Inside-Outside Algorithm for Macro Grammars

Ryuta Kambe KAMBE-RYUTA363@G.ECC.U-TOKYO.AC.JP
Naoki Kobayashi KOBA@IS.S.U-TOKYO.AC.JP
Ryosuke Sato RSATO@IS.S.U-TOKYO.AC.JP
The University of Tokyo

Ayumi Shinohara AYUMIS@TOHOKU.AC.JP
Ryo Yoshinaka RYOSHINAKA@TOHOKU.AC.JP
Graduate School of Information Sciences, Tohoku University

Editors: Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, Adam Jardine, and Menno van Zaanen

Abstract

We propose an inside-outside algorithm for stochastic macro grammars. Our approach is based on types, which has been inspired by type-based approaches to reasoning about functional programs and higher-order grammars. By considering type derivations instead of ordinary word derivation sequences, we can naturally extend the standard inside-outside algorithm for stochastic context-free grammars to obtain the algorithm for stochastic macro grammars. We have implemented the algorithm and confirmed its effectiveness through an application to the learning of macro grammars.

Keywords: macro grammars, stochastic grammars, inside-outside algorithm, types

1. Introduction

The inside-outside algorithm (Baker, 1979) is a popular algorithm for learning stochastic context-free grammars (CFGs), and it has been used in various applications (Jelinek, 1985; Dodd, 1988). There are also many extensions so that the algorithm can handle various grammars, such as stochastic lexicalized tree-adjoining grammars (Schabes, 1992), projective and non-projective dependency grammars (Eisner, 1996; Koo et al., 2007), and multiple context-free grammars (MCFGs) (Kato et al., 2006).

In this paper, we extend the inside-outside algorithm for (stochastic, OI) macro grammars (Fischer, 1968), where each non-terminal may take parameters. Macro grammars (MGs) can generate non-context free languages, including non-semilinear languages, like $\{ww \mid w \in \{\mathtt{a},\mathtt{b}\}^+\}$, $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n > 0\}$ and $\{\mathtt{a}^{2^n} \mid n \geq 0\}$. The class of languages generated by MGs coincides with the class of indexed languages and also with the level-2 language in Damm's OI-hierarchy of higher-order languages (Damm, 1982) (where level-0 and level-1 languages are regular and context-free langauges respectively). Stochastic MGs would be useful for more faithfully representing stochastic language models than less expressive grammars like CFGs.

Extending the inside-outside algorithm for MGs is non-trivial. In CFGs, non-terminals in sequences derived from the initial symbol are rewritten in an arbitrarily order and they derive strings independently. This makes derivation modes like left-most derivation and right-most derivations indifferent and allows us to define *derivation trees*, based on which the inside-outside algorithm can be constructed. It is the case for TAGs and MCFGs as well. However, the situation is quite different when considering MG derivations, where a

non-terminal may copy other non-terminals and those copies may derive different terms. Thus, it is tricky to come up with the right notion of derivation trees for designing an inside-outside algorithm.

Our key idea for extending the inside-outside algorithm is to use *intersection types*, which have been proved useful for reasoning about higher-order grammars (Kobayashi, 2013b; Kobayashi et al., 2013). By replacing the role of word derivations trees in the inside-outside algorithm for stochastic CFGs with that of *type derivation trees*, we can naturally obtain an inside-outside algorithm for MGs (and possibly also for arbitrary higher-order grammars (Damm, 1982; Kobele and Salvati, 2015), though it is left for future work).

We have implemented the algorithm and applied it to learn (non-stochastic) MGs. The learning of MGs proceeds as follows. Given a set of words, the goal is to find an MG that provides a good explanation of the words. To this end, we first fix a set of non-terminals and prepare a set of all the possible rules (in certain normal form), and run the inside-outside algorithm to maximize the probability that the words are generated. We then remove the rules whose probabilities are below a certain threshold. Finally, by removing the probabilities of the remaining rules, we obtain a non-stochastic MG. We have applied this procedure to several non-context free languages including non-semilinear ones and successfully obtained appropriate MGs.

Another potential application of our inside-outside algorithm is grammar-based data compression. Stochastic CFGs have been used for data compression, where a word is compressed as (the arithmetic coding of) its derivation sequence (Cameron, 1988; Tarhio, 2001; Naganuma et al., 2020). We expect that, by using stochastic MGs, we can achieve more compact coding of word data.

Related Work. As already mentioned, the inside-outside algorithm has been originally proposed for (stochastic) CFGs (Baker, 1979) and adapted later to other grammars (Schabes, 1992; Eisner, 1996; Koo et al., 2007; Kato et al., 2006). To our knowledge, the inside-outside algorithm for MGs is new. A distinguishing, novel feature of our inside-outside algorithm is the use of types, which allowed us to naturally extend the algorithm for CFGs. Type-based techniques have recently been used for higher-order grammars, albeit for different purposes such as higher-order model checking (Kobayashi, 2013b), pumping lemmas (Kobayashi, 2013a; Asada and Kobayashi, 2017), and data compression (Kobayashi et al., 2013).

Structure of the Paper. The rest of this paper is structured as follows. Section 2 gives the definition of stochastic macro grammars. Section 3 describes the inside-outside algorithm for stochastic macro grammars. Section 4 reports experiments, and Section 5 concludes the paper. The details omitted in this paper are found in an extended version of this paper (Kambe et al., 2021).

2. Stochastic Macro Grammars

In this section, we review the notion of macro grammars (Fischer, 1968) and define its stochastic extension.

Macro grammars are an extension of context-free grammars, where non-terminals may take parameters. For example, a macro grammar may contain a rule like $F\ x \to x \cdot x$, where

the non-terminal F takes a parameter x and $\cdot$ denotes the word concatenation (we omit $\cdot$ when there is no danger of confusion). Using the rule, F `a` and F `b` are rewritten to `aa` and `bb`.

