_S$.*

Proof. "$\Rightarrow$": Let $s^{-1}L \in \mathcal{P}_L$. As E is p-exclusive for L there is $e \in E$ with $e \in s^{-1}L$ but $e \notin s'^{-1}L$ for all $s' \in S$ with $s'^{-1}L \subseteq s^{-1}L$ but $\neg(s' \equiv_L s)$. As we have $s'^{-1}L \subseteq s^{-1}L \Leftrightarrow s' \sqsubseteq s$ the row of s must be prime in the table.
"$\Leftarrow$": Let $row(s) \in \mathcal{P}_S$. This implies that there is $e \in E$ with $obs(s, e) = 1$ but $obs(s', e) = 0$ for all $s' \in S$ with $s' \sqsubseteq s$ but $s' <> s$. For each $\gamma \in \mathcal{P}_L$ with $\gamma \subseteq s^{-1}L$ there is an element $s'' \in S$ with $s''^{-1}L = \gamma$ and $s'' \sqsubseteq s$ due to the fact that S is R-representative. If we had $s^{-1}L \notin \mathcal{P}_L$ then for each $e' \in E$ with $obs(s, e') = 1$ there would be $s'' \in S$ with $s'' \sqsubseteq s$ but $s'' <> s$ and $obs(s'', e') = 1$, which contradicts the existence of e in E. ■

We will say that T is *faithful* for L to express that S is R-representative and E exclusive and p-exclusive for L, along with the consequences stated above,

i.e., that all prime residual languages of L and all transitions and inclusion relations between them are mirrored by components of T. Recall that L has a unique state-minimal saturated RFTA $\mathcal{R}_L = \langle \Sigma, Q_L, F_L, \delta_L \rangle$, and observe the exact parallelism between the definitions of δ_L and δ_T. The following lemmata are based on the fact that due to our preconditions the FTA $\mathcal{R}_T$ derived from T exactly imitates all actions of $\mathcal{R}_L$. In order to clarify the connection between $\mathcal{R}_T$ and $\mathcal{R}_L$ we also prove two lemmata with respect to $\mathcal{R}_L$.

Lemma 13 *We have $y \subseteq t^{-1}L$ for all $t \in \mathbb{T}_\Sigma$ and all $y \in \delta_L^*(t)$.*

Proof. By induction over the depth of t. If $t = a$ for some $a \in \Sigma_0$ then the claim follows directly from the definition of δ_L. Let $t = f(t_1, \ldots, t_n)$. Then $\delta_L^*(t) = \{y \in Q_L \mid \exists \langle f, y_1 \cdots y_n \rangle \mapsto y \in \delta_L : \forall i \in \{1, \ldots, n\} : y_i \in \delta_L^*(t_i)\}$. Let $y \in \delta_L^*(t)$ and assume the claim to hold for $t_1, \ldots, t_n$. By the definition of δ_L there are states $y_1, \ldots, y_n \in Q_L$ such that $y_i \in \delta_L^*(t_i)$ for $1 \leq i \leq n$ and trees $t'_1, \ldots, t'_n \in \mathbb{T}_\Sigma$ such that $y_i = t_i'^{-1}L$ and $y \subseteq f(t'_1, \ldots, t'_n)^{-1}L$. By the induction assumption we have $y_i \subseteq {t_i}^{-1}L$. This yields $t_i'^{-1}L \subseteq {t_i}^{-1}L$ for $1 \leq i \leq n$ and also $f(t'_1, \ldots, t'_n)^{-1}L \subseteq f(t_1, \ldots, t_n)^{-1}L$ due to Lemma 6, and thus $y \subseteq t^{-1}L$. ■

Corollary 2 *We have $q \sqsubseteq row(s)$ for all $s \in Subt(L)$ and all $q \in \delta_T^*(s)$.*

Proof. For $s \in S$ this holds by Lemma 13 and the R-consistency of T, and it holds for all $s \in Subt(L)$ by the additional fact that T is faithful for L. ■

Lemma 14 *For all $y \in Q_L$ and all $t \in \mathbb{T}_\Sigma$ with $t^{-1}L = y$ we have $y \in \delta_L^*(t)$.*

Proof. As y is prime there are $e \in y$ such that $e \notin y'$ for all y' with $y' \subsetneq y$. Let $t \in \mathbb{T}_\Sigma$ with $t^{-1}L = y$ and assume $y \notin \delta_L^*(t)$. By Lemma 13 we have $y'' \subseteq y$ for all $y'' \in \delta_L^*(t)$. By the argument above no $y'' \in \delta_L^*(t)$ can contain e. However, since $y'' = \mathcal{C}_{y''}$ this would imply that $\mathcal{R}_L$ does not accept $e[\![t]\!]$, which contradicts the fact that $\mathcal{L}(\mathcal{R}_L) = L$ (proven in [20], see Appendix A.3). ■

Corollary 3 *For all $s \in Subt(L)$ with $row(s) \in Q_T$ we have $row(s) \in \delta_T^*(s)$.*

Proof. By Lemma 14 and the fact that T is faithful for L. ■

Lemma 15 *For all $q \in Q_T$ with $q = row(s)$ for some $s \in S$, for all $e \in \mathbb{C}_\Sigma$, $e \in s^{-1}L \Leftrightarrow e \in \mathcal{C}_q$.*

Proof. This also follows directly from the faithfulness of T: By Lemma 12 $s^{-1}L$ is a prime residual language of L; since S is R-representative for L all transitions between prime residual languages of L are represented by elements of S, and since E is exclusive for L the inclusion relations are exactly mirrored as well such that δ_T contains the transitions that are necessary in order to parse the context e correctly. ■

Lemma 16 *For all $s \in Subt(L)$ and all $e \in s^{-1}L$ there is $q \in \delta_T^*(s)$ with $e \in \mathcal{C}_q$.*

Proof. If $s = a$ for some $a \in \Sigma_0$ then $s \in S$ because S is R-representative for L, and $\delta_T^*(s) = \mathcal{P}_a$ by the definition of δ_T. By Lemma 12 we have $row(s') \in \mathcal{P}_a \Leftrightarrow s'^{-1}L \in \mathcal{P}_a(L)$ for all $s' \in S$, and the claim follows from Lemma 15.
Let $s = f(s_1, \ldots, s_n)$ and assume the claim to hold for $s_1, \ldots, s_n$. For $1 \leq i \leq n$, we define the context $e_i \in \mathbb{C}_\Sigma$ such that e_i is obtained by replacing s_i by $\Box$ in $e[\![s]\!]$. By the induction assumption there is $q_i \in \delta_T^*(s_i)$ with $e_i \in \mathcal{C}_{q_i}$ for all i with $1 \leq i \leq n$. Since T is faithful for L and by Lemma 15 there is $q \in Q_T$ with $e \in \mathcal{C}_q$ and a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_T$, and thus the claim is shown. ■

Lemma 17 *For all $s \in Subt(L)$ we have $s \in L \Leftrightarrow \mathcal{R}_T(s) = 1$.*

Proof. "⇒": Let $s \in L$. Then the claim follows immediately from Lemma 16 since there must be $q \in \delta_T^*(s)$ with $\Box \in \mathcal{C}_q$ and thus $q \in F_T$.
"⇐": If $\mathcal{R}_T(s) = 1$ then there is a state $q \in \delta_T^*(s)$ with $q(\Box) = 1$ by the definition of F_T, and since $q \sqsubseteq row(s)$ by Corollary 2 this yields $obs(s, \Box) = 1$ and $s \in L$ by the definition of obs. ■

Lemma 18 establishes the connection between the inclusion relation of prime rows in the table and the sets of contexts defined by the corresponding states:

Lemma 18 *For all $q_1, q_2 \in Q_T$ we have $q_1 \sqsubseteq q_2 \Leftrightarrow \mathcal{C}_{q_1} \subseteq \mathcal{C}_{q_2}$.*

Proof: "⇒": Let $q_1 \sqsubseteq q_2$ and $e \in \mathcal{C}_{q_1}$. If $e = \Box$ then $q_1(\Box) = 1$ and, as $q_1 \sqsubseteq q_2$, we have $q_2(\Box) = 1$ as well and thus $q_2 \in F_T$ and $\Box \in \mathcal{C}_{q_2}$.
Otherwise, as E is exclusive for L, $q_1 \sqsubseteq q_2$ implies ${s_1}^{-1}L \subseteq {s_2}^{-1}L$ for all $s_1, s_2 \in S$ with $row(s_1) = q_1$ and $row(s_2) = q_2$. This entails $e \in {s_2}^{-1}L$ and we obtain $e \in \mathcal{C}_{q_2}$ by Lemma 15.
"⇐": Assume $\neg(q_1 \sqsubseteq q_2)$. By the definition of $\sqsubseteq$ there exists $e \in E$ with $q_1(e) = 1$ but $q_2(e) = 0$. We have $e \in \mathcal{C}_{q_1}$ by Lemma 15. For all $s \in Subt(L)$

with $row(s) = q_2$ we have $q_2 \in \delta_T^*(s)$ by Corollary 3. Since $\mathcal{R}_T$ cannot accept $e[\![s]\!]$ due to Lemma 17 we can conclude that $e \notin \mathcal{C}_{q_2}$, and hence e is an excluding context proving that $\mathcal{C}_{q_1} \not\subseteq \mathcal{C}_{q_2}$. ■

Remark Observe how Lemma 13 and Corollary 2 evoke Lemma 8, Lemma 14 and Corollary 3 evoke Lemma 11, Lemma 15 evokes Lemma 9, and Lemma 18 evokes Lemma 10. However, also note that in the cases of Corollaries 2 and 3 we could extend the claims from elements of S to elements of $Subt(L)$ thanks to the precondition that T is faithful for L. ♢

The inclusion $L \subseteq L_R$ is a direct consequence of Lemma 17. In order to show that the converse $L_R \subseteq L$ holds as well it remains to show that $\mathcal{R}_T$ does not accept any tree that is not a subtree of L.

Lemma 19 *For all $s \notin Subt(L)$ we have $\delta_T^*(s) = \emptyset$.*

Proof. Obviously, we have $row(s)(e) = 0$ for all $e \in E$.
If $s = a$ for some $a \in \Sigma_0$ then the claim follows directly from the definition of δ_T since there cannot be a prime row in $row(S)$ that is covered by $row(a)$. Let $s = f(s_1, \ldots, s_n)$ and assume the claim to hold for $s_1, \ldots, s_n$. If there is s_j with $s_j \notin Subt(L)$ for some $j \in \{1, \ldots, n\}$ then we have $\delta_T^*(s_j) = \emptyset$ by the induction assumption, and we obtain $\delta_T^*(s) = \emptyset$ directly by definition of δ_T^*. Assume $s_1, \ldots, s_n \in Subt(L)$. Clearly for any $f(s'_1, \ldots, s'_n) \in Subt(L)$ there is at least one index $k \in \{1, \ldots, n\}$ such that ${s'_k}^{-1}L \not\subseteq {s_k}^{-1}L$ because otherwise we would have $f(s_1, \ldots, s_n) \in Subt(L)$. Moreover, since E is exclusive for L, if the tree $f(s'_1, \ldots, s'_n)$ is in S then there is an excluding context $e \in E$ with $obs(s'_k, e) = 1$ and $obs(s_k, e) = 0$, i.e., we have $\neg(s'_k \sqsubseteq s_k)$ in T.
Let $q_1, \ldots, q_n \in Q_T$ with $q_i \in \delta_T^*(s_i)$ for $1 \leq i \leq n$. Then there are elements $t_1, \ldots, t_n \in S$ with $q_i = row(t_i)$ for $1 \leq i \leq n$. However, by the argument above $t = f(t_1, \ldots, t_n)$ cannot be a subtree of L, which implies $row(t)(e) = 0$ for all $e \in E$ and $\mathcal{P}_t = \emptyset$, and hence there is no suitable prime row $q \in Q_T$ that could be assigned as a state to s. Thus, $\delta_T^*(s) = \emptyset$. ■

The **Proof of Theorem 5** can now be concluded as follows.

We show that $\mathcal{R}_T$ is an RFTA by proving
$\mathcal{C}_{row(s)} = s^{-1}L_R$ for all $s \in S$ with $row(s) \in Q_T$:

"$\subseteq$": We have $row(s) \in \delta_T^*(s)$ by Corollary 3, which implies $\mathcal{C}_{row(s)} \subseteq s^{-1}L_R$.

"$\supseteq$": We have $q \sqsubseteq row(s)$ and $\mathcal{C}_q \subseteq \mathcal{C}_{row(s)}$ for all $q \in \delta_T^*(s)$ by Corollary 2 and Lemma 18, which yields $s^{-1}L \subseteq \mathcal{C}_{row(s)}$, and thus $\mathcal{C}_{row(s)} = s^{-1}L_R$.

Moreover, Lemma 12 implies that $s^{-1}L_R$ is a prime residual language of L, which in turn implies that $\mathcal{R}_T$ is a state-minimal RFTA. The fact that $L_R = L$

follows directly by Lemmata 17 and 19. Finally, as S is R-representative for L the RFTA $\mathcal{R}_T$ must be saturated, and thus $\mathcal{R}_T$ is isomorphic to $\mathcal{R}_L$. ■

Remark Considering the proof of Lemma 19 it is easy to see that as long as E is exclusive for L allowing S to contain trees $s \notin Subt(L)$ is not of consequence, i.e., that requiring a subset of S to be R-representative is enough (compare the remark in Subsection 3.2.2 right below Theorem 2). ♢

Prime-reduced RFTA

We briefly introduce and discuss yet another kind of FTA based on the concept of a residual language which is unique for every regular language $L \subseteq \mathbb{T}_\Sigma$. Consider the following two definitions:

Definition 43 *Let $\gamma \in \mathcal{C}_L$ be a residual language of L.*
The minimal access tree $t_\gamma \in \mathbb{T}_\Sigma$ *for γ is the tree such that*
$$t_\gamma{}^{-1}L = \gamma \text{ and there is no } t' \in \mathbb{T}_\Sigma \text{ with } t'^{-1}L = t_\gamma{}^{-1}L \text{ and } t' \preceq t_\gamma.$$
Let $\gamma_p \in \mathcal{P}_L$ be the (unambiguously defined) prime residual language of L such that there is no $\gamma' \in \mathcal{P}_L \setminus \{\gamma_p\}$ with $t_{\gamma_p} \preceq t_{\gamma'}$.

Definition 44
We define the prime-reduced FTA $\mathcal{R}_L^p := \langle \Sigma, Q_p, F_p, \delta_p \rangle$ *for L by*

- $Q_p := \{\gamma \in \mathcal{C}_L \mid t_\gamma \preceq t_{\gamma_p}\}$,
- $F_p := \{y \in Q_p \mid \square \in y\}$, *and*
- $\delta_p := \{\langle f, y_1 \cdots y_n \rangle \mapsto y \mid y_1, \ldots, y_n, y \in Q_p \wedge f \in \Sigma_n \wedge \exists t_1, \ldots, t_n \in \mathbb{T}_\Sigma : \forall i \in \{1, \ldots, n\} : y_i = t_i{}^{-1}L \wedge y \subseteq f(t_1, \ldots, t_n)^{-1}L\}$.

Obviously, $\mathcal{R}_L^p$ contains between I_L and $|\mathcal{P}_L|$ states, it is saturated due to the definition of δ_p, and it is unique. We observe that $\mathcal{R}_L^p$ is the tree equivalent of a special case of RFSA for strings defined in [42] that can be obtained from the canonical DFA for a regular string language L' by saturating it and then deleting all states with a minimal access string longer than the one for any prime residual language of L'. However, our definition takes the opposite perspective since it suggests that $\mathcal{R}_L^p$ can be obtained from the canonical RFTA $\mathcal{R}_L$ by adding states and transitions for all residual languages with a minimal access tree preceding the one for the prime γ_p with respect to $\preceq$.
As a consequence, we are able show that just like the canonical RFTA $\mathcal{R}_L$ $\mathcal{R}_L^p$ is an RFTA and recognizes L. The proofs of Lemmata 37–39, established in [20] in order to prove $\mathcal{L}(\mathcal{R}_L) = L$, are reproduced in Appendix A.3.

Theorem 6 *$\mathcal{R}_L^p$ is an RFTA recognizing L.*

First of all, observe that Lemma 13 and its proof hold in an unchanged form for $\mathcal{R}_L^p$ as well (only replace δ_L by δ_p and Q_L by Q_p). This is equally true for Lemma 37 if we consider that all prime residual languages of L are states in Q_p, and that more generally, any residual language is the union of *all* residual languages it contains. Lemmata 38 and 39 rely on Lemmata 13 and 37, and can thus be transferred accordingly. Hence, $\mathcal{R}_L^p$ recognizes L.

We define $T_p := \{t_\gamma \in \mathbb{T}_\Sigma \mid \gamma \in \mathcal{C}_L \wedge t_\gamma \preceq t_{\gamma_p}\}$.

Lemma 20 *$\mathcal{R}_L^p$ is an RFTA.*

Proof. Let $y \in Q_p$ and $t \in T_p$ with $t^{-1}L = y$. We have $t^{-1}L = \bigcup \delta_p^*(t)$ by Lemma 37. Since all residual languages of L that are defined by elements of T_p are states in Q_p and all transitions between them are represented in δ_p we have $t^{-1}L = y \in \delta_p^*(t)$ (also compare the proof of Lemma 1 in Subsection 3.2.2). Since $\mathcal{R}_p$ recognizes L we have $t^{-1}L = \bigcup\{\mathcal{C}_{y'} \subseteq \mathbb{C}_\Sigma \mid y' \in \delta_p^*(t)\}$, and hence, $\mathcal{C}_y \subseteq t^{-1}L$. On the other hand, for all $y' \in \delta_p^*(t)$ we have $y' \subseteq y$ by Lemma 13, and thus $\mathcal{C}_{y'} \subseteq \mathcal{C}_y$ by Lemma 39. Since the union of all those sets $\mathcal{C}_{y'}$ equals $t^{-1}L$ we obtain $t^{-1}L \subseteq \mathcal{C}_y$ and $t^{-1}L = \mathcal{C}_y$.
Thus, for each $y \in Q$ the set $\mathcal{C}_y$ corresponds to a residual language of L that is defined by some tree in T_p, and $\mathcal{R}_p$ is an RFTA. ■

This concludes the **Proof of Theorem 6**.

Algorithmic instantiations

In the next three subsections we are going to propose and study learning algorithms for the three settings that were favourable to the deterministic approach treated in Sections 3.2 and 3.3. We intend to use as many structural elements and subroutines of the meta-algorithm GENDET described in Subsection 3.3.1 as possible. As mentioned in the introduction of this section, an algorithm learning (string) RFSA from EQs and MQs has been presented in [17], and [42] contains an algorithm learning RFSA from a positive and a negative sample. We attempt to adapt those algorithms to the case of trees,[13] but before we do so we are going to provide and discuss an algorithm for the missing third constellation of importance, namely learning from MQs and a positive sample, in order to complete the picture.
For the three subsections to come, we fix a finite ranked alphabet Σ as the smallest alphabet such that the target tree language L fulfils $L \subseteq \mathbb{T}_\Sigma$.

3.4.4 The algorithm RESI

In this subsection we present a learning algorithm that tries to infer a regular tree language $L \subseteq \mathbb{T}_\Sigma$ from MQs and a finite positive sample of L and returns an FTA. If the given positive sample fulfils certain conditions (stated below) then the FTA is isomorphic to $\mathcal{R}_L$. RESI is of polynomial complexity due to a technique based on those used in [14, 46] for DFTA which we have adapted to and verified for the more intricate residual case by major modifications.

Remark The study of this setting and of a corresponding learning algorithm seems crucial in order to understand the construction of the canonical RFTA $\mathcal{R}_L$ for L and its algorithmic learnability. Therefore this subsection contains a considerable amount of semi-formal explanations. ♢

Thus, our learner has access to a perfect membership oracle and receives a finite positive sample $X_+ \subseteq L$. We use the input format of GENDET, and the technical remarks preceding the description of GENDET in Subsection 3.3.1 apply here as well. The pseudo-code of RESI is given below. Observe that the procedure UPDATE (called in lines 9 and 15) cannot modify the sets BLUE and WHITE due to the fact that they are initialized with the empty set in line 2. Also note that in this setting we do not supply Σ in an additional parameter since we depend on X_+ being representative for L.

[13] In [20] extensions of algorithms learning regular string languages from positive and negative data via RFSA as described for example in [42] to the tree case have been announced but to our knowledge no such algorithm has yet been published.

For a tree $f(t_1, \ldots, t_n) \in \mathbb{T}_\Sigma$ and $j \in \{1, \ldots, n\}$ we define $f(t_1, \ldots, t_n)_j^\square$ as a context $f(t'_1, \ldots, t'_n) \in \mathbb{C}_\Sigma$ with $t'_j = \square$ and $t'_i = t_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$.

Input: $IP = \langle 0, 1, X_+, \emptyset \rangle$. **Output:** An FTA.

1 $S := Subt(X_+)$; $E := Cont(X_+)$;
2 RED $:= S$; BLUE $:= \emptyset$; WHITE $:= \emptyset$;
3 **while** T is not R-consistent **do**
4 find $f(s_1, \ldots, s_n), f(s'_1, \ldots, s'_n) \in S$ and $e \in E$ such that
5 $\exists j \in \{1, \ldots, n\} : s_j \sqsubseteq s'_j \ \wedge\ \forall i \in \{1, \ldots, n\} \setminus \{j\} : s_i \approx s'_i \ \wedge$
6 $obs(f(s_1, \ldots, s_n), e) = 1 \ \wedge\ obs(f(s'_1, \ldots, s'_n), e) = 0 \ \wedge$
7 $\mathcal{O}(e[\![f(s_1, \ldots, s_n)_j^\square]\!][\![s_j]\!]) = 1 \ \wedge\ \mathcal{O}(e[\![f(s_1, \ldots, s_n)_j^\square]\!][\![s'_j]\!]) = 0$;
8 $E := E \cup \{e[\![f(s_1, \ldots, s_n)_j^\square]\!]\}$;
9 UPDATE;
10 **for** $s \in S$ **do**
11 **if** $\exists f(t_1, \ldots, t_n) \in S : \exists j \in \{1, \ldots, n\} : \exists e \in E :$
12 $t_j \approx s \ \wedge\ \mathcal{O}(e[\![f(t_1, \ldots, t_n)_j^\square]\!][\![s]\!]) = 1 \ \wedge$
13 $\forall s' \in S \setminus \{s\} : s' \sqsubseteq s \ \Rightarrow\ \mathcal{O}(e[\![f(t_1, \ldots, t_n)_j^\square]\!][\![s']\!]) = 0$ **then**
14 $E := E \cup \{e[\![f(t_1, \ldots, t_n)_j^\square]\!]\}$;
15 UPDATE;
16 **return** $\mathcal{R}_T$.

Assume the sample X_+ to be representative for L. The following lemma is essential for the learner's progress because its claim constitutes a necessary condition to ensure that as long as the table is not R-consistent we can derive an excluding context for two individual candidates directly from the table itself (i.e., that the search in line 4 will always succeed). It is based on a corresponding one from [14] for DFTA in the same setting but adapted to and proven for the more intricate residual case. Trivially note that since X_+ is representative for L the set $S = Subt(X_+)$ must be as well.

Lemma 21 *As long as T is not R-consistent, there are $f(s_1, \ldots, s_n), f(s'_1, \ldots, s'_n) \in S$ with $s_i \sqsubseteq s'_i$ for all i with $1 \leq i \leq n$ and $\neg(f(s_1, \ldots, s_n) \sqsubseteq f(s'_1, \ldots, s'_n))$ such that there is an index $j \in \{1, \ldots, n\}$ with ${s_j}^{-1}L \not\subseteq {s'_j}^{-1}L$ but ${s_k}^{-1}L = {s'_k}^{-1}L$ for all other $k \in \{1, \ldots, n\} \setminus \{j\}$.*

Proof. We prove this by a contradiction. Assume $e \in \mathbb{C}_\Sigma$ to be a context such that $e[\![s]\!] \in L$ and $e[\![s']\!] \notin L$ for some $s, s' \in S$ with $s \sqsubseteq s'$ but $s^{-1}L \not\subseteq {s'}^{-1}L$ (which must exist due to the fact that T is not R-consistent and Lemma 6) and the depth $cdp(e)$ to be minimal. The fact that e cannot be in E but $\square$ is implies $e \neq \square$ and thus there are contexts $e_1, e_2 \in \mathbb{C}_\Sigma$ with $e_1[\![e_2]\!] = e$ and $e_2 = f(s_1, \ldots, s_n)_j^\square$ for some trees $s_1, \ldots, s_n \in \mathbb{T}_\Sigma$, $f \in \Sigma_n$, and $n \geq 1$. Clearly, $cdp(e_1) < cdp(e)$. Since we require the depth of e to be minimal the

context e_1 cannot fulfil the role of an excluding context for any pair of trees $t, t' \in S$ with $t \sqsubseteq t'$ but $t^{-1}L \not\subseteq t'^{-1}L$.
As $e[\![s]\!] = e_1[\![e_2[\![s]\!]]\!]$ is in L we know that $e_2[\![s]\!]$ is in $Subt(L)$, and as X_+ is representative for L there is an element $t = f(t_1, \ldots, t_n) \in S$ with $t_i \equiv_L s_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ and $t_j \equiv_L s$ and $t \equiv_L e_2[\![s]\!]$. Since $t \in Subt(X_+)$ there is a context $e_3 \in Cont(X_+) \subseteq E$ such that $e_3[\![t]\!] \in X_+ \subseteq L$. Moreover, the context $e_3[\![e_4]\!]$ with $e_4 = f(t_1, \ldots, t_n)^{\Box}_j$ is also in $Cont(X_+) \subseteq E$. Obviously, $e_3[\![e_4[\![s]\!]]\!] \in L$ and $obs(s, e_3[\![e_4]\!]) = 1$. The precondition $s \sqsubseteq s'$ implies that we also have $obs(s', e_3[\![e_4]\!]) = 1$ and $e_3[\![e_4[\![s']\!]]\!] \in L$.
Note that $e_4[\![s']\!] \equiv_L e_2[\![s']\!]$ and hence $e_2[\![s']\!]$ must be a subtree of L as well. The fact that e is excluding for s and s' entails $e_2[\![s]\!]^{-1}L \not\subseteq e_2[\![s']\!]^{-1}L$. Again, as X_+ is representative for L there is $t' = f(t'_1, \ldots, t'_n) \in S$ with $t'_i \equiv_L t_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ and $t'_j \equiv_L s'$ and $t' \equiv_L e_2[\![s']\!]$. In contradiction to the claim, let us suppose $f(t_1, \ldots, t_n) \sqsubseteq f(t'_1, \ldots, t'_n)$. In that case the context e_1 would fulfil the role of an excluding context for the pair t, t' with $t \sqsubseteq t'$ but $t^{-1}L \not\subseteq t'^{-1}L$ since with $t \equiv_L e_2[\![s]\!]$ and $t' \equiv_L e_2[\![s']\!]$ we have $e_1[\![t]\!] \in L$ but $e_1[\![t']\!] \notin L$. However, this would violate the assumption that the depth of e fulfilling such a role be minimal and thus the claim is proven. ■

For the following explanations, recall that $\mathcal{E}_L$ is the set of all equivalence classes of L under $\equiv_L$, and let us define an *L-transition* as an expression $\epsilon := f(\chi_1, \ldots, \chi_n)$ with $f \in \Sigma_n$ for some $n \geq 1$, $\chi_j = \{\Box\}$ for some index $j \in \{1, \ldots, n\}$ and $\chi_i \in \mathcal{E}_L$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ such that there is $\chi \in \mathcal{E}_L$ with $\epsilon[\![\chi]\!] \subseteq Subt(L)$, where $\epsilon[\![\chi]\!]$ denotes the set

$$\{f(t_1, \ldots, t_n) \in \mathbb{T}_\Sigma \mid t_j \in \chi \wedge \forall i \in \{1, \ldots, n\} \setminus \{j\} : t_i \in \chi_i\}.$$

Observe that an L-transition can be naturally represented by a context that is obtained by instantiating χ_j by $\Box$ and each χ_i in ϵ for $i \in \{1, \ldots, n\} \setminus \{j\}$ with an arbitrary tree from χ_i.
The *application* of ϵ to a tree $s \in Subt(L)$ is the set $\epsilon[\![[s]_L]\!]$. The *behaviour* of $\epsilon[\![[s]_L]\!]$ with respect to a context $e \in \mathbb{C}_\Sigma$ is the truth value of $e[\![\epsilon[\![[s]_L]\!]]\!] \subseteq L$. We say that s has the L-transition ϵ (or that ϵ is an L-transition of s) if $\epsilon[\![[s]_L]\!] \subseteq Subt(L)$. In terms of the canonical DFTA $\mathcal{A}^\circ_L = \langle \Sigma, Q_\circ, F_\circ, \delta_\circ \rangle$ (without a failure state) this corresponds to the existence of a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_\circ$ with $q_j = [s]_L$ and $q_i = \chi_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$. The behaviour of $\epsilon[\![[s]_L]\!]$ with respect to e is the truth of $\delta^+_\circ(q, e) \cap F_\circ \neq \emptyset$.

Assume the sample X_+ to be representative for L. We will give some intuitive explanations for the termination and correctness of RESI and then establish a corresponding theorem.

Let us call a pair of trees $s, s' \in S$ *deceiving* if $s \sqsubseteq s'$ but $s^{-1}L \not\subseteq s'^{-1}L$. An R-inconsistency in the table caused by two elements $t = f(s_1, \ldots, s_n)$ and $t' = f(s'_1, \ldots, s'_n)$ in S with $s_i \sqsubseteq s'_i$ for all $i \in \{1, \ldots, n\}$ but $\neg(t \sqsubseteq t')$ implies by Lemma 6 that there must be at least one pair deceiving s_j, s'_j for some index $j \in \{1, \ldots, n\}$ for which there is an excluding context exactly of depth $d + 1$ where d is the depth of an arbitrary context $e' \in E$ fulfilling $obs(t, e') = 1$ and $obs(t', e') = 0$. However, the actual ability of RESI to construct such a context from the (finite!) table fundamentally relies on Lemma 21 which in turn relies on the sample X_+ being representative for L. Intuitively, Lemma 21 implies that for each "multiple" R-inconsistency, i.e., where there is more than one index $j \in \{1, \ldots, n\}$ such that ${s_j}^{-1}L \not\subseteq {s'_j}^{-1}L$, the table also features a "simple" R-inconsistency for each of the deceiving pairs in the sense that the latter involves only a single deceiving pair s, s' which moreover fulfils $s \equiv_L s_j$ and $s' \equiv_L s'_j$ and all other associated pairs of direct subtrees in that R-inconsistency are equivalent under $\equiv_L$.

Changing the point of view, such a simple R-inconsistency can be seen as a comparison of the behaviour with respect to the contexts in E when the *same* L-transition (represented by e_2 in the proof of Lemma 21) is applied to the members of a deceiving pair s_j, s'_j – with the outcome that there is a difference. Note that since X_+ is representative for L the finite set E contains a positive context for *each* L-transition of s_j. Also note that a learner taking the deterministic approach in settings with $MQ = 1$ (as given in [14, 46]) when trying to resolve an inconsistency revealed by two trees $t = f(s_1, \ldots, s_n)$ and $t' = f(s'_1, \ldots, s'_n)$ in S can rely on the fact that the results of applying *any* L-transition to two non-equivalent elements cannot show the same behaviour with respect to E such that the choice is arbitrary, and it can thus pick the ones represented by t and t' themselves. However, for two residual languages where one does not include the other the results of applying some, albeit not all, L-transitions may still feature behaviours where one subsumes the other (which is linked to the fact that RFTA are generally non-deterministic).

Hence, assuming we want to add at least one excluding context in each execution of the while-loop, when RESI finds two trees $t = f(s_1, \ldots, s_n)$ and $t' = f(s'_1, \ldots, s'_n)$ with $s_j \sqsubseteq s'_j$ for some $j \in \{1, \ldots, n\}$ and $s_i \approx s'_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ but $\neg(t \sqsubseteq t')$ it has to perform an additional check in order to verify whether there is a context $e \in E$ such that $e[\![f(s_1, \ldots, s_n)^{\square}_j]\!]$ is actually excluding for s_j, s'_j since it might also be the case that $f(s_1, \ldots, s_n)^{\square}_j$ does not represent a suitable L-transition due to the fact that there are other deceiving pairs – and for the same reason s_j, s'_j might not be deceiving at all. On the other hand, if the R-inconsistency retrieved by the checks in lines 5 and 6 is indeed a simple one then the condition in line 7 will be fulfilled as well

and RESI is sure to add an excluding context to E in line 8. Clearly, when all simple R-inconsistencies are resolved there cannot remain any multiple R-inconsistencies in the table either. Since the finite set S is never modified there is only a finite number of possible covering relations between rows in the table, and obviously by no addition to the elements of E can this number be increased. Thus, the termination of the while-loop is ensured.

