Mach Learn (2014) 96:5–31
DOI 10.1007/s10994-013-5403-2

Distributional learning of parallel multiple context-free grammars

Alexander Clark · Ryo Yoshinaka

Received: 8 December 2012 / Accepted: 26 July 2013 / Published online: 3 October 2013

Abstract Natural languages require grammars beyond context-free for their description. Here we extend a family of distributional learning algorithms for context-free grammars to the class of Parallel Multiple Context-Free Grammars (PMCFGs). These grammars have two additional operations beyond the simple context-free operation of concatenation: the ability to interleave strings of symbols, and the ability to copy or duplicate strings. This allows the grammars to generate some non-semilinear languages, which are outside the class of mildly context-sensitive grammars. These grammars, if augmented with a suitable feature mechanism, are capable of representing all of the syntactic phenomena that have been claimed to exist in natural language.

We present a learning algorithm for a large subclass of these grammars, that includes all regular languages but not all context-free languages. This algorithm relies on a generalisation of the notion of distribution as a function from tuples of strings to entire sentences; we define nonterminals using finite sets of these functions. Our learning algorithm uses a nonprobabilistic learning paradigm which allows for membership queries as well as positive samples; it runs in polynomial time.

Keywords Mildly context-sensitive · Grammatical inference · Semilinearity

1 Introduction and motivation

Natural languages present some particular challenges for machine learning—primarily the fact that the classes of representations that will ultimately be required for a satisfactory description of natural language syntax are clearly much richer than the simple Markov models

Editors: Jeffrey Heinz, Colin de la Higuera, and Tim Oates.

A. Clark (✉)
Department of Philosophy, King's College London, London, UK
e-mail: alexander.clark@kcl.ac.uk

R. Yoshinaka
Graduate School of Informatics, Kyoto University, Kyoto, Japan
e-mail: ry@i.kyoto-u.ac.jp

that underpin much machine learning work. Indeed, even context-free grammars are insufficiently powerful to represent many linguistic phenomena. Accordingly it is important to be able to develop learning algorithms that are capable of learning the sorts of dependencies that we observe in natural languages.

We situate this problem in the field of grammatical inference. In its purest form, we are interested in algorithms which receive as input a sequence of strings of symbols, and are required to infer a grammar that represents a formal language: a set of strings that is typically infinite. We recall at this point the classic negative results of Gold (1967), who considered the situation where the learner only has access to the strings that are in the language, and there are no nontrivial restrictions on the sequences of examples that the learner must learn from. Within such a restrictive framework we can only learn classes of languages that have some language theoretic closure properties; see for example the structurally very similar algorithms for learning subclasses of regular languages given by Angluin (1982), for learning subclasses of context-free grammars by Clark and Eyraud (2007) and for subclasses of multiple context-free grammars (MCFGs) by Yoshinaka (2011a). These classes of languages become increasingly limited as we ascend the hierarchy: indeed, in the case of the MCFG learning approach, even some languages consisting of only a single string are not learnable. It seems important therefore to consider a somewhat weaker and less restrictive learning model. As an alternative to considering a probabilistic learning model, which enlarges the class of languages that can be learned by limiting the data sequences and the convergence criterion in a realistic way, we approach this problem by allowing the learner an additional very high quality source of information: we consider an active learning model where the learner can ask membership queries. In other words, the learner is not entirely a passive recipient of examples but can query an oracle as to whether a particular string of symbols is in the language or not. This takes us partly towards the minimally adequate teacher model (MAT) introduced by Angluin (1987); also called exact query learning. We do not however go this far—we keep a stream of positive examples as part of the learning model.

We are motivated at a high level by a desire to understand first language acquisition—in general by attempting to operationalise and extend the discovery procedures of American structuralist linguistics, for which we use the umbrella term “distributional learning”. Clearly the situation of language acquisition is quite different from the highly idealised learning models we consider in this paper, notably in that the child learner can interact with the environment in a number of ways not captured by the simple idealisation of a membership query and in the importance of semantics or meaning in the acquisition process. This raises a number of methodological issues that are outside the scope of this paper: see Clark and Lappin (2011) for a further justification of the approach in this article, but see also Berwick et al. (2011) for an opposing view.

There are two fundamental language-theoretic boundaries that are closely related: the first is the boundary between regular languages and non-regular languages, the second is between semilinear languages and non-semilinear languages. Semilinear languages are, roughly speaking, those where the lengths of the strings in the language are linear combinations of a finite set of fixed lengths.[1] Joshi et al. (1991) say that semilinearity:

> Is intended to be an approximate characterization of the linguistic intuition that sentences of a natural language are built from a finite set of clauses of bounded structures using certain simple linear operations.

[1] See Michaelis and Kracht (1997) for a precise definition.

These two boundaries are clearly related because of the following theorem: a language is semilinear iff it is letter equivalent to a regular language. Clearly the class of semilinear languages is not directly useful as it is uncountable and thus contains undecidable languages, but it serves to help demarcate the class(es) of mildly context-sensitive (MCS) languages (Joshi et al. 1991). All standardly used grammatical formalisms are semilinear—regular grammars, context-free grammars, multiple context-free grammars, tree adjoining grammars and so on all define subsets of the class of semilinear languages. Examples of non-semilinear languages include $\{a^{2^n} \mid n > 0\}$ and $\{a^{n^2} \mid n > 0\}$, which can be parsed in linear time, yet cannot be expressed by MCS formalisms. Formalisms that can define non-semilinear languages include Elementary Formal Systems (Smullyan 1961), Range Concatenation Grammars (Boullier 1999), Literal Movement Grammars (Groenink 1995) and the representation we will use in this paper, Parallel Multiple Context-Free Grammars (Seki et al. 1991), that we define in Sect. 3.

Recent work in grammatical inference has made significant progress in learning semilinear, non-regular languages using representations such as context-free grammars (Clark and Eyraud 2007) and multiple context-free grammars (Yoshinaka 2011a). Crucially, these representations just use concatenation—substrings are combined, but never copied. The richer operations used by MCFGs are just generalisations of concatenation to tuples of strings; these include for example various types of intercalation where a string can be inserted into a gap in another string.

There is a broad consensus that natural language string sets are *semilinear*, and so attention has focused largely on properties of formalisms that generate semilinear languages. We review these formalisms in Sect. 2. However there are a number of cases where linguistic data suggest that there are richer processes involved, processes that either require or might benefit from a more powerful formalism. These data, which we examine in detail in the second half of Sect. 2, are still controversial. However, regardless of what the final determinations on these examples are, it is still useful to have richer learning algorithms available since even if these formalisms are not strictly speaking necessary, the additional descriptive power that they give us may allow for a more compact and succinct grammar than we could obtain with a semilinear formalism.

In this paper we extend distributional learning to the inference of non-semilinear languages; the major technical detail is the extension of the notion of context. We give an intuitive explanation of this in Sect. 4, and present the technical details of the learning target and algorithm together with the proof of its correctness in Sect. 5.

2 Language theory and linguistics

For many years, a default assumption in computational linguistics has been that context-free grammars are more or less adequate for defining natural language syntax. In linguistics on the other hand, the orthodox view has been that they are clearly inadequate. The starting point for this debate is invariably the Chomsky hierarchy (Chomsky 1956). While seminal, and an immensely important contribution, this hierarchy is now showing its age in a number of respects. Most importantly since it predates the discovery or invention of the theory of computational complexity, there is no natural characterisation within this family of the class of efficiently recognizable languages—under standard assumptions this is the class PTIME. Similarly, the class of context-sensitive grammars is far too powerful to be of any practical use, and moreover it is difficult to define classes between context-free grammars and context-sensitive grammars within this family of formalisms. Finally, it is hard to attach semantics to a top-down derivation: as Kracht (2011) argues, in our view convincingly, semantic interpretation is a process that can only be viewed naturally as a process of composition proceeding bottom-up.

Recently a broad consensus has been forming that it is more appropriate to base a language hierarchy on a family of bottom-up systems, as in Smullyan (1961), rather than the top-down string rewriting systems in the original Chomsky hierarchy. Accordingly in this paper we consider a family of formalisms that are bottom-up in this sense. A long-standing debate in computational linguistics is over the exact language-theoretic position of natural languages, with respect to the original Chomsky hierarchy or its more modern derivatives. There are several important distinctions that we wish to keep distinct. Suppose we have a natural language like English or Kayardild, and we assume that we can in some way stipulate a sharp dividing line between grammatical and ungrammatical utterances, given a presumably infinite set of possible grammatical sentences. For each such language and a putative class of grammars $\mathcal{G}$ we can ask the following questions. Firstly, whether the language, considered as a set of strings, lies in the class generable by the formalism—this is the weakest claim; and therefore a negative answer to the question provides the strongest evidence that $\mathcal{G}$ is inadequate. Secondly, supposing that we can find a grammar that generates the right set of strings, we can ask further whether there is a grammar that generates the right set of structures. For example, we might have a language like Dutch, which is apparently "weakly" context-free, in the sense that the set of strings is a context-free language, but not strongly context-free, in the sense that no context-free grammar can generate an adequate set of structures. Finally, we can ask whether we can find a reasonably sized grammar that generates it.

It is important to remember that so far nobody has been able to construct an adequate grammar for any natural language in any formalism. Therefore our answers to these questions above will only be partial. We can come up with convincing negative answers: it is possible to show that a given formalism is inadequate using various mathematical techniques, typically exploiting closure properties of the formalism, such as closure under intersection with regular languages. However, we cannot at the moment come up with a definitive positive argument that a particular natural language is in a given class, since that would require producing a completely adequate grammar for that language, where adequacy is defined in one of the senses above. The absence of an argument showing that a formalism $\mathcal{G}$ is inadequate can of course be taken as defeasible evidence that the class is adequate for natural language description. We review here the current status of this debate, and take an integrated view of the various examples that appear to be weakly or strongly beyond the expressive power of CFGs. Though our focus in this paper is on learning languages as sets of strings, we nonetheless want to be able to produce grammars that give reasonable structures.

The Chomsky hierarchy in its original form has the (nontrivial) classes of regular, context-free and context-sensitive grammars. In Fig. 1 we show a fraction of the more modern hierarchy of bottom-up systems.

We start by giving an informal ostensive definition of a format for our rules (Groenink 1997), before giving a formal definition in Sect. 3. We assume we have some fixed alphabet Σ whose elements we will call *letters*, though they may in fact correspond to phonemes or words. We will use the symbols a, b, c here to refer to elements of Σ. We assume we have a set of nonterminals which we label $A, B, C, \ldots$, together with some variables which we write as x_i, y_i. We will proceed from the simplest rules to the most complex. Every production is written in Horn clause notation and consists of a single clause on the left and a possible empty sequence of clauses on the right. Terminal symbols can only appear on the left hand side of the rule. The most basic production is one with an empty clause on the

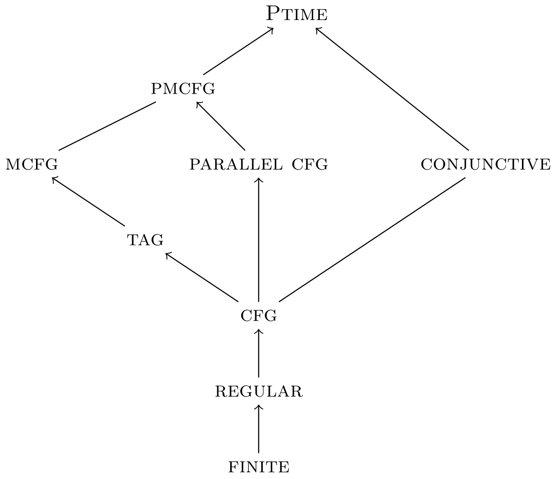


Fig. 1 A view of a part of the language-theoretic hierarchy below PTIME. All inclusions are proper

right:

$$A(a) :- .$$

This asserts that a terminal symbol a can be derived from the nonterminal symbol A. The next most basic production is one like

$$A(ax_1) :- B(x_1).$$

This means that if an arbitrary string u is derived from the nonterminal symbol B, then A may derive au. Here we constrain the rules to have only one variable, and the body of the clause on the left hand side can only be of length 2, with the terminal symbol occurring first. We call this type of production *regular*. The next type is a type exemplified by

$$A(x_1ay_1b) :- B(x_1), C(y_1).$$

If B and C derive strings u and v, respectively, A may derive $uavb$. We call this a *context-free* production. It is not limited to the case where we only have two nonterminals on the right of the rule, but allows an unbounded number, and we may have unlimited numbers of terminal symbols on the right. Note that there is a bijection between the variables that occur on the left and the variables that occur on the right hand side of the rule, and that they are distinct.