Definition 1 (Macro grammars) *A* macro grammar *(MG, for short) is a quadruple $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$ where:*

(i) $\mathcal{A}$ is a finite set of symbols called terminals*; we use meta-variables $a, b, c, \ldots$ for terminals;*

(ii) $\mathcal{N}$ is a map from a finite set of symbols called non-terminals *to the set of non-negative integers; $\mathcal{N}(F)$, called the arity of F, represents how many parameters F takes; we use meta-variables $F, G, S, \ldots$ for non-terminals;*

(iii) $\mathcal{R}$ is a finite set of rewriting rules of the form $F x_1 \ldots x_{\mathcal{N}(F)} \to t$ where F is a non-terminal, $x_1, \ldots, x_{\mathcal{N}(F)}$ are distinct variables, and t ranges over the set of terms *defined by the following BNF:*

$$t ::= x_i \mid a \mid t_1 \cdot t_2 \mid G\, t_1 \cdots t_{\mathcal{N}(G)}$$

We give higher precedence to concatenations ($\cdot$) than to applications, so that $F\, x \cdot x$ means $F(x \cdot x)$ instead of $(F\, x) \cdot x$. We sometimes insert parentheses explicitly to avoid confusion. By using the notation of λ-calculus, we sometimes write $F \to \lambda x_1, \ldots, x_k.t$ for the rule $F\ x_1\ \cdots\ x_k \to t$, and use meta-variables $\alpha, \beta, \cdots$ for the part $\lambda x_1, \ldots, x_k.t$ (thus, the rule is written $F \to \alpha$);

(iv) $S \in dom(\mathcal{N})$ is the start symbol, which satisfies $\mathcal{N}(S) = 0$;

For an MG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$, the (outermost, leftmost[1]) rewriting relation $t \Rightarrow_{\mathcal{G},r} t'$ (where $r \in \mathcal{R}$) is inductively defined by:

- $F\, t_1 \cdots t_k \Rightarrow_{\mathcal{G}, F \to \lambda x_1, \ldots, x_k.t} [t_1/x_1, \ldots, t_k/x_k]t$ *if* $F \to \lambda x_1, \ldots, x_k.t$.
- $t_1 \cdot t_2 \Rightarrow_{\mathcal{G},r} t_1' \cdot t_2$ *if* $t_1 \Rightarrow_{\mathcal{G},r} t_1'$.
- $t_1 \cdot t_2 \Rightarrow_{\mathcal{G},r} t_1 \cdot t_2'$ *if* $t_2 \Rightarrow_{\mathcal{G},r} t_2'$ *and* t_1 *contains no non-terminals.*

Here, $[t_1/x_1, \ldots, t_k/x_k]t$ denotes the term obtained from t by simultaneously replacing each occurrence of x_i with t_i. We omit the subscripts $\mathcal{G}$ and/or r when they are clear from the context or unimportant. We write $\Rightarrow^*_{\mathcal{G}, r_1 \cdots r_k}$ for the relational composition $\Rightarrow_{\mathcal{G},r_1} \cdots \Rightarrow_{\mathcal{G},r_k}$ (where k may be 0, in which case $\Rightarrow^*_{\mathcal{G}, r_1 \cdots r_k}$ is the identity relation on terms). Note that, given a term t and a sequence of rules $r_1, \ldots, r_k \in \mathcal{R}$, there exists at most one term t' such that $t \Rightarrow^*_{\mathcal{G}, r_1 \cdots r_k} t'$.

The language generated by $\mathcal{G}$, written $\mathcal{L}(\mathcal{G})$, is defined as $\{w \in \mathcal{A}^* \mid S \Rightarrow^*_{\mathcal{G}} w\}$.

Remark 2 *Fischer (1968) actually introduced two kinds of macro grammars: OI and IO. The MGs defined above are OI macro grammars. For the sake of simplicity, we excluded out the empty word ϵ from the set of terms. This is not a fundamental restriction, because any MG $\mathcal{G}$ with ϵ can be transformed to another MG $\mathcal{G}'$ without ϵ such that $\mathcal{L}(\mathcal{G}) \setminus \epsilon = \mathcal{L}(\mathcal{G}')$ (see, e.g., Asada and Kobayashi, 2016).*

Below we often write $\tilde{x}$ for a sequence of variables $x_1 \cdots x_k$.

1. We need to fix the rewriting strategy to properly count the probability of a word being generated. We could alternatively choose the rightmost derivation.

Example 1 *Consider the MG* $\mathcal{G}_{2^n} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$ *where* $\mathcal{A} = \{\mathtt{a}\}$, $\mathcal{N} = \{S \mapsto 0, F \mapsto 1\}$, *and* $\mathcal{R}$ *consists of:*

$$S \to \mathtt{a} \mid F\,\mathtt{a} \qquad\qquad F\,x \to x \cdot x \mid F(x \cdot x)$$

(Here, we use the convention of writing multiple rules $F\,\tilde{x} \to t_1, \ldots, F\,\tilde{x} \to t_\ell$ *that share the same left-hand side as* $F\,\tilde{x} \to t_1 \mid \cdots \mid t_\ell$*.) The word* $\mathtt{a}^4 = \mathtt{aaaa}$ *can be generated as follows.*

$$S \Rightarrow_{S \to F\,\mathtt{a}} F\,\mathtt{a} \Rightarrow_{F\,x \to F(x \cdot x)} F(\mathtt{a} \cdot \mathtt{a}) \Rightarrow_{F\,x \to x \cdot x} \mathtt{aaaa}.$$

The language $\mathcal{L}(\mathcal{G}_{2^n})$ *is* $\{\mathtt{a}^{2^n} \mid n \geq 0\}$. □

Definition 3 (Stochastic macro grammars) *A* stochastic macro grammar *(SMG, for short) is a quintuple* $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$*, where* $(\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$ *is an MG, and* $\phi : \mathcal{R} \to (0, 1]$ *assigns a non-zero probability to each rule. We require that* $\sum_{F \to \alpha \in \mathcal{R}} \phi(F \to \alpha) = 1$ *holds for each non-terminal* $F \in dom(\mathcal{N})$*.*