However, the mere achievement of R-consistency in a table constructed from some positive sample of L would not suffice to ensure its correctness: First of all, if the set of contexts labeling the columns of the final table is to be exclusive for L then we have yet to verify that for *any* pair of trees $s, s' \in \mathit{Subt}(L)$ fulfilling $s^{-1}L \not\subseteq s'^{-1}L$ but $s \sqsubseteq s'$ in the initial table there is a corresponding R-inconsistency which will then be resolved at some point during the process. The truth of this statement relies on the condition that X_+ be representative for L as well and will be shown by induction over the depth of excluding contexts in the proof of Theorem 7 below.

Moreover, even if E is exclusive for L after the while-loop has been exited E may not be p-exclusive for L and hence the FTA derived from the table at that point may not be isomorphic to $\mathcal{R}_L$. Therefore we add an additional loop in lines 10–15 in order to retrieve all elements $s \in S$ for which we can find a suitable representative of an L-transition that has been applied to s in S and a matching positive context in E which when combined into another context and added to E cause the row of s to become prime in the table. Note that if E is exclusive for L then it is also separative for L which implies that we can translate the $\approx$-relation into the $\equiv_L$-relation and that thus any L-transition found in line 12 for some t_i with $t_i \approx s$ is an L-transition of s. If $s^{-1}L$ is indeed a prime residual language of L then s has an L-transition ϵ such that for all contexts $e \in \mathbb{C}_\Sigma$ with $e[\![\epsilon[\![[s]_L]\!]]\!] \subseteq L$ we have $e[\![\epsilon[\![[s']_L]\!]]\!] \not\subseteq L$ for all trees $s' \in \mathbb{T}_\Sigma$ with $s'^{-1}L \subsetneq s^{-1}L$. As X_+ is representative for L the set S contains a tree representing $\epsilon[\![[s]_L]\!]$, and E contains a positive context for that tree. As a consequence, RESI is sure to add enough contexts such that when the for-loop is exited there is an exact one-to-one correspondence between prime rows in the table and prime residual languages of L.

Theorem 7 *If X_+ is representative for L*
then RESI terminates and returns an FTA which is isomorphic to $\mathcal{R}_L$.

Proof. Assume that RESI performs m executions of the while-loop in total, and let $T_k = \langle S_k, E_k, \mathit{obs}\rangle$ be the table obtained after k executions for $k \geq 0$. Clearly we have $S_{k'-1} = S_{k'}$ and $E_{k'-1} \subsetneq E_{k'}$ for all k' with $1 \leq k' \leq m$. Note that the definition of the relation symbol $\sqsubseteq$ differs depending on the set E_k currently in question – we will write $\sqsubseteq_k$ for disambiguation.

When the while-loop terminates, (a) T_m is R-consistent due to the termination criterion, and (b) the set E_m is exclusive for L, i.e., for any $t, t' \in \mathbb{T}_\Sigma$ with $t^{-1}L \not\subseteq t'^{-1}L$ there is a context $e \in E_m$ with $obs(t, e) = 1$ but $obs(t', e) = 0$.

(b): First of all, observe that if $t \notin Subt(L)$ then there is no $t' \in \mathbb{T}_\Sigma$ such that $t^{-1}L \not\subseteq t'^{-1}L$. If $t' \notin Subt(L)$ then either $t \in Subt(L)$ or t cannot fulfil $t^{-1}L \not\subseteq t'^{-1}L$. In the former case, as X_+ is representative for L and as we have $X_+ \subseteq L$ there is $s \in S_0$ with $s \equiv_L t$ and $e \in Cont(X_+) = E_0$ with $obs(s, e) = obs(t, e) = 1$, and thus $\neg(t \sqsubseteq_m t')$ is ensured.

Let $t, t' \in Subt(L)$. As X_+ is representative for L there are $s, s' \in S_0$ with $s \equiv_L t$ and $s' \equiv_L t'$. We prove (b) by induction over excluding contexts.[14]

If $s \in L$ but $s' \notin L$ then the claim is true since we have $\Box \in Cont(X_+) = E_0$. If the canonical DFTA $\mathcal{A}_L^\circ$ contains a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_\circ$ with $q_j = [s]_L$ for some $j \in \{1, \ldots, n\}$ but no $\langle f, q'_1 \cdots q'_n \rangle \mapsto q' \in \delta_\circ$ with $q'_j = [s']_L$ then the claim holds as well due to the fact that X_+ is representative for L and $X_+ \subseteq L$ which implies the existence of a context $e \in Cont(X_+) = E_0$ and $e_1, e_2 \in \mathbb{C}_\Sigma$ with $e = e_1[\![e_2]\!]$ and $e_2 = f(s_1, \ldots, s_n)_j^\Box$ and $[s_i]_L = q_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ such that $obs(s, e) = 1$ and $obs(s', e) = 0$.

Remark Intuitively stated, if two elements do not share any L-transition featuring the same root symbol then they cannot be deceiving a priori. $\diamondsuit$

For the remaining cases, let $e_s \in \mathbb{C}_\Sigma$ be an excluding context for s and s' with $cdp(e_s) = d + 1$ for some $d \geq 0$, and choose $k \leq m$ such that we already have $\neg(s_k \sqsubseteq_k s'_k)$ for all $s_k, s'_k \in S_k$ with ${s_k}^{-1}L \not\subseteq {s'_k}^{-1}L$ for which there is an excluding context $e_k \in \mathbb{C}_\Sigma$ with $cdp(e_k) \leq d$. Moreover, assume $s \sqsubseteq_k s'$.
Let there be a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_\circ$ with $q_j = [s]_L$ for some $j \in \{1, \ldots, n\}$. Since we have $s \sqsubseteq_k s'$ there is a context $e \in Cont(X_+) \subseteq E_k$ and $e_1, e_2 \in \mathbb{C}_\Sigma$ such that $e = e_1[\![e_2]\!]$ and $e_2 = f(s_1, \ldots, s_n)_j^\Box$ with $q_i = [s_i]_L$ for $i \in \{1, \ldots, n\} \setminus \{j\}$ fulfilling $obs(s, e) = obs(s', e) = 1$. With $Cont(X_+) \subseteq E_k$ we also have $e_1 \in E_k$. Since e is a positive context both for s and s' there are $f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n) \in S_k$ with $t_j \equiv_L s$ and $t'_j \equiv_L s'$ and $t_i \equiv_L s_i \equiv_L t'_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$. We have $\neg(f(t_1, \ldots, t_n) \sqsubseteq f(t'_1, \ldots, t'_n))$ by the induction assumption which implies that T_k must be R-inconsistent, i.e., there must be $e' \in E_k$ with $obs(f(t_1, \ldots, t_n), e') = 1$ but $obs(f(t'_1, \ldots, t'_n), e') = 0$.

Hence, as long as there is a pair $t, t' \in Subt(L)$ with $t^{-1}L \not\subseteq t'^{-1}L$ but $t \sqsubseteq t'$ the table cannot be R-consistent. By Lemma 21, as long as the table is not R-consistent one can derive a context from it that eliminates a covering relation between two rows of S. RESI will find such a context and add it to E.

[14] The basic idea of this induction is inspired by the original one in [3] for the simpler case of deterministic finite-state string automata in the same setting (with X_+ representative).

Since S is finite there can be only a finite number of such pairs and thus the while-loop terminates. As X_+ is representative for L and $S_m = \mathit{Subt}(X_+)$, on exiting the while-loop E_m must be exclusive for all of L.

Lines 10–15: For every $t \in \mathbb{T}_\Sigma$ with $t^{-1}L \in \mathcal{P}_L$ the canonical DFTA $\mathcal{A}_L^\circ$ features a transition $\langle f, q_1 \cdots q_n \rangle \mapsto q \in \delta_\circ$ with $q_j = [t]_L$ for some $j \in \{1, \ldots, n\}$ such that for no $t' \in \mathbb{T}_\Sigma$ with $t'^{-1}L \subsetneq t^{-1}L$ there is $\langle f, q'_1 \cdots q'_n \rangle \mapsto q' \in \delta_\circ$ with $q_j = [t']_L$ and $q'_i = q_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$.
As X_+ is representative for L there is $f(s_1, \ldots, s_n) \in S$ with $s_j \equiv_L t$ and $s_i \in q_i$ for $i \in \{1, \ldots, n\} \setminus \{j\}$, and there is $e \in E$ with $\mathit{obs}(f(s_1, \ldots, s_n), e) = 1$. Obviously, the context $e[\![f(s_1, \ldots, s_n)^{\Box}_j]\!]$ is such that it causes the row of s to be prime when included in E. As E_m is exclusive for L in the present table the $\equiv_L$-relation and the $\approx$-relation coincide and RESI is able to construct such a context and add it to E. Therefore, the final set E is also p-exclusive for L and we obtain $s \in \mathcal{P}_S \Leftrightarrow s^{-1}L \in \mathcal{P}_L$ for all elements $s \in S$.
The claim of Theorem 7 follows directly from Theorem 5. ■

Query complexity of RESI

Naturally, the complexity of RESI crucially depends on the given sample X_+. Let $m_+ := \sum_{t \in X_+} |t|$ be the size of all trees in X_+ added up, which is also the maximal cardinality of $\mathit{Subt}(X_+)$ and of $\mathit{Cont}(X_+)$. Let Σ' be the smallest alphabet such that $X_+ \subseteq \mathbb{T}_{\Sigma'}$, and let ρ be the maximal rank in Σ' such that $\Sigma'_\rho \neq \emptyset$. Observe that ρ is bounded by m_+. Assume the sample X_+ to be representative for L. Then Σ' equals the smallest alphabet Σ such that $L \subseteq \mathbb{T}_\Sigma$ and the index I_L is bounded by ρ and thus by m_+ as well.

The initial table can contain at most m_+ elements in S and at most m_+ elements in E. In a worst case we may have to eliminate the covering relation between the rows of each pair of trees in S, and thus the number of while-loop executions needed to make E exclusive for L is bounded by ${m_+}^2$. In each execution of the while-loop a single context is added to E. In order to make E also p-exclusive for L we may have to extend E by another set of contexts whose cardinality is bounded by m_+. We assume that the results of MQs are stored so that no query has to be asked twice. This implies that the number of MQs needed to successively fill the final table is bounded by $m_+(m_+ + m_+^2 + m_+) = 2m_+^2 + m_+^3$.

Moreover, we ask two additional MQs in line 7. The objects of those queries are constructed using two different trees from S and a context from E, and since during the while-loop the size of E is bounded by $m_+ + {m_+}^2$ the overall number of additional MQs caused by that loop is bounded by

$2({m_+}^3 + {m_+}^4)$. Furthermore, another two MQs are asked in lines 12–13. During the for-loop the size of E is bounded by $2m_+ + {m_+}^2$, and thus the overall number of additional MQs caused by the executions of that loop is bounded by ${m_+}^2(2m_+ + {m_+}^2)(m_+ + 1) = {m_+}^5 + 3{m_+}^4 + 2{m_+}^2$.

Discussion

Although the query complexity of RESI as discussed above is still polynomial one might remark that in comparison to a learner taking the deterministic approach (as given in [14]) RESI asks a disconcerting amount of MQs. This is mostly due to the additional intricacy of the residual approach.

The deterministic learner in [14] searches the table for an inconsistency involving two trees $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ in S with $s_i \approx s'_i$ for all $i \in \{1, \ldots, n\}$ but $s <> s'$ and a (single!) distinguishing context $e \in E$ for s and s', and adds all contexts $e[\![f(s_1, \ldots, s_n)_i^{\Box}]\!]$ for $i \in \{1, \ldots, n\}$ to E without asking further MQs. If X_+ is representative for L then this learner terminates successfully after at most I_L loop executions, and the number of contexts added in each execution depends only linearly on the rank ρ.
Assume that we naively try to avoid the additional MQs caused by executions of the two loops in RESI via the following strategy: For each R-inconsistency in the initial table involving two trees $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ in S with $s_j \sqsubseteq s'_j$ for some $j \in \{1, \ldots, n\}$ and $s_i \approx s'_i$ for all $i \in \{1, \ldots, n\} \backslash \{j\}$ but $\neg(s \sqsubseteq s')$ we simply add the set

$$\{e[\![f(s_1, \ldots, s_n)_j^{\Box}]\!] \mid e \in E \wedge obs(s, e) = 1 \wedge obs(s', e) = 0\}$$

to E. However, it is easy to see that this would not yield the desired reduction since we merely shift the entire query complexity towards the procedure filling in the cells of the table, and we still have to check every element of E.

The situation changes if we know for sure that in addition to being representative for L the sample X_+ fulfils certain favourable conditions. If we were able to identify simple R-inconsistencies in T then any excluding context e for $f(s_1, \ldots, s_n)$ and $f(s'_1, \ldots, s'_n)$ is suitable in order to construct an excluding context for the only deceiving pair of subtrees s_j, s'_j (see the explanations preceding Theorem 7), and we can pick the first such context that we find. As it is, our learner cannot reliably check equivalence under $\equiv_L$.
We define $mSubt(X_+) := \{s \in Subt(X_+) \mid \forall s' \in Subt(X_+) \cap [s]_L : s \preceq s'\}$ as in Subsection 3.3.2 and we formulate a condition

$$\text{(A) } \Sigma(mSubt(X_+)) \cap Subt(L) \subseteq Subt(X_+).$$

This condition ensures that if two non-equivalent elements $t, t' \in \mathbb{T}_\Sigma$ share an L-transition then the application of that L-transition to each of those trees is represented in S using the *same* subtrees at least once, and thus for each

deceiving pair s_j, s'_j in the table we can find a pair of trees in S representing the results of applying the exact same L-transition to s_j and s'_j, respectively. In other words, we can find a simple R-inconsistency where the non-deceiving pairs are not only equivalent but represented by the same trees.

We can then define a modified version of RESI that relies on condition (A) and searches for two trees $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ in S with $s_j \sqsubseteq s'_j$ for some j and $s_i = s'_i$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ but $\neg(s \sqsubseteq s')$. This obviously represents a simple R-inconsistency and we can pick an arbitrary excluding context for s and s' in order to construct an excluding context for s_j and s'_j without having to ask any additional MQs in the while-loop. Moreover, we do not need additional MQs in the for-loop in order to identify and mark the representatives of prime residual languages of L either: Define $mr(S) := \{s \in S \mid \forall s' \in S : s' \approx s \Rightarrow s \preceq s'\}$. Since we rely on E being separative for L after the while-loop has been exited we assume that the $\approx$-relation and the $\equiv_L$-relation coincide and that hence the set $mr(S)$ contains exactly one minimal access tree for each equivalence class $\chi \in \mathcal{E}_L$ with $\chi \subseteq Subt(L)$, and that each element of $mr(S)$ is a minimal access tree for some equivalence class of L. Since we also rely on condition (A), for each $s \in mr(S)$ we look for a concrete tree $e'[\![s]\!]$ as a representative for the application of some L-transition ϵ to s and a positive context e for $e'[\![s]\!]$ such that for all other $s' \in mr(S)$ with $s' \sqsubseteq s$ the function obs either yields $*$ for $e'[\![s']\!]$, which due to our conditions implies that s' does not have the L-transition ϵ at all, or 0, which implies that e is not a positive context for any representative of $\epsilon[\![s']\!]$. If s is prime then such an L-transition exists and the table contains a suitable tree $e'[\![s]\!]$ and a positive context e for $e'[\![s]\!]$, and obviously the addition of the context $e[\![e']\!]$ to E causes the row of s to become prime in the table.

---

3 **while** T `is not R-consistent` **do**
4 `find` $f(s_1, \ldots, s_n), f(s'_1, \ldots, s'_n) \in S$ `and` $e \in E$ `such that`
5' $\exists j \in \{1, \ldots, n\} : s_j \sqsubseteq s'_j \ \wedge\ \forall i \in \{1, \ldots, n\} \setminus \{j\} : s_i = s'_i \ \wedge$
6 $obs(f(s_1, \ldots, s_n), e) = 1 \ \wedge\ obs(f(s'_1, \ldots, s'_n), e) = 0$;
8 $E := E \cup \{e[\![f(s_1, \ldots, s_n)^{\square}_j]\!]\}$;
9 `UPDATE`;
10' **for** $s \in mr(S)$ **do**
11' **if** $\exists f(t_1, \ldots, t_n) \in mr(S) \cup (\Sigma(mr(S)) \cap S) : \exists j \in \{1, \ldots, n\} : \exists e \in E :$
12' $t_j = s \ \wedge\ obs(f(t_1, \ldots, t_n), e) = 1 \ \wedge$
13' $\forall s' \in mr(S) \setminus \{s\} : s' \sqsubseteq s \ \Rightarrow\ obs(f(t_1, \ldots, t_n)^{\square}_j[\![s']\!], e) \neq 1$ **then**
14 $E := E \cup \{e[\![f(t_1, \ldots, t_n)^{\square}_j]\!]\}$;
15 `UPDATE`;
16 **return** $\mathcal{R}_T$.

The maximal number of MQs is thus reduced to $2{n_+}^2 + {n_+}^3$ due to the fact that we were able to eliminate several search processes depending on the maximal rank ρ and on the size of E, both of which depend in turn on n_+.

Remark Condition (A) resembles Condition (A1) in Subsection 3.3.2 but we do not need the second line as the learner has access to a membership oracle. In both cases these conditions ensure that transitions are marked using certain "canonical representatives" of equivalence classes of L in the given set of positive data, thus anticipating some important aspect of the learner's strategy – in the RPNI case the fact that the learner processes the candidates in a certain order and in the case of our modifed version of RESI the fact that it relies on the existence of *obviously* simple R-inconsistencies. ♢

3.4.5 The algorithm RRPNI

In this subsection we present a learning algorithm that tries to infer a regular tree language $L \subseteq \mathbb{T}_\Sigma$ from a positive and a negative finite sample of L and returns an FTA. If the pair of given samples fulfils certain conditions (stated below) then the returned FTA is isomorphic to $\mathcal{R}_L$. Again, we use the input format of GENDET, and the technical remarks preceding the description of GENDET in Subsection 3.3.1 apply as well. As in the previous subsection (and for the same reason) we do not provide a parameter for the alphabet Σ.

Input: $IP = \langle 0, 0, X_+, X_- \rangle$. **Output:** An FTA.

```
1    RINIT;
2    RNEXTSTATE;
3    return R_T.
```

procedure RINIT

```
4    O := ⟨Σ, ∅, ∅, ∅⟩;
5    RED := Subt(X+);
6    BLUE := ∅;
7    E := Cont(X+);
8    WHITE := ∅.
```

The oracle and the components of the table are initialized by the procedure RINIT. For the residual approach we construct the oracle using an adaptation of an algorithm for strings from [42] to the tree case which entails that, in contrast to the deterministic learner for this setting, we start out with the empty automaton instead of $STA(X_+)$. Hence, the FTA $\mathcal{O}$ is initialized accordingly. The sets RED, BLUE, E, and WHITE are initialized as for the algorithm RESI described in the previous subsection, thus immediately moving all available

candidates to RED. The actual learning process is carried out in line 2 by a procedure RNEXTSTATE, which is given below.

We assume $S = \text{RED} = \{s_1, \ldots, s_m\}$ with $m := |Subt(X_+)|$ to be ordered with respect to the relation $\preceq$ (see Subsection 3.3.1). We write $s \lhd s'$ for two trees $s, s' \in \mathbb{T}_\Sigma$ if there is no context $e \in \mathbb{C}_\Sigma$ such that $e[\![s]\!] \in X_+$ and $e[\![s']\!] \in X_-$, and we write $s \bowtie s'$ if $s \lhd s'$ and $s' \lhd s$. (Compare this to the definition of $\sqsubseteq$ where for two trees $s, s' \in S$ we cannot find an excluding context in E.)
The procedure RNEXTSTATE picks the next element s_i from S with respect to $\preceq$. If we can find an element $s \in \text{RED}$ such that the given samples do not disprove the equivalence $s_i \equiv_L s$ (i.e., if we have $s_i \bowtie s$) then we assume that the equivalence class and thus the residual language defined by s_i is already represented in RED by s. In that case we delete s_i and all trees containing s_i as a subtree from the pool of candidates (note that the subtree-closedness of S stays preserved). If there is no such element for s_i in RED then s_i is added as a new state to the oracle $\mathcal{O}$ via the procedure ADDSTATE. If s_i is in X_+ then that state is stored as accepting, and the transition function $\delta_\mathcal{O}$ is updated in accordance with the interrelations between s_i and the remaining elements in S with respect to $\lhd$. Observe how the construction of $\delta_\mathcal{O}$ (as determined by line 17 in ADDSTATE) emulates the definition of δ_T in $\mathcal{R}_T$ (see Subsection 3.4.2) – however, since at this stage the cells of the table are entirely void of information, instead of relying on the $\sqsubseteq$-relation we base $\delta_\mathcal{O}$ on the $\lhd$-relation which implies that at least the given samples do not disprove an inclusion between the residual languages that are defined by two elements thus compared. If the resulting oracle is consistent with the given samples (which is tested by the procedure COMPSAMP in line 12) then the termination criterion is set to 1 and we delete all elements greater than s_i in order to minimize the set of candidates but restore those which contribute transitions from elements that were already processed and not deleted. Since we have $\tau = 1$ the subsequent call of UPDATE will consult the oracle in order to fill in all cells of the table.

```
procedure RNEXTSTATE
9     for i = 1,...,m do
10        if ∃s ∈ RED : s_i ⋈ s then RED := RED \ {s' ∈ RED | s_i ∈ Subt(s')};
11        else ADDSTATE(O, s_i);
12        if COMPSAMP(O) then τ := 1;
13           RED := (RED \ {s_j ∈ RED | j > i}) ∪
                                   (Σ({s_k ∈ RED | k ≤ i}) ∩ Subt(X_+));
14           UPDATE.
```

procedure ADDSTATE($\mathcal{A}, t$) [$\mathcal{A} = \langle \Sigma, Q, F, \delta \rangle$, $t \in \mathbb{T}_\Sigma$]
15 $Q := Q \cup \{t\}$;
16 **if** $t \in X_+$ **then** $F := F \cup \{t\}$;
17 $\delta := \delta \cup \{\langle f, t_1 \cdots t_n \rangle \mapsto t \mid t_1, \ldots, t_n \in Q \ \wedge$
$f(t_1, \ldots, t_n) \in S \ \wedge \ t \lhd f(t_1, \ldots, t_n)\} \ \cup$
$\{\langle f, t_1 \cdots t_n \rangle \mapsto t' \mid t_1, \ldots, t_n, t' \in Q \ \wedge \ \exists i \in \{1, \ldots, n\} : t_i = t \ \wedge$
$f(t_1, \ldots, t_n) \in S \ \wedge \ t' \lhd f(t_1, \ldots, t_n)\}$.

procedure COMPSAMP($\mathcal{A}$)
18 **if** $\forall t \in X_+ : \mathcal{A}(t) = 1 \ \wedge \ \forall t' \in X_- : \mathcal{A}(t') \neq 1$ **then return** `true`;
19 **else return** `false`.

As stated above, the pair of given samples has to fulfil certain properties in order to make the learner identify L correctly. Recall that t_γ denotes the minimal access tree for a residual language $\gamma \in \mathcal{C}_L$, and that $\gamma_p \in \mathcal{P}_L$ is the unique prime residual language such that t_{γ_p} is maximal with respect to $\preceq$. Define the set $T_p := \{t \in \mathbb{T}_\Sigma \mid \exists \gamma \in \mathcal{C}_L : t = t_\gamma \wedge t_\gamma \preceq t_{\gamma_p}\}$, and assume the given pair $\langle X_+, X_- \rangle$ to fulfil the three conditions (C1)–(C3) as listed below:

(C1) $T_p \cup (\Sigma(T_p) \cap \mathit{Subt}(L)) \subseteq \mathit{Subt}(X_+)$ and $T_p \cap L \subseteq X_+$,

(C2) $\forall t_1 \in T_p : \forall t_2 \in T_p \cup (\Sigma(T_p) \cap \mathit{Subt}(L)) :$
$t_1{}^{-1}L \not\subseteq t_2{}^{-1}L \Rightarrow \exists e \in \mathbb{C}_\Sigma : e[\![t_1]\!] \in X_+ \wedge e[\![t_2]\!] \in X_-$, and

(C3) $\forall \gamma \in \mathcal{P}_L : \exists t \in T_p : \exists e \in \mathbb{C}_\Sigma : t^{-1}L = \gamma \wedge e[\![t]\!] \in X_+ \wedge$
$\forall \gamma' \in \mathcal{C}_L : \gamma' \subsetneq \gamma \Rightarrow e \notin \gamma'$.

Observe that condition (C1) causes $\mathit{Subt}(X_+)$ to be R-representative for L, condition (C2) causes $\mathit{Cont}(X_+)$ to be R-exclusive for L, and condition (C3) causes $\mathit{Cont}(X_+)$ to be p-exclusive for L (see Subsection 3.4.3). In addition, conditions (C1)–(C3) ensure that all residual languages of L with a minimal access tree preceding t_{γ_p} with respect to $\preceq$ and all transitions between them be represented using elements from T_p, and that those representatives appear explicitly together with significant contexts in the given samples.

Remark Conditions (C1)–(C3) can be compared to Conditions (A1) and (A2) in Subsection 3.3.2 above, and fulfil a similar purpose by anticipating the learner's strategy of processing the candidates in a certain order – also see the remark at the end of Subsection 3.4.4, and the discussion below. $\diamondsuit$

Theorem 8 *When RRPNI terminates,*

(a) the oracle $\mathcal{O}$ is isomorphic to $\mathcal{R}_L^p$ (and thus recognizes L), and

(b) $\mathcal{R}_T$ is isomorphic to $\mathcal{R}_L$.

Before we show this we are going to prove two lemmata (of which the first can be compared to Lemma 13 and Corollary 2 and the second to the "$\Leftarrow$"-direction in Lemma 9) concerning the oracle which hold at any time during the process up to and including the point when the minimal access tree t_{γ_p} for γ_p is processed. First of all, observe that any state in $Q_\mathcal{O}$ is also an element of the set T_p since the minimal access tree for any residual language $\gamma \in \mathcal{C}_L$ is processed before any other access tree for γ, and when a non-minimal access tree for some $\gamma' \in \mathcal{C}_L$ is processed it is deleted due to its apparent equivalence to the previously processed tree $t_{\gamma'}$. Moreover, as soon as the minimal access tree for any residual language $\gamma \in \mathcal{C}_L$ is processed it must be established as a state of the oracle since condition (C2) causes the test in line 10 to fail. Hence, we have $Q_\mathcal{O} \subseteq T_p \subseteq S$ at any time due to condition (C1).

Lemma 22 *For each $t \in Subt(L)$ and all $t' \in \delta_\mathcal{O}^*(t)$ we have $t'^{-1}L \subseteq t^{-1}L$.*

Proof. This is shown by induction over the depth of t.
If $t = a$ for some $a \in \Sigma_0$ then the claim follows directly from the definition of $\delta_\mathcal{O}$ since a must be an element of S due to condition (C1).
Let $t = f(t_1, \ldots, t_n)$. Then, by the definition of $\delta_\mathcal{O}^*$,
$\delta_\mathcal{O}^*(t) = \{t' \in Q_\mathcal{O} \mid \exists \langle f, t'_1 \cdots t'_n \rangle \mapsto t' \in \delta_\mathcal{O} : \forall i \in \{1, \ldots, n\} : t'_i \in \delta_\mathcal{O}^*(t_i)\}$.
Let $t' \in \delta_\mathcal{O}^*(t)$ and assume the claim to hold for $t_1, \ldots, t_n$. By the definition of $\delta_\mathcal{O}$ there are states $t'_1, \ldots, t'_n \in Q_\mathcal{O}$ with $t'_i \in \delta_\mathcal{O}^*(t_i)$ for $1 \leq i \leq n$ such that $s = f(t'_1, \ldots, t'_n) \in S$ and $t' \lhd s$. By the induction assumption we have $t'^{-1}_i L \subseteq {t_i}^{-1} L$ for all $i \in \{1, \ldots, n\}$. This yields $s^{-1}L \subseteq t^{-1}L$ by Lemma 6. By condition (C2), $t' \lhd s$ implies $t'^{-1}L \subseteq s^{-1}L$, and thus $t'^{-1}L \subseteq t^{-1}L$. ■

Lemma 23 *For each $e \in Cont(X_+)$ and all $s \in Q_\mathcal{O}$, $e[\![s]\!] \notin L$ implies $e \notin \mathcal{C}_s$.*

Proof. Let $s \in Q_\mathcal{O}$. We prove this by induction over the depth of e.

For $e = \Box$ we have $s \in X_+$ due to condition (C1), and
we have $s \in X_+ \Leftrightarrow s \in F_\mathcal{O} \Leftrightarrow \Box \in \mathcal{C}_s$, and the claim is shown.

Let $e = e'[\![e'']\!]$ be such that $e', e'' \in \mathbb{C}_\Sigma$ and $e'' = f(t_1, \ldots, t_n)$ with $t_j = \Box$ for some $j \in \{1, \ldots, n\}$. Note that e' must be an element of $Cont(X_+)$ as well. Assume the claim to hold for e'.
Let $e[\![s]\!] \notin L$. Choose states $s_1, \ldots, s_n \in Q_\mathcal{O}$ with $s_j = s$ and $s_i \in \delta_\mathcal{O}^*(t_i)$ for $i \in \{1, \ldots, n\} \setminus \{j\}$. If there is no element $f(s_1, \ldots, s_n) \in S$ then there is

no transition $\langle f, s_1 \cdots s_n \rangle \mapsto s' \in \delta_{\mathcal{O}}$, and clearly $\mathcal{O}$ cannot accept $e[\![s]\!]$ given the chosen state assignment. Assume that we have $f(s_1, \ldots, s_n) \in S$. Then $s_i^{-1}L \subseteq t_i^{-1}L$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ due to Lemma 22. Moreover, we have $f(s_1, \ldots, s_n)^{-1}L \subseteq e''[\![s]\!]^{-1}L$ due to Lemma 6. If there is no element $s' \in Q_{\mathcal{O}}$ with $s' \lhd f(s_1, \ldots, s_n)$ then there is no transition $\langle f, s_1 \cdots s_n \rangle \mapsto s' \in \delta_{\mathcal{O}}$ and $\mathcal{O}$ cannot accept $e[\![s]\!]$ via this choice of states either. If there is $s' \in Q_{\mathcal{O}}$ with $s' \lhd f(s_1, \ldots, s_n)$ then there is a transition $\langle f, s_1 \cdots s_n \rangle \mapsto s' \in \delta_{\mathcal{O}}$. Condition (C2) yields $s'^{-1}L \subseteq f(s_1, \ldots, s_n)^{-1}L$, and thus $s'^{-1}L \subseteq e''[\![s]\!]^{-1}L$. Obviously, $e[\![s]\!] = e'[\![e''[\![s]\!]]\!] \notin L$ entails $e'[\![s']\!] \notin L$. We obtain $e' \notin \mathcal{C}_{s'}$ by the induction assumption. Since $s_1, \ldots, s_n$ were chosen arbitrarily except for $s_j = s$ we can deduce $e'[\![s']\!] \notin L$ and $e' \notin \mathcal{C}_{s'}$ for all states s' that are reachable from s by a transition featuring the symbol f, and consequently $e \notin \mathcal{C}_s$. ■

We are now able to give the **Proof of Theorem 8** as follows.