We can now extend the rules in three qualitatively distinct ways. First we may allow the nonterminals to take more than one argument. This we call an *mcfg* rule.

$$A(x_1y_1, x_2) :- B(x_1, x_2), C(y_1).$$

Here the nonterminals A and B take two arguments—we say that they are of dimension 2. Whenever B derives a pair of strings (u, v) and C derives w, the nonterminal A derives the pair (uw, v). The second generalisation is that we allow a variable to occur more than once on the left hand side of the production.

$$A(x_1ay_1bx_1) :- B(x_1), C(y_1).$$

We call this a *parallel cfg* rule or a copying rule. This has the semantic effect, which we define later, of copying the substring x_1. So if B derives u and C derives v, then A will derive $uavbu$. The third generalisation is where we allow a variable to occur more than once on the right hand side of the production.

$$A(x_1) :- B(x_1), C(x_1).$$

We call this a *conjunctive* rule: here if B derives w *and* C also derives w then A derives w. We do not consider this type of rule in this paper, though we remark briefly on it in the conclusion.

If we allow all three types of rule together, allowing a rule which may have multiple symbols occur both on the right and on the left, then we have the class of all unrestricted rules, of which we give the following illustrative example.

$$A(x_1x_1, by_1y_2) :- B(x_1, y_1), C(y_1, y_2).$$

Corresponding to this syntactic hierarchy of productions we have a corresponding strict hierarchy of grammars and languages shown in Fig. 1. At the top we have the class PTIME, which is the set of all languages that can be defined using grammars of this type (Ljunglöf 2005; Groenink 1997). In this paper we target the class of PMCFGs which allows all but conjunctive rules. If we allow only conjunction together with CFG rules, then we obtain the class of Conjunctive grammars (Okhotin 2001). If we allow only *mcfg* rules, then we obtain the class of Multiple context-free grammars (MCFGs), which are equivalent, modulo some minor technical details, to the class of Linear Context-Free Rewriting Systems (Vijay-Shanker et al. 1987). We note that the class of four convergent mildly context-sensitive formalisms studied by Vijay-Shanker and Weir (1994) are equivalent to a class intermediate between MCFG and CFG which we mark with TAG in the diagram; this is equivalent to the class of well-nested MCFGs of dimension 2; indeed there is a complex infinite hierarchy on the arc between CFG and MCFG (Seki et al. 1991). As we shall see we will use two parameters, the dimension and rank, to describe the classes of grammars that we use, together with a third which controls the degree of copying.

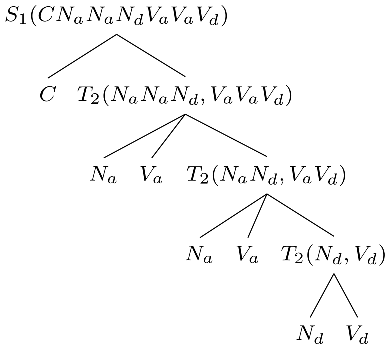
We now consider various phenomena which motivate the use of a formalism more powerful than CFGs.

2.1 Displacement/movement

A phenomenon which was taken to indicate the necessity for a formalism more powerful than context-free grammars is displacement or movement—typified by wh-movement in English. For example, we have the declarative sentence ‘John liked that restaurant’. We can form a question from this using what is called wh-movement: ‘Which restaurant did John like’?

We emphasize that these examples do not show that the set of strings is not a context-free language. One can stay within the class of context-free languages, and represent these by using a richer formalism that is capable of modeling these structures by using richly structured nonterminals, using for example the meta-grammar approach of GPSG (Gazdar et al. 1985). However from a learnability point of view it seems to be desirable to learn these using a richer formalism, which can directly represent the displacement, rather than folding it into some feature system. More precisely, within the framework of distributional learning, displacement causes a significant problem with the inference of context-free grammars, because it causes a potentially exponential increase in the number of nonterminals we need in a grammar. In addition, the sorts of derivation trees that we get for these examples seem to be inappropriate for supporting natural language interpretation.

Fig. 2 Derivation tree for the Swiss German example



2.2 Cross-serial dependencies

The example which definitively established that CFGs were not weakly adequate was the case of cross-serial dependencies in Swiss German (Huybrechts 1984; Shieber 1985). We present here the data in a form very close to the original presentation. In the particular dialect of Swiss German considered by Shieber, the data concerns a sequence of embedded clauses.

Let's abstract this a little bit and consider a formal language for this non context-free fragment of Swiss German. We consider that we have the following words or word types: N_a, N_d which are respectively accusative and dative noun phrases, V_a, V_d which are verb phrases that require accusative and dative noun phrases respectively, and finally C which is a complementizer which appears at the beginning of the clause. Thus the "language" we are looking at consists of sequences like CN_aV_a and CN_dV_d and $CN_aN_aN_dV_aV_aV_d$, but crucially does not contain examples where the sequence of accusative/dative markings on the noun sequence is different from the sequence of requirements on the verbs. So it does not contain CN_dV_a, because the verb requires an accusative and it only has a dative, nor does it include $CN_aN_dV_dV_a$, because though there are the right number of accusative and dative arguments (one each) they are in the wrong order—the reverse order.[2] More precisely, for a string w in $\{V_a, V_d\}^+$, we write $\overline{w}$ for the corresponding string in $\{N_a, N_d\}^*$: formally we define $\overline{V_a} = N_a$, $\overline{V_d} = N_d$, $\overline{V_a\alpha} = N_a\overline{\alpha}$ and $\overline{V_d\alpha} = N_d\overline{\alpha}$. The sublanguage we are concerned with is the language $L_{sg} = \{C\overline{w}w \mid w \in \{V_a, V_d\}^+\}$. This language is defined through intersection of the original language with a suitable regular language and a homomorphism relabelling the strings. Since CFGs are closed under these operations, and L_{sg} is clearly not context-free, this establishes the non-context-freeness of the original language.

We present a PMCFG for this fragment: since we do not use any copying this is formally equivalent to an MCFG. Figure 2 gives a derivation tree for the string $CN_aN_aN_dV_aV_aV_d$ with respect to this grammar.

Example 1 This grammar has two nonterminals S of dimension 1, and D of dimension 2, and the following set of productions:

$$S(Cx_1x_2) :- D(x_1, x_2)$$

$$D(N_ax_1, V_ax_2) :- D(x_1, x_2)$$

[2] Actually this is incorrect: according to Shieber the nested orders are acceptable as well. We neglect this for ease of exposition.

$$D(N_d x_1, V_d x_2) :- D(x_1, x_2)$$

$$D(N_a, V_a):-$$

$$D(N_d, V_d):-$$

2.3 Copying

While semilinearity is generally considered to be a property that holds for all natural languages, there is an increasing amount of evidence that there are some phenomena that take natural languages out of the class of semilinear languages. None of the arguments here are as conclusive as Shieber's argument that natural languages are not weakly context-free (Shieber 1985) but are nonetheless suggestive. In what follows we will describe fragments of languages that are not semilinear and will provide toy PMCFGs for these fragments.

2.4 Reduplication

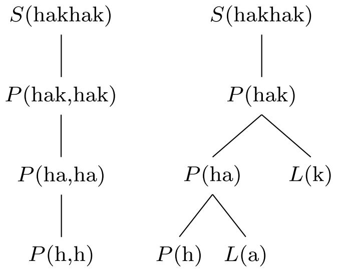
Another important area where (non-recursive) copying operations may occur is in morphology and phonology (Inkelas and Zoll 2005; Inkelas 2008). This can range from duplication of a limited amount of material at the beginning of a word to the complete copying of a full stem.

The classic example of this is in Indonesian where plurals are formed by duplication of a stem. Thus the singular form of the word 'man' is 'orang' and the plural is 'orangorang'. In principle there is no limit to the length of material that can be copied. Similarly in Dyirbal we have the same construction (Inkelas 2008):

Singular	Plural	Gloss
midi	midi-midi	'lots of little ones'
gulgiri	gulgiri-gulgiri	'lots of prettily painted men'

Language-theoretically we need to be clear about the exact status of this copying; we focus on the most interesting case of full-stem reduplication. Let us assume that we are dealing with a language where the plural is formed by duplicating the entire stem. There are three positions: (A) one could say that the lexicon is finite, and as a result the learner need only memorise the correct plural form for each word, and thus there is no language-theoretic issue at all; (B) one could say that it is sufficient to have a grammar which gives the copy language over some finite alphabet of phonemes; or finally (C) one could say that the learner must have a true primitive copying operation. There are two related differences between these three approaches—the first is the size of the grammars. The Type A learner will produce a grammar with one rule per lexical item. The Type B learner will have a grammar with one rule per phoneme; but the Type C learner has only one rule in total. The second difference is the rapidity of learning, or equivalently the generalisation ability of the algorithm. For example, if a Type A learner is confronted with an unseen word, then it will fail to generalise correctly, whereas a Type B learner will be able to generalize correctly. Correspondingly if we give a Type B learner a new phoneme, it will fail to generalise—only the Type C learner has learned a real copying rule. Thus deciding whether it is appropriate to require a Type A, B or C learner in each case depends on the ability of the extent to which we require the learner to generalise. If we are interested in modeling language acquisition then the three different learners make different predictions about the behaviour: Type A learners

Fig. 3 Derivation trees for the derivations of the word 'hakhak' (*rights*) with respect to two grammars of the Indonesian plural. On the left we have the MCFG without copying and on the right we have the PMCFG which does use copying



will not be able to produce the correct plural for a novel word. Type B learners will be unable to generalise if it is presented with a word containing a novel phoneme. Only Type C learners can generalise fully.

In the specific case of reduplication these three learners correspond to three different levels of the hierarchy: Type A learners can just use list grammars, Type B can use MCFGs, and Type C requires a PMCFG. We claim that a full explanation of acquisition may require a PMCFG acquisition model that has a true copying operation even if from a purely descriptive language-theoretic approach we only need a much weaker model (see Chandlee and Heinz 2012 for an alternative view). Thus in this model we consider both full stem and partial reduplication as instances of the same copying process.

We present two grammars for this morphological process; a more realistic model would use phonemes or some other level of phonological representation, but for ease of understanding we use letters instead. Indonesian is written with the Roman alphabet. The first grammar uses an MCFG without copying, and the second uses a PMCFG. Figure 3 shows example derivations for these two grammars.