The rewriting relation $\Rightarrow_{\mathcal{G},r}$ *for* $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ *is the one for the underlying MG* $(\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$*.*

The probability of a rewriting sequence $t \Rightarrow^*_{\mathcal{G},r_1 \cdots r_k} t'$*, written* $Prob(t \Rightarrow^*_{\mathcal{G},r_1 \cdots r_k} t')$*, is defined as* $\prod_{i=1}^{k} \phi(r_i)$*. For terms* t *and* t'*, the probability that* t' *is obtained from* t*, written* $Prob(t \Rightarrow^*_{\mathcal{G}} t')$*, is defined as the sum of the probabilities of all the rewriting sequences from* t *to* t'*, i.e.,* $\sum_{r_1 \cdots r_k \in \mathcal{R}^*, t \Rightarrow^*_{\mathcal{G},r_1 \cdots r_k} t'} Prob(t \Rightarrow^*_{\mathcal{G},r_1 \cdots r_k} t')$*. For a word* $W \in \mathcal{A}^*$*, we write* $Prob_{\mathcal{G}}(W)$ *for* $Prob(S \Rightarrow^*_{\mathcal{G}} W)$*. We write* $\mathcal{L}(\mathcal{G})$ *for the set* $\{W \in \mathcal{A}^* \mid Prob_{\mathcal{G}}(W) > 0\}$*.*

For an SMG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$, we write $(\mathcal{G})^{\#}$ for the underlying MG $(\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$. Note that $\mathcal{L}(\mathcal{G}) = \mathcal{L}((\mathcal{G})^{\#})$ holds.

Example 2 *Consider the SMG* $\mathcal{G}'_{2^n} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ *where* $(\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$ *is as given in Example 1, and* ϕ *is given by:*

$$\phi(S \to \mathtt{a}) = 0.3, \qquad \phi(S \to F\,\mathtt{a}) = 0.7,$$
$$\phi(F\,x \to x \cdot x) = 0.4, \quad \phi(F\,x \to F(x \cdot x)) = 0.6.$$

Then, the probability for the derivation

$$S \Rightarrow_{S \to F\,\mathtt{a}} F\,\mathtt{a} \Rightarrow_{F\,x \to F(x \cdot x)} F(\mathtt{a} \cdot \mathtt{a}) \Rightarrow_{F\,x \to x \cdot x} \mathtt{aaaa}$$

is $0.7 \times 0.6 \times 0.4 = 0.168$*. Since this is the only derivation sequence for generating* $\mathtt{a}^4$*,* $Prob_{\mathcal{G}'_{2^n}}(\mathtt{a}^4) = 0.168$*.* □

In the rest of the paper, we impose a restriction to SMGs, so that for each word $W \in \mathcal{A}^*$, the set of rewriting sequences $S \Rightarrow^*_{\mathcal{G},r_1 \cdots r_k} W$ is finite. For a term t, we write $\|t\|_0$ for the number of occurrences of symbols (terminals, non-terminals, and variables) in t, $\|t\|_1$ for the number of occurrences of symbols excluding non-terminals of non-zero arity. For example, if $\mathcal{N}(F) = 1$ and $\mathcal{N}(G) = 0$, then $\|F(\mathtt{a} \cdot \mathtt{a} \cdot G)\|_0 = 4$ while $\|F(\mathtt{a} \cdot \mathtt{a} \cdot G)\|_1 = 3$. We define the measure $\|t\|$ of t by:

$$\|t\| ::= (\|t\|_0, \|t\|_1).$$

For example, $\|F(x \cdot \mathtt{a})\| = (3, 2)$. We define the strict partial order $<$ on $\|t\|$ as the lexicographic order: $(m, n) < (m', n')$ iff $m < m'$ or $m = m' \wedge n < n'$.

Definition 4 (Expansive grammars) *An SMG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ is* expansive *if (I) the start symbol S does not occur on the right-hand side of any rule and (II) for every rule $F\, x_1 \cdots x_k \to t \in \mathcal{R}$ such that $F \neq S$, (i) each variable x_i occurs at least once in t, and (ii) $\|F\, x_1 \ldots x_k\| < \|t\|$.*

Note that the grammar $\mathcal{G}'_{2^n}$ in Example 2 is expansive. The following property of expansive SMGs is important for our inside-outside algorithm. A proof is given in Kambe et al. (2021).

Lemma 5 *Let $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ be an expansive SMG. For every word $W \in \mathcal{A}^*$, the set of rewriting sequences $S \Rightarrow^*_{\mathcal{G}, r_1 \cdots r_k} W$ is finite.*

Henceforth, we consider only expansive SMGs, and call them just SMGs.

Remark 6 *Every context-free grammar in Chomsky normal form is expansive. To our knowledge, there is no such normal form for MGs. The definition of expansive grammars above is non-standard and ad hoc. We do not know whether every macro grammar can be transformed to an equivalent expansive grammar.*

3. The Inside-Outside Algorithm for Macro Grammars

This section introduces our inside-outside algorithm for SMGs. Given an SMG $\mathcal{G}$ and a finite set of words $\mathcal{W} = \{W_1, \ldots, W_Q\}$, the goal of the algorithm is to obtain a probability function ϕ that (locally) maximize $\prod_{q=1}^{Q} \mathit{Prob}_{\mathcal{G}}(W_q)$ by iteratively updating ϕ.[2] At each iteration, the value of $\phi(A \to \alpha)$ is updated to $\dfrac{c(A \to \alpha)}{\sum_{A \to \beta} c(A \to \beta)}$, where $c(A \to \alpha)$ is the expected number of times the rule $A \to \alpha$ is used to derive the words in $\mathcal{W}$.

In the inside-outside algorithm for stochastic context-free grammars (SCFGs), $c(A \to \alpha)$ is obtained by considering the *inside probability* $\mathit{inside}(A, i, j)$ and *outside probability* $\mathit{outside}(A, i, j)$. For a word $W = a_1 \cdots a_n$, $\mathit{inside}(A, i, j)$ (where $1 \leq i \leq j \leq n$) denotes the probability that $a_i \cdots a_j$ is generated from A (or, the probability that the derivation tree for $a_i \cdots a_j$ whose root is A is obtained), and $\mathit{outside}(A, i, j)$ denotes the probability that $a_1 \cdots a_{i-1} A a_{j+1} \cdots a_n$ can be derived from the start symbol S: see Figure 1 (a) for an illustration.