(a): We first show that as long as the minimal access tree t_{γ_p} for γ_p has not been processed the oracle $\mathcal{O}$ cannot be consistent with the given data.
We have $t_{\gamma_p} \in \mathit{Subt}(X_+)$ and there is $e_p \in \mathit{Cont}(X_+)$ such that $e_p \notin \gamma'$ for all $\gamma' \in \mathcal{C}_L$ with $\gamma' \subsetneq \gamma_p$ and $e_p[\![t_{\gamma_p}]\!] \in X_+$ due to conditions (C1) and (C3). We have $s^{-1}L \subseteq {t_{\gamma_p}}^{-1}L = \gamma_p$ for all $s \in \delta_{\mathcal{O}}^*(t_{\gamma_p})$ by Lemma 22. If t_{γ_p} has not been processed yet then this implies that we also have $s^{-1}L \subsetneq {t_{\gamma_p}}^{-1}L = \gamma_p$ and $e_p \notin s^{-1}L$ for all $s \in \delta_{\mathcal{O}}^*(t_{\gamma_p})$. Moreover, we have $e_p \notin \mathcal{C}_s$ by Lemma 23, and hence $\mathcal{O}(e_p[\![t_{\gamma_p}]\!]) \neq 1$, i.e., $\mathcal{O}$ wrongly classifies $e_p[\![t_{\gamma_p}]\!] \in X_+ \subseteq L$.
On the other hand, as soon as the tree t_{γ_p} has been processed we have $S = T_p \cup (\Sigma(T_p) \cap \mathit{Subt}(L))$ due to condition (C1), to the arguments preceding Lemma 22, and to line 13. We also have $Q_{\mathcal{O}} = T_p$ due to the same arguments and lines 10 and 11, and $\mathcal{O}$ must be isomorphic to $\mathcal{R}_L^p$ due to the respective definitions of $\delta_{\mathcal{O}}$ and δ_p and to conditions (C1) and (C2) which entail that the $\lhd$-relation between the elements of S (relevant for the definition of $\delta_{\mathcal{O}}$) and the subset relation between the residual languages of L defined by those elements (relevant for the definition of δ_p) coincide.

(b): We show that $\mathcal{R}_T$ is isomorphic to $\mathcal{R}_L$.
The final table T is filled in completely and correctly because on termination $\mathcal{O}$ is a perfect oracle for L. As we have $S = T_p \cup (\Sigma(T_p) \cap \mathit{Subt}(L))$ the set S is R-representative for L. The set E is p-exclusive for L due to condition (C3). Observe that due to condition (C2) and to the fact that all non-members of $T_p \cup (\Sigma(T_p) \cap \mathit{Subt}(L))$ were deleted from S the set E is also exclusive for L with respect to all elements in S at least, and we can apply Theorem 5. ■

Complexity and discussion

Naturally, the complexity of RRPNI essentially depends on the size of the given samples as well. Let $m_+ := \sum_{t \in X_+} |t|$ and $m_- := \sum_{t \in X_-} |t|$ be the size of all trees in X_+ and X_- added up, respectively, and let ρ be the maximal rank in Σ such that $\Sigma_\rho \neq \emptyset$. The oracle we are developing can have at most m_+ states and $m_r := |\Sigma| {m_+}^{\rho+1}$ transitions. Testing if the FTA developed so far (wrongly) accepts an element of X_- takes a number of steps bounded by $O(m_r m_- \rho)$ (see [93]), and it is easy to see that the procedures of checking two elements of S with respect to the relation $\bowtie$ (line 10) and the construction of the transition function in ADDSTATE based on the $\lhd$-relation (line 17) are of a polynomial complexity with respect to m_+ and m_- as well. In addition, UPDATE uses the oracle once to fill a table of a size which is bounded by ${m_+}^2$ since its rows are labeled with candidates from $Subt(X_+)$ and the columns with contexts from $Cont(X_+)$. Thus, polynomial complexity with respect to the size of the given input as defined above is ensured.

Note that we must rely on the elements of $T_p \cup (\Sigma(T_p) \cap Subt(L))$ appearing with significant contexts explicitly because RRPNI does not have access to a membership oracle. Conditions (C1)–(C3) may seem rather strong but at least we do not require X_+ to be representative for *all* equivalence classes and transitions of L nor to be exclusive for *all* residual languages of L, and thus in favourable cases the given positive sample may be exponentially smaller than the one needed by RPNI (and RESI) to ensure correct identification, and when comparing condition (C2) to (A2) as required for RPNI in Subsection 3.3.2 one can see that the same is true for the negative sample. Also note that in favourable cases we can choose our data such that condition (C3) does not increase the size of a minimal sample which is due to the fact that a single context can be excluding for several residual languages and/or their unions. In [42] the authors let their algorithm derive the prime-reduced RFSA (for strings) instead of the canonical state-minimal one and accordingly can do without an equivalent of condition (C3). We imitate their algorithm for the construction of the oracle but we prefer to provide the additional information enforced by condition (C3) via the positive sample in order to obtain a table which when filled in correctly actually represents the canonical RFTA $\mathcal{R}_L$.

The approach of RRPNI differs from all other algorithms considered in this chapter inasmuch as this time the set E stays unchanged throughout the process but we even have to delete elements from S as otherwise the FTA derived from the table may contain wrong transitions. However, note that just like the other algorithms RRPNI still features the desirable property that the learner is sure to identify the target correctly from any superset of a

suitable sample as long as conditions (C1)–(C3) are fulfilled. We can adopt the perspective that instead of trying to establish a set of experiments which will make the right distinctions between the residual languages represented by the subtrees in the sample, in this case the relevant residual languages are represented by *contexts* in the given data which will stay fixed and it is the learner's task to find a matching (minimal) set S of trees that is fit to represent the relevant states and transitions between them.

3.4.6 The algorithm MATRES

In this subsection we propose a learning algorithm that tries to infer a regular tree language $L \subseteq \mathbb{T}_\Sigma$ from EQs and MQs and in case of termination returns the canonical RFTA $\mathcal{R}_L$. MATRES is based on the main body and various subprocedures of the meta-algorithm GENDET as described in Subsection 3.3.1 with some necessary modifications to adapt them to the residual case. The technical remarks preceding the description of GENDET still apply, and the main body of MATRES is given below. Lines that differ (in more or less significant details) from GENDET are marked by an inverted comma. The input is instantiated by $\langle 1, 1, \emptyset, \emptyset \rangle$ (for EQs and MQs), and since we return $\mathcal{R}_T$ instead of $\mathcal{A}_T$ obviously the output can no longer be specified to be a DFTA.

In [17] Bollig et al. have given an algorithm learning RFSA for strings in the same setting which stays as close as possible to Angluin's algorithm LSTAR for DFA [5]. In staying close to GENDET we attempt to reproduce the switch from the deterministic to the residual approach for the tree case. However, we will argue that due to certain intrinsic properties of finite-state automata for trees such a straightforward adaptation causes considerable complications. A better algorithmic solution is yet to be presented.

Input('): $IP = \langle 1, 1, \emptyset, \emptyset \rangle$, a finite ranked alphabet Σ.
Output('): An FTA.

```
1'      INIT; c := □; RCLCS;
2       τ := 0;
3'      while τ = 0 do
           RNEXT; RCLCS;
4'      return R_T.
```

The procedures INIT, MQORACLE, POOL, and UPDATE can be applied unchanged – however, for reasons of simplicity we do not use the set C but initialize a global variable c by $\square$, which will then successively be replaced by the last counterexample given by the teacher. We replace CLOSURE in lines

1' and 3' by a procedure RCLCS and NEXTDIST by a procedure RNEXT (for descriptions see below). As soon as the while-loop has been exited the algorithm MATRES returns the FTA $\mathcal{R}_T$ as defined in Subsection 3.4.2.

---

procedure RNEXT
23' $c := \texttt{EQ}(\mathcal{R}_T)$;
24' **if** $c = \square$ **then** $\tau := 1$;
25' **else** $E := E \cup \mathit{Cont}(c)$; RED $:=$ RED $\cup \mathit{Subt}(c)$;
26' UPDATE.

The procedure RNEXT submits an equivalence query concerning the current hypothesis automaton $\mathcal{R}_T$ and instantiates c with the answer. If the teacher returns $\square$ then the learning process is considered concluded and the termination criterion τ is set to 1 which will cause MATRES to exit the main loop. Otherwise the learner adds $\mathit{Cont}(c)$ to E and $\mathit{Subt}(c)$ to RED to create new states and/or eliminate at least one more covering relation between rows of the table (see the correctness proof below). Note that the latter may include covering relations between an individual row and a join of rows with the consequence that the individual row will become prime. Also note that by adding $\mathit{Subt}(c)$ to RED we achieve S-composure for E, which is a precondition of Theorem 4 (Subsection 3.4.3).
After the call of RNEXT in the main loop the learner applies the procedure RCLCS to make the updated table R-closed and R-consistent again.

---

procedure RCLCS
r0 **while** (T is not R-closed) $\vee$ (T is not R-consistent) **do**
18' **if** T is not R-closed **then**
19' find $s \in$ BLUE such that $\mathit{row}(s) \in \mathcal{P}_S \setminus \mathcal{P}_{\text{RED}}$;
20 RED $:=$ RED $\cup \{s\}$;
21 BLUE $:=$ BLUE $\setminus \{s\}$;
22 UPDATE;
r1 **if** T is not R-consistent **then**
r2 **for** $f(s_1, \ldots, s_n), f(s'_1, \ldots, s'_n) \in S$:
$\forall i \in \{1, \ldots, n\} : s_i \sqsubseteq s'_i \wedge \neg(f(s_1, \ldots, s_n) \sqsubseteq f(s'_1, \ldots, s'_n))$ **do**
r3 **for** $j = 1, \ldots, n$ **do**
r4 **if** $\exists e \in E : \exists m \in \mathbb{N} \setminus \{0\} : \exists g \in \Sigma_m : \exists t_1, \ldots, t_m \in S :$
$\exists k \in \{1, \ldots, m\} : \mathcal{O}(e[\![g(t_1, \ldots, t_m)^{\square}_k [\![s_j]\!]]\!]) = 1 \ \wedge$
$\mathcal{O}(e[\![g(t_1, \ldots, t_m)^{\square}_k [\![s'_j]\!]]\!]) = 0$ **then**
r5 $E := E \cup \{e[\![g(t_1, \ldots, t_m)^{\square}_k]\!]\}$;
r6 UPDATE.

The goal of RCLCS is to ensure that the table is R-closed and R-consistent.

Given the definition of R-closedness, lines 18'–22 are straightforward and can be compared to the corresponding ones of the procedure CLOSURE.

Remark Since closedness implies R-closedness we could also have reused the procedure CLOSURE but we aim to reduce the number of elements that are promoted to RED. However, also note that while in the deterministic case we were able to ensure that any element promoted to RED would stay obviously different from all other RED elements throughout the learning process, in this framework the row of any element promoted to RED due to the criterion that its row is prime may become composed again later on when BLUE is filled up with new elements. ♢

Moreover, in contrast to the tables built by GENDET, in the residual case T cannot be guaranteed to be R-consistent a priori. This is due to the fact that for DFTA we only need a single representative for a certain state in RED such that we could eliminate the eventuality of an inconsistency altogether. For RFTA we generally have to allow and even need rows covering other rows in S and hence the preconditions for an R-inconsistency are given (line r1).

As discussed in Subsection 3.4.4, an R-inconsistency in T caused by two trees $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ in S with $s_i \sqsubseteq s'_i$ for all $i \in \{1, \ldots, n\}$ but $\neg(s \sqsubseteq s')$ implies that there is at least one index $j \in \{1, \ldots, n\}$ such that ${s_j}^{-1}L \not\subseteq {s'_j}^{-1}L$ and there is an excluding context exactly of depth $d+1$ for s_j and s'_j where d is the depth of an arbitrary context $e \in E$ fulfilling $obs(s, e) = 1$ and $obs(s', e) = 0$. However, in the present setting there is no knowing if S is representative for L, which entails that (a) S may not contain the necessary subtrees in order to represent the application of an L-transition ϵ of s to the tree s such that ϵ is not an L-transition of s' (see the explanations in Subsection 3.4.4), and (b) even if there is such a tree in S then E may not contain a positive context for it. Unfortunately, this implies that RCLCS may not terminate since it is possible that the table may still contain R-inconsistencies after the execution of the two for-loops in lines r2–r6 and that none of them is solvable even after solving all other ones.

Remark This differs from the string case where we can simply generate all conceivable L-transitions for an element from the alphabet Σ without having to recur to the equivalence classes that are already represented in S. It also differs from the deterministic approach for trees where an inconsistency was tantamount to the existence of a suitable transition and context in the table due to the fact that the offending elements and the remedy coincide. ♢

Whether there exists an efficient algorithmic solution to overcome this difficulty remains yet to be studied. Also see the discussion in Section 3.5.

In lines r2–r6, our learner processes all R-inconsistencies in the current table in some arbitrary but fixed order and checks for each pair of subtrees s_j and s'_j if it can construct a representative $e' \in \mathbb{C}_\Sigma$ of an L-transition for s_j using a symbol from Σ and elements of S as subtrees such that there is a positive context e for $e'[\![s_j]\!]$ in E and the context $e[\![e']\!]$ is negative for s'_j. Obviously, if it succeeds then adding $e[\![e']\!]$ to E eliminates the covering relation between s_j and s'_j, and the R-inconsistency under consideration disappears.

Remark We could try to improve our chances by rotating the order in which the R-inconsistencies are processed since by resolving one of them suitable positive contexts for others might be introduced into the table. ♢

Let C be the set of counterexamples obtained from the teacher throughout a run of MATRES, and let ζ be the maximal tree size featured in C, i.e., we have $|c| \leq \zeta$ for all $c \in C$. Let ρ be the maximal rank in Σ such that $\Sigma_\rho \neq \emptyset$.

Theorem 9 *If MATRES terminates then*
it has asked a number of EQs bounded by ${I_L}^2$ *and*
filled a number of cells bounded by $(\zeta I_L + (\zeta I_L)^\rho |\Sigma|) \cdot \zeta {I_L}^2$ *via MQs and returns an FTA which is isomorphic to* $\mathcal{R}_L$.

Proof.[15] First of all, if MATRES terminates then the output FTA is the state-minimal saturated RFTA for L by Theorem 4 since obviously the components of T fulfil conditions (1) and (2), and $\mathcal{R}_T$ fulfils condition (3) and recognizes L due to the fact that the last EQ must have been answered in the positive.

We show that if RCLCS terminates then MATRES terminates,
and that MATRES asks at most ${I_L}^2$ EQs.

Define values $\alpha := |row(S)|$, $\beta := |row(\text{RED})|$, $\pi := |\mathcal{P}_S|$, and
$\iota := |\{\langle r, r' \rangle \mid r, r' \in row(S) \wedge r \sqsubseteq r' \wedge r <> r'\}|$, i.e., ι is the number of row pairings such that the second row strictly covers the first.

We examine how these values evolve during a run of MATRES. Clearly the values of α, β, and π cannot increase beyond I_L. Moreover, α and β can never decrease since we do not delete elements of S and E, and no subsequent extension of the table can make two already distinguished rows identical again. Each new distinction causes at least one of those values to change. We will show that none of those values can change infinitely often and that each EQ leads to a new distinction of at most ${I_L}^2$ possible ones.

If T is not R-closed then after an execution of the while-loop in RCLCS where line 18' is entered β increases by 1. Simultaneously, α may increase by some measure $k > 0$ due to the call of UPDATE in line 22, and the value of ι may

[15] This proof has been adapted from [17] to the tree case.

increase by at most k times the old value of α (which is the maximal number of strict covering relations between new rows and old rows) plus $k(k-1)/2$ (which is the maximal number of strict covering relations between new rows).

If T is not R-consistent then after an execution of the while-loop in RCLCS where line r1 is entered β stays unchanged. However, α may increase by some measure $k' > 0$, and ι may increase by at most k' times the old value of α plus $k'(k'-1)/2$ (see above). If α does not increase then this means that any pair of elements $s, s' \in S$ with $s \approx s'$ in the old table still fulfils $s \approx s'$ in the new table. Thus, no new strict covering relation can have been introduced and ι cannot increase. However, if in that loop execution we have added a context to E *eliminating* a covering relation then ι must decrease by at least 1.

If RCLCS terminates then the current table $T = \langle S, E, obs \rangle$ is R-closed and R-consistent, and the learner submits an EQ for $\mathcal{R}_T$ in the next execution of RNEXT. If this results in a counterexample $c \neq \square$ then MATRES constructs a new table $T' = \langle S', E', obs \rangle$ with $S' = S \cup Subt(c)$ and $E' = E \cup Cont(c)$. Either α increases or it does not. If α increases by some measure $k'' > 0$ then ι may increase by at most k'' times the old value of α plus $k''(k''-1)/2$.
If α does not increase then ι cannot increase either (as explained in the previous paragraph). We add $Cont(c)$ to E. Note that T' must be R-consistent because the introduction of an R-inconsistency would entail an increase of α.
If c is a positive counterexample: Observe that for each symbol $a \in \Sigma_0$ and context $e \in E$ with $e[\![a]\!] \in L$ the tree $e[\![a]\!]$ is accepted by $\mathcal{R}_T$ due to the fact that $\delta_T^*(a) = \{q \in Q_T \mid q \sqsubseteq row(a)\}$ and Lemma 9 (and note that for the members of Σ_0 we do not need T-consistency to ensure the "$\Rightarrow$"-direction). Thus, if we had $Cont(c) \subseteq E$ then $\mathcal{R}_T$ would accept c. Therefore, adding $Cont(c)$ to E must change the values of ι and/or π: Suppose that both stay unchanged. Then $\mathcal{R}_T$ and $\mathcal{R}_{T'}$ would be isomorphic since α has not changed either. However, as we have $Cont(c) \subseteq E'$ the automaton $\mathcal{R}_{T'}$ must correctly accept c whereas $\mathcal{R}_T$ rejects it, a contradiction to our assumption.
If c is a negative counterexample: The fact that c is wrongly accepted by $\mathcal{R}_T$ implies that there is a context $e' \in Cont(c)$ and some state $q \in Q_T$ such that $e' \in \mathcal{C}_q$ but $q(e') = 0$. Moreover, we have $q(e) = 0 \Rightarrow e \notin \mathcal{C}_q$ for all $e \in E$ due to the "$\Leftarrow$"-direction in Lemma 9 (and recall that for this direction we do not depend on $\mathcal{R}_T$ being T-consistent at all). Again, adding $Cont(c)$ to E must change the values of ι and/or π: Suppose that both stay unchanged. Then $\mathcal{R}_T$ and $\mathcal{R}_{T'}$ would be isomorphic since α has not changed either. However, as we have $e' \in E'$ the automaton $\mathcal{R}_{T'}$ must correctly reject c due to Lemma 9 whereas $\mathcal{R}_T$ accepts it, a contradiction to our assumption.
Therefore, ι decreases or π increases or both.

In summary we observe that after each run of lines 18'–22, r1–r6, or 25'–26', under the assumption that RCLCS terminates, either (a) β is increased or (b) α is increased by some measure $k > 0$ and simultaneously ι is increased by at most $k\alpha + k(k-1)/2$ or (c) α stays the same and ι does not increase but ι decreases or π increases. Recall that the values of α, β, and π cannot increase beyond I_L. When α and β cannot increase anymore then ι decreases or π increases and hence MATRES terminates. Furthermore, we have shown that each EQ leads to a change of at least one of the values α, ι, or π. Since such a change indicates a new distinction between rows and since there are at most ${I_L}^2$ distinctions to be made the number of EQs is bounded by ${I_L}^2$.

Concerning the number of MQs, we fill a table containing at most $m_S = \zeta I_L + (\zeta I_L)^\rho |\Sigma|$ elements in S and $m_E = \zeta {I_L}^2$ elements in E since for each distinction between two rows of the table at most ζ elements were added to E and, given that the distinction had to be obtained via an EQ, also to S. Moreover, in each execution of the inner for-loop in lines r3–r6 the learner asks two additional MQs, and thus the overall number of MQs caused by an execution of the outer for-loop is bounded by $2\rho^2|\Sigma|{m_S}^{\rho+2}m_E$. ■

Remark Bollig et al. [17] give an example which demonstrates that merely adding all substructures of the counterexample to S as would be suggested by an (even more) direct adaptation of LSTAR [5] to the residual case would not ensure termination, not even for strings (see Appendix F in [17]). ♢

Remark The algorithm in [17] learning string RFSA from MQs and EQs needs ${I_L}^2$ EQs to infer the target automaton in a worst case but the authors can show that in practice for many languages and example runs their learner returns the correct solution after a much smaller number of queries than even Angluin's theoretically superior LSTAR [5]. ♢

We will continue the discussion in Section 3.5 below.

3.5 Discussion

The literature shows that the learning algorithms for strings as described in [5, 3, 94] taking the deterministic approach in the three settings which were of consequence above were relatively easy to adapt to trees – see [46, 14, 93]. Moreover, the algorithms for the settings including a positive sample fulfilling certain conditions were also comparatively straightforward to adapt to the residual approach as shown in Subsections 3.4.4 and 3.4.5 above. One might have expected that an equivalence oracle in combination with a membership oracle is at least as favourable to polynomial inference as any setting including a non-void positive sample – however, it seems that a representative sample of L bears essential information for the residual approach due to the double role of the subtrees in the given sample as candidates for representatives of states but also as components for the representation of transitions in L.

In the deterministic case, an inconsistency revealed by two trees $t, t' \in S$ can always be resolved due to the arguments in Subsection 3.4.4 – and we are even spared the effort of trying out all possible L-transitions that can be derived from the table in order to construct a distinguishing context for two direct subtrees of t and t' due to the fact that the L-transition represented by t itself and any positive context for it is suitable – while we might not be able to find a counterexample on our own. Note that for the design of GENDET we have chosen to maintain the set of all one-symbol extensions in the setting of MQs and a positive sample as well in order to be able to translate any inconsistency into an explicit counterexample (see Subsection 3.3.1). In the residual case, however, without S containing a representative sample we may not even be able to solve some R-inconsistencies at all due to missing representatives for a suitable L-transition, see Subsection 3.4.6.

The crucial point is that between EQs the learner must always be able to compute its next hypothesis in a polynomially bounded number of steps. Judging from all other algorithms learning from MQs and EQs considered so far, which all basically imitate the actions of Angluin's LSTAR [5]), an eligible strategy to ensure this seemed to be to comply with the principle that any hypothesis must fulfil essential properties of the agreed canonical description, i.e., a deterministic or a residual automaton, respectively, before submitting it to the teacher – even if the oracle would also return an answer for an arbitrary FTA. This implies that when the residual approach is taken in the setting of MQs and EQs the difficulty arises even before the learner has the opportunity of applying the powerful instrument of an EQ.

Trivially, since any algorithmic learner of the kind considered here settles on a specific type of description for its hypotheses the range of descriptions that can ever be constructed and returned by the learner is restricted. The difficulty outlined above may be traced back to the result from [40] that the description class of non-deterministic automata as such is not polynomially characterizable, i.e., that the construction of a set which when included into the available data is sure to make the learner identify the target correctly is not of polynomial complexity. The result has been reproduced for residual automata in [42]. When learning deterministic and residual string automata and deterministic tree automata this problem could be circumvented due to the specific arguments elaborated above (for strings we only depend on the candidates already present in the table and on the alphabet, and for DFTA the existence of a representative for the application of a suitable L-transition in the table is ensured) but in general the elimination of (R-)inconsistencies can be seen as part of the process of building a characteristic sample.

On the other hand, provided that T can always be made R-consistent, we observe that a learner learning from MQs and EQs significantly benefits from the fact that it will receive a counterexample as long as the table does not represent the canonical RFTA for the target since it entails that eventually, for each prime residual language of L the table must contain an element representing it with a prime row. This can be seen as follows: Assume that there is a prime residual language such that there is no element in T representing it with a prime row. Then either there is a counterexample in T itself proving that $\mathcal{R}_T$ does not yet correspond to $\mathcal{R}_L$ – or $\mathcal{R}_T$ is T-consistent but then $\mathcal{R}_T$ is a state-minimal saturated RFTA by Theorem 4. However, since there still exists a prime residual language of L not having a prime row in T the language recognized by $\mathcal{R}_T$ cannot equal the target language L.
Moreover, note that while all equivalence classes of L have to appear in the table in order to represent the deterministic canonical FTA $\mathcal{A}_L$, as can be seen from the discussion of RRPNI in Subsection 3.4.5 the set S does not have to be representative for all of L to encode $\mathcal{R}_L$ and the learner may even benefit from the absence of certain representatives since they may introduce misleading covering relations. And although the learner does not know which of the representatives in the current table are indispensable and which are not, due to the powerful device of an equivalence query the learner is notified immediately in case of success. This is also the reason why in many individual cases the algorithm by Bollig et al. [17] will terminate after having asked even less queries than Angluin's LSTAR [5].

As mentioned above, the algorithm RRPNI can do with less data than its deterministic counterpart RPNI [93] since we do not require the positive sample to be representative for all of L – provided that the given samples fulfil conditions (C1)–(C3). In contrast, one might argue that those conditions are rather restrictive and that with MQs and a full representative sample the algorithm RESI is able to construct all the necessary distinctions on its own. Note that just like the deterministic counterpart of RESI presented in [14], while in theory RESI asks a large number of MQs its query complexity is still not exponential with respect to the maximal rank ρ in Σ due to the fact that we only use subtrees that are already contained in the table, whereas in the case of both the deterministic and the residual learner for MQs and EQs the procedure of filling up the set BLUE with all one-symbol extensions of RED is exponential in ρ. As a motivation for this setting, it has been argued in [14] that for the representation of a learning process the combination of MQs and a positive sample seems to be the most natural and beneficial one, and we may second this claim considering the fact that it features a good balance between the strength of the conditions that have to be fulfilled by the input and the complexity of the computation that is performed.

Remark Since any exclusive sample for L is also separative for L, if we have membership queries and a negative finite sample of L then as before we can hope that the sample is non-void and compute an upper bound for I_L in order to construct a potentially representative positive sample for L. However, we would obviously still encounter the same exponentiality issues mentioned for MQs and a negative sample in Subsections 3.3.1, 3.3.2, and 3.3.3 above. $\diamondsuit$

Outlook

State-minimal deterministic and saturated state-minimal residual automata are not the only kinds of canonical automata for a regular language L that recognize L and can be characterized using the equivalence classes or residual languages of L. Another instance is the *universal automaton* where the set of states is given by all possible intersections of any number of residual languages of L, see [56, 89] for the string case. The authors of [56] present a learning algorithm which infers a correct nondeterministic finite-state automaton from a finite positive and negative sample fulfilling certain properties with respect to the universal automaton of the target language but the output may not be the universal automaton itself and may vary depending on the given input. And recently Brzozowski and Tamm [18] have defined a generalization of deterministic, residual, and universal finite-state automata (stripped of inaccessible and failure states) for strings, so-called *atomic automata*, where each

state can be characterized by the *atoms* of the language L that is recognized. An atom of L is an intersection such that for each residual language of L either the residual language itself or its complement is a component of the intersection. The canonical atomic automaton recognizing L, its *átomaton*, is an automaton where the set of states corresponds to the set of atoms of L. A further option to explore are *biautomata* (see [84]), which can read a string alternatively from the right or from the left. The states of the canonical biautomaton for a regular language L correspond to the residual languages of L with respect to the notion of both-sided string contexts, see Definition 2. As we have seen that in learning theory the existence of a canonical member in a given class of descriptions is a useful property for convergence it would be interesting to explore whether those canonical automata can be inferred in the settings and by the methods considered in this work without major modifications. If this is answered in the positive then an extension of those results to trees should be straightforward as well.

In fact, the range of potential candidates for future work includes any kind of automaton that can be unambiguously characterized using the equivalence classes and/or residual languages of the recognized language L and the operations of complement, union, and intersection (possibly combined with other restrictions such as the condition defining the prime-reduced residual automaton $\mathcal{R}_p$, see Subsection 3.4.3) such that the respective sets of access elements and/or contexts for its states satisfy some maximality condition with the effect that each of those sets must contain instances that we can use in order to distinguish between pairs of individual states. A research project intending to explore that direction in more depth with Johanna Björklund and Henning Fernau is in preparation.

Chapter 4

Distributional learning

This chapter mainly serves to give an impression of a cluster of approaches to the inference of formalisms beyond regularity or even beyond context-freeness with certain favourable properties which has been rediscovered and formalized by Alexander Clark based on linguistic work in American structuralism [112, 65], and has been essentially advanced by Clark and Ryo Yoshinaka and others during the past few years, for example in [32, 30, 120, 34, 116, 117, 118], including a paper by Yoshinaka and the author [122].

We base our preliminary explanations on a summary of the arguments in the literature from the author's perspective in Section 4.1, and then we describe the generic strategy of a range of learning algorithms that have been developed for the distributional approach so far in Section 4.2. Most of those distributional learners can be attributed to one of two subapproaches, and we establish a kind of formal meta-synopsis over several algorithms for one of them (in Subsection 4.2.1) and give a version of a previously published algorithm in our own notation for the other (Subsection 4.2.2). The underlying learning model in these two subsections is identification in the limit. We also briefly discuss some distributional one-shot learning algorithms for the inference of context-free grammars in the settings that were of interest in Chapter 3, which we have adapted to the tree case in Subsection 4.2.3. In fact, most of the formal notions and all algorithms in this chapter will be formulated with respect to trees, thereby generalizing a range of algorithms presented in the literature. We do not cover every detail and at some points we refer the reader to corresponding articles for proofs.

This chapter owes a great deal to previous personal communication with Ryo Yoshinaka and Alexander Clark. For a comprehensive impression of the general principles of distributional learning in the string case we also refer the reader to their articles [26, 29, 30] and [118].

4.1 Finitely characterizable distributions

For the general explanations to follow, let us fix that if $L \subseteq \Sigma^*$ is a string language over some unranked alphabet Σ then

- $\equiv_L^>$ denotes the equivalence relation with $w \equiv_L^> w'$ for $w, w' \in \Sigma^*$ if $wv \in L \Leftrightarrow w'v \in L$ for all suffixes $v \in \Sigma^*$ and
- $\equiv_L$ denotes the equivalence relation with $w \equiv_L w'$ for $w, w' \in \Sigma^*$ if $e[\![w]\!] \in L \Leftrightarrow e[\![w']\!] \in L$ for all both-sided string contexts e in the sense of Definition 2 (see Subsection 2.1.1),

and if $L \subseteq \mathbb{T}_\Sigma$ is a tree language over some ranked alphabet Σ then

- $\equiv_L^>$ denotes the equivalence relation such that $t \equiv_L^> t'$ for $t, t' \in \mathbb{T}_\Sigma$ if $e[\![t]\!] \in L \Leftrightarrow e[\![t']\!] \in L$ for all tree contexts e in the sense of Definition 10 (see Subsection 2.2.1) and
- $\equiv_L$ denotes the equivalence relation such that $s \equiv_L s'$ for $s, s' \in \mathbb{S}_\Sigma^k$ with $k \geq 0$ if $e[\![s]\!] \in L \Leftrightarrow e[\![s']\!] \in L$ for all tree environments $e \in \mathbb{E}_\Sigma^k$ (see Definitions 11 and 12).

The learnability results for regular languages given and discussed in the previous chapter intrinsically rely on the existence of a canonical description for the target language where each component (in our case: each state) can be characterized by a single member of the language in question such that the description still only has a finite number of components due to the respective Myhill-Nerode theorems for strings and trees. The description of choice in the previous chapter is a finite-state automaton but it is a well-known result (see [67, 58, 57]) that any automaton $\mathcal{A}$ can be directly translated into a regular grammar in normal form, i.e., with rules of the form $A \to a$ and $A \to Ba$ for nonterminals A, B and $a \in \Sigma$ in the string case and $A \to f(B_1, \ldots, B_n)$ for nonterminals $A, B_1, \ldots, B_n$ and $f \in \Sigma_n$ in the case of trees, where the set of nonterminals is given by the states of $\mathcal{A}$ and the rules are determined by the transitions in $\mathcal{A}$. This implies that among the different kinds of conceivable canonical finite-state automata for a regular language L we can choose for example the state-minimal deterministic automaton $\mathcal{A}_L$ and transform it into a regular grammar where the nonterminals correspond to equivalence classes under the relation $\equiv_L^>$. As a desirable outcome, the constructed grammar is unambiguously defined for L and can thus be adopted as another canonical description since it also correctly generates L due to the fact that $\equiv_L^>$ is a right congruence, i.e., whenever $w \equiv_L^> w'$ then $wa \equiv_L^> w'a$ for all $a \in \Sigma$ in the string case, and whenever $t_i \equiv_L^> t_i'$ for $1 \leq i \leq n$ then

$f(t_1, \ldots, t_n) \equiv_L^> f(t'_1, \ldots, t'_n)$ for all $f \in \Sigma_n$ in the case of trees. Observe how this condition is exactly reflected by our canonical grammar in normal form. In other words, the Myhill-Nerode theorem is responsible for the fact that the number of equivalence classes under $\equiv_L^>$ is finite and the property making $\equiv_L^>$ a congruence relation is responsible for the fact that it is possible to construct a finite grammar which generates L using those equivalence classes as nonterminals while observing the restrictions of a certain normal form.