Example 2 The MCFG has two nonterminals, S of dimension 1, and P of dimension 2. We only give the rules for the letters 'a' and 'z' and suppress the other 24 rules of the two types.

$$S(x_1x_2) :- P(x_1, x_2)$$

$$P(x_1a, x_2a) :- P(x_1, x_2)$$

$$P(x_1z, x_2z) :- P(x_1, x_2)$$

$$P(a, a):-$$

$$P(z, z):-$$

Example 3 The second grammar uses a primitive copying operation, and only has nonterminals of dimension 1.

$$S(x_1x_1) :- P(x_1)$$

$$L(a) :-$$

$$L(z) :-$$

$$P(x_1x_2) :- P(x_1), L(x_2)$$

$$P(x_1) :- L(x_1)$$

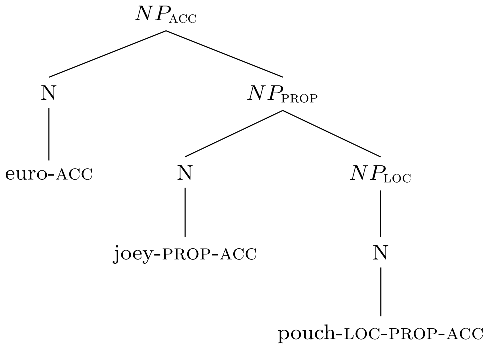


Fig. 4 Example from Martuthunira (Sadler and Nordlinger 2006). The word 'thara' receives 3 distinct suffixes, a locative, proprietive and accusative case markers

2.5 Case-stacking

Case-stacking (or *Suffixaufnahme*) is a comparatively rare phenomenon that occurs in some European and several Australian languages such as Martuthunira and Kayardild. In languages with case-stacking a single word may receive more than one case-marker—a suffix that indicates the grammatical status of a word. In languages without case-stacking, like German, a noun may receive a marking that depends on the case it receives from the verb. This serves to indicate the syntactic relationship between the noun and the verb which governs it. In English for example, which has only a vestigial case system, in the noun phrase "John's dog" the proper noun "John" bears a genitive suffix or clitic which indicates that John is the possessor of the dog. In the noun phrase, "John's father's dog" (the dog of the father of John), the word "John" is embedded deeply but still only receives one suffix.

In other languages however, such as Martuthunira, a noun can receive multiple case markers (Andrews 1996; Sadler and Nordlinger 2006) if it is embedded deeply. Figure 4 gives an example. In Martuthunira, as in Kayardild, it seems that the same noun cannot be marked more than once with the same case marker, and therefore there is a finite bound on the number of case markers that an individual noun can receive. This means that from a weak language-theoretic perspective there is no problem—there are a finite number of possible sequences of case markers and we can capture each of these with a different state or symbol in our grammar. However, the same argument that we used in the previous section applies here: in order to adequately account for learning we will still need to represent this compactly rather than exhaustively listing all of the exponentially many options.

Consider the following example from Kayardild (example 1–15 of Evans 1995)

- maku-wa yalawu-jarra yakuri-na dangka-karra-nguni-na mijil-nguni-na
- woman-NOM catch-PST fish-MABL man-GEN-INSTR-MABL net-INSTR-MABL
- The woman caught fish in the man's net.

Here the word 'dangka' receives three case markers—karra GEN (a genitive marker), nguni INSTR (an instrumental case marker) and MABL a modal ablative which indicates that it is past.

A particular example of Suffixaufnahme has received recent theoretical attention: Old Georgian is a now dead language that has a particularly extreme form of suffix-stacking. The exact status of the data is controversial (Michaelis and Kracht 1997; Bhatt and Joshi 2004); unfortunately Old Georgian is extinct so there is no way of verifying the exact data. Here we assume that the arguments are valid.

For reasons of space we will just describe the string set concerned, rather than attempt to describe the linguistic data: n can be thought of as a noun, g a genitive suffix and v a verb. We have a string set over three letters $\{n, g, v\}$ where the language is:

$\{nv, nngv, nngnggv, nngnggngggv, \ldots\}$ More formally, defining $u_i = ng^i$ this is the language: $L_{\mathrm{OG}} = \{nu_1 \cdots u_k v \mid k \geq 0\}$. This is not semilinear, since the total number of occurrences of g will be a quadratic function of the number of ns in the string. We can describe this string set with the following grammar.

Example 4 This grammar has just two nonterminals S and N of dimension 2.

$$S(x_1x_2v) :- N(x_1, x_2);$$

$$N(x_1x_2n, x_2g) :- N(x_1, x_2);$$

$$N(n, \lambda) :- .$$

We can see that $\mathcal{L}(G, N) = \{(n, \lambda)\} \cup \{nu_1 \cdots u_k n, g^{k+1} \mid k \geq 0\}$.

2.6 Yoruba

Kobele (2006) argues that Yoruba, a Nigerian language, has a certain type of recursive copying in relative clauses. Yoruba can form relative clauses by copying entire verb phrases—the verb phrases can have nouns which can have relative clauses; the end result of this is a language which under intersection with a suitable regular language, and after homomorphism gives the language $\{a^{2^n} \mid n \geq 0\}$. This means that Yoruba cannot be adequately represented by an MCFG. Yoruba has noun phrases of the form (Example 4.48 of Kobele 2006) ‘rira NP ti Ade ra NP’ (the fact that Ade bought NP) where NP is a noun phrase which must be copied; the two occurrences of NP must be identical.[3]

Noun phrases can also be formed into sentences like ‘Ade ra NP’ (Ade bought NP) or ‘NP ko da’ (NP is not good).

Example 5 We can represent this tiny non-semilinear fragment of Yoruba with the grammar:

$$S(x_1x_2) :- NP(x_1), VP(x_2);$$

$$NP(n) :-;$$

$$NP(v_n x_1 t x_2 r x_1) :- NP(x_1), NP(x_2);$$

$$VP(v_f) :-,$$

where we write n for nouns like ‘adiẹ’ (chicken) and proper nouns like ‘Ade’, v_n for nonfinite verbs like ‘rira’ (buying), v_f for finite verb phrases, and t, r for ‘ti’ and ‘ra’.

We can verify that this language is not semilinear by counting the number of occurrences of t in grammatical sentences. Similar phenomena occur widely in other West African languages, such as Wolof, though Yoruba has perhaps the most complex system of this type.

[3] There can apparently be slight differences in the NPs which we neglect here.

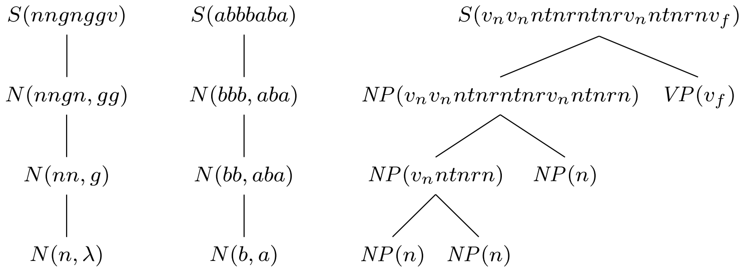


Fig. 5 Derivation trees for these examples; *from left to right*: Old Georgian, Chinese numbers, Yoruba

2.7 Chinese number names

In Mandarin Chinese, a certain subset of number names can be formed from 'wu' (5) and 'zhao' (10^{12}). Here we will write a for 'wu' and b for 'zhao'. The well-formed expressions intersected with a suitable regular language form the language $L_{\mathrm{CN}} = \{ab^{k_1}ab^{k_2}\cdots ab^{k_n} \mid k_1 > k_2 > \cdots > k_n \geq 0\}$. This data is controversial as it is not clear whether the well-formedness of number expressions should form part of the syntax of the language. Here we assume that it does, in which case the language is not semilinear (Radzinski 1991). A grammar for this is:

Example 6

$$\begin{aligned} S(ax_1x_2) &:- N(x_1, x_2);\\ N(bx_1, x_2) &:- N(x_1, x_2);\\ N(bx_1, ax_1x_2) &:- N(x_1, x_2);\\ N(\lambda, \lambda) &:- . \end{aligned}$$

Figure 5 shows an example derivation tree for each of the three examples here.

3 Preliminaries

We now define our formalism more precisely, starting with some basic definitions.

The sets of non-negative and strictly positive integers are denoted by $\mathbb{N}$ and $\mathbb{N}_+$, respectively. A sequence over an alphabet Σ is called a *word*. The empty word is denoted by λ. Σ^* denotes the set of all words and $\Sigma^+ = \Sigma^* - \{\lambda\}$. Any subset of Σ^* is called a *language (over Σ)*. An *m-word* is an m-tuple of words and we denote the set of m-words by $\mathcal{S}_m$ for $m \in \mathbb{N}$. Any m-word is a *multiword*. We define $\mathcal{S}_{\leq m} = \bigcup_{i \leq m} \mathcal{S}_i$ and $\mathcal{S}_* = \bigcup_{i \in \mathbb{N}} \mathcal{S}_i$. We note that the only 0-word is the empty tuple. We usually identify a 1-word (u) with a word u, and a string of length 1 with an element of Σ.

We fix a countably infinite set X of *variables* $x_1, x_2, \ldots$. We use y, y', y_i etc. as meta-variables for variables in X. A *pattern* is a string over $\Sigma \cup X$. For a pattern π, we denote by X_π the set of variables that occur in π. An *m-pattern* π is a pattern such that $|X_\pi| = m$. Hence a 0-pattern is a synonym of a word. A pattern is said to be *n-copying* if each variable occurs at most n times in it. An *m-context* is an m-pattern π such that $X_\pi = \{x_1, \ldots, x_m\}$. We denote the set of m-contexts by $\mathcal{C}_m$ and that of n-copying m-contexts by $\mathcal{C}_{m,n}$. Note that every element of $\mathcal{C}_{m,n}$ contains *exactly* m variables, each of which occurs *at most* n times. In preceding papers on distributional learning algorithms (e.g. Clark and Eyraud 2007), a context is defined to be a pair (l, r) of words. Those correspond to lx_1r in our notation and particularly the empty context (λ, λ) is denoted by x_1 in this paper. One can see an m-context as a function that takes an m-word as an argument and returns a word formed from the components of the m-word and other words that form part of the m-context. For example, from a 2-context $ax_1bx_2x_1$ and a 2-word (c, d), one will get a word $acbdc$. We formally define the composition of an m-context and an m-word through a substitution. A *substitution* θ is a finite partial function from X to Σ^*, which is extended to the homomorphism $\hat{\theta}$ from $(\Sigma \cup X)^*$ to $(\Sigma \cup X)^*$ such that $\hat{\theta}(y) = \theta(y)$ if y is in the domain of θ, and $\hat{\theta}(y) = y$ otherwise for $y \in \Sigma \cup X$. We identify $\hat{\theta}$ and θ if no confusion arises. A substitution θ is often denoted as a suffix operator $[y_1 \mapsto \theta(y_1), \dots, y_k \mapsto \theta(y_k)]$ where $\{y_1, \dots, y_k\}$ is the domain of θ. When the domain is understood, particularly when the domain is $\{x_1, \dots, x_k\}$, it is denoted by $[\theta(y_1), \dots, \theta(y_k)]$ omitting the domain. E.g., for $\theta = \{x_1 \mapsto c, x_2 \mapsto d\}$, we write

$$\theta(ax_1bx_2x_1) = ax_1bx_2x_1[c, d] = acbdc.$$

(The homomorphic extension of) a substitution operation is naturally generalised for sets $C \subseteq \mathcal{C}_k$ and $K \subseteq \mathcal{S}_k$ as $C[K] = \{w[\mathbf{v}] \mid w \in C, \mathbf{v} \in K\}$. For example, for $C = \{ax_1bx_2x_1\}$ and $K = \{(c, d), (e, f)\}$, we have

$$C[K] = \{acbdc, aebfe\}.$$

In what follows, we will consider a fixed language $L \subseteq \Sigma^*$ and we denote the set of m-words that every m-context in a set $C \subseteq \mathcal{C}_m$ accepts with respect to the language L by

$$C^\dagger = \bigl\{\mathbf{v} \in \mathcal{S}_m \mid \pi[\mathbf{v}] \in L \text{ for all } \pi \in C\bigr\}.$$