In the case of SMG, the notion of a "derivation tree" is unclear, due to the existence of parameters. Our insight is that, by considering a "type derivation tree" corresponding to a derivation of a word, we can naturally define the *inside probability* and *outside probability*, and obtain an inside-outside algorithm based on those probabilities. For those who are familiar with type systems, Figure 1 (b) shows an illustration; it will be explained later. The reason why "types" come to play is that an SMG may be viewed as a (probabilistic) functional program; thus, a type system can be used to analyze the behavior of an SMG. In

2. As in the case for stochastic context-free grammars, our inside-outside algorithm may be viewed as an instance of EM-algorithm (Dempster et al., 1977; Wu, 1983). Therefore, actually we can only guarantee that the value of $\prod_{q=1}^{Q} \mathit{Prob}_{\mathcal{G}}(W_q)$ increases monotonically at each iteration, and the algorithm converges to the value at a stationary point of $\prod_{q=1}^{Q} \mathit{Prob}_{\mathcal{G}}(W_q)$ (as a function of ϕ), not necessarily to a local maximum.

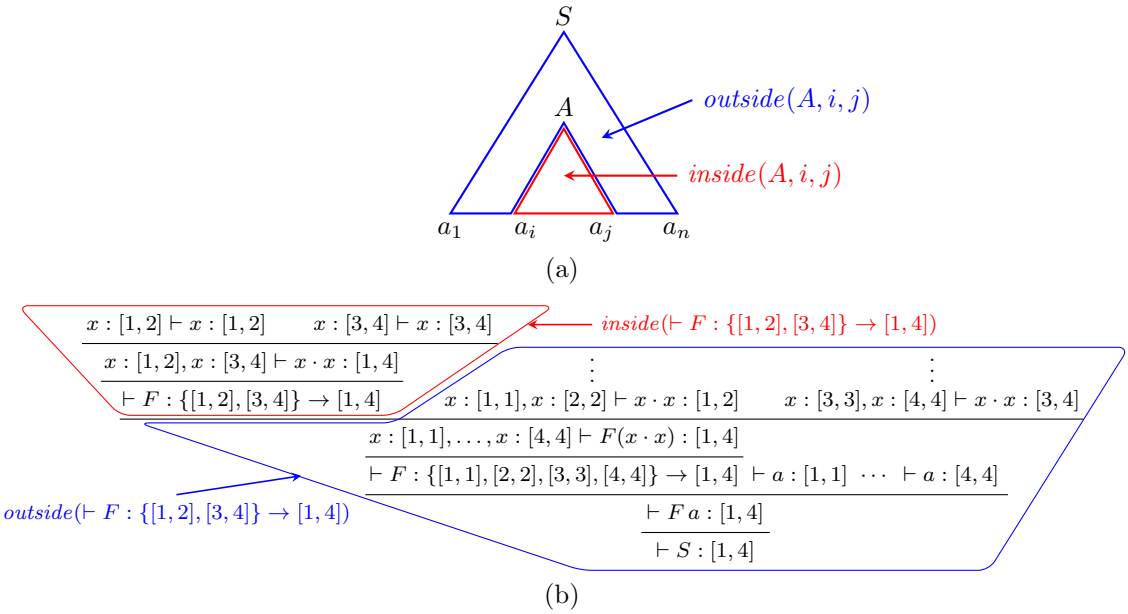

Figure 1: Inside and outside probabilities for SCFG (a) and SMG (b)

fact, types have recently been used for analyzing higher-order grammars (of which an MG can be considered a special case) (Kobayashi, 2013b; Kobayashi et al., 2013).

Remark 7 *It would be possible to formalize an inside-outside algorithm without using the notion of types, but the resulting formalization would be more awkward. Also, we expect that the use of types allows us to extend the inside-outside algorithm for higher-order grammars of arbitrary orders, although it is left for future work.*

We now define a type system. Below we fix an SMG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ and a word $W = a_1 \cdots a_n \in \mathcal{A}^*$. We write $W[i,j]$ (where $1 \leq i \leq j \leq n$) for the subword $a_i \cdots a_j$.

The set of types, ranged over by τ, is defined by:

$$\tau ::= [i,j] \mid \sigma \to \tau \qquad \sigma ::= \{[i_1,j_1], \ldots, [i_k,j_k]\}.$$

Intuitively, $[i,j]$ is the type of a term that may be used to generate $W[i,j]$. For example, if $W = \mathtt{a}^4$, then $\mathtt{aa}$ has types $[1,2]$, $[2,3]$, and $[3,4]$. The type $\sigma \to \tau$ describes a function that takes an argument that has every type $\tau' \in \sigma$, and returns a value of type τ. For example, if there is a rule $F\ x \to x \cdot x$, F has type $\{[1,2],[3,4]\} \to [1,4]$. We require that a function of type $\sigma \to \tau$ must use the argument once for each $\tau' \in \sigma$. Thus, the function G with rule $G\ x \to \mathtt{a} \cdot \mathtt{a} \cdot x$ has type $\{[3,4]\} \to [1,4]$, but not $\{[1,2],[3,4]\} \to [1,4]$ (because x is used to generate only $W[3,4]$, not $W[1,2]$). A set of types of the form $\{[i_1,j_1], \ldots, [i_k,j_k]\}$ is so called an *intersection type* in the terminology of programming languages; we sometimes write $[i_1,j_1] \wedge \cdots \wedge [i_k,j_k]$ for it. We also just write $[i,j]$ for $\{[i,j]\}$, and $\top$ for $\emptyset$.