Moving on from regular to (strictly) context-free languages, the situation becomes more intricate. Of course there are various descriptions for context-free languages with a finite number of components such as push-down automata and context-free grammars. However, there are no efficient learnability results for those description classes as a whole based on the same principles as above which is basically due to the fact that even if we settle on some normal form there may be infinitely many distinct choices when trying to establish a minimal set of nonterminals and to determine corresponding rules, such that there is no unambiguous link to the syntactic structure of the target language. To illustrate the difficulty of such a claim, Clark [29] turns the observations that were exploited in order to obtain the established learnability results for regular string languages – in essence, their decomposability into sets of prefixes and suffixes and the interrelations of those sets when recombined by concatenation – into a first approach to the context-free case based on the argument that given two strings w and w', if we have evidence for $w \equiv_L w'$ then this suggests the existence of a grammar for L featuring a nonterminal which generates both w and w'. He accordingly constructs a kind of canonical context-free grammar for every context-free string language L in a modified Chomsky normal form (see [67]) with multiple initial symbols where the set of nonterminals corresponds to the set of equivalence classes under $\equiv_L$ and every possible concatenation between two classes $[w]_L$ and $[w']_L$ is reflected by a rule of the form $[ww']_L \to [w]_L[w']_L$. Nonterminals are finally instantiated by terminating rules of the form $[a]_L \to a$ for individual symbols $a \in \Sigma$ and $[\varepsilon]_L \to \varepsilon$ for the empty string ε. The important aspect of this construction is that even if we restrict our nonterminals to the set of equivalence classes consisting of substrings of L, if L is not regular then although in theory L is generated correctly the resulting grammar will not be finite.

This strongly suggests that the class of context-free languages as such is not efficiently learnable in the models and settings considered so far. In fact, it has been shown that it is computationally intractable to infer a context-free grammar even in favourable environments involving membership queries and labeled data, see [10]. Consequently, Clark [26, 29, 30] suggests to concentrate on the inference of "*objective*" grammars, i.e., where there exists a specific finitely characterizable connection between the nonterminals and

the syntactic structure of the generated language while still satisfying the constraint that the number of components of that grammar be finite as well, with the intrinsic property that efficient learning in the sense discussed above becomes a feasible task. Note that as a consequence of such a restriction the class of languages that are generated by a grammar meeting it may not correspond to the class of context-free languages in general anymore.

In line with these observations, a range of similar properties of context-free formalisms have been accumulated and shown to be favourable to algorithmic learning in the literature (for references see Figure 4.1). Basically, in a finite context-free grammar $\mathcal{G}$ fulfilling one or several of the properties below each nonterminal is characterizable by some finite set of substructures of $\mathcal{L}(\mathcal{G})$ or by a finite set of positive contexts for those substructures. The set of contexts for a certain substructure within $\mathcal{L}(\mathcal{G})$, i.e., its residual language, is also called its *distribution*, hence the term '*distributional learning*'. As the paper [122] by Yoshinaka and the author, which contributes to this range of results, performs a generalization from classic context-free string grammars (CFGs) to SCFTGs (see Subsection 2.2.1) we will list those properties with respect to an SCFTG since that formalism includes CFGs as a special case. Note that we could also have listed them with respect to an MCFG (see [103], [117, 120] for a definition and learnability results) as another generalization of CFGs – however, MCFGs generate strings, not trees.[1]

Definition 45 *An SCFTG $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ is said to be*

- substitutable *if the existence of an environment $e \in \mathbb{E}^m_\Sigma$ with $e[\![s]\!] \in \mathcal{L}(\mathcal{G}) \wedge e[\![s']\!] \in \mathcal{L}(\mathcal{G})$ implies $e'[\![s]\!] \in \mathcal{L}(\mathcal{G}) \Leftrightarrow e'[\![s']\!] \in \mathcal{L}(\mathcal{G})$ for all $e' \in \mathbb{E}^m_\Sigma$, $s, s' \in \mathbb{S}^m_\Sigma$, and $m \geq 0$,*
- congruential *if we have $e[\![s]\!] \in \mathcal{L}(\mathcal{G}) \Leftrightarrow e[\![s']\!] \in \mathcal{L}(\mathcal{G})$ for all $e \in \mathbb{E}^m_\Sigma$, $s, s' \in \mathcal{L}_A$, $A \in N_m$, and $m \geq 0$,*
- context-deterministic *if $S \Rightarrow^*_{\mathcal{G}} e[\![A(\square_1, \ldots, \square_m)]\!]$ for $S \in I$ implies $\mathcal{L}_A = \{s \in \mathbb{S}^m_\Sigma \mid e[\![s]\!] \in \mathcal{L}(\mathcal{G})\}$ for all $e \in \mathbb{E}^m_\Sigma$, $A \in N_m$, and $m \geq 0$.*

Furthermore, $\mathcal{G}$ is said to have the

- finite kernel property (k-FKP) *for some $k \in \mathbb{N} \setminus \{0\}$ if for all $m \geq 0$, each $A \in N_m$ admits a finite set $K_A \subseteq \mathcal{L}_A$ with $0 < |K_A| \leq k$ such that for all $e \in \mathbb{E}^m_\Sigma$, if we have $e[\![s']\!] \in L$ for all $s' \in K_A$ then this implies $e[\![s]\!] \in L$ for all $s \in \mathcal{L}_A$ – we call K_A a* k-kernel of A,

[1] Roughly, an MCFG is a grammar for the generation of strings (or 1-words) where each nonterminal generates a set of multi-words (see Subsection 2.1.2) and rules are of the form $A \to \mathsf{f}(B_1, \ldots, B_n)$ where $A, B_1, \ldots, B_n$ are nonterminals and f is a function that rearranges the components of the multi-words generated by $B_1, ..., B_n$ into a new one.

- finite context property (x-FCP) *for some* $x \in \mathbb{N} \setminus \{0\}$ *if for all* $m \geq 0$, *each* $A \in N_m$ *admits a finite set* $X_A \subseteq \mathbb{E}_\Sigma^m$ *with* $0 < |X_A| \leq x$ *such that* $\mathcal{L}_A = \{s \in \mathbb{S}_\Sigma^m \mid \forall e \in X_A : e[\![s]\!] \in \mathcal{L}(\mathcal{G})\}$.
  – *we call* X_A *an* x-context of A.

A language L is said to have any of the properties given above if there is a grammar $\mathcal{G}$ having the respective property and fulfilling $\mathcal{L}(\mathcal{G}) = L$.

Note that congruentiality implies the k-FKP and context-determinism implies the x-FCP (for any values of k and x) while the property of substitutability is so restrictive that it implies all other four, with corresponding effects on the inclusion relations between the respective language classes.

We give some selected examples from the literature with respect to CFGs over the alphabet $\Sigma = \{a, b, c\}$ to help on the intuition.

- *Substitutable* string languages are Σ^*, all singletons over Σ, the language $\{a^n \mid n > 0\}$, and $\{wc\bar{w} \mid w \in \Sigma^*\}$ where $\bar{w}$ is w reversed, see [32].
  Not substitutable are for example $\{a, aa\}$ and $\{a^n b^n \mid n > 0\}$. Observe: In the finite language $\{a, aa\}$ the strings a and aa share the context $\varepsilon\Box\varepsilon$ but the positive context $\varepsilon\Box a$ for a is not a positive context for aa.

- *Congruential* languages are $\{a^n b^n \mid n > 0\}$ and the Dyck language [26].
  Not congruential are $\{a^n b^m \mid 0 < n < m\}$ and $\{a^m b^n \mid 0 < n < m\}$, $\{a^n b^n \mid n > 0\} \cup \{a^n b^{2n} \mid n > 0\}$ and the language $\{w \in \Sigma^* \mid w = \bar{w}\}$ of all palindromes over Σ, which is due to the fact that those languages are unions of infinitely many classes under the congruence relation $\equiv_L$.
  Also note that the Dyck language is not *context-deterministic* since for example the context $a\Box b$ is positive for ab as well as for ba, see [26].

- *FKP*: The palindrome language $\{w \in \Sigma^* \mid w = \bar{w}\}$ and all languages $L_m = \{a^n b^{ln} \mid n \geq 0 \wedge 1 \leq l \leq m\}$ for some $m \geq 2$ have the 2-FKP but not the 1-FKP.
  Moreover, for each $k \geq 3$ the k-FKP is separated from the $(k-1)$-FKP by the language $L_k = \{a_1^{n_1} \cdots a_k^{n_k} \mid \exists i, j \in \{1, \ldots, k\} : i \neq j \wedge n_i = n_j\}$ over an alphabet of symbols $\{a_1, \ldots, a_k\}$, see [118].

Some inclusion and incomparability relations with respect to CFGs:

- The examples above show that the class of substitutable languages is incomparable with the finite and with the regular class.

- All regular languages are congruential, some context-free ones are not.

- The context-deterministic and the congruential class are incomparable.

Depending on the underlying context-free formalism – for instance, CFGs, MCFGs, or SCFTGs – the class of grammars meeting one of those properties can be associated with a string language class that does not feature a one-to-one correspondence with one of the established classes and may even cross-cut the Chomsky hierarchy. Of course a CFG can only generate a context-free string language but there are classes of MCFGs that allow the derivation of string languages beyond context-freeness within the *mildly context-sensitive* family which was first proposed by Joshi [72] and is of particular importance in computational linguistics, and the same can be stated for some SCFTG classes when the yield operation is interposed. On the other hand, there may be context-free string languages that are not included in the generated class. For example, for some alphabet $\{a, b, c, d, e\}$,

- the string languages $\{a^m b^n c^m d^n \mid m, n > 0\}$, $\{wcwc \mid w \in \{a, b\}^*\}$, and $\{a^n b^n c \mid n > 0\} \cup \{a^n b^{2n} d \mid n > 0\}$ can be generated by a congruential MCFG but $\{a^m b^n \mid 0 < m \leq n\}$ and $\{a^n b^n \mid n > 0\} \cup \{a^n b^{2n} \mid n > 0\}$ cannot – see [120], and

- a non-contextfree string language generated by a substitutable SCFTG is $\{a^n b^n c d^n e^n \mid n > 0\}$, see [122].

As can be observed from Definition 45, the components of a substitutable, congruential, or context-deterministic grammar can be characterized by the distribution of a single element (a substructure or context) with respect to $\mathcal{L}(\mathcal{G})$, and for the FKP and the FCP the same holds at least for a finite set. Note that there may not be a unique canonical grammar for each language with the respective property since there may be several distinct choices for those characterizing components. However, this finiteness is a most essential condition which can be exploited by a learner. Existing distributional learnability results for various grammar types, settings, and properties from the ones listed above are summarized in Figure 4.1.[2]

Remark The columns of the table in Figure 4.1 can be related to the three axes evoked in the motivation at the end of Chapter 1 in the sense that the first and second axis determine the language class, the second also determines the object type, and the third fixes the available information sources. ♢

[2]BFGs = Binary Feature Grammars; ITGs = Inversion Transduction Grammars; k, l-substitutable: A less restrictive form of substitutability for CFGs, see [115].

property	*grammar type*	*identification from*	*reference*
substitutable	CFGs	a stream of positive data	[32]
k, l-substitutable	CFGs	a stream of positive data	[115]
substitutable	MCFGs	a stream of positive data	[117]
substitutable	SCFTGs	a stream of positive data	[122]
congruential	CFGs	MQs and EQs	[26]
congruential	MCFGs	MQs and EQs	[120]
context-det.	CFGs	MQs and EQs	[107]
FCP and 1-FKP	BFGs	a stream of positive data and MQs	[34]
1-FKP	MCFGs	a stream of positive data and MQs	[116]
FKP	CFGs	a stream of positive data and MQs	[118]
FCP	CFGs	a stream of positive data and MQs	[28, 118]
FCP	SCFTGs	a stream of positive data and MQs	[122]
	ITGs	a stream of positive data	[31]

Figure 4.1: Some distributional learning results

In this overview we concentrate on grammars of the classic context-free type (CFGs, MCFGs, SCFTGs) because their form nicely evokes the notion of decomposing an object into exactly one substructure and its context by the fact that in each rule there is a single nonterminal on the left-hand side. Moreover, the grammars we consider will be in a normal form that is similar to the Chomsky normal form which evokes the third ingredient of importance, a well-defined concatenation operation between two substructures.

Another grammar formalism that is favourable to distributional learning and in fact has been designed by Clark et al. [33, 34] as a way to model the lattice structure of the distribution of certain sets of substrings in a language are *Contextual Binary Feature Grammars (CBFGs)*. The string languages that can be generated by CBFGs include some context-free as well as some mildly context-sensitive ones, and the subclass of grammars that is shown to be inferable by distributional techniques in [34], *exact CBFGs*, generates all regular string languages and some richly structured context-free ones.

Remark The authors of [34] define an FKP for CFGs using an existential instead of a universal quantifier in the sense that each nonterminal A is to admit a finite set K_A where for each $w \in \mathcal{L}_A$, *there exists* an element $w' \in K_A$ such that for all contexts e with $e[\![w']\!] \in L$ we also have $e[\![w]\!] \in L$. Let us call this property the FKP$^\exists$. A language generated by a grammar $\mathcal{G}$ with the FKP$^\exists$ has the 1-FKP in the sense of Definition 45 due to the fact that (a) for each nonterminal A and any kernel K_A of A in the sense of [34], the elements of K_A induce a partition on the set $\mathcal{L}_A$ such that each block consists of elements that have at least the same contexts as some element of K_A, and (b) one can find an equivalent grammar $\mathcal{G}'$ in which the nonterminals of $\mathcal{G}$ are split up accordingly such that for each of those blocks there exists a nonterminal generating it in $\mathcal{G}'$. Obviously the grammar $\mathcal{G}'$ has the 1-FKP in the sense of Definition 45 since each of its nonterminals meets the condition for the FKP with respect to a singleton subset of K_A. $\diamondsuit$

Motivated by the search for a representation that can handle phenomena in natural language syntax while still being efficiently inferable from given data, Clark [25, 28, 27] develops the formalism of *Distributional Lattice Grammars (DLGs)* which is based even more directly on the so-called *syntactic concept lattice* of a formal language and generates some but not all context-free string languages as well as some mildly context-sensitive ones of major importance for computational linguistics. In a syntactic concept lattice for a language L, each element is a pair $\langle S, C\rangle$ where S is a set of substructures of L, and C is a set of positive contexts for those substructures, and S and C fulfil certain maximality conditions with respect to each other. Using these lattices Clark [27] also shows the learnability of some inherently ambiguous context-free

string languages, thereby disproving a postulated barrier for CFG inference. Instead of concentrating on the traditional classes on the Chomsky hierarchy, Clark [25] rather propagates the slogan "Put learnability first!", i.e., design formalisms with properties that are favourable to algorithmic learnability and then study the language classes that they generate, as a general directive for future research in formal language theory under linguistic motivations.

4.2 The learner's strategy

Thus, if a language L is such that there exists a context-free grammar $\mathcal{G}$ that generates L and fulfils one of the favourable distributional properties above and if a learner is aware of that property and has access to suitable sources of information then it can exploit this knowledge to construct a grammar $\mathcal{G}_*$ whose nonterminals and rules reflect the distributional interrelations it can observe in the available data, with the outcome that $\mathcal{G}_*$ generates the same language as $\mathcal{G}$, namely L. Note that the goal is to learn the language L – we do not require the learner to produce exactly the reference description $\mathcal{G}$.

The strategies of most distributional learning algorithms developed so far (see Figure 4.1 for references) can be classified into one of the following two categories, which were first suggested in [30]. Informally speaking,

- a *primal* approach uses (sets of) substructures of the available data to represent a tentative set of nonterminals and then investigates their environments in order to determine which of the conceivable rules specifying how to combine those postulated nonterminals are valid and which are not, whereas

- a *dual* approach derives (sets of) environments from the data to represent nonterminals and then studies the substructures that can occur in those environments in order to establish a set of valid rules.

When studying existing distributional learning algorithms (for non-substitutable cases) one can observe the general principle that under the primal approach an increase of the substructures that constitute the nonterminals of the current hypothesis grammar leads to an increase of the language that it generates whereas an increase of the environments taken into account leads to a decrease of that language, while under the dual approach the effects of increasing sets of substructures and environments are switched. This is the reason for the fact that given an increasing amount of data those learners will gradually converge to the target, up to the point where eventually the set of data seen so far contains all the necessary information to exclude both under-

and overgeneralization. On closer examination, among the properties listed in Definition 45 congruentiality and the FKP promote the former approach whereas context-determinism and the FCP promote the latter. The property of substitutability is so restrictive that it permits both approaches and also allows to avoid overgeneralization altogether such that an increase of the given data does not have to lead to a decrease of the language generated by the current hypothesis grammar at all.

Remark All algorithms from the previous chapter, both for the deterministic and the residual case, take the primal approach because the components (i.e., states) of the respective hypotheses are representable by a single subtree. $\diamondsuit$

Since it is one of the objectives of this work to develop a generalizing perspective on a specific range of topics in Grammatical Inference, we will once more attempt a kind of formal meta-synopsis over a selection of previously established learnability results. Since moreover our main object of interest is the tree our chosen description will be the SCFTG as another contribution to the completion of the field, which had also been the intention in [122].
Subsection 4.2.1 covers three primal learning algorithms for substitutability, congruentiality, and the k-FKP for some $k > 0$ inspired by various algorithms from [122], [26], and [118]. In Subsection 4.2.2 we describe a learner that infers SCFTGs with the x-FCP for some $x > 0$ by the dual approach, which generalizes the one for the 1-FCP in [122] and the one for strings in [118].

We fix a finite tree labeling alphabet Σ and the following notations:

Definition 46 *For a set of trees $L \subseteq \mathbb{T}_\Sigma$ and $m \in \mathbb{N}$, we let*

- $Sub^m(L) := \{s \in \mathbb{S}_\Sigma^m \mid \exists e \in \mathbb{E}_\Sigma^m : e[\![s]\!] \in L\}$, *and*
- $Env^m(L) := \{e \in \mathbb{E}_\Sigma^m \mid \exists s \in \mathbb{S}_\Sigma^m : e[\![s]\!] \in L\}$.

For some given $r \in \mathbb{N}$, we let

- $Sub^{\leq r}(L) := \bigcup\{Sub^m(L) \mid m \leq r\}$, *and*
- $Env^{\leq r}(L) := \bigcup\{Env^m(L) \mid m \leq r\}$.

For two stubs $s_1 \in \mathbb{S}_\Sigma^m$, $s_2 \in \mathbb{S}_\Sigma^{m'}$ with $m, m' \in \mathbb{N}$ and for $j \in \mathbb{N}$ with $j \leq m$, we define a concatenation operation $\odot^j$ such that $s_1 \odot^j s_2$ with $j \geq 1$ denotes the result of replacing the jth occurrence of $\Box$ in s_1 by s_2, and $s_1 \odot^0 s_2 := s_1$. The operation can be extended to pairs of sets $S_1 \subseteq \mathbb{S}_\Sigma^m$, $S_2 \subseteq \mathbb{S}_\Sigma^{m'}$ such that $S_1 \odot^j S_2 := \{s \in \mathbb{S}_\Sigma \mid \exists s_1 \in S_1 : \exists s_2 \in S_2 : s = s_1 \odot^j s_2\}$.
For a stub $s \in \mathbb{S}_\Sigma^m$ and $t_1, \ldots, t_m \in \mathbb{T}_\Sigma \cup \mathbb{S}_\Sigma$, we define the insertion operation $\odot$ such that $s \odot \langle t_1, \ldots, t_m \rangle$ denotes the result of replacing the j'th occurrence of $\Box$ in s by $t_{j'}$ for $0 \leq j' \leq m$.

4.2.1 Primal learning algorithms

We fix an r-SCFTG $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ for some given $r \in \mathbb{N}$ to represent our learning target. We assume that $\mathcal{G}$ is in *normal form*, i.e., that for each rule $A \to s \in P$ with $A \in N_m$ and $s \in \mathbb{S}^m_{\Sigma \cup N}$ for some $m \geq 0$ we have

(a) $s = f(\Box_1, \ldots, \Box_m)$ for some $f \in \Sigma_m$, or

(b) $s = B(\Box_1, \ldots, \Box_{m'}) \odot^j C(\Box_1, \ldots, \Box_{m''})$
for $B \in N_{m'}$, $C \in N_{m''}$ with $j \leq m'$ such that
$m' = m$ if $j = 0$ and $m' + m'' - 1 = m$ otherwise.

Note the parallels to the Chomsky normal form for CFGs.[3] As stated in [122], one can show that every r-SCFTG admits an equivalent r-SCFTG in normal form. We also assume that no nonterminal or rule is unreachable or useless, i.e., each nonterminal or rule appears in at least one derivation of a member of $\mathcal{L}(\mathcal{G})$ and each nonterminal generates at least one substub in $Sub^{\leq r}(\mathcal{L}(\mathcal{G}))$. Moreover, we assume $\mathcal{G}$ to fulfil the property P where

$$\mathsf{P} \in \{\textit{substitutable, congruential, } k\textit{-FKP}\}$$

and $k > 0$. For $\mathsf{P} = \textit{substitutable}$ or $\mathsf{P} = \textit{congruential}$ we fix $k = 1$.

We will first indicate how the learner constructs its hypothesis grammar from the currently available data, and then describe the generic learning algorithm.

Suppose the learner is aware of P and has access to a finite set $D \subseteq \mathcal{L}(\mathcal{G})$ of positive data, and suppose that it has derived two sets $K \subseteq Sub^{\leq r}(D)$ and $X \subseteq Env^{\leq r}(D)$ of substructures and contexts, respectively, from that set D. Suppose that K fulfils the condition $Sub^{\leq r}(\{s\}) \subseteq K$ for all $s \in K$.
We define $K_m := Sub^m(D) \cap K$ and $X_m := Env^m(D) \cap X$.

Remark $Sub^{\leq r}(D)$ and $Env^{\leq r}(D)$ can be computed from D in polynomial time with respect to the overall number $\sum_{t \in D} |t|$ of nodes in D, see [122]. ♢

The learner's current hypothesis grammar $\mathcal{G}_*$ is an r-SCFTG

$$\mathcal{G}(K, X, \mathcal{O}) = \langle \Sigma, N', I', P' \rangle$$

which is constructed with the aid of a membership oracle $\mathcal{O}$.
If $\mathcal{G}$ is substitutable then we let $\mathcal{O}$ be a perfect oracle for D (for example, the subtree automaton $STA(D)$), otherwise we let $\mathcal{O}$ be a perfect oracle for $\mathcal{L}(\mathcal{G})$.

[3] Recall that while there are CFGs generating the empty string the stub $\Box$ is not a valid member of any tree language generated by some SCFTG we consider. However, also note that we do enable chain rules by allowing $i = 0$ in the second case. This is in line with other literature on distributional learning – see for example [26, 29] where Clark does not disallow the nonterminals of a CFG in Chomsky normal form to derive the empty string.

Remark We do so because substitutable languages have been shown to be learnable from positive data only whereas for the weaker properties of congruentiality and the k-FKP we need a perfect membership oracle in addition. Note that as substitutability implies congruentiality which in turn implies the k-FKP obviously any learner for the k-FKP is also a learner for the other two – however, since the purpose of this synopsis is to generalize over several existing algorithms we would like to provide the learner with the minimum amount of assistance that is assumed in the literature depending on P. $\diamondsuit$

The components of $\mathcal{G}_*$ are determined as follows.

- N' is defined as the set of all subsets of K of cardinality at most k sharing the same contexts from X such that for each $m \leq r$,
  $N'_m := \{[S] \mid S \subseteq K_m \wedge 1 \leq |S| \leq k \wedge \forall e \in X_m :$
  $(\exists s \in S : \mathcal{O}(e[\![s]\!]) = 1 \Rightarrow \forall s' \in S : \mathcal{O}(e[\![s']\!]) = 1)\}$, and
- $I' := \{[S] \in N'_0 \mid \forall s \in S : \mathcal{O}(s) = 1\}$.

The production rules of $\mathcal{G}_*$ are all of the following two types, where the elimination of nonterminals and the introduction of terminals are handled by rules of Type I while the rules of Type II express all possibilities to rewrite a nonterminal into the concatenation of two other ones that we have reason to believe correct provided that the target grammar has indeed the property P we have chosen to rely on. In the following list, recall that for $\mathsf{P} \neq k\text{-}FKP$ the elements of N' are represented by singletons.

I. $[S] \to s$
   with $S \subseteq K_m$ and $s = f(\square_1, \ldots, \square_m)$ for some $f \in \Sigma_m$
   if, depending on P,
   - *(substitutable)*
     there is $e \in X_m$ such that
     $\mathcal{O}(e[\![s']\!]) = 1$ for $S = \{s'\}$ and $\mathcal{O}(e[\![s]\!]) = 1$;
   - *(congruential)*
     for all $e \in X_m$ we have $\mathcal{O}(e[\![s]\!]) = 1 \Leftrightarrow \mathcal{O}(e[\![s']\!]) = 1$
     for $S = \{s'\}$;
   - *(k-FKP)*
     for all $e \in X_m$ such that $\mathcal{O}(e[\![s']\!]) = 1$ for all $s' \in S$
     we have $\mathcal{O}(e[\![s]\!]) = 1$.

II. $[S] \to [S_1](\square_1, \ldots, \square_{m'}) \odot^j [S_2](\square_1, \ldots, \square_{m''})$
   for $S \subseteq K_m$, $S_1 \subseteq K_{m'}$, $S_2 \subseteq K_{m''}$ with $0 \leq j \leq m'$
   and $m' = m$ if $j = 0$ and $m' + m'' - 1 = m$ otherwise

if, depending on P,

- *(substitutable)*
  there is $e \in X_m$ such that $\mathcal{O}(e[\![s]\!]) = 1$ and $\mathcal{O}(e[\![s_1 \odot^j s_2]\!]) = 1$ for $S = \{s\}$, $S_1 = \{s_1\}$, and $S_2 = \{s_2\}$;
- *(congruential)*
  for all $e \in X_m$ we have $\mathcal{O}(e[\![s]\!]) = 1 \Leftrightarrow \mathcal{O}(e[\![s_1 \odot^j s_2]\!]) = 1$ for $S = \{s\}$, $S_1 = \{s_1\}$, and $S_2 = \{s_2\}$;
- *(k-FKP)*
  for all $e \in X_m$ such that $\mathcal{O}(e[\![s]\!]) = 1$ for all $s \in S$ we have $\mathcal{O}(e[\![s_1 \odot^j s_2]\!]) = 1$ for all $s_1 \in S_1$ and $s_2 \in S_2$.

Note that $\mathcal{G}_*$ is in normal form. Also observe that for all $m \geq 0$ and for each $f(\Box_1, \ldots, \Box_m) \in K_m$ with $f \in \Sigma_m$ there is a rule

$$[S] \to f(\Box_1, \ldots, \Box_m) \in P'$$

of Type I for some $S \subseteq K_m$ such that $f(\Box_1, \ldots, \Box_m) \in S$,
and for each $s \in K_m$ and each decomposition of s into two stubs $s_1 \in K_{m'}$ and $s_2 \in K_{m''}$ such that $s = s_1 \odot^j s_2$ for some $j \in \{1, \ldots, m'\}$ there is a rule

$$[S] \to [S_1](\Box_1, \ldots, \Box_{m'}) \odot^j [S_2](\Box_1, \ldots, \Box_{m''}) \in P'$$

of Type II such that $s \in S$, $s_1 \in S_1$, and $s_2 \in S_2$.
This ensures that each nonterminal $[S] \in N'$ generates at least the set $S \backslash \{\Box\}$, and that the grammar $\mathcal{G}_*$ generates at least the set $K \cap D$.
Informally stated, in addition to establishing these trivial derivabilities the learner includes into its hypothesis grammar all possible rules such that $\mathcal{G}_*$ fulfils the property P – to the best of its knowledge, i.e., under the assumption that $\mathcal{G}_*$ generates the target language and that the interrelations of the finite sets K and X within the language accepted by the oracle $\mathcal{O}$ (i.e., the set D for the substitutable case and $\mathcal{L}(\mathcal{G})$ otherwise) exactly reflect the interrelations between the substructures and contexts within the target language. To see this, observe how the definitions of rules of Type I and II above exactly reflect the respective definition of P. For example, for $\mathsf{P} =$ *substitutable* we obtain a rule of Type I if the learner finds a common environment for s and s' and can thus conclude that those two stubs are substitutable for each other, and accordingly a rule of Type II if the same occurs for s and $s_1 \odot^j s_2$. Thereby we aim to ensure that eventually each substub $s' \in \mathit{Sub}^m(\mathcal{L}(\mathcal{G}))$ for some $m \leq r$ is generated by a nonterminal $[\{s\}] \in N'_m$ such that s and s' are substitutable for each other in all contexts in $\mathit{Env}^m(\mathcal{L}(\mathcal{G}))$. For the congruential case, the learner postulates a rule of Type II if the stubs s and $s_1 \odot^j s_2$ share exactly the same contexts from X where X is the set of all contexts that the learner is currently aware of, with the intention that eventually each nonterminal

only generates stubs that share exactly the same contexts in $Env^{\leq r}(\mathcal{L}(\mathcal{G}))$, and for the k-FKP the learner postulates a rule of Type II if $s_1 \odot^j s_2$ has at least all contexts of s in X with the intention that at some point, for each nonterminal $[S] \in N'$ the set S is a k-kernel of $[S]$.

For $\mathsf{P} = \textit{substitutable}$ and the construction of a hypothesis grammar $\mathcal{G}_*$ as indicated above, one can show that $\mathcal{G}_*$ invariably fulfils

Lemma 24 $\mathcal{L}(\mathcal{G}_*) \subseteq \mathcal{L}(\mathcal{G})$

by induction on the length of derivations in $\mathcal{G}_*$, see [122].
For $\mathsf{P} = \textit{congruential}$ and $\mathsf{P} = k\text{-}FKP$ one can show a couple of monotonicity lemmata (along the lines of similar ones for strings in [118]) in the sense that

Lemma 25 *For two sets $K, K' \subseteq Sub^{\leq r}(\mathcal{L}(\mathcal{G}))$ and a set $X \subseteq Env^{\leq r}(\mathcal{L}(\mathcal{G}))$, if $K \subseteq K'$ then $\mathcal{L}(\mathcal{G}(K, X, \mathcal{O})) \subseteq \mathcal{L}(\mathcal{G}(K', X, \mathcal{O}))$.*

Proof (from [118]): Every rule of $\mathcal{G}(K, X, \mathcal{O})$ is a rule of $\mathcal{G}(K', X, \mathcal{O})$. ■

Lemma 26 *For two sets $X, X' \subseteq Env^{\leq r}(\mathcal{L}(\mathcal{G}))$ and a set $K \subseteq Sub^{\leq r}(\mathcal{L}(\mathcal{G}))$, if $X \subseteq X'$ then $\mathcal{L}(\mathcal{G}(K, X', \mathcal{O})) \subseteq \mathcal{L}(\mathcal{G}(K, X, \mathcal{O}))$.*

Proof (from [118]): Every rule of $\mathcal{G}(K, X', \mathcal{O})$ is a rule of $\mathcal{G}(K, X, \mathcal{O})$. ■

Remark Note that this break between substitutability and the weaker two properties is due to the quantifiers in the definitions of rules of Type I and II: Whereas in the substitutable case the learner can establish a rule as soon as it has found a single common context for two corresponding stubs and will not have to delete that rule again, in the other two cases the universal quantifier may be contradicted when more data is available (also see [30] where Clark distinguishes between *a priori* rules, *certain* rules, and *defeasible* rules). ♢

We will now describe a generic learner for the three properties above in order to illustrate the principle of a distributional learning algorithm taking the primal approach. As in the references [122, 118], the chosen learning model will be identification in the limit from a continuous stream of positive data for the target grammar $\mathcal{G}$ such that each element of $L := \mathcal{L}(\mathcal{G})$ is given to the learner at least once, and a membership oracle which for substitutability we will instantiate with an oracle that exactly predicts the set of data seen so far, and with a perfect membership oracle $\mathcal{O}_L$ for L in the other cases.