By definition, $C[K] \subseteq L$ iff $K \subseteq C^\dagger$ for any $K \subseteq \mathcal{S}_m$. Note that if $C = \{x_1\}$ then $C^\dagger = L$. For a set of strings $D \subseteq \Sigma^*$, we let $\mathrm{Sub}_{\leq p}(D)$ and $\mathrm{Con}_{\leq p,r}(D)$ denote the sets consisting of m-words and of r-copying m-contexts with $m \leq p$ that are "extracted" from strings in D, respectively. Those are formally defined as

$$\mathrm{Sub}_{\leq p}(D) = \bigl\{\mathbf{v} \in \mathcal{S}_m \mid \pi[\mathbf{v}] \in D \text{ for some } \pi \in \mathcal{C}_{m,1} \text{ with } m \leq p\bigr\},$$

$$\mathrm{Con}_{\leq p,r}(D) = \bigl\{\pi \in \mathcal{C}_{m,r} \mid \pi[\mathbf{v}] \in D \text{ for some } \mathbf{v} \in \mathcal{S}_m \text{ with } m \leq p\bigr\}.$$

Typically D here will be a finite set of example strings drawn from some target infinite language. When $p = 1$, $\mathrm{Sub}_{\leq p}(D)$ is just the set of substrings of strings in D. Note that $\mathrm{Con}_{\leq 0,r}(L) = L$ and that replacing $\mathcal{C}_{m,1}$ in the definition of $\mathrm{Sub}_{\leq p}(L)$ by $\mathcal{C}_m$ gives an equivalent definition.

3.1 Parallel multiple context-free grammars

A *ranked alphabet* is a pair $\langle N, \dim\rangle$ of an alphabet N and a function $\dim : N \to \mathbb{N}_+$. The number $\dim(A)$ is called the *dimension* of A. We often simply express a ranked alphabet $\langle N, \dim\rangle$ by N if no confusion arises. By N_d we denote the subset of N whose elements have dimension d.

Seki et al. (1991) introduced *parallel multiple context-free grammars* (PMCFGs) as a generalization of context-free grammars. A PMCFG is a tuple $G = \langle \Sigma, N, S, P\rangle$ where Σ is an alphabet whose letters are called *terminals*, N is a ranked alphabet whose elements are called *nonterminals*, $S \in N$ is a special nonterminal of dimension 1 called the *start symbol*, and P is a set of *production rules*.

Production rules in P have the following form:[4]

$$B_0(\pi_1, \dots, \pi_{d_0}) :- B_1(y_{1,1}, \dots, y_{1,d_1}), \dots, B_k(y_{k,1}, \dots, y_{k,d_k})$$

where $B_0, B_1, \dots, B_k \in N$ for some $k \geq 0$, $d_i = \dim(B_i)$ for each $i \in \{0, \dots, k\}$, variables $y_{1,1}, \dots, y_{k,d_k}$ are distinct, and each π_j for $j = 1, \dots, d_0$ is a pattern such that

$$\bigcup_{1 \leq j \leq d_0} X_{\pi_j} = \{y_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq d_i\}.$$

If $k = 0$ then the right-hand side is empty, and the production is of the form $B(\mathbf{v}) :-$ where $\mathbf{v} \in \mathcal{S}_{\dim(B)}$. We say that a rule is *r-copying* if the concatenation of the patterns on its left-hand side is r-copying, that is, no variables occur more than r times on the left-hand side.

Example 7 A tuple $G_1 = \langle \{a\}, N, S, P \rangle$ where $N = N_1 = \{S\}$ and P consists of the two rules

$$S(x_1 x_1) :- S(x_1); \qquad S(a) :-$$

is a PMCFG.

A tuple $G_2 = \langle \{a\}, N, S, P \rangle$ where $N = \{S, A\}$ with $N_1 = \{S\}$ and $N_2 = \{A\}$ and P consists of the three rules

$$S(x_1 x_2 a) :- A(x_1, x_2); \qquad A(x_1 x_2 a, x_2 aa) :- A(x_1, x_2); \qquad A(\lambda, \lambda) :-$$

is a PMCFG.

Another PMCFG is $G_3 = \langle \{a\}, N, S, P \rangle$ where $N = \{S, A, B\}$ with $N_1 = \{S, A\}$ and $N_2 = \{B\}$ and P consists of the 6 rules

$$\begin{aligned} S(x_1 x_{2,1} x_1 x_{2,2}) &:- A(x_1), B(x_1, x_2); \\ A(x_1 x_2) &:- A(x_1), A(x_2); \qquad A(a) :- ; \qquad A(b) :- \\ B(x_1 c, x_2 d) &:- B(x_1, x_2); \qquad B(\lambda, \lambda) :- . \end{aligned}$$

We define the derivation process of a PMCFG G by derivation trees. A node labeled by $A(\mathbf{v})$ is a derivation tree of G if $A(\mathbf{v}) :-$ is a rule of G. We naturally generalise the domain of a substitution θ to tuples $\boldsymbol{\pi}$ of patterns as $\theta(\boldsymbol{\pi}) = (\theta(\pi_1), \dots, \theta(\pi_k))$ for $\boldsymbol{\pi} = (\pi_1, \dots, \pi_k)$. If we have derivation trees t_i whose roots are labeled by $B_i(\mathbf{v}_i)$ for $i = 1, \dots, k$ and G has a rule

$$B_0(\pi_1, \dots, \pi_{d_0}) :- B_1(\mathbf{y_1}), \dots, B_k(\mathbf{y}_k),$$

then the tree whose root is labeled by

$$B_0\big(\theta(\pi_1, \dots, \pi_{d_0})\big)$$

[4] The notation adopted in this paper follows Smullyan's elementary formal systems (1961) rather than Seki et al. (1991). We only consider non-deleting productions in this paper.

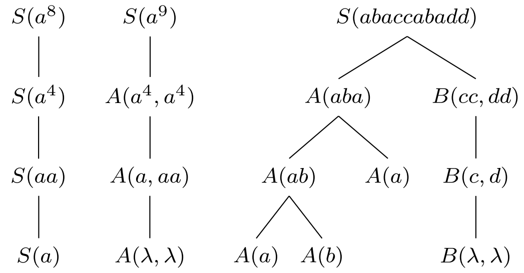


Fig. 6 Example derivation trees of G_1, G_2, G_3 *from left to right*

where $\theta(\mathbf{y_i}) = \mathbf{v}_i$ for all $i = 1, \ldots, k$ with immediate subtrees $t_1, \ldots, t_k$ is also a derivation tree. We will abbreviate this substitution θ as $[\mathbf{v}_1, \ldots, \mathbf{v}_k]$. If there is a derivation tree whose root is labeled by $A(\mathbf{v})$, we write $\vdash_G A(\mathbf{v})$.

We then define the *language of* A by

$$\mathcal{L}(G, A) = \left\{\mathbf{v} \in \mathcal{S}_{\dim(A)} \mid \vdash_G A(\mathbf{v})\right\}.$$

The *language of* G is $\mathcal{L}(G) = \mathcal{L}(G, S)$.

Figure 6 shows some examples of derivation trees of grammars G_1, G_2 and G_3 from Example 7. For the grammar G_1, we have $\vdash_{G_1} S(a^{2^n})$ for all $n \in \mathbb{N}$ and in fact $\mathcal{L}(G_1) = \{a^{2^n} \mid n \in \mathbb{N}\}$. For the grammar G_2, it is easy to see that we have $\vdash_{G_2} A(a^{n^2}, a^{2n})$ for all $n \in \mathbb{N}$ and in fact $\mathcal{L}(G_2) = \{a^{n^2} \mid n \in \mathbb{N}_+\}$. For G_3, it is clear that $\mathcal{L}(G_3, A) = \{a, b\}^+$, $\mathcal{L}(G_3, B) = \{(c^n, d^n) \mid n \in \mathbb{N}\}$ and $\mathcal{L}(G_3, S) = \{wc^n wd^n \mid w \in \{a, b\}^+$ and $n \in \mathbb{N}\}$. See Sect. 2 for some more examples together with some derivation trees.

The following lemma states that the pattern in a rule is easily reconstructed from a word derived using that rule.

Lemma 1 *Suppose that a rule* $B(\boldsymbol{\pi}) :\!- B_1(\mathbf{x}_1), \ldots, B_k(\mathbf{x}_k)$ *is used to obtain* $u \in \mathcal{L}(G)$. *Then* u *can be represented by* $u = \pi_0[\theta(\boldsymbol{\pi})]$ *for some* $\pi_0 \in \mathcal{C}_{\dim(B),1}$ *and some substitution* θ *whose domain consists of the variables of* $\mathbf{x}_1, \ldots, \mathbf{x}_k$ *exactly.*

Proof (Sketch) Suppose that we have $\vdash_G B(\theta(\boldsymbol{\pi}))$ following $\vdash_G B_i(\mathbf{v}_i)$ for $i = 1, \ldots, k$ where $\theta = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$. The derivation process solely consists of concatenation operations. Strings obtained during the derivation process are never deleted or split. Therefore, it is easily seen that if some derivation tree for $\vdash_G A(\mathbf{u})$ contains the derivation tree corresponding to $\vdash_G B(\theta(\boldsymbol{\pi}))$, then $\mathbf{u}$ can be represented as $\mathbf{u} = \boldsymbol{\pi}'[\theta(\boldsymbol{\pi})]$ for some tuple of patterns $\boldsymbol{\pi}'$. Particularly for $u \in \mathcal{L}(G)$, we have $u = \pi'_0[\theta(\boldsymbol{\pi})]$ for some $\pi'_0 \in \mathcal{C}_{d,*}$ with $d = \dim(B)$. By replacing all but one occurrences of a variable x by $\theta(x)$ in π'_0, one obtains $\pi_0 \in \mathcal{C}_{d,1}$ with the desired property. □

We denote by $\mathbb{G}(p, q, r)$ the class of PMCFGs such that the dimension of every nonterminal is at most p and every production rule has at most q nonterminals on the right-hand side and is r-copying. For the grammars in Example 7, we have $G_1 \in \mathbb{G}(1, 1, 2)$, $G_2 \in \mathbb{G}(2, 1, 2)$ and $G_3 \in \mathbb{G}(2, 2, 2)$.

Theorem 1 (Seki et al. 1991) *The uniform membership problem for* $\mathbb{G}(p, q, r)$ *is solvable in polynomial time whose degree is linear in* pq.

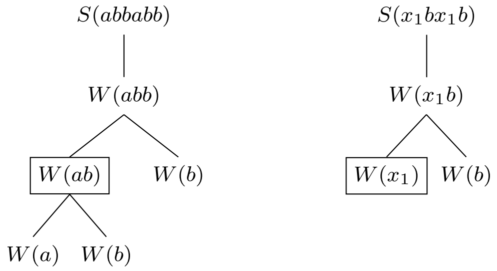


Fig. 7 Example of a context as a function from strings to strings

4 Intuition

We will now give an informal introduction to the extension of distributional learning to these formalisms; the basic idea is quite natural but may be obscured by the unavoidable complexity of the notation.