We restrict the shape of types as follows. We define $\tau^\#$ by:

$$[i,j]^\# = \{x \in \mathbf{Nat} \mid i \leq x \leq j\}$$
$$(\{[i_1,j_1], \ldots, [i_k,j_k]\} \to \tau)^\# = \tau^\# \setminus ([i_1,j_1]^\# \cup \cdots \cup [i_k,j_k]^\#)$$

$$\frac{}{x : [i,j] \vdash_W x : [i,j]} \quad \text{(T-Var)} \qquad \frac{W[i,i] = a}{\vdash_W a : [i,i]} \quad \text{(T-Term)}$$

$$\frac{\Gamma_1 \vdash_W t_1 : [i,j] \qquad \Gamma_2 \vdash_W t_2 : [j+1,k]}{\Gamma_1 \uplus \Gamma_2 \vdash_W t_1 \cdot t_2 : [i,k]} \quad \text{(T-Concat)}$$

$$\frac{\vdash_W F : \sigma_1 \to \cdots \to \sigma_{\mathcal{N}(F)} \to [i,j] \qquad \Gamma_{k,\tau} \vdash_W t_k : \tau \text{ for each } k \in \{1,\ldots,\mathcal{N}(F)\},\ \tau \in \sigma_k}{\uplus_{k \in \{1,\ldots,\mathcal{N}(F)\}, \tau \in \sigma_k} \Gamma_{k,\tau} \vdash_W F\, t_1 \cdots t_{\mathcal{N}(F)} : [i,j]} \quad \text{(T-App)}$$

$$\frac{F\, x_1 \cdots x_k \to t \in \mathcal{R} \qquad x_1 : \sigma_1, \ldots, x_k : \sigma_k \vdash_W t : [i,j]}{\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i,j]} \quad \text{(T-Rule)}$$

Figure 2: Typing rules

where **Nat** is the set of natural numbers. Intuitively, $\tau^{\#}$ denotes the set of positions generated by a term of type τ. We require: (i) in $[i,j]$, it must be the case that $i \leq j$, and (ii) in $\{[i_1,j_1],\ldots,[i_k,j_k]\} \to [i,j]$, $[i_\ell,j_\ell]^{\#} \cap [i_{\ell'},j_{\ell'}]^{\#} = \emptyset$ for any $1 \leq \ell < \ell' \leq k$, and $(\bigcup_{\ell \in \{1,\ldots,k\}} [i_\ell,j_\ell]^{\#}) \subseteq [i,j]^{\#}$. Henceforth, we consider only types that satisfy the above restriction.

We consider a *type judgment* of the form $\Gamma \vdash_W t : \tau$, where Γ, called a *type environment*, is a set of bindings of the form $x : [i,j]$. Each binding $x : [i,j]$ in a type environment means that x must be used as a value of type $[i,j]$, i.e., may be used to generate $W[i,j]$. A type environment may contain more than one binding for each variable; for example, $\{x:[1,2], x:[3,4]\}$ means that x must be used for generating $W[1,2]$, and also for generating $W[3,4]$. We sometimes write $x : \sigma$ to mean the type environment $\bigcup_{[i,j] \in \sigma} \{x : [i,j]\}$. The judgment $\Gamma \vdash_W t : \tau$ means that assuming that the variables in t are used according to Γ, t behaves as a value of type τ. For example, $\{x : [1,2], x : [3,4]\} \vdash_{\mathtt{a}^4} x \cdot x : [1,4]$ is a valid judgment. We often omit brackets and just write $x_1 : [i_1,j_1],\ldots,x_k : [i_k,j_k]$ for $\{x_1 : [i_1,j_1],\ldots,x_k : [i_k,j_k]\}$. When Γ is empty, we just write $\vdash_W t : \tau$ for $\emptyset \vdash_W t : \tau$. We usually fix W and omit the subscript W.

The typing rules for deriving valid type judgments are given in Figure 2. In the figure, $\uplus$ denotes the disjoint union: $\Gamma_1 \uplus \Gamma_2$ is defined to be $\Gamma_1 \cup \Gamma_2$ only if $\Gamma_1 \cap \Gamma_2 = \emptyset$.

Figure 1 (b) shows a type derivation tree for $\vdash_W S : [1,4]$ for the SMG in Example 2 and $W = \mathtt{a}^4$. The type derivation tree corresponds to the derivation sequence $S \Rightarrow_{S \to F\,\mathtt{a}} F\,\mathtt{a} \Rightarrow_{F\,x \to F(x \cdot x)} F(\mathtt{a} \cdot \mathtt{a}) \Rightarrow_{F\,x \to x \cdot x} \mathtt{aaaa}$. The judgment $\vdash F : \{[1,2],[3,4]\} \to [1,4]$ in the type derivation tree corresponds to the second use of F (in $F(\mathtt{a} \cdot \mathtt{a})$) and represents that the argument $\mathtt{a} \cdot \mathtt{a}$ is used for generating the subwords $W[1,2]$ and $W[3,4]$. This is not a coincidence. Every type derivation tree for $\vdash_W S : [1,|W|]$ corresponds to a derivation sequence for W, and vice versa. Thus, we can calculate the probability $\mathit{Prob}_{\mathcal{G}}(W)$ by summing up the probabilities of type derivation trees, instead of considering derivation sequences, as discussed below.

We write T for a type derivation tree, and $\mathbf{DT}(\Gamma \vdash t : \tau)$ for the set of type derivation trees whose conclusion is $\Gamma \vdash t : \tau$.[3] We write $occ(T, F \to \alpha)$ for the number of times the rule $F \to \alpha$ is used in (applications of the rule T-Rule) T. The probability $Prob(T)$ of a type derivation tree T is defined by:

$$Prob(T) = \prod_{F \to \alpha \in \mathcal{R}} \phi(F \to \alpha)^{occ(T, F \to \alpha)}.$$

For a type judgment $\Gamma \vdash_W t : \tau$, its *inside probability* $inside(\Gamma \vdash_W t : \tau)$ is defined by:

$$inside(\Gamma \vdash_W t : \tau) = \sum_{T \in \mathbf{DT}(\Gamma \vdash_W t : \tau)} Prob(T).$$

Thus, the inside probability $inside(\Gamma \vdash_W t : \tau)$ is analogous to the inside probability for SCFG, where derivation trees have been replaced by type derivation trees. The theorem below guarantees that our inside probability correctly models the probability that a word is generated. A proof is given in Kambe et al. (2021).