Remark Note that we generalize the setting of learning from positive data to learning from positive data and any membership oracle by letting the learner build its own oracle in the substitutable case – this can be related to our way of generalizing the algorithm RPNI [41, 93] in Section 3.3. ♢

We let the continuous stream of data divide time into distinct steps and we assume that the learning process starts at time step 0 whereas the first datum is received by the learner at time step 1. A learner is said to identify a target language L in the limit from a continuous stream of data that includes all elements of L, possibly with the aid of membership queries, if there is a time step n such that for all n' with $n \leq n'$ the learner's hypothesis grammar at step n' equals the one for step n and moreover generates L.

We assume that r is fixed. The learner maintains a triple $T = \langle D, K, X \rangle$ where D is the set of data seen so far and $K \subseteq Sub^{\leq r}(D)$ and $X = Env^{\leq r}(D)$. By $T_i = \langle D_i, K_i, X_i \rangle$ we denote the triple T between time steps i and $i+1$. The learner starts out with $T_0 = \langle \emptyset, \emptyset, \emptyset \rangle$. At each time step i with $i > 0$ the learner receives a datum s_i. If $\mathsf{P} =$ *substitutable* then the oracle is updated to $\mathcal{O}_i$ such that it exactly predicts $D_i = D_{i-1} \cup \{s_i\}$, otherwise we let $\mathcal{O}_i = \mathcal{O}_L$. The learner then calls the procedure below using $T_{i-1} = \langle D_{i-1}, K_{i-1}, X_{i-1} \rangle$, datum s_i, and the oracle $\mathcal{O}_i$ as input.[4] From the resulting triple $\langle D', K', X' \rangle$ the learner constructs $\mathcal{G}_i = \mathcal{G}(K_i, X_i, \mathcal{O}_i)$ with $K_i = K'$ and $X_i = X'$.

Input: `A triple` $\langle D, K, X \rangle$`, a datum` s`, a membership oracle` $\mathcal{O}$`.`
Output: `A triple` $\langle D', K', X' \rangle$`.`

1 $D' := D \cup \{s\}$; $X' := Env^{\leq r}(D')$;
2 **if** $D' \notin \mathcal{L}(\mathcal{G}(K, X', \mathcal{O}))$ **then** $K' := Sub^{\leq r}(D')$; **else** $K' := K$;
3 **return** $\langle D', K', X' \rangle$.

Let $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ be the target grammar.
In the following, we separately sketch sufficient conditions to ensure that the learner correctly identifies $L := \mathcal{L}(\mathcal{G})$ for $\mathsf{P} =$ *substitutable* on the one hand and for $\mathsf{P} =$ *congruential* or $\mathsf{P} = k$-*FKP* on the other.

Let $\mathsf{P} =$ *substitutable*. For each $m \geq 0$ and each nonterminal $A \in N_m$, let $e_A \in Env^m(L)$ be the minimal environment with respect to the tree ordering relation $\preceq$ (for some arbitrary lexical value of $\square$, see Subsection 3.3.1) such that $B \Rightarrow_{\mathcal{G}}^* e_A[\![A(\square_1, \ldots, \square_m)]\!]$ for some $B \in I$, and let $s_A \in Sub^m(L)$ be the minimal stub with respect to $\preceq$ such that $A \Rightarrow_{\mathcal{G}}^* s_A$. Define

$$D_{\mathcal{G}} := \{e_A[\![f(\square_1, \ldots, \square_m)]\!] \mid A \to f(\square_1, \ldots, \square_m) \in P \wedge f \in \Sigma_m\} \cup \{e_A[\![s_B \odot^j s_C]\!] \mid A \to B(\square_1, \ldots, \square_{m'}) \odot^j C(\square_1, \ldots, \square_{m''}) \in P \wedge B, C \in N\}.$$

[4] Clark and Yoshinaka often let the data stream as a whole be part of the input which gives their learners an a little different outward appearance but of course strictly speaking input for a computable algorithm cannot be infinite – rather, at each step the learner calls a well-defined subroutine to process the finite amount of data that is available at the time.

Observe that $D_{\mathcal{G}}$ is at most of cardinality $|P|$.
Our learner fulfils the following lemma shown in [122]:

Lemma 27 *At any time step $i \geq 0$ such that $D_{\mathcal{G}} \subseteq D_i$ and $Sub^{\leq r}(D_{\mathcal{G}}) \subseteq K_i$ we have $L \subseteq \mathcal{L}(\mathcal{G}_i)$.*

Proof (from [122]). We show that for each nonterminal A of $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ there is a nonterminal $[\{s_A\}]$ of $\mathcal{G}_i = \langle \Sigma, N^i, I^i, P^i \rangle$ simulating A. Recall that we assume the target grammar $\mathcal{G}$ to be in normal form.
If $A \in I$ then $s_A \in D_i$ and thus $[\{s_A\}] \in I^i$.
For each rule $A \to f(\square_1, \ldots, \square_m) \in P$ for some $f \in \Sigma_m$ we have

$$e_A[\![s_A]\!], e_A[\![f(\square_1, \ldots, \square_m)]\!] \in D_{\mathcal{G}}.$$

Then by definition, there is a rule $[\{s_A\}] \to f(\square_1, \ldots, \square_m)$ of Type I in P^i.
Let $A \to B(\square_1, \ldots, \square_{m'}) \odot^j C(\square_1, \ldots, \square_{m''}) \in P$. We have

$$e_A[\![s_A]\!], e_A[\![s_B \odot^j s_C]\!] \in D_{\mathcal{G}}.$$

Then by definition, there is a corresponding rule

$$[\{s_A\}] \to [\{s_C\}](\square_1, \ldots, \square_{m'}) \odot^j [\{s_C\}](\square_1, \ldots, \square_{m''})$$

of Type II in $\mathcal{G}_i$. Thus, every derivation in $\mathcal{G}$ can be reproduced in $\mathcal{G}_i$. ■

Since in the substitutable case the learner never overgeneralizes (Lemma 24), by the first time step i at which the set D_i of data seen so far contains $D_{\mathcal{G}}$ as a subset and enough positive counterexamples such that $Sub^{\leq r}(D_{\mathcal{G}}) \subseteq K_i$ the learner's hypothesis is correct due to Lemma 27. Note that such a time step must exist since we assume that in the continuous data stream each element of L appears at least once. Moreover, from that step onwards the learner does not change the set K anymore due to the test in line 2 and Lemma 24, and as the rules of the hypothesis cannot be contradicted by an increasing set X due to the existential quantifiers in their definitions the learner's hypothesis will not change anymore either. Consequently, the learner correctly identifies L in the limit. For a sketch of an example run in the substitutable case, see Example 3 from [122] below.

For $\mathsf{P} = \textit{congruential}$ and $\mathsf{P} = k\textit{-FKP}$ one can formulate sufficient conditions for the exclusion of undergeneralization and overgeneralization as well. In order to avoid the former, the learner has to be provided with a sufficient amount of material from which to establish its nonterminals.

- For congruentiality, we say that a finite set $D_{\mathcal{G}} \subseteq L$ is $\mathcal{G}$-*representative* if in order to reproduce all elements of $D_{\mathcal{G}}$ each rule of P has to be applied at least once.
- For the k-FKP, we say that a finite set $D_{\mathcal{G}} \subseteq L$ is $\mathcal{G}$-*representative* if for each $A \in N$, some subset of $Sub^{\leq r}(D_{\mathcal{G}})$ is a k-kernel of A.

Lemma 28 (in analogy to [120]/[118]) *At any time step $i \geq 0$ such that $D_{\mathcal{G}} \subseteq D_i$ and $Sub^{\leq r}(D_{\mathcal{G}}) \subseteq K_i$ for some $\mathcal{G}$-representative set $D_{\mathcal{G}}$, we have $L \subseteq \mathcal{L}(\mathcal{G}_i)$.*

Remark The proofs in [120] (congruentiality) and [118] (k-FKP) are given for hypothesis grammars from the classes of MCFGs and CFGs, respectively, but due to our definitions of stubs and environments an adaptation to trees is not difficult. All proofs in question are based on induction over the length of derivations in the target grammar. Also note that [120] assumes a different learning setting but at this stage the way the sets K and X are obtained is irrelevant for the construction of the hypothesis. ♢

For the exclusion of overgeneralization, one can show the following.
We say that a rule in a hypothesis grammar $\mathcal{G}(K, X, \mathcal{O})$ constructed as specified above for two finite sets $K \subseteq Sub^{\leq r}(L)$ and $X \subseteq Env^{\leq r}(L)$ is *incorrect* if there is an environment $e \in Env^{\leq r}(L) \setminus X$ contradicting that rule.

Lemma 29 (in analogy to [118]/[26])
For any finite set $K \subseteq Sub^{\leq r}(L)$ one can find a finite set $X \subseteq Env^{\leq r}(L)$ such that $\mathcal{G}_* := \mathcal{G}(K, X, \mathcal{O})$ does not contain any incorrect rules.
As a consequence, $\mathcal{G}_*$ fulfils $\mathcal{L}(\mathcal{G}_*) \subseteq L$ (compare [118]).
Moreover, the same is true for all $\mathcal{G}(K, X', \mathcal{O})$ with $X' \subseteq Env^{\leq r}(L)$ such that $X \subseteq X'$ (compare [26], Lemma 6).

Remark The existence of such a set X for a given K follows directly from the definition of $\mathcal{G}_*$: For every combination of at most three nonterminals derived from K, a single context is enough to prevent a rule involving them. Thus, if we let N' be the set of nonterminals postulated by the learner then it is easy to see that there is a suitable context set X of cardinality at most $(r|\Sigma| + 1)|N'| + r|N'|^3$ by considering the respective definitions of the two admissible types of rules I and II. ♢

From the explanations above we can conclude that our learner identifies the language generated by the grammar $\mathcal{G}$ after a finite number of steps in the congruential and the k-FKP case as well: There will be a first time step i at which the set of data seen so far contains a $\mathcal{G}$-representative subset $D_{\mathcal{G}}$ and enough counterexamples such that we have $Sub^{\leq r}(D_{\mathcal{G}}) \subseteq K_i$. From that point, the learner does not expand the set K anymore due to the test in the second line of the subroutine and Lemma 28. At some later step i' with $i' \geq i$, the set $D_{i'}$ will contain a corresponding set X for that specific K such that any incorrect rules are excluded by Lemma 29 and the learner's hypothesis will be correct. Due to the second part of Lemma 29 the hypothesis will not change anymore either, and consequently the learner identifies L in the limit.

In analogy to [122, 118] the learner can be shown to construct its hypothesis in a polynomially bounded number of steps with respect to the overall number of nodes in D for all possible values of the property P.

Let $n_D := \sum_{t \in D} |t|$ and $n_S := \sum_{t \in Sub^{\leq r}(D)} |t|$ and $n_E := \sum_{t \in Env^{\leq r}(D)} |t|$.

As stated in [122], $Sub^{\leq r}(D)$ and $Env^{\leq r}(D)$ can be computed from D in a polynomial number of steps with respect to n_D where the degree of the polynomial linearly depends on r, and n_S and n_E are bounded by ${n_D}^{r+1}$. Moreover, the cardinalities of K and X are bounded by $|Sub^{\leq r}(D)|$ and $|Env^{\leq r}(D)|$, respectively, and an upper bound for the number of nonterminals that the learner derives from K is $|K+1|^k$ (the number of subsets of K of cardinality at most k). As a consequence, only a polynomial number of MQs with respect to n_D is needed to compute all possible rules for the set of nonterminals currently postulated by the learner (and in order to improve performance in practice we would store the results of MQs such that no tree has to be queried twice). Furthermore, one can show that the membership problem for a given tree t and an r-SCFTG $\mathcal{G}_*$ can be decided in polynomial time with respect to $|t|$ and to the number of nonterminals and rules in $\mathcal{G}_*$, see [122]. All in all, the learner constructs its conjecture in a polynomial number of steps with respect to n_D.

Remark We have assumed that the parameter r is fixed. Alternatively, we can also derive r from the given data but then r depends on n_D as well which would lead to an exponential overall complexity with respect to n_D. ♢

We also remark that in the substitutable case the size of the set $D_{\mathcal{G}}$ for the target grammar $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ is bounded by $|P|$ and that the size of the elements of $D_{\mathcal{G}}$ is minimal by definition, and in the other two cases there always exists a set of at most $k|N|$ examples which suffices to provide the learner with a kernel for each nonterminal in N, and a set of at most $|N|^3$ environments which suffices to make the learner suppress all incorrect rules.

Example 3 *Let $\mathcal{G} = \langle \Sigma_0 \cup \Sigma_3, N_0 \cup N_3, I, P \rangle$ be the target grammar where*

- $\Sigma_0 = \{a, b, c, d, f\}$, $\Sigma_3 = \{g, h\}$,
- $N_0 = \{A\}$, $N_3 = \{B\}$, *and*
- *P consists of the following three rules (also see Figure 4.2):*
  $A \to g(a, B(b, f, c), d)$
  $B \to h(\square, \square, \square)$
  $B \to g(a, B(b, g(\square, \square, \square), c), d)$.

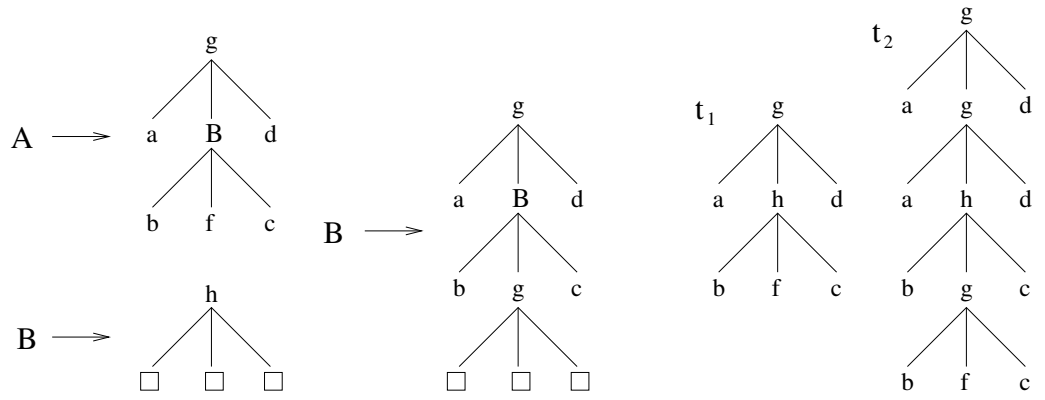

Figure 4.2: The rules of $\mathcal{G}$ from Example 3, and a sample of two trees

$\mathcal{G}$ is not in normal form but since the learner is only required to generate a grammar equivalent to $\mathcal{G}$ we choose this representation for compactness.
Suppose the learner gets t_1 in Figure 4.2 as its first datum. The learner will react by constructing a grammar $\mathcal{G}_1 = \{\Sigma, N^1, I^1, P^1\}$ as specified by the subroutine above where $I^1 = [\{t_1\}]$ and all rules in P^1 only serve to recompose t_1 from its substructures. Thus, $\mathcal{G}_1$ generates only t_1 itself.
Suppose the next datum is t_2 in Figure 4.2. This results in several additional nonterminals and one more start symbol $[\{t_2\}]$. The learner observes that the stubs $h(\Box,\Box,\Box)$ and $g(a,h(b,g(\Box,\Box,\Box),c),d)$ are substitutable for each other due to the environment $e = \langle g(a,\Box,d), \langle b,f,c\rangle\rangle$ since we have

$$e[\![h(\Box,\Box,\Box)]\!] = t_1 \in D_2, \text{ and}$$
$$e[\![g(a,h(b,g(\Box,\Box,\Box),c),d)]\!] = t_2 \in D_2.$$

The learner accordingly constructs a rule

$$[\{h(\Box,\Box,\Box)\}] \to [\{g(a,\Box,d)\}](\Box) \odot^1 [\{h(b,g(\Box,\Box,\Box),c)\}](\Box,\Box,\Box)$$

based on the decomposition

$$g(a,h(b,g(\Box,\Box,\Box),c),d) = g(a,\Box,d) \odot^1 h(b,g(\Box,\Box,\Box),c).$$

The successive application of trivial rules results in a partial derivation

$$\begin{aligned}[\{h(\Box,\Box,\Box)\}] &\Rightarrow_{\mathcal{G}_2} [\{g(a,\Box,d)\}]([\{h(b,g(\Box,\Box,\Box),c)\}](\Box,\Box,\Box)) \\ &\Rightarrow^*_{\mathcal{G}_2} g(a,[\{h(\Box,\Box,\Box)\}](b,g(\Box,\Box,\Box),c),d)\end{aligned}$$

which simulates the rule $B \to g(a,B(b,g(\Box,\Box,\Box),c),d)$ in $\mathcal{G}$, thus enabling the generation of infinitely many trees. As a consequence, $\mathcal{L}(\mathcal{G}_2) = \mathcal{L}(\mathcal{G})$.

4.2.2 A dual learning algorithm

In this subsection we present a distributional learning algorithm for SCFTGs with the x-FCP in our own notation which was previously published in [122]. In principle, the architecture of such an algorithm taking the dual approach is quite symmetric to those taking the primal one for the k-FKP but at some essential points the roles of substructures and environments are switched.

As in the subsection above, we fix an r-SCFTG $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ for some given $r \in \mathbb{N}$ to represent our learning target. We assume that $\mathcal{G}$ is in normal form, and we assume that no nonterminal or rule is unreachable or useless. Moreover, we assume that $\mathcal{G}$ fulfils the x-FCP. Again, we will first indicate how the learner constructs its hypothesis from the currently available data, and then describe the learning algorithm itself. We abbreviate $\mathcal{L}(\mathcal{G})$ to L. Suppose the learner is aware of the target fulfilling the x-FCP and has access to a finite positive data set $D \subseteq L$, and suppose that it has derived two sets $K \subseteq Sub^{\leq r}(D)$ and $X \subseteq Env^{\leq r}(D)$ of substructures and contexts from D. Suppose that K fulfils the condition $Sub^{\leq r}(\{s\}) \subseteq K$ for all $s \in K$.
The learner's current hypothesis grammar $\mathcal{G}_*$ is an r-SCFTG

$$\mathcal{G}(K, X, \mathcal{O}) = \langle \Sigma, N', I', P' \rangle$$

which is constructed with the aid of a perfect membership oracle $\mathcal{O}$ for L. The components of $\mathcal{G}_*$ are determined as follows.

- N' is defined as the set of all subsets of X of cardinality at most x extending the same stubs from K into L such that for each $m \leq r$,
  $N'_m := \{[E] \mid E \subseteq X_m \wedge 1 \leq |E| \leq x \wedge \forall s \in K_m :$
  $(\exists e \in E : \mathcal{O}(e[\![s]\!]) = 1 \Rightarrow \forall e' \in E : \mathcal{O}(e'[\![s]\!]) = 1)\}$, and
- $I' := \{[E] \in N'_0 \mid \langle \square, \langle\rangle \rangle \in E\}$.

The production rules of $\mathcal{G}_*$ are all of the following two types, where the elimination of nonterminals and the introduction of terminals are handled by rules of Type I while the rules of Type II express all possibilities to rewrite a nonterminal into the concatenation of two other ones that we have reason to believe correct provided that the target grammar has indeed the x-FCP.

I. $[E] \to f(\square_1, \ldots, \square_m)$ with $E \subseteq X_m$ and $f \in \Sigma_m$
if we have $\mathcal{O}(e[\![f(\square_1, \ldots, \square_m)]\!]) = 1$ for all $e \in E$;

II. $[E] \to [E_1](\square_1, \ldots, \square_{m'}) \odot^j [E_2](\square_1, \ldots, \square_{m''})$
for $E \subseteq X_m$, $E_1 \subseteq X_{m'}$, $E_2 \subseteq X_{m''}$ with $0 \leq j \leq m'$
and $m' = m$ if $j = 0$ and $m' + m'' - 1 = m$ otherwise if,

for all $e \in E$ and all $s_1, s_2 \in K$ such that
$\mathcal{O}(e_1[\![s_1]\!]) = 1$ and $\mathcal{O}(e_2[\![s_2]\!]) = 1$ for all $e_1 \in E_1$ and $e_2 \in E_2$,
we have $\mathcal{O}(e[\![s_1 \odot^j s_2]\!]) = 1$.

The learner postulates a rule of Type II if the contexts in E are positive for every concatenation of two stubs s_1 and s_2 from the set K such that all contexts from E_1 and E_2 are positive for s_1 and s_2, respectively. Thereby the learner aims to ensure that eventually each nonterminal $[E]$ in its hypothesis grammar $\mathcal{G}_*$ only generates a stub s if s has at least all contexts in the set E such that as a consequence $\mathcal{G}_*$ fulfils the x-FCP – assuming that the learner's hypothesis correctly generates the target language L. Observe that although in this case the nonterminals can be *characterized* by sets of contexts, the languages that they generate still consist of stubs, and not of environments. Accordingly, one can show a couple of monotonicity lemmata for the dual approach (along the lines of similar ones for strings in [118]) stating that

Lemma 30 *For two sets $X, X' \subseteq Env^{\leq r}(\mathcal{L}(\mathcal{G}))$ and a set $K \subseteq Sub^{\leq r}(\mathcal{L}(\mathcal{G}))$, if $X \subseteq X'$ then $\mathcal{L}(\mathcal{G}(K, X, \mathcal{O})) \subseteq \mathcal{L}(\mathcal{G}(K, X', \mathcal{O}))$.*

Lemma 31 *For two sets $K, K' \subseteq Sub^{\leq r}(\mathcal{L}(\mathcal{G}))$ and a set $X \subseteq Env^{\leq r}(\mathcal{L}(\mathcal{G}))$, if $K \subseteq K'$ then $\mathcal{L}(\mathcal{G}(K, X, \mathcal{O})) \subseteq \mathcal{L}(\mathcal{G}(K', X, \mathcal{O}))$.*

Observe the exact symmetry to Lemmata 25 and 26. The proofs run parallel to the ones for Lemmata 25 and 26 as well.

We describe a learning algorithm for the x-FCP that identifies the target language in the limit from a stream of positive data and membership queries.

We assume that r is fixed. The learner maintains a triple $T = \langle D, K, X \rangle$ where D is the set of data seen so far and $K = Sub^{\leq r}(D)$ and $X \subseteq Env^{\leq r}(D)$. By $T_i = \langle D_i, K_i, X_i \rangle$ we denote the triple T between time steps i and $i+1$. The learner starts out with $T_0 = \langle \emptyset, \emptyset, \emptyset \rangle$. At each time step i with $i > 0$ the learner receives a datum s_i. The learner then calls the procedure below for $T_{i-1} = \langle D_{i-1}, K_{i-1}, X_{i-1} \rangle$, datum s_i, and oracle $\mathcal{O}$, and from the output $\langle D', K', X' \rangle$ constructs $\mathcal{G}_i = \mathcal{G}(K_i, X_i, \mathcal{O})$ with $K_i = K'$ and $X_i = X'$.

Input: `A triple` $\langle D, K, X \rangle$`, a datum` s`, a membership oracle` $\mathcal{O}$.
Output: `A triple` $\langle D', K', X' \rangle$.

1 $D' := D \cup \{s\}$; $K' := Sub^{\leq r}(D')$;
2 **if** $D' \not\subseteq \mathcal{L}(\mathcal{G}(K', X, \mathcal{O}))$ **then** $X' := Env^{\leq r}(D')$; **else** $X' := X$;
3 **return** $\langle D', K', X' \rangle$.

Again, note the exact symmetry to the learner from the previous subsection. Let $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ be the target grammar. We sketch sufficient conditions

for the exclusion of undergeneralization and overgeneralization, respectively. As with the primal approach, to avoid the former the learner must have access to a sufficient amount of material from which to establish its nonterminals. Under the dual approach, we say that a finite set $D_{\mathcal{G}} \subseteq L$ is *$\mathcal{G}$-representative* if for each $A \in N$, some subset of $Env^{\leq r}(D_{\mathcal{G}})$ is an x-context of A.

Lemma 32 (in analogy to [118]) *At any time step $i \geq 0$ such that $D_{\mathcal{G}} \subseteq D_i$ and $Env^{\leq r}(D_{\mathcal{G}}) \subseteq X_i$ for some $\mathcal{G}$-representative set $D_{\mathcal{G}}$, for some $\mathcal{G}$-representative set $D_{\mathcal{G}}$, we have $L \subseteq \mathcal{L}(\mathcal{G}_i)$.*

In order to avoid overgeneralization the learner needs enough information to suppress incorrect rules. Under the dual approach, we say that a rule in a hypothesis grammar $\mathcal{G}(K, X, \mathcal{O})$ constructed as specified above for two finite sets $K \subseteq Sub^{\leq r}(L)$ and $X \subseteq Env^{\leq r}(L)$ is *incorrect* if there is a stub $s \in Sub^{\leq r}(L) \setminus K$ contradicting that rule. One can show the following:

Lemma 33 (in analogy to [118])
For any finite set $X \subseteq Env^{\leq r}(L)$ one can find a finite set $K \subseteq Sub^{\leq r}(L)$ such that $\mathcal{G}_* := \mathcal{G}(K, X, \mathcal{O})$ does not contain any incorrect rules.
As a consequence, $\mathcal{G}_*$ fulfils $\mathcal{L}(\mathcal{G}_*) \subseteq L$ (compare [118]).
The same is true for all $\mathcal{G}(K', X, \mathcal{O})$ with $K' \subseteq Sub^{\leq r}(L)$ such that $K \subseteq K'$.

Remark If we let N' be the set of nonterminals that the learner has derived from X then there is such a set K of cardinality at most $(r|\Sigma|+1)|N'|+r|N'|^3$ by a similar argument as for the primal case in Subsection 4.2.1 above. $\diamondsuit$

We can conclude that this dual learner identifies the language generated by the target grammar $\mathcal{G}$ after a finite number of steps as well: There will be a first step i at which the set of data seen so far contains a $\mathcal{G}$-representative subset $D_{\mathcal{G}}$ and enough counterexamples such that we have $Env^{\leq r}(D_{\mathcal{G}}) \subseteq X_i$. From that point on, the learner does not expand X anymore due to the test in the second line of its subroutine and Lemma 32. At some later step i' with $i' \geq i$, the data set $D_{i'}$ will contain a corresponding set K for that specific X such that any incorrect rules are excluded by Lemma 33, and the learner's hypothesis will be correct. This hypothesis will not be changed anymore due to the second part of Lemma 33. Thus, we have identification in the limit.

Remark The learner for CFGs with the x-FCP given in [28] only uses a set of contexts to represent a nonterminal if it fulfils an equivalent of the condition for N' in the construction of $\mathcal{G}_*$ above and is *of maximal cardinality*. In [118], Yoshinaka observes that under the primal approach it is possible to construct a suitable hypothesis grammar using only maximal sets of substructures fulfilling an equivalent of the condition for N' in Subsection 4.2.1 as well but

that these maximality conditions require modified strategies which moreover differ from each other for the two approaches in a way such that the parallels between the respective construction schemes for the learner's hypothesis and the monotonicity lemmata are lost. ♢

Concerning complexity one can show that this learner constructs its hypothesis in a polynomially bounded number of steps with respect to the overall number of nodes in D as well – an upper bound for the number of nonterminals that the learner derives from X is $|X|^x$ and the rest of the argument runs parallel to the one for the primal case in Subsection 4.2.1 above. Moreover, there always exists a set of at most $x|N|$ examples which suffices to provide the learner with an x-context for each nonterminal in $\mathcal{G}$, and a set of at most $|N|^3$ substructures to make the learner suppress all incorrect rules.

4.2.3 One-shot learning settings

As in most of the earlier work by Clark and Yoshinaka on distributional learning, the learning model of choice in the previous two subsections was identification in the limit from positive data and (possibly) membership queries. Corresponding algorithms have been shown to identify a target language L correctly as soon as the set of data seen so far meets certain conditions with respect to some reference grammar $\mathcal{G}$ with $\mathcal{L}(\mathcal{G}) = L$. However, also observe that $\mathcal{G}$ is not defined in a canonical way: Any grammar for L in normal form fulfilling the distributional property under consideration will do.

In this subsection we consider distributional learning in the one-shot settings that were of interest in Chapter 3, i.e., learning from MQs and EQs, from MQs and a finite positive sample, and from a finite positive and a finite negative sample. In particular, we will describe an algorithm for the inference of a congruential SCFTG from MQs and EQs based on similar ones for CFGs and MCFGs in [26] and [120], and two algorithms that infer a congruential or a context-deterministic SCFTG from a positive and a negative finite sample based on an unpublished draft [36] for CFGs.

First of all, we observe that considering the conditions for the exclusion of under- and overgeneralization stated in Subsection 4.2.1, the development of an algorithm that infers r-SCFTGs with $\mathsf{P} \in \{$*substitutable, congruential, k-FKP*$\}$ from MQs and a finite positive sample is straightforward: Let L be the target language. Present the learner with a suitable set $D \subseteq L$ of positive data with respect to some reference grammar $\mathcal{G}$ generating L, i.e., D must be such that $D_{\mathcal{G}} \subseteq D$ and $Env^{\leq r}(D)$ contains a suitable set of environments. The learner will construct a hypothesis $\mathcal{G}(K, X, \mathcal{O})$ from $K := Sub^{\leq r}(D)$ and $X := Env^{\leq r}(D)$ as specified in Subsection 4.2.1 using a polynomial number

of MQs, and the resulting grammar must correctly generate the target L by the corresponding Lemmata 24, 27, 28, and 29. A dual learner for the x-FCP can be defined in an analogous way, relying on Lemmata 32 and 33.
However, the conditions on D in both cases are rather restrictive – also see the remarks on complexity at the end of the part on MQs and EQs below.

MQs and EQs

Distributional learning results for the setting joining MQs and EQs can be found in [107, 26, 120]. In the following we will sketch a primal distributional learner that infers a congruential SCFTG from MQs and EQs along the lines of comparable algorithms for congruential (M)CFGs as described in [26, 120].

For the explanations below, we recursively define an *A-derivation tree* for an r-SCFTG $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ in normal form and a nonterminal $A \in N_m$ as a tree $\tau \in \mathbb{T}_P$ where the nodes are labeled with rules from P such that

- $\tau = \pi \in P$ with $\pi = A \to f(\square_1, \ldots, \square_m)$ for some $f \in \Sigma_m$, or
- $\tau = \pi(\tau_0)$ with $\pi = A \to B(\square_1, \ldots, \square_m) \in P$ and τ_0 is a B-derivation tree,[5] or
- $\tau = \pi(\tau_1, \tau_2)$ with $\pi = A \to B(\square_1, \ldots, \square_{m'}) \odot^j C(\square_1, \ldots, \square_{m''}) \in P$ and $j \geq 1$ and τ_1 is a B-derivation tree and τ_2 is a C-derivation tree.

The element $s \in \mathcal{L}_A$ thus derived, denoted by $\gamma\delta(\tau)$, is called the *yield* of τ.

Let $r \in \mathbb{N}$ be fixed, and let L be the target language generated by some congruential r-SCFTG. The learner has access to an equivalence oracle that answers queries of the form '$\mathcal{L}(\mathcal{G}_*) = L$?' correctly for any r-SCFTG $\mathcal{G}_*$, and to a perfect membership oracle for L.
Basically, the learner constructs a hypothesis $\mathcal{G}_* = \mathcal{G}(K, X, \mathcal{O})$ from two sets $K \subseteq \mathbb{S}_\Sigma$ and $X \subseteq \mathbb{E}_\Sigma$ of substructures and contexts as for $\mathsf{P} = \textit{congruential}$ in Subsection 4.2.1 but we will use a simplified scheme corresponding to the one in [120] for the sake of readability. Thus, in the grammar $\mathcal{G}_* = \langle \Sigma, N', I', P' \rangle$,

- N' is defined by $N'_m := \{[s] \mid s \in K_m\}$ for each $m \leq r$, and
- $I' := \{[s] \in N'_0 \mid \mathcal{O}(s) = 1\}$.

The production rules of $\mathcal{G}_*$ are all of the following three types.

I'. $[s] \to s$ with $s \in K_m$ and $s = f(\square_1, \ldots, \square_m)$ for some $f \in \Sigma_m$,

[5] Observe that rules of this form fulfil condition (b) in our definition of the normal form for SCFTGs in Subsection 4.2.1 since we admit a concatenation operation $\odot^j$ with $j = 0$.

II'. $[s] \to [s_1](\square_1, \ldots, \square_{m'}) \odot^j [s_2](\square_1, \ldots, \square_{m''})$
with $s \in K_m$, $s_1 \in K_{m'}$, $s_2 \in K_{m''}$
for $1 \leq j \leq m'$ and $m' + m'' - 1 = m$, and $s_1 \odot^j s_2 = s$,

III'. $[s] \to [s'](\square_1, \ldots, \square_m)$ for $s, s' \in K_m$ if $\forall e \in X_m : \mathcal{O}(e[\![s]\!]) = \mathcal{O}(e[\![s']\!])$.