In distributional learning we typically consider a context (l, r) which we can wrap around a substring u to give a complete string lur. Consider this context rather as a function f from a substring to a full sentence. $u \mapsto lur$, which in our notation is represented by what we call a 1-copying 1-context lx_1r, an element of $\mathcal{C}_{1,1}$.

In the derivation of a string with respect to a CFG, these functions correspond to the operation that takes the yield of a nonterminal and integrates into the rest of the sentence: given a derivation like $S \overset{*}{\Rightarrow} lNr \overset{*}{\Rightarrow} lur$, we can consider the derivation $S \overset{*}{\Rightarrow} lNr$ to be applying a function $f \in \mathcal{C}_{1,1}$ to the yield of N.

In a parallel CFG, we might again have a nonterminal N that derives a string u. However, the part of the derivation that produces the whole sentence from u may include rules that copy u.

Example 8 Consider for example the language $\{ww \mid w \in \{a, b\}^+\}$. This could be defined as a parallel CFG with two nonterminals S and W, together with productions:

$$S(x_1x_1) :\!-\, W(x_1); \qquad W(a) :\!-; \qquad W(b) :\!-; \qquad W(x_1x_2) :\!-\, W(x_1), W(x_2).$$

Figure 7 shows a derivation tree of $abbabb$; we can pick one node in the tree (marked with a box). If we consider the "context" of the node $W(ab)$, then this is not a simple context, but rather the function $x_1 \mapsto x_1bx_1b$. We can see this by replacing this node in the tree with a variable x_1, as on the right hand side of Fig. 7.

Therefore, with this richer class of grammars we need to consider a larger class of functions that correspond to r-copying contexts, and when we consider tuples of strings in the full PMCFG formalism, to r-copying d-contexts: the class $\mathcal{C}_{d,r}$. Given such a set of functions we can consider the 'distribution' in this extended sense of a substring in a language to be the set of functions that when applied to that substring give an element of the language.

In this paper we use a dual approach—the nonterminals are defined by small finite sets of patterns/functions, and incorrect rules will be eliminated by strings or tuples of strings.

In Example 8, we can see how the nonterminals that we need can be picked out. The symbol S will correspond as usual to the single simple pattern x_1—the identity function which corresponds to the empty context (λ, λ) in distributional learning of CFGs (e.g. Clark and Eyraud 2007). The symbol W corresponds to the 2-copying context x_1x_1. It is easy to see that the set of strings generated by W is exactly the same as the set of strings which can occur in the context x_1x_1. In the notation we defined earlier we have a singleton set $C_W = \{x_1x_1\}$

such that $\mathcal{L}(G, W) = \{v \in \Sigma^* \mid C_W[v] \subseteq \mathcal{L}(G)\}$. Note that for any grammar, the start symbol S will be characterised by the single context x_1. We shall show that languages that have this nice property—that the languages defined by each nonterminal can be picked out by a small set of contexts—are learnable by a straightforward algorithm, very similar to ones that have been used before for distributional learning of CFGs.

4.1 Algorithm description

We will now give an informal description of the algorithm. The algorithm receives some examples of strings from a target language: we use D to denote the finite set of examples. In addition we assume that the learner can ask membership queries (MQs). This means that the learner, in addition to passively observing the examples it is given can construct a string and query whether it is in the language being learned or not. We discuss below in Sect. 5.2 the implications of this choice of learning model.

We are interested in modelling the relationship between sets of contexts and sets of subyields. The algorithm therefore considers every possible decomposition of the positive examples into contexts and subyields, which can be done in polynomial time: this gives us a set of subyields, which are tuples of strings, K, and a set of contexts, F. We then construct a grammar which has a nonterminal for every small subset of F; every subset of size at most s. Each such subset C defines a set of strings or tuples of strings in K: namely those which when inserted into the context give a string in the language: we will denote this set $C^{(K)}$, which is finite and can be computed using a polynomial number of membership queries. We then construct all possible rules, of bounded complexity, that can be made using these nonterminals. Suppose we have a possible rule like

$$[\![C_0]\!](\boldsymbol{\pi}) :- [\![C_1]\!](\mathbf{x}_1), \ldots, [\![C_k]\!](\mathbf{x}_k). \tag{1}$$

This rule is correct if when we apply the rule $\boldsymbol{\pi}$ to each of the strings in $C_i^{(K)}$ the result is in fact in $C_0^\dagger$. So we take all of the strings or tuples of strings that correspond to the nonterminals on the right hand side of the rule, combine them using the recipe in the pattern $\boldsymbol{\pi}$ and then test, using membership queries that the resulting tuple can occur in each of the contexts in C_0. The final grammar that we construct consists of all rules that pass this test.

5 Learning target and algorithm

Definition 2 We say that a PMCFG G has *the* (r, s)*-finite context property* ((r, s)-FCP) if each nonterminal $A \in N_d$ admits a nonempty set $C_A \subseteq \mathcal{C}_{d,r}$ of r-copying d-contexts such that $|C_A| \leq s$ and

$$\mathcal{L}(G, A) = C_A^\dagger = \big\{\mathbf{v} \in \mathcal{S}_d \mid C_A[\mathbf{v}] \subseteq \mathcal{L}(G)\big\}.$$

Such a set C_A is called a *characterising set* of A.

By $\mathbb{G}(p, q, r, s)$ we denote the subclass of $\mathbb{G}(p, q, r)$ where grammars have the (r, s)-FCP. The class of languages generated by grammars in $\mathbb{G}(p, q, r, s)$ is denoted by $\mathbb{L}(p, q, r, s)$.

Clearly the above definition is a generalisation of the s-FCP (Clark 2010). All regular languages are in $\mathbb{L}(1, 1, 1, 1)$ and the Dyck language is in $\mathbb{L}(1, 2, 1, 1)$.

Recall the grammar G_1 in Example 7, which has only one nonterminal symbol. We have $\mathcal{L}(G_1) = \{a^{2^n} \mid n \geq 0\} \in \mathbb{L}(1, 1, 2, 1)$ since the 1-context x_1 always characterises the start symbol S.

There are context-free languages which are not in $\mathbb{L}(p, q, r, s)$ for any values of p, q, r, s such as for example, the language $\{a^n b^m \mid n, m > 0, m \neq n\}$. This is because the required nonterminals cannot be picked out by any finite set of contexts.

5.1 Linguistic examples

We now consider the examples in Sect. 2 with respect to this representational assumption.

In the case of Swiss German, we do not need any copying operations, and we need to be able to define the two nonterminals in the grammar in Example 1: S which is of dimension 1, and D which is of dimension 2. S is trivially definable using the single context x_1. D could be defined using the single context $\pi = CN_a x_1 N_d V_a x_2 V_d$, which is a 1-copying 2-context. We can easily verify that $\{\pi\}^\dagger$ which is the set of 2-words that can occur in this context is the set $\{(\bar{w}, w) \mid w \in \{V_a, V_d\}^*\}$. This differs slightly from the set of strings generated by the nonterminal D in the specific grammar we defined earlier, in that it includes the empty bi-word (λ, λ). However we can modify that grammar slightly to get the following slightly larger grammar:

Example 9 This grammar generates the same language as Example 1, but is slightly larger.

$$S(CN_a x_1 V_a x_2) :- D(x_1, x_2)$$

$$S(CN_d x_1 V_d x_2) :- D(x_1, x_2)$$

$$D(N_a x_1, V_a x_2) :- D(x_1, x_2)$$

$$D(N_d x_1, V_d x_2) :- D(x_1, x_2)$$

$$D(\lambda, \lambda) :-$$

Note that now $\mathcal{L}(G, D) = \{CN_a x_1 N_d V_a x_2 V_d\}^\dagger$ and as a result the language is in $\mathbb{L}(2, 1, 1, 1)$.

In the Old Georgian case, Example 4, we can use the single context $\pi = x_1 x_2 n x_2 g v$ to pick out the multiwords generated by the nonterminal N of dimension 2. We recall that $\mathcal{L}(G, N) = \{(n, \lambda)\} \cup \{nu_1 \cdots u_k n, g^{k+1} \mid k \geq 0\}$. Note that π is a 2-copying 2-context—the variable x_1 occurs twice. Suppose that $(w_1, w_2) \in \{\pi\}^\dagger$. This means that $w_1 w_2 n w_2 g v \in L$. By considering the number of occurrences of the symbols n and g in w_1 and w_2 we can verify that this can only happen if w_2 is a string of zero or more occurrences of g, and therefore if $(w_1, w_2) \in \mathcal{L}(G, N)$. Moreover we can see that $\pi[\mathcal{L}(G, N)] \subseteq \mathcal{L}(G)$. Therefore we have that $\{\pi\}^\dagger = \mathcal{L}(G, N)$ as desired.

In the case of the duplication in Indonesian, as shown in Examples 2 and 3, we can in both cases find appropriate sets of contexts. Taking the MCFG grammar first, the single context $x_1 x_2$ does not suffice to pick out the nonterminal of dimension 2, P, since $\{x_1 x_2\}$ includes strings like (ha, khak) as well as the desired 2-words. Indeed no single 1-copying 2-context can pick out exactly the right set of 2-words. Suppose we have a 1-copying 2-context which will be of the form $ux_1 vx_2 w$ for some strings u, v, w; then $\{ux_1 vx_2 w\}^\dagger$ will include for any string y both the 2-words $(y, wuyv)$ and $(vywu, y)$. However, we can pick out the correct set of 2-words using two distinct contexts. If we define $C = \{ax_1 ax_2, bx_1 bx_2\}$, then $C^\dagger = \{(w, w) \mid w \in \Sigma^+\}$. Therefore this grammar is in the class $\mathbb{G}(2, 1, 1, 2)$ and the language is in $\mathbb{L}(2, 1, 1, 2)$.

If we consider now the PMCFG grammar for the same language, as shown in Example 3, we can define the nonterminal of dimension 2 P using the single 2-copying 1-context x_1x_1. Therefore this grammar is $\mathbb{G}(1,2,2,1)$; indeed although we have written it using a grammar with a rule with two nonterminals on the right hand side, we could also have used a simpler grammar using only rules of rank 1, which would give a grammar in $\mathbb{G}(1,1,2,1)$.

In the Yoruba case, Example 5, the NP class is picked out by the context $x_1 v_f$ and the VP class (just the symbol v_f in this trivial example) by the context nx_1. Therefore the grammar belongs to $\mathbb{G}(2,2,2,1)$.

Finally in the Chinese number example, Example 6, we can prove that the context $abx_1ax_1x_2$ characterises the nonterminal N. It is easy to see that

$$\mathcal{L}(G,N)=\bigl\{\bigl(b^{k_0},ab^{k_1}\cdots ab^{k_n}\bigr)\mid n\geq 0, k_0>k_1>\cdots>k_n\geq 0\bigr\}$$

and

$$\pi\bigl[\mathcal{L}(G,N)\bigr]=\bigl\{abb^{k_0}ab^{k_0}ab^{k_1}\cdots ab^{k_n}\mid n\geq 0, k_0>k_1>\cdots>k_n\geq 0\bigr\}\subseteq L_{\mathrm{CN}}.$$

Thus it is enough to show that $\pi[u,v]=abuauv\in L_{\mathrm{CN}}$ implies $(u,v)\in\mathcal{L}(G,N)$. Let n be the number of occurrences of a in u. There are $k_0,\dots,k_n\in\mathbb{N}$ such that $u=b^{k_0}ab^{k_1}a\cdots ab^{k_n}$. Then $\pi[u,v]$ should be represented as $ab^{k_0+1}ab^{k_1}a\cdots ab^{k_n}ab^{k_0}aw$ for some w. The fact $k_0+1>k_1>\cdots>k_n>k_0$ implies $n=0$. That is, $u=b^{k_0}$ for some $k_0\in\mathbb{N}$. We now have $\pi[u,v]=ab^{k_0+1}ab^{k_0}v\in L_{\mathrm{CN}}$. The leftmost symbol of v cannot be b. That is, $v=ab^{k_1}ab^{k_2}\cdots ab^{k_n}$ for some $n\geq 1$ and $k_1,\dots,k_n\in\mathbb{N}$ such that $k_0>k_1>\cdots>k_n\geq 0$. Therefore $(u,v)\in\mathcal{L}(G,N)$. The grammar belongs to $\mathbb{G}(2,1,2,1)$.