Theorem 8 *Let $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ be an SMG and $W \in \mathcal{A}^*$. Then $Prob_{\mathcal{G}}(W) = inside(\vdash_W S : [1, |W|])$.*

For a type environment Γ, we write $\Gamma{\downarrow_{[i,j]}}$ for the restricted type environment $\{x : [i', j'] \in \Gamma \mid i \le i' \le j' \le j\}$; $\Gamma{\downarrow_{[i,j]}}$ is undefined if there exists $x : [i', j'] \in \Gamma$ such that $[i, j]^{\#} \cap [i', j']^{\#} \neq \emptyset$ but $[i', j']^{\#} \not\subseteq [i, j]^{\#}$. We write $\mathbf{Ty}_W$ for the set of (valid) types, where all the indexes i occurring in them must satisfy $1 \le i \le |W|$ (so that they are valid indexes for W). As in the case for SCFG, the inside probability $inside(\Gamma \vdash t : \tau)$ can be inductively calculated as follows.

$$inside(x : [i, j] \vdash_W x : [i, j]) = 1$$

$$inside(\vdash_W a : [i, i]) = \begin{cases} 1 & \text{if } a = W[i, i] \\ 0 & \text{otherwise} \end{cases}$$

$$inside(\Gamma \vdash_W t_1 \cdot t_2 : [i, j]) =$$
$$\sum_{i \le k < j} inside(\Gamma{\downarrow_{[i,k]}} \vdash_W t_1 : [i, k]) \times inside(\Gamma{\downarrow_{[k+1,j]}} \vdash_W t_2 : [k+1, j])$$

$$inside(\Gamma \vdash_W F\, t_1 \cdots t_k : [i, j]) =$$
$$\sum_{\sigma_1 \to \cdots \to \sigma_k \to [i,j] \in \mathbf{Ty}_W} \Big(inside(\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i, j]) \times \prod_{1 \le \ell \le k, [i', j'] \in \sigma_\ell} inside(\Gamma{\downarrow_{[i', j']}} \vdash_W t_\ell : [i', j']) \Big)$$

$$inside(\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i, j]) =$$
$$\sum_{F x_1 \ldots x_k \to t \in \mathcal{R}} \phi(F x_1 \ldots x_k \to t) \times inside(x_1 : \sigma_1, \ldots, x_k : \sigma_k \vdash_W t : [i, j])$$

3. We view type derivation trees as *unordered* trees, where the order of children of each node is irrelevant. For example, $\dfrac{T_1 \quad T_2}{\Gamma \vdash t : \tau}$ and $\dfrac{T_2 \quad T_1}{\Gamma \vdash t : \tau}$ are considered the same trees.

Here, $\mathit{inside}(\Gamma \vdash_W t : \tau) = 0$ if it does not match any of the above definitions (e.g., $\mathit{inside}(x : [1,2] \vdash_W x : [1,3]) = 0$). Each equation for $\mathit{inside}(\Gamma \vdash_W t : \tau)$ sums up the probabilities of all the possible derivations for $\Gamma \vdash_W t : \tau$. In the fourth clause, we assume $k > 0$; the case where t is an arity-0 non-terminal is covered by the last clause. The correctness of this calculation is discussed in Kambe et al. (2021).

Remark 9 *A careful reader may notice that the above algorithm for calculating* $\mathit{inside}(\Gamma \vdash t : \tau)$ *is circular: in the definition of* $\mathit{inside}(\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i,j])$, F *may occur in* t*, so that* $\mathit{inside}(\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i,j])$ *may show up again in the calculation of* $\mathit{inside}(x_1 : \sigma_1, \ldots, x_k : \sigma_k \vdash_W t : [i,j])$*. This is actually not a problem. Thanks to the restriction to expansive grammars (recall Definition 4), we can actually prove that in any type derivation tree* T*, each type judgment may occur at most once. Thus, when recursively calculating* $\mathit{inside}(\Gamma \vdash_W t : \tau)$*, we can consider that* $\mathit{inside}(\Gamma \vdash_W t : \tau) = 0$ *when it is encountered again. Without the restriction to expansive grammars, we would need to compute the least fixpoint of the equations above, which is computationally more expensive.*

The outside probability $\mathit{outside}(\Gamma \vdash_W t : \tau)$, which intuitively expresses the probability that $\vdash_W S : [1, |W|]$ is derived from $\Gamma \vdash_W t : \tau$, can also be calculated analogously to the case of SCFG. Let $\mathbf{Terms}_{\mathcal{G}}$ be the set of terms that occur in the rewriting rules of $\mathcal{G}$. Then $\mathit{outside}(\Gamma \vdash_W t : \tau)$ is given by:

$$
\begin{aligned}
&\mathit{outside}(\vdash_W S : [1,|W|]) = 1\,,\\
&\mathit{outside}(\Gamma \vdash_W t : [i,j]) =\\
&\quad \Big(\textstyle\sum_{t \cdot t' \in \mathbf{Terms}_{\mathcal{G}}, j+1 \le k, \Gamma'\downarrow_{[i,j]} = \Gamma}\\
&\qquad\quad \mathit{outside}(\Gamma' \vdash_W t \cdot t' : [i,k]) \times \mathit{inside}(\Gamma'\downarrow_{[j+1,k]} \vdash_W t' : [j+1,k])\Big)\\
&\quad + \Big(\textstyle\sum_{t' \cdot t \in \mathbf{Terms}_{\mathcal{G}}, k \le i-1, \Gamma'\downarrow_{[i,j]} = \Gamma}\\
&\qquad\quad \mathit{outside}(\Gamma' \vdash_W t' \cdot t : [k,j]) \times \mathit{inside}(\Gamma'\downarrow_{[k,i-1]} \vdash_W t' : [k,i-1])\Big)\\
&\quad + \Big(\textstyle\sum_{F\tilde{x} \to t \in \mathcal{R}, \Gamma = \{x_1:\sigma_1, \ldots, x_k:\sigma_k\}} \phi(F\tilde{x} \to t) \times \mathit{outside}(\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i,j])\Big)\\
&\quad + \Big(\textstyle\sum_{F\,t_1 \cdots t_k \in \mathbf{Terms}_{\mathcal{G}}, t = t_\ell, \Gamma'\downarrow_{[i,j]} = \Gamma, \sigma_1 \to \cdots \to \sigma_\ell \uplus \{[i,j]\} \to \cdots \to \sigma_k \to [i',j'] \in \mathbf{Ty}_W}\\
&\qquad\quad \mathit{outside}(\Gamma' \vdash_W F\,t_1 \cdots t_k : [i',j'])\\
&\qquad\quad \times \mathit{inside}(\vdash_W F : \sigma_1 \to \cdots \to \sigma_\ell \uplus \{[i,j]\} \to \cdots \to \sigma_k \to [i',j'])\\
&\qquad\quad \times \textstyle\prod_{1 \le m \le k, [i'',j''] \in \sigma_m} \mathit{inside}(\Gamma'\downarrow_{[i'',j'']} \vdash_W t_m : [i'',j'])\Big)\,, \qquad (\text{if } t \neq S)\\
&\mathit{outside}(\vdash_W F : \sigma_1 \to \cdots \to \sigma_k \to [i,j]) =\\
&\quad \textstyle\sum_{F\,t_1 \cdots t_k \in \mathbf{Terms}_{\mathcal{G}}, \Gamma} \mathit{outside}(\Gamma \vdash_W F\,t_1 \cdots t_k : [i,j])\\
&\qquad\quad \times \textstyle\prod_{1 \le m \le k, [i',j'] \in \sigma_m} \mathit{inside}(\Gamma\downarrow_{[i',j']} \vdash_W t_m : [i',j'])\Big)\,. \qquad (\text{if } k > 0)
\end{aligned}
$$

Although the equations above may look complicated, they can systematically be obtained from the intuition on outside probabilities. Unlike inside probabilities, the case where t is an arity-0 non-terminal is covered by the second clause since such t as a premise does not appear in T-Rule except when $t = S$.

Now we are ready to describe our inside-outside algorithm, which takes a set of words $\mathcal{W} = \{W_1, \ldots, W_Q\}$ and an SMG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ as inputs, and output an SMG $\mathcal{G}' = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi')$. The overall algorithm is shown in Algorithm 1.

In Step 1, $c(F \to \alpha)$ denotes the expected number of times the rule $F \to \alpha$ is used for deriving $W_1, \ldots, W_Q$. It is calculated by:

$$c(F \to \alpha) = \sum_{q=1}^{Q} \frac{\phi(F \to \alpha)}{Prob_{\mathcal{G}}(W_q)} \sum_{\sigma_1 \to \cdots \to \sigma_k \to [i,j] \in \mathbf{Ty}_{W_q}} \tag{1}$$
$$outside(\vdash_{W_q} F : \sigma_1 \to \cdots \to \sigma_k \to [i,j]) \times inside(x_1 : \sigma_1, \cdots, x_k : \sigma_k \vdash_{W_q} t : [i,j]).$$

In Step 2, $\phi(F \to \alpha)$ is re-estimated by:

$$\phi'(F \to \alpha) = \frac{c(F \to \alpha)}{\sum_{F \to \beta \in \mathcal{R}} c(F \to \beta)}. \tag{2}$$

The probability function ϕ is repeatedly updated until the change in $\prod_{q=1}^{Q} Prob_{\mathcal{G}}(W_q)$ is less than the given threshold ϵ.

Algorithm 1: The Inside-Outside Algorithm

Given an SMG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$, a set of words $\mathcal{W} = \{W_1, \ldots, W_Q\}$, and a threshold ϵ

1. For each $F \to \alpha \in \mathcal{R}$, calculate $c(F \to \alpha)$ by equation 1
2. For each $F \to \alpha \in \mathcal{R}$, calculate $\phi'(F \to \alpha)$ by equation 2
3. $\mathcal{G}' \leftarrow (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi')$
4. If $\frac{\prod_{q=1}^{Q} Prob_{\mathcal{G}'}(W_q)}{\prod_{q=1}^{Q} Prob_{\mathcal{G}}(W_q)} < 1 + \epsilon$, then output $\mathcal{G}'$;
   otherwise let $\mathcal{G} \leftarrow \mathcal{G}'$ and go back to step 1

Like the inside-outside algorithm for SCFG (Lafferty, 2000), our algorithm for SMG can be viewed as an instance of EM algorithm, hence satisfies the following property.

Theorem 10 (correctness of the inside-outside algorithm) *In Algorithm 1, the value of $\prod_{q=1}^{Q} Prob_{\mathcal{G}}(W_q)$ increases monotonically at each iteration.*

See Kambe et al. (2021) for a proof.

Remark 11 *If the arity of every non-terminal is 0 and all the rules in $\mathcal{R}$ are in Chomsky normal form, then our algorithm coincides with the standard inside-outside algorithm for SCFG. See Kambe et al. (2021) for more details.*

As for the complexity of the algorithm, the worst-case complexity for computing the inside probabilities for a word W is exponential in $|W|$, unfortunately. This is because in general we need to consider a type of the form $\sigma_1 \to \cdots \to \sigma_k \to [i,j]$ for each non-terminal of arity k, and each σ_i is a *set* of intervals of the form $[i', j']$. Even under the well-formed condition, there are as many as $k^{O(|W|)}$ possible types, hence also $k^{O(|W|)}$ possible type judgements to be considered when computing the inside and outside probabilities. This problem is similar to that of the extremely high complexity of higher-order model checking

(and related problems on higher-order grammars) (Ong, 2006; Kobayashi and Ong, 2011). In practice, however, we expect that we can often avoid the exponential blow-up by considering only relevant types, as in practical algorithms for higher-order model checking (Kobayashi, 2013b; Broadbent and Kobayashi, 2013; Ramsay et al., 2014).