This is essentially the same but instead of integrating the conjectured exchangeabilities of stubs from K into rules of all types they are now uniquely handled by rules of Type III' whereas the other types are based on identity. Also note that for each rule $[s] \to [s'] \in P$ we also have $[s'] \to [s] \in P$ by the definition of Type III'. This corresponds to the merging of nonterminals. The results expressed in Lemmata 25, 26, 28, and 29 stay preserved.
What necessarily differs in this setting is the way how the learner obtains the sets K and X. The learner starts out with $K := \emptyset$ and $X := \{\langle\square, \langle\rangle\rangle\}$, which results in a hypothesis $\mathcal{G}_*$ such that $\mathcal{L}(\mathcal{G}_*) = \emptyset$. For each of its hypotheses, the learner asks an equivalence query. Positive counterexamples are used to expand K in the same way as the elements of the continuous data stream in the previous two subsections. Negative counterexamples are used to eliminate incorrect rules – on the receipt of a negative counterexample c the learner calls a subroutine which will expand the set X until c is not generated anymore by the grammar $\mathcal{G}(K, X, \mathcal{O})$.
The learner's pseudocode is given below. As for GENDET in Subsection 3.3.1 we model the equivalence oracle as a blackbox routine that returns the undefined tree $\square$ if the learner's hypothesis is correct.

Input: `Membership oracle` $\mathcal{O}$`, equivalence oracle EQ, a value` $r \in \mathbb{N}$.
Output: `An` r`-SCFTG.`

```
1    K := ∅;  X := {⟨□, ⟨⟩⟩};  G* := G(K, X, O);
2    while EQ(G*) ≠ □ do
3        c := EQ(G*);
4        if c ∉ L(G*) then K := K ∪ Sub^{≤r}({c});
5        else X := X ∪ ADDCONTEXTS(K, X, c);
6        G* := G(K, X, O);
7    return G*.
```

If the learner receives a negative counterexample $c \notin L$ in line 3 then this reveals that its current hypothesis grammar $\mathcal{G}_*$ must contain at least one incorrect rule of Type III' which has been established under the assumption that two given stubs are equivalent under $\equiv_L$ (see the definition of Type III'). Hence, for each such rule involved in some derivation of c the learner must find a context e that disproves the equivalence and thus the validity of the rule,

and add it to X. This is handled by the procedure ADDCONTEXTS.

---

procedure `ADDCONTEXTS`(K, X, c) `[`$K \subseteq \mathbb{S}_\Sigma$`,` $X \subseteq \mathbb{E}_\Sigma$`,` $c \in \mathbb{T}_\Sigma$`]`

8 $\quad \mathcal{G}_* := \mathcal{G}(K, X, \mathcal{O}) = \langle \Sigma, N', I', P' \rangle$;

9 $\quad$ **while** $c \in \mathcal{L}(\mathcal{G}_*)$ **do**

10 $\quad\quad$ `find an` A`-derivation tree` τ `for some` $A \in I'$ `with` $\gamma\delta(\tau) = c$;

11 $\quad\quad e :=$ `FINDCONTEXT`$(K, X, \tau, c, \langle \Box, \langle\rangle \rangle)$;

12 $\quad\quad X := X \cup \{e\}$;

13 $\quad\quad \mathcal{G}_* := \mathcal{G}(K, X, \mathcal{O})$;

14 $\quad$ **return** X.

Since $\mathcal{G}_* = \langle \Sigma, N', I', P' \rangle$ wrongly generates c there exists at least one initial symbol $[s] \in I'$ for some $s \in K$ and an $[s]$-derivation tree τ_s with $\gamma\delta(\tau_s) = c$, and we have $s = \langle \Box, \langle\rangle \rangle [\![s]\!] \in L$ by the definition of I' but $c = \langle \Box, \langle\rangle \rangle [\![c]\!] \notin L$. Any such derivation tree τ_s must contain an incorrect rule, and the learner calls the procedure FINDCONTEXT for τ_s and c and the environment $\langle \Box, \langle\rangle \rangle$ in line 11 in order to search τ_s recursively in a top-down manner for such a rule and a context invalidating it. We will describe this procedure in prose (also compare [120]). FINDCONTEXT receives an $[s_a]$-derivation tree τ_a for some nonterminal $[s_a] \in N'_m$ with $m \geq 0$, a stub $s \in \mathbb{S}^m_\Sigma$ and an environment $e = \langle s_e, \langle t_1, \ldots, t_m \rangle \rangle \in \mathbb{E}^m_\Sigma$ such that $\gamma\delta(\tau_a) = s$ and $e[\![s_a]\!] \in L$ but $e[\![s]\!] \notin L$. We have the following case distinction.

- Let $\tau_a = \pi(\tau_b, \tau_c)$. Then π is of the form
  $$[s_a] \to [s_b](\Box_1, \ldots, \Box_{m'}) \odot^j [s_c](\Box_1, \ldots, \Box_{m''})$$
  for $[s_b], [s_c] \in N'$ with $s_b \odot^j s_c = s_a$, and, as τ_a is a derivation tree for s, there are two stubs $s' \in \mathbb{S}^{m'}_\Sigma$, $s'' \in \mathbb{S}^{m''}_\Sigma$ with $s' \odot^j s'' = s$ such that τ_b is an $[s_b]$-derivation tree for s' and τ_c is an $[s_c]$-derivation tree for s''. By the precondition for e we have $e[\![s_b \odot^j s_c]\!] \in L$ but $e[\![s' \odot^j s'']\!] \notin L$. If s_b was congruent to s' and s_c to s'' then $s_a = s_b \odot^j s_c$ and $s = s' \odot^j s''$ would share the context e and thus either s_b is not congruent to s' or s_c is not congruent to s'' or both. Accordingly, if we have $e[\![s_b \odot^j s'']\!] \in L$ then we know that there is a separating context $e' \in \mathbb{E}^{m'}_\Sigma$ for s_b and s' of the form $\langle s_e, \langle s_1, \ldots, s_{m'} \rangle \rangle$ where $s_l = t_l$ for $l < j$,
  $$s_j = s'' \odot \langle t_j, \ldots, t_{j+m''-1} \rangle, \text{ and } s_l = t_{l-m''+1} \text{ for } j < l \leq m''.$$
  In that case we recurse with τ_b and s' and e'.
  Otherwise we have a separating context e'' for s_c and s'' of the form $\langle s'_e, \langle s'_1, \ldots, s'_{m''} \rangle \rangle$ with $s'_l = t_{l+j-1}$ for $l' \in \{1, \ldots, m''\}$ and
  $$s'_e = s_e \odot \langle s' \odot \langle s''_1, \ldots, s''_{m''} \rangle \rangle$$
  where $s''_l = t_l$ for $l < j$, $s''_j = \Box$, and $s''_l = t_{k+m''-1}$ for $j < l \leq m'$, and we recurse with τ_c and s'' and e''. See Figure 4.3 for an illustration.

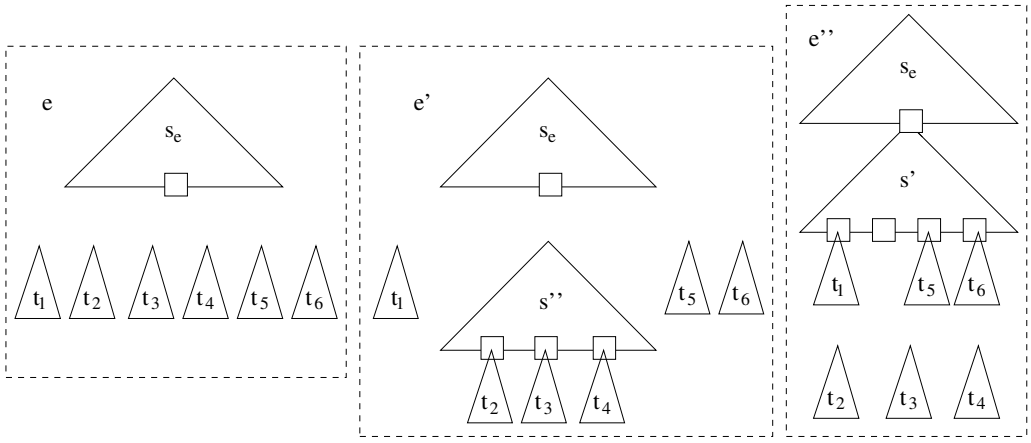

Figure 4.3: FINDCONTEXT with $m = 6$, $m' = 4$, $m'' = 3$, $j = 2$

- Let $\tau_a = \pi(\tau_b)$. Then π is of the form $[s_a] \to [s_b](\square_1, \ldots, \square_m)$ for some $[s_b] \in N'_m$ and τ_b is an $[s_b]$-derivation tree for s. We have $e[\![s_a]\!] \in L$ by assumption. If $e[\![s_b]\!] \notin L$ then π is an incorrect rule and we return e as a witness. Otherwise e must be separating for s_b and s, and we recurse with τ_b and s and e.

- τ' cannot be a single-node tree since the root of a single-node derivation tree can only be labeled by a rule of Type I' which would imply $s = s'$. Hence, the procedure must terminate.

One can show that the size of the context returned by FINDCONTEXT is bounded by $|c|\zeta_K$ where ζ_K is the maximal tree size featured in K since we successively replace nodes in c by elements from K. Moreover, the procedure involves parsing but, as stated in [122], this can be shown to be of polynomial complexity as well. Thus, FINDCONTEXT will always return a separating environment after a polynomial number of steps and MQs with respect to $|c|$ and to the number of nonterminals and rules in $\mathcal{G}_*$.
Concerning the overall complexity of the algorithm above one can show that the learner receives a positive counterexample at most $|P|$ times and that at that point the set of positive counterexamples received so far contains a $\mathcal{G}$-representative set $D_\mathcal{G}$ such that the learner has seen a witness for the application of every rule in $\mathcal{G}$. The cardinality of K is bounded by $|P|\zeta^{r+1}$ where ζ is the maximal tree size featured by any counterexample since $|Subt^{\leq r}(\{t\})|$ is bounded by $|t|^{r+1}$ for any tree $t \in \mathbb{T}_\Sigma$. Furthermore, one can show that the number of times that the learner receives a negative counterexample and the cardinality of X are both bounded by $|P|\zeta^{r+1}$ since each context that is

added to X in order to remove an incorrect rule induces a new equivalence class of which there are at most $|K|$. Thus, the algorithm returns a correct solution validated by the equivalence oracle after asking a polynomial amount of MQs and EQs and taking a polynomial amount of steps with respect to ζ and to the number of nonterminals and rules in the target grammar $\mathcal{G}$.
All complexity measures and arguments indicated above are largely based on and adapted from [120] and [122].

Remark Note that some of the learner's hypotheses may not be congruential and thus the teacher may have to answer *extended equivalence queries*, i.e., EQs for grammars that generate languages outside the learnable class. ♢

Remark Contrary to the regular language learning algorithms in the same setting studied in Chapter 3, the learner does not generate any candidates on its own. Note that a regular language learner could rely solely on positive counterexamples as well but EQs tend to be overly expensive. On the other hand, if the learner described above would generate its own candidates then its hypotheses would grow too big and possibly require exponential amounts of computation resources. The authors of [120] remark that the learner could also try to retrieve its own counterexamples from all possible combinations of elements from K and X since it may be that the current hypothesis does not classify all of them correctly but that then we would encounter a similar complexity problem as with freely generated candidates, whereas in the case of the learner above ζ is an intrinsic value given by the input. ♢

Remark The learner for congruential CFGs given in [26] emulates Angluin's LSTAR [5] more closely by establishing an adapted kind of observation table, modifying it until it meets a certain consistency condition based on a similar principle as the procedure FINDCONTEXT, and then deriving a hypothesis from that table in which nonterminals correspond to rows. The authors of [120] refrain from a consistency check due to the complexity issues addressed in the previous remark and solely use tables to store and look up values of previous MQs but nonterminals still correspond to individual elements of K which can be merged by rules of Type III'. Moreover, note that consistency is not strictly necessary since we do not disallow output grammars containing rules with the same right-hand sides for distinct nonterminals on the left.

The increased cost of a consistency check in the context-free case is also a barrier for the design of an algorithm that infers a context-free grammar $\mathcal{G}$ from MQs and a $\mathcal{G}$-representative positive sample by a method comparable to algorithms for regular languages from MQs and a representative sample (see [14] and Chapter 3) since the learner would not be able to construct a suitable set of environments in a polynomial number of steps. ♢

CPNI

The third scenario we are interested in is learning from a finite positive and a finite negative sample. Clark and Yoshinaka and the author have drafted a paper [36] on the inference of context-free string grammars in this setting by distributional techniques *(CPNI)*, and we will summarize its contents while adapting it to the SCFTG case below. In [36] we describe two CPNI learners, one for congruential and one for context-deterministic grammars.

Definition 47 *For a tree language $L \subseteq \mathbb{T}_\Sigma$ and a stub $s \in \mathbb{S}_\Sigma^m$ with $m \geq 0$, define $s^{-2}L := \{e \in \mathbb{E}_\Sigma^m \mid e[\![s]\!] \in L\}$ and $\mathcal{S}(s) := \{s' \in \mathbb{S}_\Sigma^m \mid s^{-2}L \subseteq s'^{-2}L\}$.*

Let a congruential r-SCFTG $\mathcal{G} = \langle \Sigma, N, I, P \rangle$ be the target, and $L := \mathcal{L}(\mathcal{G})$. For a finite set K such that $\mathcal{S}_\Sigma := \{f(\Box_1, \ldots, \Box_m) \mid m \geq 0 \wedge f \in \Sigma_m\} \subseteq K$, define a grammar $\mathcal{G}_K := \langle \Sigma, N', I', P' \rangle$ where

- $N'_m = \{[s] \mid s \in K_m\}$ for $m \geq 0$,
- $I' = \{[s] \mid s \in K \cap L\}$, and
- the rules in P' are of the form (I") $[s] \to s$ with $s \in \mathcal{S}_\Sigma$ or
  of the form (II") $[s] \to [s'](\Box_1, \ldots, \Box_{m'}) \odot^j [s''](\Box_1, \ldots, \Box_{m''})$
  with $[s'] \in N_{m'}$, $[s''] \in N_{m''}$ and $0 \leq j \leq m'$ such that
  $m' = m$ if $j = 0$ and $m' + m'' - 1 = m$ otherwise,
  and $\mathcal{S}(s') \odot^j \mathcal{S}(s'') \subseteq \mathcal{S}(s)$.

One can show by induction over the derivation length in $\mathcal{G}_K$ that $\mathcal{G}_K$ always fulfils $\mathcal{L}(\mathcal{G}_K) \subseteq L$. Moreover, one can show that $\mathcal{S}(s_1) \odot^j \mathcal{S}(s_2) \subseteq \mathcal{S}(s_1 \odot^j s_2)$, and using that result, one can show that if $\mathcal{S}_\Sigma \subseteq K$ and $K \cap \mathcal{L}_A \neq \emptyset$ for each nonterminal $A \in N$ in $\mathcal{G}$ then each $A \in N$ is simulated by some nonterminal in $\mathcal{G}_K$ which entails $L \subseteq \mathcal{L}(\mathcal{G}_K)$ by induction over the rules in P.

In the following, we describe a learning algorithm and the construction of two *characteristic* sets D_+ and D_- such that for any input pair $\langle X_+, X_- \rangle$ of a positive and a negative finite sample with $D_+ \subseteq X_+ \subseteq L$ and $D_- \subseteq X_- \subseteq (\mathbb{T}_\Sigma \setminus L)$, the learner returns a grammar $\mathcal{G}_K$ for some set $K \subseteq \mathit{Sub}^{\leq r}(X_+)$ meeting the conditions in the previous paragraph such that $L \subseteq \mathcal{L}(\mathcal{G}_K)$. The algorithm is given in pseudocode in the upper part of Figure 4.4.

As long as its hypothesis does not generate the entire set X_+ of given positive data, the learner retrieves the smallest positive counterexample with respect to the tree ordering relation $\preceq$ (see Subsection 3.3.1) from X_+ and uses it to expand the set K. Then it adds all conceivable rules of Type II" to P' unless the given data in $X_+ \cup X_-$ include an environment contradicting it.

Input: A finite set $X_+ \subseteq L$ and a finite set $X_- \subseteq (\mathbb{T}_\Sigma \setminus L)$
Output: An r-SCFTG.

$K := \mathcal{S}_\Sigma$;
$N' := \{[s] \mid s \in K\}$; $I' := \{[s] \mid s \in K \setminus X_-\}$; $P' := \{[s] \to s \mid s \in \mathcal{S}_\Sigma\}$;
$\mathcal{G}_* := \langle \Sigma, N', I', P' \rangle$;
while $X_+ \not\subseteq \mathcal{L}(\mathcal{G}_*)$ **do**
  $c := min_{\preceq}(X_+ \setminus \mathcal{L}(\mathcal{G}_*))$;
  $K := K \cup Sub^{\leq r}(\{c\})$;
  for $0 \leq m', m'' \leq m \leq r$ and $0 \leq j \leq m'$ and $s \in K_m$, $s' \in K_{m'}$, $s'' \in K_{m''}$ **do**
    if $(j = 0 \wedge m' = m \vee j \geq 1 \wedge m = m' + m'' - 1) \wedge \neg\exists e \in \mathbb{E}_\Sigma : (e[\![s]\!] \in X_+ \wedge e[\![s' \odot^j s'']\!] \in X_-)$ **then**
      $P' := P' \cup \{[s] \to [s'](\square_1, \ldots, \square_{m'}) \odot^j [s''](\square_1, \ldots, \square_{m''})\}$;
  $I' := \{[s] \mid s \in K \setminus X_-\}$;
return $\langle \Sigma, N', I', P' \rangle$.

Procedure 1
Input: The target language L
Output: A sample $D_+ \subseteq L$ and a sample $D_- \subseteq (\mathbb{T}_\Sigma \setminus L)$.

$D_+ := \emptyset$; $D_- := \emptyset$; $K := \mathcal{S}_\Sigma$;
$N_D := \{[s] \mid s \in K\}$; $I_D := \{[s] \mid s \in \mathcal{S}_\Sigma \cap L\}$; $P_D := \{[s] \to s \mid s \in \mathcal{S}_\Sigma\}$;
$\mathcal{G}_D := \langle \Sigma, N_D, I_D, P_D \rangle$;
while $L \not\subseteq \mathcal{L}(\mathcal{G}_D)$ **do**
  $c := min_{\preceq}(L \setminus \mathcal{L}(\mathcal{G}_D))$; $D_+ := D_+ \cup \{c\}$; $K := K \cup Sub^{\leq r}(\{c\})$;
  for $0 \leq m', m'' \leq m \leq r$ and $0 \leq j \leq m'$ and $s \in K_m$, $s' \in K_{m'}$, $s'' \in K_{m''}$ **do**
    if $(j = 0 \wedge m' = m \vee j \geq 1 \wedge m = m' + m'' - 1)$ **then**
      if $\mathcal{S}(s') \odot^j \mathcal{S}(s'') \subseteq \mathcal{S}(s)$ **then**
        $P_D := P_D \cup \{[s] \to [s'](\square_1, \ldots, \square_{m'}) \odot^j [s''](\square_1, \ldots, \square_{m''})\}$;
      else find a size-minimal $e \in \mathbb{E}_\Sigma^m$ **such that** $e[\![s]\!] \in L \wedge e[\![s' \odot^j s'']\!] \notin L$;
        $D_+ := D_+ \cup \{e[\![s]\!]\}$; $D_- := D_- \cup \{e[\![s' \odot^j s'']\!]\}$;
  for $s \in K$ **do**
    if $s \notin L$ **then** $D_- := D_- \cup \{s\}$; **else** $I_D := I_D \cup \{s\}$;
return $\langle D_+, D_- \rangle$.

Figure 4.4: CPNI for congruential SCFTGs – learner and sample construction

Again, since we do not have access to a membership oracle the learner must rely on the stubs and environments in question to appear as concatenations within the positive and negative data *explicitly*, as when inferring a DFTA or an RFTA from a positive and negative finite sample in the regular case, compare Subsections 3.3.2 and 3.4.5 in Chapter 3. For the same reason, we choose to add all stubs that are not members of X_- as initial symbols to I'.

The construction of two characteristic sets D_+ and D_- for the learner is defined by Procedure 1 in the lower part of Figure 4.4. This procedure can be seen to simulate our learning algorithm on the input sets L and $\mathbb{T}_\Sigma \setminus L$ and returns two finite sets D_+ and D_- containing all the information that is needed to make this specific learner identify the target language correctly.

Remark Note that the constructor does not only add a rule if the stubs in question share exactly the same contexts as suggested by the definition of a congruential grammar but also if the concatenation of stubs considered for the right-hand side has a subset of the contexts for the stub on the left, as suggested by the definition of $\mathcal{G}_K$. This still results in a grammar where each nonterminal $[s]$ only generates a stub s' if s' has at most all positive contexts for s, and it facilitates the design of the learner and the proofs.
Moreover, we choose size-minimal environments for some reasonable notion of minimality in order to keep down the size of the output sets. ♢

Procedure 1 is not a computable algorithm but is mathematically well-defined and terminates, and when it does the internal working grammar $\mathcal{G}_D$ correctly represents the target language. Due to the fact that both the learner and the constructor always recur to the smallest positive counterexample to construct their next grammar one can show that they compute exactly the same sequence of grammars. As a consequence, our learner returns a correct hypothesis grammar $\mathcal{G}_*$ with $\mathcal{L}(\mathcal{G}_*) = L$ after a finite number of steps which moreover is polynomial with respect to $m_+ := \sum_{t \in X_+} |t|$ and $m_- := \sum_{t \in X_-} |t|$.

We will now present a dual learner inferring a context-deterministic SCFTG.

Definition 48 *For a tree language $L \subseteq \mathbb{T}_\Sigma$ and an environment $e \in \mathbb{E}^m_\Sigma$ with $m \geq 0$, define $\mathcal{E}_L(e) := \{s \in \mathbb{S}^m_\Sigma \mid e[\![s]\!] \in L\}$.*

For a context-deterministic target r-SCFTG $\mathcal{G} = \langle \Sigma, N, I, P \rangle$, let $L := \mathcal{L}(\mathcal{G})$. For a finite set X such that $\langle \Box, \langle\rangle \rangle \in X$, define $\mathcal{G}_X := \langle \Sigma, N', I', P' \rangle$ where

- $N'_m = \{[e] \mid e \in X_m\}$ for $m \geq 0$,
- $I' = \{\langle \Box, \langle\rangle \rangle\}$, and
- the rules in P' are of the form $[e] \to s$ with $s \in \mathcal{S}_\Sigma \cap \mathcal{E}_L(e)$ or

of the form $[e] \to [e'](\square_1, \ldots, \square_{m'}) \odot^j [e''](\square_1, \ldots, \square_{m''})$
with $[e'] \in N_{m'}$, $[e''] \in N_{m''}$ and $0 \leq j \leq m'$ such that
$m' = m$ if $j = 0$ and $m' + m'' - 1 = m$ otherwise,
and $\mathcal{E}_L(e') \odot^j \mathcal{E}(e'') \subseteq \mathcal{E}(e)$.

One can show by induction over the derivation length in $\mathcal{G}_X$ that $\mathcal{G}_X$ always fulfils $\mathcal{L}(\mathcal{G}_X) \subseteq L$. Moreover, if X contains an environment e_A with $\mathcal{E}_L(e_A) = \mathcal{L}_A$ for each $A \in N$ in $\mathcal{G}$ then each $A \in N$ is simulated by some nonterminal in $\mathcal{G}_X$ which entails $L \subseteq \mathcal{L}(\mathcal{G}_X)$ by induction over the rules in P.

Figure 4.5 shows an algorithm and the construction of two characteristic sets D_+ and D_- such that for any input pair $\langle X_+, X_- \rangle$ of a positive and a negative finite sample with $D_+ \subseteq X_+ \subseteq L$ and $D_- \subseteq X_- \subseteq (\mathbb{T}_\Sigma \setminus L)$, the learner returns a grammar $\mathcal{G}_X$ for some set $X \subseteq Env^{\leq r}(X_+)$ with $\langle \square, \langle\rangle \rangle \in X$ such that $L \subseteq \mathcal{L}(\mathcal{G}_X)$. In this case, as long as its hypothesis does not generate the entire set X_+ of given positive data, the learner retrieves the smallest positive counterexample with respect to $\preceq$ from X_+ and uses it to expand the set X. Then it adds all conceivable rules as specified above to P' unless the given data in $X_+ \cup X_-$ include stubs figuring in the environments in question such that the rule is contradicted. The construction of two characteristic sets D_+ and D_- for this learner is defined by Procedure 2, which works in a fairly similar way as Procedure 1. When it terminates the grammar $\mathcal{G}_D$ correctly represents the target language. Again, due to the fact that both the learner and the constructor always recur to the smallest positive counterexample to construct their next grammar one can show that they compute exactly the same sequence of grammars, and consequently our learner returns a correct hypothesis grammar $\mathcal{G}_*$ with $\mathcal{L}(\mathcal{G}_*) = L$ after a finite number of steps which is polynomial with respect to $m_+ := \sum_{t \in X_+} |t|$ and $m_- := \sum_{t \in X_-} |t|$.

Remark As in the algorithm for MQs and EQs above we do not check for consistency with the available data: If the input includes characteristic sets then the output will trivially fulfil this condition, otherwise our algorithms may produce hypotheses which correctly generate all positive examples but may also generate some of the negative ones. This could be rectified by a test which in case of overgeneralization replaces the final hypothesis with a default grammar that merely generates the positive input set X_+. We refrain from such a modification since it seems rather artificial. ♢

Let us briefly compare the results outlined above (referred to as CPNI) to the case of learning regular languages from a positive and a negative finite sample (RPNI) as discussed in Subsections 3.3.1, 3.3.2, and 3.4.5 (Chapter 3). The following remarks are partly taken from [36].

Input: A finite set $D_+ \subseteq L$ and a finite set $D_- \subseteq (\mathbb{T}_\Sigma \setminus L)$
Output: An r-SCFTG.

$X = \{\langle \Box, \langle\rangle\rangle\}$; $N' := \{[e] \mid e \in X\}$; $I' := N'$; $P' := \emptyset$;
$\mathcal{G}_* := \langle \Sigma, N', I', P' \rangle$;
while $D_+ \nsubseteq \mathcal{L}(\mathcal{G}_*)$ **do**
  $c := min_{\preceq}(D_+ \setminus \mathcal{L}(\mathcal{G}_*))$;
  $X := X \cup Env^{\leq r}(\{c\})$;
  for $0 \leq m \leq r$ and $e \in X_m$ and $s \in \{f(\Box_1, \ldots, \Box_m) \mid f \in \Sigma_m\}$ **do**
    if $e[\![s]\!] \notin D_-$ **then** $P' := P' \cup \{[e] \to s\}$;
  for $0 \leq m', m'' \leq m \leq r$ and $0 \leq j \leq m'$ and $e \in X_m$, $e' \in X_{m'}$, $e'' \in X_{m''}$ **do**
    if $(j = 0 \wedge m' = m \vee j \geq 1 \wedge m = m' + m'' - 1) \wedge \neg(\exists s' \in \mathbb{S}_\Sigma^{m'} : \exists s'' \in \mathbb{S}_\Sigma^{m''} : (e'[\![s']\!] \in X_+ \wedge e''[\![s'']\!] \in X_+ \wedge e[\![s' \odot^j s'']\!] \in X_-))$ **then**
      $P' := P' \cup \{[e] \to [e'](\Box_1, \ldots, \Box_{m'}) \odot^j [e''](\Box_1, \ldots, \Box_{m''})\}$;
return $\langle \Sigma, N', I', P' \rangle$.

Procedure 2
Input: The target language L
Output: A sample $D_+ \subseteq L$ and a sample $D_- \subseteq (\mathbb{T}_\Sigma \setminus L)$.

$D_+ := \emptyset$; $D_- := \emptyset$; $X := \{\langle \Box, \langle\rangle\rangle\}$;
$N_D := \{[e] \mid e \in X\}$; $I_D := \{[\langle \Box, \langle\rangle\rangle]\}$; $P_D := \emptyset$;
$\mathcal{G}_D := \langle \Sigma, N_D, I_D, P_D \rangle$;
while $L \nsubseteq \mathcal{L}(\mathcal{G}_D)$ **do**
  $c := min_{\preceq}(L \setminus \mathcal{L}(\mathcal{G}_D))$; $D_+ := D_+ \cup \{c\}$; $X := X \cup Env^{\leq r}(\{c\})$;
  for $0 \leq m \leq r$ and $e \in X_m$ and $s \in \{f(\Box_1, \ldots, \Box_m) \mid f \in \Sigma_m\}$ **do**
    if $s \in \mathcal{E}_L(e)$ **then** $P_D := P_D \cup \{[e] \to s\}$;
    else $D_- := D_- \cup \{e[\![s]\!]\}$;
  for $0 \leq m', m'' \leq m \leq r$ and $0 \leq j \leq m'$ and $e \in X_m$, $e' \in X_{m'}$, $e'' \in X_{m''}$ **do**
    if $(j = 0 \wedge m' = m \vee j \geq 1 \wedge m = m' + m'' - 1)$ **then**
      if $\mathcal{E}_L(e') \odot^j \mathcal{E}_L(e'') \subseteq \mathcal{E}_L(e)$ **then**
        $P_D := P_D \cup \{[e] \to [e'](\Box_1, \ldots, \Box_{m'}) \odot^j [e''](\Box_1, \ldots, \Box_{m''})\}$;
      else find $s' \in \mathbb{S}_\Sigma^{m'}$, $s'' \in \mathbb{S}_\Sigma^{m''}$ such that $s' \in \mathcal{E}_L(e') \wedge s'' \in \mathcal{E}_L(e'') \wedge s' \odot^j s'' \notin \mathcal{E}_L(e)$ and s', s'' are minimal with respect to $\preceq$;
        $D_+ := D_+ \cup \{s', s''\}$; $D_- := D_- \cup \{s' \odot^j s''\}$;
return $\langle D_+, D_- \rangle$.

Figure 4.5: CPNI for context-deterministic SCFTGs

Contrary to settings involving an oracle where interaction is still possible during the learning process, in the RPNI and CPNI paradigms the teacher must anticipate the learner's strategy when constructing suitable samples which gives rise to conditions as (A1) and (A2) in Subsection 3.3.2, (C1) and (C2) in Subsection 3.4.5, and the choice of the smallest counterexample in Procedures 1 and 2 in order to ensure that the learner also chooses a *specific candidate* for the representation of a certain component in its hypothesis.

Moreover, in both cases the learner crucially depends on the given data to contain a sufficient amount of material from which these components can be constructed, and a sufficient amount of information on which to base its decisions for their adoption or their rejection. If one simply defines characteristic sets as finite sets such that as soon as they are included in the data the learner will identify the target language L correctly then this requirement is not restrictive for algorithmic learnability in a polynomial number of steps since if we do not impose conditions on the size of such sets then one can always design vacuous enumerative algorithms – for example, one can add an element of exponential size with respect to the number of components in a grammar for L to the data whose only purpose is to justify an overly large amount of computation. A polynomial bound on the amount of computation with respect to the size of the set of given data is meaningless without a bound on the size of that set itself. Accordingly, it is standard (see [40]) to require the size of a characteristic set to be polynomially bounded in terms of the complexity of L, i.e., with respect to the size of a smallest description.

In the RPNI case the definitions of a representative and a separative or exclusive sample could be formulated with respect to the number of states in the state-minimal canonical DFTA for the target. However, as indicated in Section 4.1 above, in the context-free case there is no comparable canonical finite description, and we cannot obtain a strict polynomial bound on the size of characteristic sets which to some extent is due to the fact that a context-free grammar can be of *exponential thickness* such that all but finitely many of the generated elements are exponential in size with respect to the number of components in the grammar. Moreover, in general such a pair of sets may not be computable even when given a concrete reference description at all.

Thus, from a technical viewpoint, the main challenge for CPNI lies in the lack of an adequate bound on the size of characteristic sets. This problem has been encountered before when moving from the regular to the context-free case (see [32, 115]) and to a certain extent is inherent to the language class in question. Part of the problem is the twin role of positive data as the learner uses them to form the nonterminals of its hypothesis as well as a tentative set of rules but some of its choices may generate the need for more data in order to make it converge to a correct solution such that we risk running into

an infinite loop when trying to construct corresponding characteristic sets where we have to take the learner's strategy into account by definition.