5.2 Learning model

The learner receives a presentation of positive data in the *identification in the limit* paradigm. We assume that our learner has in addition access to an oracle which answers *membership queries* (MQs), which says whether an arbitrary string u belongs to the learning target L_*. See for example Yoshinaka (2010) for details.

A *positive presentation* of a language L_* over Σ is an infinite sequence of words $w_1,w_2,\ldots\in\Sigma^*$ such that $L_*=\{w_i\mid i\geq 1\}$. A learner is given a positive presentation of the language $L_*=\mathcal{L}(G_*)$ of the target grammar G_* and each time a new example w_i is given, it outputs a grammar G_i computed from $w_1,\dots,w_i$ with the aid of a *membership oracle*. One may query the oracle whether an arbitrary string w is in L_*, and the oracle answers in constant time. We say that a learning algorithm *identifies G_* in the limit from positive data and membership queries* if for any positive presentation $w_1,w_2,\ldots$ of $\mathcal{L}(G_*)$, there is an integer n such that $G_n=G_m$ for all $m\geq n$ and $\mathcal{L}(G_n)=\mathcal{L}(G_*)$. Trivially every grammar admits a successful learning algorithm. An algorithm should learn a rich class of grammars in a uniform way. We say that a learning algorithm *identifies a class $\mathbb{G}$ of grammars in the limit from positive data and membership queries* if and only if it identifies all $G\in\mathbb{G}$.

We remark that as we have membership queries, learning algorithms based on exhaustive enumeration will work, hence a learner should have further properties in terms of efficiency. Accordingly we require the learner to operate in polynomial time: we assume that the membership queries can be answered in constant time (or equivalently time polynomial in the length of the example). Thus at each step t, the learner can only use time that is bounded by a polynomial of the total size of the data seen so far, $|w_1|+\cdots+|w_t|$.

We note that our interest here is in showing that a particular algorithm is correct and efficient, and not in showing that a particular class of languages is learnable with respect to a particular learning model. However, we note that this learning model is not restrictive in the sense that it is possible to find trivial enumerative algorithms that can also learn the classes $\mathbb{L}(p,q,r,s)$ we discuss in this paper using a delaying trick. The algorithm we present here does not use such tricks.

5.3 Construction of hypothesis

Hereafter we arbitrarily fix rather small natural numbers $p,q,r,s \geq 1$ and a target language $L_* \in \mathbb{L}(p,q,r,s)$ to be learnt.

Our learner is a straightforward generalisation of the one for CFGs with the s-FCP given in Yoshinaka (2011b). The learner constructs its conjecture $\hat{G} = \mathcal{G}(K,F)$ from finite sets of multiwords $K \subseteq \mathrm{Sub}_{\leq p}(D)$ and r-copying contexts $F \subseteq \mathrm{Con}_{\leq p,r}(D)$ where D is a set of positive examples. We let $K_i = K \cap \mathcal{S}_i$ for $i = 1,\dots,p$ and $F_j = F \cap \mathcal{C}_{j,r}$ for $j = 0,\dots,p$. We assume that $x_1 \in F_1$. The nonterminal set $\hat{N} = \bigcup_{1\leq i\leq p} \hat{N}_i$ of $\hat{G}$ is given by

$$\hat{N}_i = \big\{ [\![C]\!] \mid C \subseteq F_i \text{ with } 1 \leq |C| \leq s \big\},$$

where $[\![C]\!]$ simply means a symbol indexed with C. The start symbol is $[\![\{x_1\}]\!]$.

We would like each nonterminal $[\![C]\!] \in \hat{N}$ to generate $C^\dagger = \{\mathbf{v} \mid C[\mathbf{v}] \in L_*\}$. If a grammar G_* generating the target language L_* has a nonterminal A which is characterised by a context set C_A and $C_A \subseteq K$, then the nonterminal $[\![C_A]\!]$ of our grammar should be used to simulate A. Recall that the start symbol of any grammar is characterised by $\{x_1\}$, which is the reason why $[\![\{x_1\}]\!]$ is the start symbol of our grammar.

If we have a rule of the form

$$[\![C_0]\!](\boldsymbol{\pi}) \,{:}{-}\, [\![C_1]\!](\mathbf{x}_1),\dots,[\![C_k]\!](\mathbf{x}_k), \tag{2}$$

according to the semantics of the nonterminals, it should hold that $C_0^\dagger \supseteq \boldsymbol{\pi}[C_1^\dagger,\dots,C_k^\dagger]$ by the nature of the derivation. Equivalently, we should have

$$C_0\big[\boldsymbol{\pi}\big[C_1^\dagger,\dots,C_k^\dagger\big]\big] \subseteq L_*, \tag{3}$$

where $\boldsymbol{\pi}[C_1^\dagger,\dots,C_k^\dagger] = \{\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k] \mid \mathbf{v}_i \in C_i^\dagger \text{ for } i = 1,\dots,k\}$. Yet in general we cannot decide the inclusion relation (3) since we have no means to compute sets $C_i^\dagger$. We substitute $C_i^\dagger \cap K$ for $C_i^\dagger$, which is computable by the aid of MQs; for each $\mathbf{v} \in K$, we have $\mathbf{v} \in C_i^\dagger \cap K$ if and only if for all $w \in C_i$, $w[\mathbf{v}] \in L_*$. For $C \subseteq \mathcal{C}_{j,r}$, we let denote $C^\dagger \cap K_j$ by $C^{(K)}$. Since the sets $C_1^{(K)},\dots,C_k^{(K)}$ are finite, the relation

$$C_0\big[\boldsymbol{\pi}\big[C_1^{(K)},\dots,C_k^{(K)}\big]\big] \subseteq L_* \tag{4}$$

can be decided by finitely many MQs.

Consequently our grammar $\mathcal{G}(K,F)$ has rules of the form (2) if and only if the following conditions hold, where d_i are such that $[\![C_i]\!] \in N_{d_i}$ for $i = 0,\dots,k$ and $d = \sum_{1\leq i\leq k} d_i$:

- $0 \leq k \leq q$,
- $\boldsymbol{\pi}$ is a d_0-tuple of patterns whose concatenation is an r-copying d-context,
- $|\mathbf{x}_i| = d_i$ for each $i = 1,\dots,k$ and the variables from $\mathbf{x}_1,\dots,\mathbf{x}_k$ constitute $X_{\boldsymbol{\pi}}$,
- there are $\mathbf{v}_i \in \mathcal{S}_{d_i}$ for $i = 1,\dots,k$ and $\pi_0 \in \mathcal{C}_{d_0,1}$ such that $\pi_0[\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k]] \in F_0$,
- the inclusion (4) holds.

We call a rule satisfying the above conditions but the last a *potential rule*. Note that the condition to be a potential rule is subject to the target class $\mathbb{L}(p,q,r,s)$ but not to a target language; we use MQs only to decide the condition (4).

Example 10 Let $(p,q,r,s)=(1,1,2,1)$. If $a, aa \in F_0$ and $x_1 \in F_1$, we have the following potential rules among others:

$$\begin{aligned} \llbracket\{x_1\}\rrbracket(x_1x_1) &:- \llbracket\{x_1\}\rrbracket(x_1), \\ \llbracket\{x_1\}\rrbracket(a) &:-, \\ \llbracket\{x_1\}\rrbracket(x_1a) &:- \llbracket\{x_1\}\rrbracket(x_1). \end{aligned}$$

Suppose that the learning target is generated by G_1 of Example 7:

$$\mathcal{L}(G_1)=\{a^{2^n} \mid n \in \mathbb{N}\}.$$

The first two rules always satisfy (4) whatever K is. On the other hand, if $aa \in K$, the last rule does not satisfy (4) since $x_1[aa] \in \mathcal{L}(G_1)$ and $x_1a[aa] \notin \mathcal{L}(G_1)$.

Example 11 Let $(p,q,r,s)=(2,1,2,1)$. Consider the fragment of the Old Georgian (Example 4)

$$L_{\mathrm{OG}}=\{u_0\cdots u_kv \mid k \geq 0, u_i = ng^i\}.$$

If $nngv \in F_0$ and $x_1x_1x_2v, x_1x_2nx_2gv \in F_2$, we have the following potential rules:

$$\begin{aligned} \llbracket\{x_1x_1x_2v\}\rrbracket(x_1, x_2nx_1g) &:- \llbracket\{x_1x_1x_2v\}\rrbracket(x_1,x_2), \\ \llbracket\{x_1x_1x_2v\}\rrbracket(x_1, x_2g) &:- \llbracket\{x_1x_2nx_2gv\}\rrbracket(x_1,x_2), \\ \llbracket\{x_1x_2nx_2gv\}\rrbracket(x_1x_1n, x_2g) &:- \llbracket\{x_1x_2nx_2gv\}\rrbracket(x_1,x_2). \end{aligned}$$

If $(n,g) \in K$, the first rule does not satisfy (4) since $x_1x_1x_2v[n,g] \in L_{\mathrm{OG}}$ and $x_1x_1x_2v[n,gnng] \notin L_{\mathrm{OG}}$.

If $(nn,g) \in K$, the second rule does not satisfy (4) since $x_1x_2nx_2gv[nn,g] \in L_{\mathrm{OG}}$ and $x_1x_1x_2v[nn,g] \notin L_{\mathrm{OG}}$.

The third rule always satisfies (4) whatever K is.

Lemma 2 *One can construct $\mathcal{G}(K,F)$ in polynomial time in $\|D\|$.*

Proof By $F \subseteq \mathrm{Con}_{\leq p,r}(D)$ and $K \subseteq \mathrm{Sub}_{\leq p}(D)$, $\|F\|$ and $\|K\|$ are bounded by a polynomial in $\|D\|$ with a degree linear in pr and p, respectively. For each $\llbracket C\rrbracket \in \hat{N}$, the fact $|C| \leq s$ implies $|\hat{N}| \leq |F|^s$.

We first estimate the number of potential rules of the form (2). By $k \leq q$, at most $(|\hat{N}|+1)^{q+1}$ combinations of nonterminals are possible. It remains to count the number of possible $\boldsymbol{\pi}$. There must exist $u \in F_0 \subseteq D$, $\mathbf{v}_i \in \mathrm{Sub}_{\leq p}(u)$ for $i=1,\dots,k$ and $\pi_0 \in \mathrm{Con}_{\leq p,1}(u)$ such that $u=\pi_0[\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k]]$. Determining $\boldsymbol{\pi}$ can be seen as determining the left and right positions in u to which each occurrence of variables in π_0 and $\boldsymbol{\pi}$ corresponds. Note that π_0 and $\boldsymbol{\pi}$ contain at most p and pqr occurrences of variables, respectively. Thus we extract at most $|u|^{2(p+pqr)}$ variants of $\boldsymbol{\pi}$ from a string $u \in F_0$. Therefore, we have at most $(|\hat{N}|+1)^{q+1}|F_0|\ell^{2(p+pqr)}$ production rules in $\mathcal{G}(K,F)$ where ℓ is the length of a longest word in F_0.