One way to overcome this difficulty would be to present the learner with two different sets of positive data along with the set of negative ones, where one of the positive sets is explicitly designated as a source of nonterminals whereas the other should be used to lay down rules in accordance with the negative evidence. The solution chosen above allows a closer imitation of the established learning algorithms for the RPNI case inasmuch as the learner only receives a single set of positive data. Procedures 1 and 2 anticipate the learner's strategy in a very literal sense by going through the motions themselves – where the essential ingredient is the choice of the smallest positive counterexample at every step of the construction – but with full information about the target language and without the restrictions of finiteness. From a certain point of view, the particular strategies of our two learners determine two new canonical descriptions for congruential and for context-deterministic context-free languages, respectively, which are unambiguously defined as the grammars for the target language constructed within Procedures 1 and 2, or equivalently as the grammars returned by the learners when given the respective outputs of those procedures as their input data sets.

We note that we do not have a characterization for those characteristic sets other than the given procedures, which is linked to the fact that we do not have a characterization for the considered language classes in terms of certain restrictions on the form of the generating grammars themselves – however, it has been conjectured in [26] that the class of congruential context-free string languages is identical to the class of languages generated by nonterminally separated *(NTS)* CFGs (see [16, 105]) for which equivalence is decidable, and the NTS property for a CFG is decidable in a polynomial number of steps.

One may surmise that the CPNI techniques we describe can be directly transferred to classes of context-free grammars with the 1-FKP or the 1-FCP. However, a crucial difference is that while any $s \in \mathcal{L}_A$ is suitable to simulate the nonterminal A in a congruential grammar $\mathcal{G}$, the 1-FKP merely ensures the *existence* of such an element s_A for A. It may happen that one can derive an element t in $\mathcal{G}$ using A such that t contains the stub s_A and the learner's hypothesis $\mathcal{G}_*$ correctly generates t as well although $\mathcal{G}_*$ does not yet feature a nonterminal whose representing element fulfils the same conditions as s_A. In such a case, a characteristic set constructor for grammars with the 1-FKP in the style of Procedure 1 may not terminate. There can be infinitely many elements causing a further expansion of the set K while none of them meets the conditions for s_A. On the other hand, we currently do not know of any concrete example where such an infinite run is inevitable.

Remark Under the observation that the retrieval of separating contexts is closely related to the construction of characteristic samples, one may relate Procedures 1 and 2 to the procedure FINDCONTEXT for MQs and EQs, and ask once more why they terminate despite the general complexity issues. In essence, this is due to the fact that in all these procedures the previously established contexts are not extended using single symbols from the finite alphabet Σ as suggested by the regular case (where the number of equivalence classes reachable from any given class by a one-symbol transition is finite) but using elements from the fixed finite set K which is also the source for the nonterminals in the current working hypothesis. This tactical revision exactly reflects the difference between the respective normal forms for regular and for context-free grammars. ♢

Remark Certain observations in the literature on distributional learning allow us to establish a link to the difficulties encountered by the algorithm MATRES for the inference of residual finite-state tree automata from MQs and EQs described in Subsection 3.4.6. In [118], Yoshinaka points out that CFGs with the 1-FCP arise as the natural context-free counterparts of RFSA when one switches from the definition of a context as a suffix to both-sided string contexts. Yoshinaka and Clark [120] state that it seems hard to obtain learning results for distributional properties such as the FCP in the setting of MQs and EQs since it may be that there are infinitely many counterexamples whose generation is not covered by the nonterminals drawn from the learner's current set of contexts X at any time. As already conjectured in Section 3.5, we surmise that the difficulties for the residual tree case and those anticipated for the FCP in the setting of MQs and EQs may both be traced back to the fact that in general neither residual finite-state automata nor context-free grammars are polynomially characterizable, see [40, 42].
We will resume this discussion in Chapter 5. ♢

4.3 Outlook

As demonstrated in the previous sections and in [122], the distributional learning techniques which were initially developed for CFGs can be extended to context-free grammar formalisms based on more complex structures than the basic string in a natural way once we have redefined the notions of a substructure and its context along with a suitable concatenation operation (also see the concluding remarks in Chapter 5). Since the main object of interest in this work is the tree the context-free formalism we have chosen are SCFTGs, and by rephrasing all notions and results in terms of trees we have imitated the extension of learning techniques for strings to trees in the regular case.

Note that results of this kind may also be relevant in formal linguistics. For example, there exists a translation for Tree Adjoining Grammars (TAGs) into SCFTGs (see [83], where SCFTGs are called *monadic linear* CFTGs), and thus TAGs must be inferable by distributional methods using this connection. Alternatively, one could also design a distributional learner for TAGs directly by choosing a variant of TAG with suitable restrictions and modifying the definitions of a substub and environment accordingly.

Moreover, we conjecture that the shift from string to tree grammars can be continued to context-free graph generating formalisms such as *node controlled graph grammars* (see [15]) and *hyperedge replacement grammars* (see for example [63]) by establishing corresponding definitions and specifying in an analogous manner how a learner would extract sub-(hyper)graphs from given data to construct the components of its hypothesis.

We have concentrated on simple CFTGs as the most intuitive extension of CFGs to trees – a further conceivable direction for future work is the study of CFTGs that allow subtrees to be duplicated or permuted (whereas allowing the deletion of subtrees does not change the expressive power of SCFTGs, see [54], such that this is not an interesting relaxation).

Another prominent grammar formalism offering itself for the application of distributional learning techniques is given by MCFGs [103] – corresponding algorithms, which generalize the ones from [32, 26], can be found in [117, 120]. Also see [121] where the authors show distributional learnability for a range of formalisms (CFGs, *Linear Context-Free Rewriting Systems*, TAGs, SCFTGs) via an encoding in terms of *Abstract Categorial Grammars*. Furthermore, we recommend Yoshinaka's paper [119] where the primal and the dual approach are integrated into a single distributional learning strategy for CFGs.

In [31], Clark describes the first (dual) distributional learning algorithm for the inference of (non-regular) synchronous CFGs defining *transductions*, so-called *Inversion Transduction Grammars*, from a stream of positive data in the limit, and he also considers extending his results to the case of trees. Synchronous CFGs are a special case of MCFGs and are of interest for programming language compilation and machine translation within the area of natural language processing (see [23] for an introduction and references).

As mentioned before, Yoshinaka [118] observes that CFGs with the 1-FCP arise naturally as the context-free counterparts of RFSA when one switches from the definition of a context as a suffix to both-sided string contexts. In reverse, he suggests to define an x-RFSA as a finite-state automaton where each state is associated with a set $S^{-1}L := \{w \in \Sigma^* \mid \forall v \in S : vw \in L\}$ for some set S of prefixes of L with $|S| \leq x$, and conjectures that this may yield potentially even more succinct descriptions for regular languages which can be inferred by similar distributional methods as CFGs with the x-FCP.

A further direction of research could be to study the connection between the classes of context-free languages fulfilling one of the distributional properties above and the hierarchy that is formed when classifying the context-free languages by their *nonterminal complexity* [62], i.e., by the minimal number of nonterminals featured by any context-free grammar generating them.

Also of interest for more practical applications is a combination of distributional learning techniques with probabilistic strategies. In [27], Clark motivates the use of a probabilistic clustering method based on the frequency with which substrings and contexts occur in large sets of given data. Accordingly, it seems a quite attractive approach for computational linguistics to consider the application of probabilistically modified distributional techniques to large corpora of linguistic data such as the *Penn treebank* [92], i.e., a databank from which one can extract a *probabilistic CFG (PCFG)*, see [22].
Results obtained under a distributional approach to *PAC learning* (see [111]) for languages that are generated by nonterminally separated PCFGs from a stream of positive data in the limit and with polynomial bounds on data and computation steps can be found in [24], and for a parametrized extension of the NTS property in [90]. In [120], the authors state that their results for congruential MCFGs could be extended to a stochastic variant as well.

Chapter 5

Concluding reflections

The reflections below should not be understood as concluding in the sense of exhausting the topic but we let them occupy the space usually assigned to a conclusion.

Learning from structural information

In [102], Sakakibara presents a learner for the inference of context-free string grammars by exploiting the well-known result that the set of derivation trees of any CFG forms a regular tree language (see [109]) and adapting Angluin's algorithm LSTAR [5] to *structural data*, i.e., derivation trees of the context-free target grammar where only the leaves are labeled. Algorithmic solutions of this kind allow us to state the simple but fundamental principle for Grammatical Inference that the learnable class can be extended by providing the learner with additional *structural information*. In our concluding chapter, we would like to reflect a little further on the notion of 'structural information' as in essence we have strived throughout the present work to find answers to the following questions: What is structural information, in what forms can it appear, and most importantly, how can it help in a learning process?

The use of the term 'structural information' in [102] where strings appear together with unlabeled derivation trees evokes the more general conception that the learner is given an object not only as one homogeneous structure but is also provided with some additional kind of bracketing information specifying parts that belong together more closely than others in the sense that they may have been added at the same point or by the same mechanism in a conceivable construction process of the given object. This implies that the teacher already has some syntactic theory in mind, which in the realm of Grammatical Inference translates into a concrete formal description.

Accordingly, let us use the term 'structural information' as synonymous to 'information on properties of possible descriptions for the target language'. In the literature there is usually a sharp distinction between learning from "pure" data and learning with additional structural information in a similar sense as in [102], and research on Grammatical Inference generally strives to explore the minimal amount of structural information needed to identify a formal language. However, learning without structural information in the broadened sense suggested above is hardly possible. In the following list, let us summarize the various kinds of structural information as exploited by the learners considered in this work from this generalized perspective.

(a) The *underlying alphabet* is structural information.

(b) The *language class on the Chomsky hierarchy* is structural information as the original definition of the Chomsky hierarchy refers to restrictions on formal grammars, with resulting restrictions on other class-specific descriptions such as various kinds of automata or term-like expressions. In particular, the learner can deduce from the respective normal forms that in the regular case it is sufficient to maintain a hypothesis where each rule combines a nonterminal with an individual symbol from the alphabet, and two nonterminals in the context-free case. In reverse, the learner can deduce that it may safely base its conclusions on the effects of splitting off a single symbol from the data, or splitting it in two.
The traditional classes are narrowed by additional restrictions on the rule format: For example, *linearity* only admits rules with at most one nonterminal on the right, and the refinement of *even linearity* requires the nonterminal to be enclosed into an equal number of symbols to its right and left (which for context-free string grammars yields a normal form admitting only rules of the form $A \to \varepsilon$ or $A \to aBc$ where A and B are nonterminals and a, b are symbols from the alphabet, see [1]).

(c) *Numerical parameters* are structural information such as the maximal rank for SCFTGs, the *rank* and *dimension* of an MCFG (see [117]), the *dimension* of a multi-dimensional tree language, the cardinalities k and x for the k-FKP and the x-FCP, and the parameters k and l for k, l-substitutability (see [115]) i.e., all kinds of measures which further restrict the rule format within a formalism. When learning languages that are generated by one of those formalisms or associated with them via some yield operation then these numerical parameters provide a learner with information on the maximal number of discontinuous parts of an object that can belong together at most, i.e., into how many parts does the learner have to decompose the data to draw its conclusions?

In the branch of formal linguistics initiated by Joshi in [72], these numerical restrictions can be related to the syntactic condition of *limited cross-serial dependencies* – also see [85] for a study of the dependency structures in mildly context-sensitive linguistic frameworks.

(d) Many learners in the literature are provided with even more concrete structural information by being informed that there exists a description for the target language which is for example *reversible* (forwards and backwards deterministic), *k-reversible* for a given parameter k (see [4]), *f-distinguishable* for a given function f (see [53]), *simple* (see [68]), or *very simple* (see [113]). All these properties specify directly observable or testable interrelations between the components of a description and thus further restrict the conceivable sets of rules that can appear in a valid description of the target.

(e) Moving on to interrelations between the components of a description and the sets of elements that they generate, *structural reversibility*[1] and most prominently, all *distributional properties* discussed in Chapter 4 (all of which subsume the regular case studied in Chapter 3 except for substitutability) represent structural information in the sense suggested above. This view can perhaps be motivated best considering the algorithmic solution for CPNI in Subsection 4.2.3 where learner and teacher construct the same description based on a common strategy.

(f) Yet another kind of structural information may be perceived in scenarios where for each datum the learner receives another datum associated with the first – a simple example is a value from $\{0, 1\}$ indicating whether the first datum is a member of the target language but it could also be a value from a larger numerical set or even an object of the kind studied in Chapter 2 as in the case of Sakakibara's learner which infers CFGs from structural information in the original sense (see [102]).
A numerical value can be computed by a *weighted automaton* in which each transition has some weight and the automaton accumulates these weights during parses, thus assigning a certain *coefficient* to each datum which is also revealed to the learner (see for example [64, 45, 49] and references therein on the inferability of weighted tree automata).
Languages of object pairs can be generated or recognized by the rules of descriptions that process two objects simultaneously (a prominent example are so-called *transduction functions* – learnability results can

[1] A CFG in Greibach normal form with nonterminal set N is *structurally reversible* if for any pair $\alpha, \beta \in N^*$ such that α and $\beta\alpha$ each represent the sequence of nonterminals in some valid sentential form, α and $\beta\alpha$ generate disjoint sets of terminal strings, see [19].

be found for *subsequential transducers* in [95], for *weighted transducers* in [38], and for *Inversion Transduction Grammars* in [31]). Here again, one may take the perspective that in each pair $\langle a, b \rangle$ the first object a is an element of the target language under consideration and that b is merely a by-product of its generation. Ideally, each rule of the reference description should have a non-trivial impact on b as well such that the learner can draw more accurate conclusions with respect to a possible construction of a. In formal linguistics, this requirement can be related to the condition that a formalism should be *lexicalized*, i.e., each rule in it should add interpretable material to the constructed expression. See [85] on the interrelations between the dependency structures in a lexicalized grammar formalism and its generative power.

(g) A very powerful way of extending the learnable classes by structural information is a variant of (f) where the learner is set to learn a language from a simpler class but is also informed that the elements of the language it is learning can be reinterpreted as unambiguous instructions for the construction of elements of the intended target language. Stated in terms of object pairs as in (f) above, if a learner is able to produce an element b of the so-called *control language* then it can also construct the pair $\langle a, b \rangle$ and thus the desired element a of the target language as well. Note that the control language may be composed of objects of an entirely different type than the target language such that this scenario is a good illustration of the various possible interactions between the three generalization axes we have spanned in Chapter 1 ranging over object types, learning settings, and language classes, respectively.
One example for such a reinterpretation strategy is Sakakibara's [102] previously mentioned learner for context-free string languages, which learns a regular tree language and applies the yield operation for trees. The strategy can be extended to certain mildly context-sensitive string language classes by using regular multi-dimensional tree languages and the corresponding yield operation (see Subsection 2.2.2).
Another example is given by results which show the inferability of even linear context-free string grammars in normal form (see (b)) when the symbols to the left and right of the nonterminal on the right-hand side of each rule are coded into a single symbol such that the learning task is reduced to the inference of regular string grammar rules which can then be unfolded again into even linear grammar rules (see [108, 104]).

This kind of argument can be iterated by learning control languages via other control languages, and extended to more complicated formalisms than CFGs such as *simple matrix grammars* , which gives rise to various nested hierarchies of learnable language classes – see [52] for a survey of the literature and further control-based learnability results.

The most essential kind of structural information for successful identification in a finite number of steps, even prior to all other ones, is *finiteness*. Trivially, the target language has to admit a finite description – otherwise a learner would not be able to construct nor to communicate its hypothesis. The fact that infinite languages of finite objects over a finite alphabet admit a finite description is well-known but the literature also features learnability results for regular languages that consist of infinite objects (see [91] for ω-*regular strings*) and for infinite-state automata that allow a finite description (*counter automata*, see [13]). Moreover, we suggest considering a similar approach for the automata over infinite alphabets proposed in [61] which can be described as finite-state automata where some of the transitions are labeled by variables ranging over the infinite underlying alphabet.

We note that structural information does not necessarily have to point the learner to a single canonical description as long as it narrows down the set of eligible descriptions enough both among and within language classes. More precisely, structural information of the kind described by items (b)–(e) in the list above serves to determine the language class containing the target, and once the class is fixed, structural information narrows down the number of potential candidates within the class in combination with the given data. On the other hand, this is another distinction to be generalized considering that each positive and each negative datum defines a new class of languages containing it or not containing it, respectively. Also compare the notions of a *finite telltale set*, *finite thickness*, and *finite elasticity* which were proposed by Angluin [2] as conditions for learnability in the limit from positive data.

It must be possible for the learner to derive all the necessary clues from a finite amount of information, otherwise it is thrown back on guessing or simply assuming that the set of data seen so far includes all there is to know. These are exterior conditions. Concerning the learner's internal strategy, we can state that some of its choices must restrict the remaining ones enough such that the learner will reliably produce a finite description. In addition, the number of steps taken by a learner may not only be required to be finite but also to be bounded by a polynomial with respect to a given reference, which implies that each of the learner's choices may have to restrict the remaining ones even more drastically.

In summary, we observe that all learners in this work having access to some finite sample fulfilling certain informativeness conditions were able to construct a correct description in a polynomial number of steps with respect to the given sample. Note that in the tree case studied here, the polynomial bound is achieved by referring to the number of nodes in the sample trees, and not to their depth. Moreover, in the residual case (algorithms RESI and RRPNI, Subsections 3.4.4 and 3.4.5) polynomiality is achieved by defining suitable samples in terms of a description which may be exponential in size with respect to the intended target description, viz., the minimal DFTA $\mathcal{A}_L$ instead of the canonical RFTA $\mathcal{R}_L$. In the CPNI case (Subsection 4.2.3) we do not know if there is a (computable) descriptional characterization for the characteristic samples constructed by Procedures 1 and 2.

When no sample is provided the issue of polynomial characterizability is shifted from the teacher to the learner (see Section 3.5 and the last remark in Subsection 4.2.3). If a specializing learner in the setting of MQs and EQs as considered in this work is to meet the requirement of polynomiality then it must derive each new distinction in a polynomial number of steps from its own hypothesis and the available counterexamples, and the overall number of computed distinctions must be polynomially bounded with respect to the description serving as a reference for the learner's performance.

In the cases where each component of the target description corresponds to a class under an adequate congruence relation (viz., $\equiv_L^>$ in the regular and $\equiv_L$ in the context-free case as defined in Section 4.1), the computability of a next distinction in a polynomially bounded number of steps is ensured by the fact that any observable inconsistency in the hypothesis or between the hypothesis and a counterexample already offers the material needed for its solution (see the explanations in Subsection 3.4.4 for the regular case). Corresponding learning algorithms exploiting this fact have been described in Section 3.3 (for DFTA) and Subsection 4.2.3 (for congruential SCFTGs). Moreover, in these cases the target and reference description coincide.

In all remaining cases, in order to meet the requirement of polynomiality formulated above a learner must apply a more refined strategy. As argued in Subsection 3.4.4, the task of computing the next distinction is intrinsically linked to the retrieval of an inconsistency involving a number of components (states or nonterminals) of the learner's current hypothesis and some rule or transition. A learner for the inference of residual finite-state string automata can achieve polynomiality (see [17]) by the fact that in the string case all transitions for a congruence class can be constructed using the alphabet Σ, i.e., without having to recur to any other classes postulated by the learner. However, note that as in the case of RESI and of RRPNI, the polynomial bound observed by the learner in [17] refers to the size of the state-minimal

deterministic automaton, and not of a state-minimal residual one. The same shift is effected by Yokomori in [114] where he describes an algorithm that infers a non-deterministic finite-state string automaton (NFA) using MQs and EQs in a polynomial number of steps in the size of the state-minimal deterministic automaton and the longest counterexample. Not surprisingly, Yokomori suspects that his inferability results only apply to a class of NFA fulfilling the condition that their size is polynomially bounded with respect to some deterministic equivalent.

This leaves the inference of RFTA and of non-congruential context-free grammars where in the setting of MQs and EQs we seem to face particular difficulties when trying to design a learner with a polynomial performance (see Subsection 3.4.6 and the last remark in Subsection 4.2.3). By way of a first explanation, we observe that finite-state tree automata and context-free (string or tree) grammars have a common aspect which distinguishes them from finite-state string automata and from regular grammars alike: Both in FTA and in CFGs in normal form the rule format generally links one component (state or nonterminal) to a combination of several others instead of an expression involving another single component and a terminal symbol as in the regular string case, i.e., these are descriptions with an intrinsic potential for non-linearity. Thus, a learning process based on the inference of one of those description classes comprises the problem that the retrievability of a suitable transition for the computation of a next distinction crucially depends on the question if the learner's hypothesis already contains suitable components for its representation – if it does not then even structural information may not be of use as it may not restrict the search space enough. This may also account for the fact that in contrast to the regular residual case, once the non-linear barrier is overcome learnability results for context-free string formalisms can be adapted to the context-free tree case without additional difficulties as we have demonstrated in [122] and in Chapter 4.

From the psychological point of view, one might object that learning for example multi-dimensional tree languages requires providing a learner with rather specific structural information and thus does not seem very plausible as a model for natural first language acquisition. We have several arguments against such a view. First of all, as we have motivated in the present chapter, all algorithmic learnability results cited in this work can be seen to rely on structural information since the range of structural information in the wider sense outlined above coincides with basic definitory mechanisms in formal language theory, and one might as well object to the usefulness of them all. The central issue in any learning process is the conveyance of information and the structures chosen – consciously or unconsciously – for the purpose.

Any organization of the chosen data based on distributional observations is a classification of objects according to their potential to be extended into a correct expression. However, both within a learner's and a teacher's mind or system there must be some concrete mechanism constructing and testing these extensions such that the continued encoding and decoding processes of the information that is passed between learner and teacher translate into partial generating and parsing processes. Accordingly, all results presented in this work where a learner is said to learn languages from a certain class were inferability results for a previously fixed class of descriptions. Note that we consistently use the verb "to learn" in collocation with a language and "to infer" with descriptions in order to make a distinction.

Even if one assumes that a teacher does not convey any explicit structural information to the learner as the teacher may only be a speaker of the target language without any conscious knowledge of syntax, or simply serves as an abstraction of the entirety of data to be found in the learner's environment, algorithmic results involving structures as complex as multi-dimensional trees might be more plausible components in a natural language learning model than it first seems: Suppose that some of the proposed algorithmic procedures do not model the interaction between the learner and some external teacher but rather intermediate stages of the learning process within the learner's system at which the learner has already developed subhypotheses or derivations for parts of the data and tries to combine them in a way that should be as basic as possible. A result offering itself for such an interpretation is the fact that sets of derivation trees in the prominent linguistic formalism of Tree Adjoining Grammars correspond to regular multi-dimensional tree languages (see [100, 101]). This also shows the importance of defining variants of the notion of a substructure and its context and a suitable recombining operation. Thus, we imagine a learning model where the learner circulates between phases of building partial hypotheses and phases of integrating and testing those smaller units, sometimes considering them as blackboxes and sometimes decomposing and reassembling the parts in a different way if they have proven flawed, thereby constantly acting as a teacher for itself.

These are just some arguments we would like to name in favour of the study of Algorithmic Language Learning and Grammatical Inference.

Index

$<>$, 44
A-derivation tree, 144
$D_{\mathcal{G}}$, 136
L-transition, 98
S-composure, 48
T-consistent, 49
T-representative, 49
$[\ ]_L$, 23
$\Sigma(\textsc{red})$, 48
$\approx$, 44
$\bigsqcup$, 82
$\bowtie$, 106
δ^*, 22
δ^+, 22
$\equiv_L$, 23, 122
$\equiv_L^>$, 122
γ_p, 94
$\langle \Sigma, \sigma \rangle$, 18
$\mathbb{C}_\Sigma$, 20
$\mathbb{E}_\Sigma$, 21
$\mathbb{S}_\Sigma$, 20
$\mathbb{T}_\Sigma$, 19
$\mathcal{A}(t)$ (for a tree t), 22
$\mathcal{A}_L$, 24
$\mathcal{A}_L^\bullet$, 24, 79
$\mathcal{A}_L^\circ$, 24, 79
$\mathcal{A}_T$ (for a table T), 46
$\mathcal{C}_L$, 79
$\mathcal{C}_q$ (for a state q), 22
$\mathcal{E}_L$, 23
$\mathcal{E}_L(e)$ (for a context e), 151
$\mathcal{G}$-representative (FCP), 142
$\mathcal{G}$-representative (primal), 137
$\mathcal{G}(K, X, \mathcal{O})$, 131, 140
$\mathcal{G}_K$ (for a stub set K), 149
$\mathcal{G}_X$ (for a context set X), 152
$\mathcal{L}(\mathcal{A})$ (for an automaton $\mathcal{A}$), 22
$\mathcal{L}(\mathcal{G})$ (for a grammar $\mathcal{G}$), 25
$\mathcal{L}_A$ (for a nonterminal A), 25
$\mathcal{L}_q$ (for a state q), 22
$\mathcal{P}_L$, 80
$\mathcal{P}_S$, 82
$\mathcal{R}_L$, 81
$\mathcal{R}_L^p$, 94
$\mathcal{R}_T$ (for a table T), 83
$\mathcal{S}(s)$ (for a stub s), 149
$\mathcal{S}_\Sigma$, 149
$\mathcal{T}^d$, 27
$Cont(t)$ (for a tree t), 20
$Env^m(L)$, 130
$Env^{\leq r}(L)$, 130
$STA(X)$, 46
$Subt(t)$ (for a tree t), 19
$Sub^m(L)$, 130
$Sub^{\leq r}(L)$, 130
$mSubt(X_+)$, 67
$min_{\preceq}(X)$, 55
$obs(s, e)$, 44
$row(s, e)$, 44
$\odot$, 130
$\odot^j$, 130
ω-regular string, 37, 76, 163
$\preceq$, 55
$\sqcup$, 82
$\sqsubseteq$, 82
$\square$, 16, 20, 55

$\lhd$, 106
$e[\![s]\!]$, 21
$f(\Box_1, \ldots, \Box_m)$, 20
k-kernel, 124
k, l-substitutability, 126, 160
q_s (for a tree s), 61
$s^{-2}L$ (for a stub s), 149
$t^{-1}L$ (for a tree t), 79
t_γ, 94
x-context, 125

Abstract Categorial Grammars, 157

behaviourally correct, 39
Binary Feature Grammar (BFG), 128
both-sided string context, 16

characteristic set, 149, 154
closed (table), 45
complete (table), 44
composed residual language, 80
composed row, 82
congruentiality, 124
consistent (table), 45
context-determinism, 124
contradiction backtracing, 61, 184
control language, 77, 162
correction queries, 40, 76
counter automaton, 37, 76, 163
counterexample, 39
cover (a row), 82
CPNI, 149
cross-serial dependency, 161

deceiving (pair of trees), 99
depth (of a tree context), 20
depth (of a tree), 18
DFTA, 23
direct subtree, 19
distinguishing context, 44
distribution, 124
distributional learning, 121, 124
Distributional
    Lattice Grammar (DLG), 129
dual (approach), 129

environment, 21
equivalence query (EQ), 39
even linear grammar, 160, 162
excluding context, 82
exclusive (for a language), 89
extended EQ, 148
extension (of a string), 15

failure state, 24
faithful (for a language), 90
finite context property (FCP), 124
finite elasticity, 163
finite kernel property (FKP), 124
finite telltale set, 163
finite thickness, 163
FTA, 22
function-distinguishability, 77, 161

generalization-closedness, 48

hedge, 37
hyperedge
    replacement grammars, 157

identification in the limit, 39
incorrect rule (dual), 142
incorrect rule (primal), 137
index (of a language), 23
Inversion Transduction
    Grammar, 157, 162

join (of rows), 82

lexicalized grammar, 162
Linear Context-Free
    Rewriting Systems, 157
linear grammar, 160
local testability, 77
local tree, 26

matrix grammar, 77, 163
membership query (MQ), 40
mild context-sensitivity, 34, 161, 162
minimal access tree, 94
multi-dimensional tree, 25, 162
multi-word, 17
multiple context-free grammar (MCFG), 17, 124, 160

negative context, 44
node controlled graph grammar, 157
non-empty (information source), 64
non-void (information source), 64
nonterminal complexity, 158
nonterminally separated (NTS), 155
normal form (SCFTG), 131

objective grammar, 124
observation table, 44
obviously different, 44
one-shot learning, 41
oracle, 39

p-exclusive (for a language), 90
PAC learning, 10, 158
picture, 37
pol. characterizability, 118, 154, 164
positive context, 44
primal (approach), 129
prime residual language, 80
prime row, 82
prime-reduced RFTA, 94
probabilistic CFG (PCFG), 158

R-closed (table), 82
R-consistent (table), 83
R-exclusive (for a language), 90
R-representative (for a language), 89
ranked alphabet, 18
recognizable tree language, 23
representative (for a language), 47
residual language, 79
reversibility, 77, 161
RFSA, 78
RFTA, 78, 80

saturated (automaton), 81
separating context, 44
separative (for a language), 47
simple context-free tree grammar (SCFTG), 24
simple grammar, 161
specializing learner, 72, 164
structural information, 159
structural reversibility, 161
stub, 20
subsequential transducer, 162
substitutability, 124
subtree automaton (STA), 46
subtree-closedness, 44
syntactic concept lattice, 129

thickness (of a grammar), 154
transduction, 157, 161
Tree Adjoining Grammar (TAG), 25, 36, 166
tree automaton, 22
tree context, 20
tree domain, 18

very simple grammar, 161

weighted automaton, 161

yield, 25, 31, 162

Bibliography

[1] V. Amar and G. R. Putzolu. On a family of linear grammars. *Information and Control*, 7(3):283–291, 1964.

[2] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980.

[3] D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51(1):76–87, 1981.

[4] D. Angluin. Inference of reversible languages. *Journal of the Association for Computing Machinery*, 29(3):741–765, 1982.

[5] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[6] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[7] D. Angluin. Equivalence queries and approximate fingerprints. In *Proceedings of COLT 1989*, pages 134–145. Morgan Kaufmann, 1989.

[8] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[9] D. Angluin, L. Becerra-Bonache, A. H. Dediu, and L. Reyzin. Learning finite automata using label queries. In *Proceedings of ALT 2009*, volume 5809 of *LNAI*, pages 171–185. Springer, 2009.

[10] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of the 23rd annual ACM Symposium on Theory of Computing*, pages 444–454. ACM, 1991.

[11] D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *ACM Computing Surveys*, 15:237–269, 1983.

[12] J. L. Balcázar, J. Díaz, R. Gavaldà, and O. Watanabe. Algorithms for learning finite automata from queries: A unified view. In *Advances in Algorithms, Languages, and Complexity*, pages 53–72. Springer, 1997.

[13] P. Berman and R. Roos. Learning one-counter languages in polynomial time. In *Proceedings of SFCS 1987*, pages 61–67. IEEE Computer Society, 1987.

[14] J. Besombes and J.-Y. Marion. Learning tree languages from positive examples and membership queries. *Theoretical Computer Science*, 382:183–197, 2007.

[15] H. Blockeel and R. Brijder. Learning non-confluent NLC graph grammar rules. In *Proceedings of CiE 2009*, pages 60–69, 2009.

[16] L. Boasson and G. Sénizergues. NTS languages are deterministic and congruential. *Journal of Computer and System Sciences*, 31(3):332–342, 1985.

[17] B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In *Online Proceedings of IJCAI 21*, 2009.

[18] J. Brzozowski and H. Tamm. Theory of Átomata. In *Proceedings of DLT 2011*, volume 6795 of *LNCS*, pages 105–116. Springer, 2011.

[19] A. Burago. Learning structurally reversible context-free grammars from queries and counterexamples in polynomial time. In *Proceedings of COLT 1994*, pages 140–146. ACM, 1994.

[20] J. Carme, R. Gilleron, A. Lemay, A. Terlutte, and M. Tommasi. Residual finite tree automata. In *Proceedings of DLT 2003*, volume 2710 of *LNCS*, pages 171–182. Springer, 2003.

[21] R. C. Carrasco, J. Daciuk, and M. L. Forcada. Incremental construction of minimal tree automata. *Algorithmica*, 55:95–110, 2009.

[22] E. Charniak. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036, 1996.

[23] D. Chiang. An introduction to synchronous grammars, 2006.

[24] A. Clark. PAC-learning unambiguous NTS languages. In *Proceedings of ICGI 2006*, volume 4201 of *LNCS*, pages 59–71. Springer, 2006.