The algorithm verifies whether potential rules satisfy the last condition (4). To compute $C_i^{(K)}$, we call the membership oracle on at most $|C_i||K|$ words. To see whether (4) holds, it is enough to check the membership on at most $|C_0|\prod_{1\le i\le k}|C_k^{(K)}| \le s|K|^q$ words.

All in all, one can compute $\mathcal{G}(K,F)$ in polynomial time in $\|D\|$, where the degree of the polynomial linearly depends on $pqrs$. □

Just like the algorithms based on syntactic concept lattices (e.g. Clark 2010), we establish the following monotonicity lemma: expansion of F expands the hypothesised language while expansion of K shrinks the hypothesised language.

Lemma 3 (Monotonicity) *Let $\hat{G}=\mathcal{G}(K,F)$ and $\hat{G}'=\mathcal{G}(K',F')$.*

(1) *If $K\subseteq K'$ and $F=F'$, then $\mathcal{L}(\hat{G})\supseteq\mathcal{L}(\hat{G}')$.*
(2) *If $K=K'$ and $F\subseteq F'$, then $\mathcal{L}(\hat{G})\subseteq\mathcal{L}(\hat{G}')$.*

Proof (1) Every rule of $\hat{G}'$ is also a rule of $\hat{G}$. (2) Every rule of $\hat{G}$ is also a rule of $\hat{G}'$. □

We say that a rule of the form (2) is *correct* if it satisfies (3). In other words, for any $\mathbf{v}_1,\dots,\mathbf{v}_k\in\mathcal{S}_*$,

$$C_i[\mathbf{v}_i]\subseteq L_* \quad \text{for all } i\in\{1,\dots,k\} \quad\Longrightarrow\quad C_0\big[\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k]\big]\subseteq L_*.$$

If a rule is not correct, it is *incorrect*.

Lemma 4 *Every context set F admits a multiword set K of a polynomial cardinality in $\|F\|$ such that $\hat{G}=\mathcal{G}(K,F)$ has no incorrect rules.*

Proof Suppose that a rule $[\![C_0]\!](\boldsymbol{\pi}) :- [\![C_1]\!](\mathbf{x}_1),\dots,[\![C_k]\!](\mathbf{x}_k)$ is incorrect. There exist multiwords $\mathbf{v}_i\in C_i^{\dagger}$ for $i=1,\dots,k$ such that $C_0[\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k]]\nsubseteq L_*$. If $\mathbf{v}_1,\dots,\mathbf{v}_k\in K$, such a rule is suppressed. That is, at most q multiwords are enough to get rid of an incorrect rule. Recall that the number of potential rules is polynomially bounded by $|F|\ell$ with $\ell=\max\{|u|\mid u\in F_0\}$ by the proof of Lemma 2. This proves the lemma. □

We say that K is *fiducial on F* (*with respect to L_**) if $\mathcal{G}(K,F)$ has no incorrect rules. If K is fiducial on F, then so is every superset of K by definition.

Lemma 5 *If $\hat{G}=\mathcal{G}(K,F)$ has no incorrect rules, then $\mathcal{L}(\hat{G})\subseteq L_*$.*

Proof We show by induction that $\vdash_{\hat{G}} [\![C]\!](\mathbf{v})$ implies $C[\mathbf{v}]\subseteq L_*$. This implies particularly for $\vdash_{\hat{G}} [\![\{x_1\}]\!](v)$, where $[\![\{x_1\}]\!]$ is the start symbol of $\hat{G}$, we have $v\in L_*$.

Suppose that we have $\vdash_{\hat{G}} [\![C_0]\!]([\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k]])$ by the rule $[\![C_0]\!](\boldsymbol{\pi}) :- [\![C_1]\!](\mathbf{x}_1),\dots,[\![C_k]\!](\mathbf{x}_k)$ and $\vdash_{\hat{G}} [\![C_i]\!](\mathbf{v}_i)$ for $i=1,\dots,k$. By the induction hypothesis we have $C_i[\mathbf{v}_i]\subseteq L_*$ for $i=1,\dots,k$. (When $k=0$, it is the base case.) Since the rule is correct, we have $C_0[\boldsymbol{\pi}[\mathbf{v}_1,\dots,\mathbf{v}_k]]\subseteq L_*$. □

Let $G_*=\langle\Sigma,N_*,P_*,S_*\rangle\in\mathbb{G}(p,q,r,s)$ generate L_*. We say that F is *adequate* (*with respect to G_**) if F includes a characterising set C_A for every nonterminal $A\in N_*$ and F_0

contains a string $v_\rho \in \mathcal{L}(G_*)$ derived by using ρ for every rule $\rho \in P_*$. If F is adequate, then so is every superset of F.

Lemma 6 *If F is adequate, then $L_* \subseteq \mathcal{L}(\mathcal{G}(K, F))$ for any K.*

Proof Let $\hat{G} = \mathcal{G}(K, F)$. For a rule $A_0(\boldsymbol{\pi}) :- A_1(\mathbf{x}_1), \dots, A_k(\mathbf{x}_k)$ of G_*, let $C_i \subseteq F_{\dim(A_i)}$ be a characterising set of A_i for $i = 0, \dots, k$. By the assumption, there are $\mathbf{v}_i \in \mathcal{L}(G_*, A_i)$ for $i = 1, \dots, k$ and a $\dim(A_0)$-pattern π_0 such that $\pi_0[\boldsymbol{\pi}[\mathbf{v}_1, \dots, \mathbf{v}_k]] \in \mathcal{L}(G) \cap F_0$ by Lemma 1. Thus $[\![C_0]\!](\boldsymbol{\pi}) :- [\![C_1]\!](\mathbf{x}_1), \dots, [\![C_k]\!](\mathbf{x}_k)$ is a potential rule. For any $\mathbf{u}_i \in C_i^{(K)} = \mathcal{L}(G_*, A_i) \cap K$, we have $\boldsymbol{\pi}[\mathbf{u}_1, \dots, \mathbf{u}_k] \in \mathcal{L}(G_*, A_0) = C_0^\dagger$. That is, $C_0[\boldsymbol{\pi}[\mathbf{u}_1, \dots, \mathbf{u}_k]] \subseteq L_*$. Hence the rule is present in $\hat{G}$ whatever K is. □

Lemmas 2, 4–6 mean that one can construct a right grammar from a small amount of data efficiently.

5.4 Learning algorithm

Algorithm 1, $\mathcal{A}(p, q, r, s)$, wants an adequate set F and a fiducial set K on the set F, from which it can compute a grammar generating the target language. Expanding K infinitely causes no problem because it is used only for removing incorrect rules; after all incorrect rules have been removed, any further increase in K will have no effect. On the other hand, expansion of F leads to an increase in the number of nonterminal symbols and production rules, which should not happen infinitely many times. Thus $\mathcal{A}(p, q, r, s)$ expands F only when it knows that F does not include all characterising sets C_A for nonterminals A of G_*, i.e., when it observes that it is undergenerating.

Lemma 7 *If the current conjecture $\hat{G}$ is such that $L_* \not\subseteq \mathcal{L}(\hat{G})$, then the learner will discard $\hat{G}$ at some point.*

Proof At some point, some element $u \in L_* - \mathcal{L}(\hat{G})$ is given to the learner. The rule $[\![\{x_1\}]\!](u):-$ is correct but not present in $\hat{G}$. Once the learner gets u, we obtain this rule by $u \in F_0$. □

Lemma 8 *If $\mathcal{L}(\hat{G}) \not\subseteq L_*$, then the learner will discard $\hat{G}$ at some point.*

Proof By Lemma 5, the fact $\mathcal{L}(\hat{G}) \not\subseteq L_*$ implies that $\hat{G}$ has an incorrect rule $[\![C_0]\!](\boldsymbol{\pi}) :- [\![C_1]\!](\mathbf{x}_1), \dots, [\![C_k]\!](\mathbf{x}_k)$, where $C_0[\boldsymbol{\pi}[C_1^\dagger, \dots, C_k^\dagger]] \not\subseteq L_*$. That is, there are $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathcal{S}_{\le p}$

Algorithm 1 $\mathcal{A}(p, q, r, s)$

```
Data: A sequence of strings $w_1, w_2, \dots \in L_*$; membership oracle $\mathcal{O}$
Result: A sequence of PMCFGs $G_1, G_2, \dots \in \mathbb{G}(p, q, r)$
let $D := K := F := \varnothing$; $\hat{G} := \mathcal{G}(K, F)$;
for $n = 1, 2, \dots$ do
    let $D := D \cup \{w_n\}$; $K := \mathrm{Sub}_{\le p}(D)$;
    if $D \not\subseteq \mathcal{L}(\hat{G})$ then
        let $F := \mathrm{Con}_{\le p,r}(D)$;
    end if
    output $\hat{G} = \mathcal{G}(K, F)$ as $G_n$;
end for
```

such that $C_i[\mathbf{v}_i] \subseteq L_*$ for all $i = 1, \dots, k$ and $C_0[\boldsymbol{\pi}[\mathbf{v}_1, \dots, \mathbf{v}_k]] \nsubseteq L_*$. By $\mathbf{v}_i \in \mathrm{Sub}_{\leq p}(L_*)$, at some point the learner will have $D \subseteq L_*$ such that $\mathbf{v}_i \in \mathrm{Sub}_{\leq p}(D)$ for all $i = 1, \dots, k$. For $K = \mathrm{Sub}_{\leq p}(D)$ we have $\mathbf{v}_i \in C_i^{(K)}$ and $C_0[\boldsymbol{\pi}[C_1^{(K)}, \dots, C_k^{(K)}]] \nsubseteq L_*$. The incorrect rule must be removed. □

Theorem 3 *The learner $\mathcal{A}(p, q, r, s)$ identifies $\mathbb{G}(p, q, r, s)$ in the limit.*

Proof Let $L_* \in \mathbb{L}(p, q, r, s)$ be the learning target. By Lemmas 7 and 8, the learner never converges to a wrong hypothesis. It is impossible that the set F is changed infinitely many times because F is monotonically expanded and at some point F will become adequate with respect to a target grammar G_* generating L_*, in which case the learner never updates F any more by Lemma 6. Then sometime K will be fiducial on F by Lemmas 8 and 4, where $\hat{G}$ has no incorrect rules. Thereafter no rules will be added to or removed from $\hat{G}$ any more. □

6 Discussion and conclusion

6.1 Related work

There is very little work that this paper can be directly compared to; it is of course, as noted earlier, an extension of recent work on distributional learning of context-free and multiple context-free grammars; in particular it subsumes the dual approaches to learning context-free grammars taken in Clark (2010) as corrected by Yoshinaka (2011b). Under a different learning model, the minimally adequate teacher (MAT) model, Yoshinaka and Clark (2012) show that a class of multiple context-free grammars can be learned using distributional techniques. The class of grammars there is based on a different representational assumption: each nonterminal of dimension d corresponds to an equivalence class of d-words that are distributionally identical. As in the case of context-free grammars, this representational assumption limits the class of languages that can be learned significantly.

We can contrast the algorithm here with the approach taken in Shinohara (1994), which concerns the inference of elementary formal systems in the Smullyan sense, in a number of respects. First, the algorithm presented here is polynomial, whereas the Shinohara result studies learnability without constraints on the computational resources of the learner, and indeed includes languages which are not in PTIME. On the other hand, Shinohara considers a learning model where the learner only has positive examples, whereas we allow the learner to ask membership queries. Finally, Shinohara obtains learnability by bounding the number of clauses in the grammar. Here we do not need a priori bounds on the number of clauses—as we can learn grammars of unbounded complexity—but we do put bounds on the parameters that define the language-theoretic hierarchy.