[25] A. Clark. A learnable representation for syntax using residuated lattices. In *Proceedings of the 14th Conference on Formal Grammar*, 2009.

[26] A. Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In *Proceedings of ICGI 2010*, volume 6339 of *LNCS*, pages 24–37. Springer, 2010.

[27] A. Clark. Efficient, correct, unsupervised learning of context-sensitive languages. In *Proceedings of CoNLL 2010*, pages 28–37, 2010.

[28] A. Clark. Learning context-free grammars with the syntactic concept lattice. In *Proceedings of ICGI 2010*, volume 6339 of *LNCS*, pages 38–51. Springer, 2010.

[29] A. Clark. Three learnable models for the description of language. In *Proceedings of LATA 2010*, volume 6031 of *LNCS*, pages 16–31. Springer, 2010.

[30] A. Clark. Towards general algorithms for grammatical inference. In *Proceedings of ALT 2010*, volume 6331 of *LNAI*, pages 11–30. Springer, 2010.

[31] A. Clark. Inference of inversion transduction grammars. In *Proceedings of ICML 2011*, 2011.

[32] A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007.

[33] A. Clark, R. Eyraud, and A. Habrard. A polynomial algorithm for the inference of context-free languages. In *Proceedings of ICGI 2008*, volume 5278 of *LNCS*, pages 29–42. Springer, 2008.

[34] A. Clark, R. Eyraud, and A. Habrard. Using contextual representations to efficiently learn context-free languages. *Journal of Machine Learning Research*, 11:2707–2744, 2010.

[35] A. Clark and S. Lappin. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell, 2011.

[36] A. Clark, R. Yoshinaka, and A. Kasprzik. Context-free inference from positive and negative evidence. Unpublished manuscript, 2011.

[37] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. Online publication, 2007.

[38] C. Cortes and M. Mohri. Learning with weighted transducers. In *Post-Proceedings of FSMNLP 2008*, pages 14–22. IOS Press, 2009.

[39] B. Courcelle. *Resolution of Equations in Algebraic Structures*, chapter 'On Recognizable Sets and Tree Automata'. Academic, 1989.

[40] C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.

[41] C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[42] F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using RFSA. In *Proceedings of ALT 2001*, volume 2225 of *LNCS*, pages 348–363. Springer, 2001.

[43] F. Denis, A. Lemay, and A. Terlutte. Residual finite state automata. *Fundamentae Informaticae*, 51:339–368, 2002.

[44] F. Drewes. *Grammatical Picture Generation – A Tree-Based Approach*. Texts in Theoretical Computer Science (EATCS series). Springer, 2006.

[45] F. Drewes. MAT learners for recognizable tree languages and tree series. *Acta Cybernetica*, 19, 2009.

[46] F. Drewes and J. Högberg. Learning a regular tree language from a teacher. In *Proceedings of DLT 2003*, volume 2710 of *LNCS*, pages 279–291. Springer, 2003.

[47] F. Drewes and J. Högberg. Extensions of a MAT learner for regular tree languages. In *Proceedings of SAIS 2006*, pages 35–44, 2006.

[48] F. Drewes and J. Högberg. Query learning of regular tree languages: How to avoid dead states. *Theory of Computing Systems*, 40(2):163–185, 2007.

[49] F. Drewes, J. Högberg, and A. Maletti. MAT learners for tree series: an abstract data type and two realizations. *Acta Informatica*, 48:165–189, 2011.

[50] J. Engelfriet and G. Rozenberg. Graph grammars based on node rewriting: An introduction to NLC graph grammars. In *Graph Grammars and Their Application to Computer Science*, pages 12–23, 1990.

[51] A. F. Fahmy and R. S. Roos. Efficient learning of real time one-counter automata. In *Proceedings of ALT 1995*, volume 997 of *LNCS*, pages 25–40. Springer, 1995.

[52] H. Fernau. Even linear simple matrix languages: Formal language properties and grammatical inference. *Theoretical Computer Science*, 289(1):425–456, 2002.

[53] H. Fernau. Identification of function distinguishable languages. *Theoretical Computer Science*, 290(3):1679–1711, 2003.

[54] A. Fujiyoshi. Linearity and nondeletion on monadic context-free tree grammars. *Information Processing Letters*, 93(3):103–107, 2005.

[55] A. Fujiyoshi and T. Kasai. Spinal-formed context-free tree grammars. *Theory of Computing Systems*, 33(1):59–83, 2000.

[56] P. García, M. Vázquez de Parga, G. I. Álvarez, and J. Ruiz. Universal automata and NFA learning. *Theoretical Computer Science*, 407:192–202, 2008.

[57] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, Hungary, 1984.

[58] F. Gécseg and M. Steinby. *Handbook of formal languages, vol. 3*, chapter 'Tree languages', pages 1–68. Springer, 1997.

[59] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[60] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[61] O. Grumberg, O. Kupferman, and S. Sheinvald. Variable automata over infinite alphabets. In *proceedings of LATA 2010*, volume 6031 of *LNCS*, pages 561–572. Springer, 2010.

[62] J. Gruska. Descriptional complexity of context-free languages. In *Proceedings of MFCS 1973*, pages 71–83, 1973.

[63] A. Habel and H. Kreowski. Some structural aspects of hypergraph languages generated by hyperedge replacement. In *Proceedings of STACS 1987*, pages 207–219, 1987.

[64] A. Habrard and J. Oncina. Learning multiplicity tree automata. In *Proceedings of ICGI 2006*, volume 4201 of *LNCS*, pages 268–280. Springer, 2006.

[65] Z. Harris. Distributional structure. *Word*, 10(2–3):146–162, 1954.

[66] T. Head, S. Kobayashi, and T. Yokomori. Locality, reversibility, and beyond: Learning languages from positive data. In *Proceedings of ALT 1998*, volume 1501 of *LNCS*, pages 191–204. Springer, 1998.

[67] J. E. Hopcroft and J. D. Ullmann. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Longman, 1990.

[68] H. Ishizaka. Polynomial time learnability of simple deterministic languages. *Machine Learning*, 5:151–164, 1990.

[69] S. Jain. Hypothesis spaces for learning. *Information and Computation*, 209:513–527, 2011.

[70] S. Jain and E. Kinber. Learning languages from positive data and a finite number of queries. *Information and Computation*, 204(1):123–175, 2006.

[71] S. Jain, D. Osherson, J. Royer, A. Sharma, and S. Weinstein. *Systems That Learn*. MIT Press, 1999.

[72] A. K. Joshi. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural description. In *Natural Language Processing*. Cambridge University Press, 1985.

[73] A. Kasprzik. Two equivalent regularizations for Tree Adjoining Grammars. Master's thesis, University of Tübingen, Germany, 2007.

[74] A. Kasprzik. A learning algorithm for multi-dimensional trees, or: Learning beyond context-freeness. In *Proceedings of ICGI 2008*, volume 5278 of *LNAI*, pages 111–124. Springer, 2008.

[75] A. Kasprzik. Generalizing over several learning settings. Technical report, 09-2, University of Trier, 2009.

[76] A. Kasprzik. Learning residual finite-state automata using observation tables. Technical report, 09-3, University of Trier, 2009.

[77] A. Kasprzik. Making finite-state methods applicable to languages beyond context-freeness via multi-dimensional trees. In *Post-Proceedings of FSMNLP 2008*, pages 98–109. IOS Press, 2009.

[78] A. Kasprzik. Generalizing over several learning settings. In *Proceedings of ICGI 2010*, volume 6339 of *LNAI*, pages 288–292. Springer, 2010.

[79] A. Kasprzik. Learning residual finite-state automata using observation tables. In *Proceedings of DCFS 2010*, volume 31 of *EPTCS*, pages 205–212, 2010.

[80] A. Kasprzik. Polynomial learning of regular multi-dimensional tree languages in different settings: A meta-algorithm. Technical report, 10-1, University of Trier, 2010.

[81] A. Kasprzik. Learning residual finite-state tree automata from membership queries and finite positive data. Technical report, 11-1, University of Trier, 2011.

[82] M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory*. MIT Press, 1994.

[83] S. Kepser and J. Rogers. The equivalence of Tree Adjoining Grammars and Monadic Linear Context-Free Tree Grammars. In *Proceedings of MOL 10*, pages 129–144, 2009.

[84] O. Klíma and L. Polák. On biautomata. In *Proceedings of NCMA 2011*, pages 153–164, 2011.

[85] M. Kuhlmann. *Dependency Structures and Lexicalized Grammars*. PhD thesis, Saarland University, Saarbrücken, Germany, 2007.

[86] K. Lang, B. Pearlmutter, and R. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *Grammatical Inference*, volume 1433 of *LNCS*, pages 1–12. Springer, 1998.

[87] S. Lange and S. Zilles. Formal language identification: Query learning vs. Gold-style learning. *Information Processing Letters*, 91:285–292, 2004.

[88] S. Lange and S. Zilles. Relations between Gold-style learning and query learning. *Information and Computation*, 203:211–237, 2005.

[89] S. Lombardy and J. Sakarovitch. The universal automaton. In *Logic and Automata*, pages 457–504, 2008.

[90] F. M. Luque and G. G. Infante López. PAC-learning unambiguous k, l-NTS$^{\leq}$ languages. In *Proceedings of ICGI 2010*, number 6339 in LNCS, pages 122–134. Springer, 2010.

[91] O. Maler and A. Pnueli. On the learnability of infinitary regular sets. *Information and Computation*, 118(2):316–326, 1995.

[92] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[93] J. Oncina and P. García. Inference of recognizable tree sets. Technical report, DSIC II/47/93, Universidad de Valencia, 1993.

[94] J. Oncina and P. García. Identifying regular languages in polynomial time. In *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific, 2002.

[95] J. Oncina, P. García, and E. Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458, 1993.

[96] R. Parekh and V. Honavar. On the relationship between models for learning in helpful environments. In *Proceedings of ICGI 2000*, volume 1891 of *LNCS/LNAI*, pages 207–220. Springer, 2000.

[97] R. Parekh, C. Nichitiu, and V. Honavar. A polynomial time incremental algorithm for learning DFA. In *Proceedings of ICGI 1998*, volume 1433 of *LNAI*, pages 37–49. Springer, 1998.

[98] L. Pitt. Inductive inference, DFAs, and computational complexity. In *Proceedings of AII 1989*, pages 18–44. Springer, 1989.

[99] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103(2):299–347, 1993.

[100] J. Rogers. Syntactic structures as multi-dimensional trees. *Research on Language and Computation*, 1:265–305, 2003.

[101] J. Rogers. wMSO theories as grammar formalisms. *Theoretical Computer Science*, 293:291–320, 2003.

[102] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76(2–3):223–242, 1990.

[103] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229, 1991.

[104] J. M. Sempere and P. García. A characterization of even linear languages and its application to the learning problem. In *Proceedings of ICGI 1994*, volume 862 of *LNCS*, pages 38–44. Springer, 1994.

[105] G. Sénizergues. The equivalence and inclusion problems for NTS languages. *Journal of Computer and System Sciences*, 31(3):303–331, 1985.

[106] E. Y. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.

[107] H. Shirakawa and T. Yokomori. Polynomial-time MAT learning of c-deterministic context-free grammars. *Transactions of the information processing society of Japan*, 34:380–390, 1993.

[108] Y. Takada. Grammatical interface for even linear languages based on control sets. *Information Processing Letters*, 28(4):193–199, 1988.

[109] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[110] C. Tîrnăucă. A note on the relationship between different types of correction queries. In *Proceedings of ICGI 2008*, volume 5278 of *LNCS*, pages 213–223. Springer, 2008.

[111] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[112] R. S. Wells. Immediate constituents. *Language*, 23(2):81–117, 1947.

[113] T. Yokomori. Polynomial-time learning of very simple grammars from positive data. In *Proceedings of COLT 1991*, pages 213–227, 1991.

[114] T. Yokomori. Learning non-deterministic finite automata from queries and counterexamples. In *Machine Intelligence 13*, pages 169–189, 1994.

[115] R. Yoshinaka. Identification in the limit of k, l-substitutable context-free languages. In *Proceedings of ICGI*, volume 5278 of *LNCS*, pages 266–279. Springer, 2008.

[116] R. Yoshinaka. Polynomial-time identification of multiple context-free languages from positive data and membership queries. In *Proceedings of ICGI 2010*, volume 6339 of *LNAI*, pages 230–244. Springer, 2010.

[117] R. Yoshinaka. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412(19):1821–1831, 2011.

[118] R. Yoshinaka. Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In *Proceedings of DLT 2011*, volume 6795 of *LNCS*, pages 429–440. Springer, 2011.

[119] R. Yoshinaka. Integration of the dual approaches in the distributional learning of context-free grammars. In *Proceedings of LATA 2012*, volume 7183 of *LNCS*. Springer, 2012.

[120] R. Yoshinaka and A. Clark. Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In *Proceedings of the 15th Conference on Formal Grammar*, 2010.

[121] R. Yoshinaka and M. Kanazawa. Distributional learning of abstract categorial grammars. In *Proceedings of LACL 2011*, volume 6736 of *LNCS*, pages 251–266. Springer, 2011.

[122] R. Yoshinaka and A. Kasprzik. Distributional learning of simple context-free tree grammars. In *Proceedings of ALT 2011*, volume 6925 of *LNAI*, pages 398–412. Springer, 2011.

Appendix

A.1 $MQ = 1$: 4 ways to use a counterexample

There are various options that we could have chosen for GENDET of how to "milk" a counterexample for $MQ = 1$. Let us compare four of them:

Lemma 34 *Let $MQ = 1$, let $T = \langle S, E, obs\rangle$ with $S = \text{RED} \cup \text{BLUE}$, and let c be a counterexample for $\mathcal{A}_T$. Assume that all elements of $Subt(c) \cap \Sigma(\text{RED})$ are in* BLUE. *Each of the following methods allows the creation of at least one more distinct row in T labeled by a* RED *element in at most one more step:*

a. Join $Subt(c)$ to RED.

b1. Join $Cont(c)$ to E.

b2. Find $s \in \text{BLUE}$ and $e \in \mathbb{C}_\Sigma$ with $c = e[\![s]\!]$ and join
$$Cont(c) \setminus \{e[\![s']\!] \in \mathbb{C}_\Sigma \mid s' \in Cont(s)\} \text{ to } E.$$

c. (MINIMIZE) If there is $s \in \text{BLUE}$ and $e \in \mathbb{C}_\Sigma \setminus \{\Box\}$ with $c = e[\![s]\!]$ but no $r \in \text{RED}$ with $\neg(r <> s)$ such that $e[\![r]\!]$ is a counterexample for $\mathcal{A}_T$ as well then add $\{e\}$ to E. Otherwise find $s' \in \text{BLUE}$, $e' \in \mathbb{C}_\Sigma \setminus \{\Box\}$, and $r' \in \text{RED}$ with $c = e'[\![s']\!]$ and $\neg(r' <> s')$ such that $e'[\![r']\!]$ is a counterexample for $\mathcal{A}_T$ and repeat the procedure MINIMIZE with input $e'[\![r']\!]$.

Proof. (a): Either such a row is created directly if E already contains a suitable separating context, or the table becomes inconsistent. To see the latter, assume that no element of $Subt(c)$ is obviously different from every RED element. As the automaton derived from the new table including $Subt(c)$ can evidently assign a different state to c than $\mathcal{A}_T$ although no new distinct row representing a separate state has been created this automaton must be nondeterministic, and the new table inconsistent. A consistency check (which is necessary with this method) would repair that in the following loop execution by adding to E a context distinguishing two RED elements that have not been obviously different before, thus creating another distinct row (also see [5]).

(c): Suppose MINIMIZE returns $\langle s, e\rangle$ with $s \in$ BLUE. As T is closed there is $r \in$ RED with $\neg(r <> s)$ but $e[\![s]\!]$ is a counterexample whereas $e[\![r]\!]$ is not. Consequently s and r should represent distinct states such that the context e leads to an accepting state from one of them but there is no such accepting state for the other. Obviously s and r will be distinguished by adding e to E. Note that we could also add s to RED (as done in [46]) but then we would need a consistency check to retrieve a separating context as for (a) because RED elements would no longer be sure to be pairwise obviously different.
(b1)/(b2): Consider (c) and the fact that the context added to E in (c) is an element of $Cont(c) \setminus \{e[\![s']\!] \in \mathbb{C}_\Sigma \mid s' \in Cont(s)\} \subseteq Cont(c)$ for some $s \in$ BLUE and $e \in \mathbb{C}_\Sigma$ with $c = e[\![s]\!]$. ■

The equivalent of method (a) is the one used in the original description of LSTAR for strings by Angluin [5]. Method (b1) was suggested for strings in a footnote in [91]. Method (b2) is based on the reflection that there is at least one suitable context in $Cont(\{c\})$ that does not have an element of $Cont(\{s\})$ as a subtree and thus we can avoid creating redundant columns by excluding contexts with subtrees in $Cont(\{s\})$ from the set of contexts we join to E. Method (c) which we have chosen for GENDET is inspired by a similar one for MQs and EQs in [46] which in turn is based on the technique of *contradiction backtracing* developed by Shapiro [106] but among other details our method differs from the one in [46] in that we have aimed to reduce the number of recursive calls to MINIMIZE by looking for an irreplaceable BLUE candidate first, and we do not add it to RED immediately but pass it on along with the distinguishing context which we have derived from the counterexample.

Remark Method (b1) obviously introduces all separating contexts that can be derived from c into the table at once but this may also lead to redundant columns. We proceed differently: We add one separating context at a time but we keep c as long as it can be a counterexample and hence eventually we will have added all separating contexts derivable from c as well. ♢

Remark When using methods (b2) and (c) the set of contexts E easily fails to be generalization-closed and/or S-composed but, as stated in Subsection 3.2.2, these are not essential properties for the extraction of an automaton from an observation table. The fact that E fulfils it both in [5, 46] (LSTAR) and [14] (learning from MQs and a finite positive sample) is just a by-product of the way how those algorithms compute their next distinction. ♢

A.2 About $\langle 1, 0, X_+, \emptyset \rangle$ and $\langle 1, 0, \emptyset, X_- \rangle$

The task of learning in these two settings is not of polynomial complexity. We will first sketch some arguments for the difficulty of designing a strategy that comes as close as possible to our learner's approach in those cases in which polynomial inference is ensured, and then give a proof for the complexity increase in the present constellations.

- $\langle 1, 0, X_+, \emptyset \rangle$: Let us suppose that we wanted to handle this case in a way analogous to those cases in which successful identification by GENDET performing a polynomial number of steps or queries is guaranteed. Assume $X_+ \neq \emptyset$. We would initialize the oracle $\mathcal{O}$ with the subtree automaton $STA(X_+)$ and then test the mergeability of states in $\mathcal{O}$ via EQs. If X_+ is representative then a positive counterexample implies the existence of states that should be merged, and a negative one the existence of states that should not have been. When we query the result of a merge (even without eliminating non-determinism by further merges) and receive a positive counterexample we could either repeat the same EQ and wait for a negative one – but the number of possible positive counterexamples may be infinite. Or we could query the next merge – but when (if!) we eventually obtain a negative counterexample we do not know which of the previous merges was illegitimate and should be undone. So this method is just as complex as it would be to ignore all counterexamples and simply query the result of every possible *set of merges* among the states of $\mathcal{O}$, and the number of those sets is exponential in $|Q_{\mathcal{O}}|$. Therefore, since we cannot proceed as in cases where polynomial inference is ensured we have chosen to eclipse this case from GENDET by the corresponding case distinctions.

- $\langle 1, 0, \emptyset, X_- \rangle$: This case is equally problematic to include in the present framework. First, observe that if X_- is separative then negative counterexamples do not contribute additional information, and their number may be infinite at any step. Second, the set of positive counterexamples obtained so far may not be representative so that we could not reliably detect an illegitimate merge because there may be accepting states of the solution that are not even represented in the current version of $\mathcal{O}$ such that the compatibility check is too weak. If we effect the merge then we might have to undo it on the receipt of another positive counterexample, which is a situation we want to avoid. Therefore this case is eclipsed from GENDET by its case distinctions as well.

The following proof was given for strings but we assume that the corresponding terminology is known to the reader. Recall that since strings can be seen as a special kind of non-branching trees negative learnability results for strings imply negative results for trees.
Suppose that there is an algorithm A learning from EQs and negative data such that for every regular language L, if A is run with an equivalence oracle for L and a finite negative sample $X_- \subseteq \Sigma^* \setminus L$ then A terminates and returns a DFA $\mathcal{A}$ fulfilling $\mathcal{L}(\mathcal{A}) = L$. We fix an alphabet $\Sigma = \{a, b\}$ for concreteness.

Theorem 10
The number of EQs asked by A cannot be bounded by a polynomial $p(n, m, s)$ where n is the index of the target language L and m is the maximum length of any counterexample and s is the sum of the lengths of the elements in X_-.

Proof. We show that this would imply the existence of an algorithm A' that learns regular languages using a number of EQs bounded by a polynomial $q(n, m)$, which has been shown to be impossible in [7].
Let $L' \subseteq \Sigma^*$ be the target language for A'. We first describe A' under the assumption that n is known to A' and that L' does not contain the empty word, and we sketch how to remove these assumptions later on. We extend Σ by a symbol c and simulate running A with a negative sample $X_- = \{c^n\}$.
For every EQ of A involving a DFA $\mathcal{A}_1$ the algorithm A' constructs a new DFA $\mathcal{A}_2$ by removing all c-transitions from $\mathcal{A}_1$ and then consults the oracle about $\mathcal{A}_2$ itself. If the answer is positive then we return $\mathcal{A}_2$ as the solution. If the answer is negative and we receive a counterexample w then w is also a counterexample for $\mathcal{A}_1$ and we return w to A as the answer to its query.
Let $\mathcal{A}'$ be the state-minimal DFA for L', and let it have n states. Since we assume $\varepsilon \notin L'$ the start state of $\mathcal{A}'$ is non-accepting. Use $\mathcal{A}'$ to define a DFA $\mathcal{A}''$ over the alphabet $\{a, b, c\}$ by adding a single cycle of c-transitions that starts from and ends in the start state and visits all non-accepting states before all accepting ones. Assume $\mathcal{L}(\mathcal{A}'') = L$.
Since the start state is non-accepting, the string c^n is not accepted by $\mathcal{A}''$ and thus $\{c^n\}$ is a negative sample of L. To see that it is also separative for L, note the following: For $n = 1$ any negative sample is separative. If $n > 1$ then there are both accepting and non-accepting states in $\mathcal{A}''$, and for every pair of distinct states q_1, q_2 in $\mathcal{A}''$ there is a suffix of c^n distinguishing them – if one is accepting and the other is not then the empty suffix suffices, and if both are accepting then consider the shortest suffix that takes one of them back to the (non-accepting) start state which will obviously take the other to an accepting state further ahead in the cycle of c-transitions. The argument runs analogously for two non-accepting states.

Thus the sample given to A is separative for L, and all EQs of A are answered correctly with respect to L. The algorithm A' only stops answering EQs for A as soon as it has successfully identified L'. If we assume that A' asks a number of EQs that is bounded by $p(n, m, s)$ then the number of EQs asked by A is bounded by $q(n, m) = p(n, m, n)$ since we have $s = n$.

To remove the assumption that n is known to A' we could run A' successively for all $n \geq 1$ until it succeeds. To remove the assumption that the empty word $\square$ is not in L there is an easy reduction that simulates A' on $L \setminus \{\square\}$ and correctly identifies L. (■)[2]

A.3 Proofs for $\mathcal{R}_L$

We reproduce the proofs for the following theorem from [20], at the same time adapting them to our notation and sometimes adding steps for clarity.

Theorem 11 *$\mathcal{R}_L$ is the state-minimal saturated RFTA recognizing L.*

For convenience, we repeat the two lemmata from [20] which were given as Lemmata 6 and 7 in Subsection 3.4.1 above (see [20] for the proofs):

Lemma 35 *Let $t_1, \ldots, t_n, t'_1 \ldots, t'_n \in \mathbb{T}_\Sigma$ for some $n \geq 1$.*
If we have ${t_i}^{-1}L \subseteq {t'_i}^{-1}L$ for all $i \in \{1, \ldots, n\}$ then we also have
$$f(t_1, \ldots, t_n)^{-1}L \subseteq f(t'_1 \ldots, t'_n)^{-1}L \text{ for all } f \in \Sigma_n.$$

Lemma 36 *Let $t_1, \ldots, t_n \in \mathbb{T}_\Sigma$ for some $n \geq 1$, and*
for each $i \in \{1, \ldots, n\}$, let $X_i \subseteq \mathbb{T}_\Sigma$ be a set of trees such that
${t_i}^{-1}L = \bigcup\{t^{-1}L \in \mathcal{C}_L \mid t \in X_i\}$. Let $f \in \Sigma_n$. We have
$$f(t_1, \ldots, t_n)^{-1}L = \bigcup\{f(t'_1, \ldots, t'_n)^{-1}L \in \mathcal{C}_L \mid \forall i \in \{1, \ldots, n\} : t'_i \in X_i\}.$$

The authors of [20] prove Theorem 11 by a sequence of lemmata as follows.

Lemma 37 *For all $t \in \mathbb{T}_\Sigma$, we have $t^{-1}L = \bigcup \delta_L^*(t)$.*

Proof. This is shown by induction over the depth of t.
For $t = a \in \Sigma_0$ this follows directly from the definition of δ_L as any residual language of L equals the union of the prime residual languages it includes.

Now let $t = f(t_1, \ldots, t_n)$ and $dpt(t) = d + 1$ for some $d \geq 0$ and assume the claim to hold for all trees t' with $dpt(t') \leq d$, which implies that we have ${t_i}^{-1}L = \bigcup \delta_L^*(t_i)$ for all $i \in \{1, \ldots, n\}$. By Lemma 36 we obtain

[2] This proof is due to an anonymous reviewer of one of the author's submissions.

$$t^{-1}L = \bigcup\{\gamma \in \mathcal{C}_L \mid \exists t'_1, \ldots, t'_n : \forall i \in \{1, \ldots, n\} : {t'_i}^{-1}L \in \delta_L^*(t_i) \wedge \gamma = f(t'_1, \ldots, t'_n)^{-1}L\}.$$

Any residual language of L equals the union of the prime residual languages it includes, so for all $t'_1, \ldots, t'_n \in \mathbb{T}_\Sigma$ with ${t'_i}^{-1}L \in \delta_L^*(t_i)$ for all $i \in \{1, \ldots, n\}$ we have $f(t'_1, \ldots, t'_n)^{-1}L = \bigcup\{\gamma \in \mathcal{P}_L \mid \gamma \subseteq f(t'_1, \ldots, t'_n)^{-1}L\}$. This implies

$$t^{-1}L = \bigcup\{\gamma \in \mathcal{C}_L \mid \exists t'_1, \ldots, t'_n : \forall i \in \{1, \ldots, n\} : {t'_i}^{-1}L \in \delta_L^*(t_i) \wedge \gamma = \bigcup\{\gamma' \in \mathcal{P}_L \mid \gamma' \subseteq f(t'_1, \ldots, t'_n)^{-1}L\}\}.$$

Since we have $\gamma' \subseteq f(t'_1, \ldots, t'_n)^{-1}L$ if and only if there exists a transition $\langle f, {t'_1}^{-1}L \cdots {t'_n}^{-1}L\rangle \mapsto \gamma' \in \delta$ we obtain $t^{-1}L = \bigcup \delta_L^*(t)$. ■

Lemma 38 *The FTA $\mathcal{R}_L$ recognizes L, i.e., for all $t \in \mathbb{T}_\Sigma$ we have*

$$(\exists y \in F_L : y \in \delta_L^*(t)) \Leftrightarrow t \in L.$$

Proof. Let $y \in F_L$ and $y \in \delta_L^*(t)$. Then $y \subseteq t^{-1}L$ by Lemma 13. Since $y \in F_L$ we have $\Box \in y$ and thus $\Box \in t^{-1}L$ as well, which implies $t \in L$.
Let $t \in L$. Then $\Box \in t^{-1}L$, and there is $y' \in Q_L$ with $y' \in \delta_L^*(t)$ and $\Box \in y'$ due to Lemma 37, which implies $y' \in F_L$ and thus $\mathcal{R}_L(t) = 1$. ■

Lemma 39 *Let $y, y' \in Q_L$. Then $y \subseteq y'$ implies $\mathcal{C}_y \subseteq \mathcal{C}_{y'}$.*

Proof. Let $t, t' \in \mathbb{T}_\Sigma$ with $t^{-1}L = y$ and ${t'}^{-1}L = y'$.
Observe that for all $t_1, \ldots, t_n \in \mathbb{T}_\Sigma$, $f \in \Sigma_n$ and all $i \in \{1, \ldots, n\}$ we have $(f(t_1, \ldots, t_n)_i^{\Box}[\![t]\!])^{-1}L \subseteq (f(t_1, \ldots, t_n)_i^{\Box}[\![t']\!])^{-1}L$ by Lemma 35, and that the definition of δ_L implies $\langle f, y_1 \cdots y_n \mapsto y_0\rangle \in \delta_L \Leftrightarrow {t_0}^{-1}L \subseteq f(t_1, \ldots, t_n)^{-1}L$ with ${t_i}^{-1}L = y_i$ for all $i \in \{0, \ldots, n\}$.
For each $\langle f, y_1 \cdots y_n \mapsto y''\rangle \in \delta_L$ with $y_j = y'$ for some $j \in \{1, \ldots, n\}$ we have $y'' \subseteq (f(t_1, \ldots, t_n)_j^{\Box}[\![t']\!])^{-1}L$, which implies $y'' \subseteq (f(t_1, \ldots, t_n)_j^{\Box}[\![t]\!])^{-1}L$ due to $t^{-1}L \subseteq {t'}^{-1}L$, and hence there exists a transition $\langle f, y'_1 \cdots y'_n \mapsto y''\rangle \in \delta_L$ with $y'_j = y$ and $y'_i = y_i$ for all $i \in \{1, \ldots, n\}$. Thus the claim is shown. ■

Lemma 40 *$\mathcal{R}_L$ is an RFTA.*

Proof. Let $y \in Q_L$ and $t \in \mathbb{T}_\Sigma$ with $t^{-1}L = y$. We have $t^{-1}L = \bigcup \delta_L^*(t)$ by Lemma 37. If $t^{-1}L$ would strictly contain all prime residual languages in $\delta_L^*(t)$ then it would be composite. Hence, $t^{-1}L = y \in \delta_L^*(t)$. Since $\mathcal{R}_L$ recognizes L we have $t^{-1}L = \bigcup\{\gamma \subseteq \mathbb{C}_\Sigma \mid \exists y' \in \delta_L^*(t) : \gamma = \mathcal{C}_{y'}\}$. Consequently, $\mathcal{C}_y \subseteq t^{-1}L$. On the other hand, for all $y' \in \delta_L^*(t)$ we have $y' \subseteq y$ by Lemma 13, and thus $\mathcal{C}_{y'} \subseteq \mathcal{C}_y$ by Lemma 39. Since the union of all those sets $\mathcal{C}_{y'}$ equals $t^{-1}L$ we obtain $t^{-1}L \subseteq \mathcal{C}_y$ and $t^{-1}L = \mathcal{C}_y$. Consequently, for each state $y \in Q$ the set $\mathcal{C}_y$ corresponds to a (prime) residual language of L and $\mathcal{R}_L$ is an RFTA. ■

Lemma 41 *$\mathcal{R}_L$ is state-minimal.*

Proof. Let $t \in \mathbb{T}_\Sigma$ be such that there is no $y \in Q_L$ with $t^{-1}L = \mathcal{C}_y$. We have $t^{-1}L = \bigcup \delta_L^*(t)$ by Lemma 37. Since we have $\mathcal{C}_{y'} = y'$ for all $y' \in Q_L = \mathcal{P}_L$ (consider the arguments in the proof of the previous lemma) and since there is no $y \in Q_L$ with $t^{-1}L = \mathcal{C}_y = y$ the residual language $t^{-1}L$ must be a union of prime residual language that it strictly contains, i.e., $t^{-1}L$ is composed. Thus, for all $\gamma \in \mathcal{P}_L$ there is $y \in Q_L$ with $\gamma = \mathcal{C}_y$ and the RFTA $\mathcal{R}_L$ has $|\mathcal{P}_L|$ states, which is the smallest number possible. ■

The FTA $\mathcal{R}_L$ is saturated by the definition of δ_L, and Theorem 11 is proven.