A different class of representations, Marcus contextual grammars, is studied by Oates et al. (2006). These are incomparable to the classes of representations that we use, but are capable of representing some non-context-free languages, though the class studied there cannot represent all regular languages. Again this is a learning approach which only considers positive data as a source, but in this case the algorithm is also efficient, although the class of languages that can be learned is rather small. We use patterns in the definition of our rules, but the languages are very different from the pattern languages studied by Angluin (1980), though it is worth noting that every such pattern language can be defined easily by a PMCFG.

6.2 Hardness of primal approach

Among two different types of approaches in distributional learning, this paper takes the so-called *dual* approach for learning PMCFGs in the sense of Yoshinaka (2011b). Dual approaches are those where the nonterminals are defined by sets of contexts or generalisations of contexts. One might expect that the other, called *primal* approach, would work as well; these approaches define nonterminals using sets of yields of nonterminals—sets of tuples of strings. For example, using the same learning model that we use here, Yoshinaka (2010) presents a learner for learning some MCFGs using a primal approach. There the nonterminals are defined using sets of tuples of strings, and the contexts are used to eliminate the incorrect rules.

A primal counterpart to the s-FCP could be defined as follows:

> Let us say that a grammar G has the s-FKP (finite kernel property) if every nonterminal A admits a finite string set K_A of cardinality at most s such that $\pi[K_A] \subseteq \mathcal{L}(G)$ iff $\pi[\mathcal{L}(G, A)] \subseteq \mathcal{L}(G)$ for any context π.

However, the simplest non-linear grammar with the rule set $\{S(x_1x_1) :\!- S(x_1), S(a){:}-\}$ does not have the 1-FKP, which contrasts with the fact that every grammar with a single nonterminal has the 1-FCP. This grammar still has the 2-FKP, but the authors did not yet find a non-semilinear language generated by a grammar with the 1-FKP.

An even more serious problem is in the hardness of avoiding overgeneralisation while still only using polynomial-time computation (cf. Lemma 5). To get a primal learner for PMCFGs with the s-FKP, it seems a natural idea to combine the techniques proposed in this paper and by Yoshinaka (2011b). Nonterminal symbols should be indexed by string sets K of cardinality at most s. A rule of the form

$$[\![K_0]\!](\boldsymbol{\pi}) :\!- [\![K_1]\!](\mathbf{x}_1), \dots, [\![K_k]\!](\mathbf{x}_k)$$

would be said to be *correct* if $\pi_0[\boldsymbol{\pi}[K_1, \dots, K_k]] \subseteq L_*$ for every π_0 such that $\pi_0[K_0] \subseteq L_*$ where L_* is our learning target. However, to avoid exponential growth of computation time, we have to consider only r-copying contexts as π_0 for some fixed number r. Suppose, for example, $r = 2$ and $L_* = \{a^4, b^4, ad, bd, cd\}$. Since $\pi[\{a, b\}] \subseteq L_*$ iff $\pi[\{c\}] \subseteq L_*$ for every 2-copying context π, though $x_1^4[\{a, b\}] \subseteq L_*$ and $x_1^4[\{c\}] \nsubseteq L_*$, our learner will construct a rule $[\![\{a, b\}]\!](x_1) :\!- [\![\{c\}]\!](x_1)$. Together with other rules $[\![\{a^4, b^4\}]\!](x_1x_1) :\!- [\![\{aa, bb\}]\!](x_1)$, $[\![\{aa, bb\}]\!](x_1x_1) :\!- [\![\{a, b\}]\!](x_1)$ and $[\![\{c\}]\!](c){:}-$, we can derive $c^4 \notin L_*$. Thus, we can even find a *finite* language that this primal approach fails to learn. In the dual approach we have taken to learn PMCFGs, strings are used to exclude incorrect rules, and gathering all substrings from positive data can be done in polynomial time. On the other hand, in the primal approach, extracting all contexts from positive data is computationally intractable and thus we have to consider only r-copying contexts, though actually strings may be recursively copied any number of times during a derivation process. This limits our ability to control overgeneralisation in the primal approach.

6.3 Conclusion

In this paper, we have extended distributional learning to the inference of non-semilinear languages. This result also includes as a corollary a significant extension of the learnable classes of MCFGs where the nonterminals are based on contexts: a dual model in the sense of Yoshinaka (2011b).

These algorithms are all polynomial but the degree, as stated earlier, depends on the product of the parameters p, q, r and s. As a result, these algorithms will not be practical, in their naive forms, for anything other than very small values for these parameters: 1 or 2 at most. Of course, there are numerous heuristic modifications that could be made to only restrict the format of the rules that are considered to some more limited subset.

The combination of these two extensions gives, for the first time, an efficiently learnable class of languages that plausibly includes all natural languages, even under the worst case that all of the questionable examples in Sect. 2 are valid; more precisely, a class where there are no arguments that suggest that there is a natural language which is *not* in the class. In particular we are able to learn the particular example from Swiss German, which motivated the development of the theory of mildly context-sensitive grammars. Previous primal algorithm for MCFGs were not able to learn this precise case, though they could learn some closely related languages.

This leaves open two interesting issues: finding an appropriate learnable feature calculus to represent the large set of nonterminals required, and the more fundamental question of whether these grammars are also strongly adequate: adequate not just in terms of the sets of strings that they generate but in terms of the sets of structural descriptions.

From a technical point of view, it is naturally to ask whether this learning approach can be extended beyond the class of PMCFGs to use conjunction as well. The linguistic motivations for this extension do not seem to be particularly compelling, though there may be reasons to study this in other application domains such as program induction.

Acknowledgements We would like to thank the anonymous reviewers for helpful comments.

References

Andrews, A. (1996). Semantic case-stacking and inside-out unification. *Australian Journal of Linguistics*, *16*(1), 1–55.

Angluin, D. (1980). Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, *21*(1), 46–62.

Angluin, D. (1982). Inference of reversible languages. *Journal of the Association for Computing Machinery*, *29*(3), 741–765.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, *75*(2), 87–106.

Berwick, R., Pietroski, P., Yankama, B., & Chomsky, N. (2011). Poverty of the stimulus revisited. *Cognitive Science*, *35*, 1207–1242.

Bhatt, R., & Joshi, A. (2004). Semilinearity is a syntactic invariant: a reply to Michaelis and Kracht 1997. *Linguistic Inquiry*, *35*(4), 683–692.

Boullier, P. (1999). Chinese numbers, MIX, scrambling, and range concatenation grammars. In *Proceedings of the 9th conference of the European chapter of the association for computational linguistics (EACL 99)* (pp. 8–12).

Chandlee, J., & Heinz, J. (2012). Bounded copying is subsequential: implications for metathesis and reduplication. In *Twelfth meeting of the ACL special interest group on computational morphology and phonology, association for computational linguistics* (pp. 42–51).

Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, *2*(3), 113–124.

Clark, A. (2010). Learning context free grammars with the syntactic concept lattice. In *Sempere and García (2010)* (pp. 38–51).

Clark, A., & Eyraud, R. (2007). Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, *8*, 1725–1745.

Clark, A., & Lappin, S. (2011). *Linguistic nativism and the poverty of the stimulus*. New York/Oxford: Wiley/Blackwell Sci.

Evans, N. (1995). *A grammar of Kayardild: with historical-comparative notes on Tangkic* (Vol. 15). Berlin: de Gruyter.

Gazdar, G., Klein, E., Pullum, G., & Sag, I. (1985). *Generalised phrase structure grammar*. Oxford: Blackwell Sci.

Gold, E. M. (1967). Language identification in the limit. *Information and Computation*, *10*(5), 447–474.

Groenink, A. (1995). Literal movement grammars. In *Proceedings of the seventh conference of the European chapter of the association for computational linguistics*, University College, Dublin (pp. 90–97).

Groenink, A. (1997). Mild context-sensitivity and tuple-based generalizations of context-grammar. *Linguistics and Philosophy*, *20*(6), 607–636.

Huybrechts, R. A. C. (1984). The weak inadequacy of context-free phrase structure grammars. In G. de Haan, M. Trommelen, & W. Zonneveld (Eds.), *Van Periferie naar Kern*, Dordrecht: Foris.

Inkelas, S. (2008). The dual theory of reduplication. *Linguistics*, *46*(2), 351–401.

Inkelas, S., & Zoll, C. (2005). *Reduplication: doubling in morphology*. Cambridge: Cambridge University Press.

Joshi, A., Vijay-Shanker, K., & Weir, D. (1991). The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Shieber, & T. Wasow (Eds.), *Foundational issues in natural language processing* (pp. 31–81). Cambridge: MIT Press.

Kobele, G. (2006). *Generating copies: an investigation into structural identity in language and grammar*. PhD thesis, University of California Los Angeles.

Kracht, M. (2011). *Interpreted languages and compositionality*. Berlin: Springer.

Ljunglöf, P. (2005). A polynomial time extension of parallel multiple context-free grammar. In P. Blache, E. Stabler, J. Busquets, & R. Moot (Eds.), *Lecture notes in computer science: Vol. 3492. Logical aspects of computational linguistics* (pp. 177–188). Berlin: Springer.

Michaelis, J., & Kracht, M. (1997). Semilinearity as a syntactic invariant. In C. Retoré (Ed.), *Logical aspects of computational linguistics* (pp. 329–345). Berlin: Springer.

Oates, T., Armstrong, T., Becerra-Bonache, L., & Atamas, M. (2006). Inferring grammars for mildly context sensitive languages in polynomial-time. In Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, & E. Tomita (Eds.), *Lecture notes in computer science* (Vol. 4201, pp. 137–147). Berlin: Springer.

Okhotin, A. (2001). Conjunctive grammars. *Journal of Automata, Languages and Combinatorics*, *6*(4), 519–535.

Radzinski, D. (1991). Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, *17*(3), 277–299.

Sadler, L., & Nordlinger, R. (2006). Case stacking in realizational morphology. *Linguistics*, *44*(3), 459–487.

Seki, H., Matsumura, T., Fujii, M., & Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, *88*(2), 191–229.

Sempere, J. M. & García, P. (Eds.) (2010). Grammatical inference: theoretical results and applications. In *10th International Colloquium, ICGI 2010*. Berlin: Springer.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, *8*, 333–343.

Shinohara, T. (1994). Rich classes inferrable from positive data—length-bounded elementary formal systems. *Information and Computation*, *108*(2), 175–186.

Smullyan, R. (1961). *Theory of formal systems*. Princeton: Princeton University Press.

Vijay-Shanker, K., & Weir, D. J. (1994). The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, *27*(6), 511–546.

Vijay-Shanker, K., Weir, D. J., & Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th annual meeting of association for computational linguistics*, Stanford (pp. 104–111).

Yoshinaka, R. (2010). Polynomial-time identification of multiple context-free languages from positive data and membership queries. In *Sempere and García (2010)* (pp. 230–244).

Yoshinaka, R. (2011a). Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, *412*(19), 1821–1831.

Yoshinaka, R. (2011b). Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In G. Mauri & A. Leporati (Eds.), *Lecture notes in computer science: Vol. 6795. Developments in language theory* (pp. 429–440). Berlin: Springer.

Yoshinaka, R., & Clark, A. (2012). Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In P. Groote & M. J. Nederhof (Eds.), *Lecture notes in computer science: Vol. 7395. Formal grammar* (pp. 192–207). Berlin: Springer.