4. Experiments

We have implemented the inside-outside algorithm for SMGs, and applied it to learning of MGs. Given a set of words $\mathcal{W} = \{W_1, \ldots, W_Q\}$, the goal is to obtain an SMG $\mathcal{G} = (\mathcal{A}, \mathcal{N}, \mathcal{R}, S, \phi)$ such that the probability $\prod_{q=1}^{Q} \mathit{Prob}_{\mathcal{G}}(W_q)$ is high. To this end, we apply the following procedure.

1. Fix $\mathcal{N}$, the set of non-terminals and their arities.
2. Let $\mathcal{R}$ be the set of all the possible rewriting rules (modulo certain constraints on the shape of rules, to ensure that the number of rules is finite; see below for details).
3. Randomly assign the initial probability to each rule.
4. Run the inside-outside algorithm (Algorithm 1, with the threshold $\epsilon = 10^{-5}$).
5. Remove the rules whose probabilities are 10^2 times smaller than the average of the probabilities of rules of the same non-terminal, and output the resulting grammar.

Since the inside-outside algorithm is not guaranteed to converge to the global maximum, we ran the steps 3-5 above ten times, and output the grammar with the largest value of $\prod_{q=1}^{Q} \mathit{Prob}_{\mathcal{G}}(W_q)$.

As the benchmark set, we picked the languages $L_{\mathtt{a}^{2^n}} = \{\mathtt{a}^{2^n} \mid n \geq 0\}$, $L_{ww} = \{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^+\}$, $L_{w\mathtt{a}w} = \{w\mathtt{a}w \mid w \in \{\mathtt{a}, \mathtt{b}\}^+\}$, $L_{\mathtt{a}ww} = \{\mathtt{a}ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^+\}$, and $L_{www} = \{www \mid w \in \{\mathtt{a}, \mathtt{b}\}^+\}$, and for each language, we set $\mathcal{W}$ to the subset of the language consisting of words of length up to 9. For example, $\mathcal{W}$ was set to $\{\mathtt{a}, \mathtt{aa}, \mathtt{a}^4, \mathtt{a}^8\}$ for $L_{\mathtt{a}^{2^n}}$. We fixed $\mathcal{N}$ to $\{S \mapsto 0, F \mapsto 1\}$.

To ensure that the number of rewriting rules is finite, we restricted the right-hand side of each rule to one of the following forms for certain constants c_1 and c_2:

(i) the concatenation $t_1 \cdot \ldots \cdot t_k$ of terms $t_1, \ldots, t_k$, where $k \leq c_1$, and at most one of $t_1, \ldots, t_k$ may be an application $F\, u_1\, \ldots\, u_\ell$ (with $u_1, \ldots, u_\ell$ are variables or terminals); all the other t_i's must be variables or terminals.

(ii) $F\, t_1\, \ldots\, t_k$, where the number of symbols occurring in the term is less than or equal to c_2.

In the experiments, c_1 was set to 2 for $L_{\mathtt{a}^{2^n}}$, and 3 for the other languages. We set $c_2 = 3$ for all the languages. For example, for $L_{\mathtt{a}^{2^n}}$, possible right-hand sides that match (i) are:

$$x,\ \mathtt{a},\ x \cdot x,\ x \cdot \mathtt{a},\ \mathtt{a} \cdot x,\ \mathtt{a} \cdot \mathtt{a},\ x \cdot F\, x,\ \mathtt{a} \cdot F\, x,\ x \cdot F\, \mathtt{a}, \ldots,$$

and those that match (ii) are:

$$F\, x,\ F\, \mathtt{a},\ F(x \cdot x),\ F(\mathtt{a} \cdot x),\ F(x \cdot \mathtt{a}),\ \ldots$$

These are ad hoc restrictions introduced to avoid an explosion of the number of rewriting rules; our algorithm itself works without these restrictions (as long as the number of rules is finite).

Table 1 shows the results of our experiments. The column "ideal grammars" shows (the rewriting rules of) a simple (expansive) MG for each language. The column "learned grammars" shows the grammar obtained by our procedure (with the probability of each rule being omitted). The P column gives the value of $\prod_{W \in \mathcal{W}} Prob_{\mathcal{G}}(W)$ for the learned grammar. The column "time" gives the total running time of ten runs, measured in seconds.

As seen in the table, our procedure returned correct grammars for all the five languages, although the learned grammars are a little more complex than ideal ones. Out of the ten runs of Steps 3–5 of the procedure, the grammars shown in the table were obtained seven times for L_{ww}, and ten times for other languages. Interestingly, we found that for all the five languages, the learned grammars have larger values of $\prod_{W \in \mathcal{W}} Prob_{\mathcal{G}}(W)$ than the ideal grammars (with optimal probability assignment). For example, for the language $L_{\mathtt{a}^{2^n}}$, the value of $\prod_{W \in \mathcal{W}} Prob_{\mathcal{G}}(W)$ for the ideal grammar is 1.6×10^{-3}, which is smaller than that of the learned grammar. Thus, the learned grammar may actually be preferable to the "ideal" grammar for the purpose of data compression based on stochastic grammars (Naganuma et al., 2020).

Table 1: The experimental results

L	ideal grammar	learned grammar	P	time
$L_{\mathtt{a}^{2^n}}$	$S \to \mathtt{a} \mid F\,\mathtt{a}$ $F\,x \to F\,(x \cdot x) \mid x \cdot x$	$S \to \mathtt{a} \mid \mathtt{a} \cdot \mathtt{a} \mid F\,\mathtt{a} \mid F\,(\mathtt{a} \cdot \mathtt{a})$ $F\,x \to F\,(x \cdot x) \mid x \cdot x$	2.3×10^{-3}	1.1×10^{4}
L_{ww}	$S \to F\,\mathtt{a} \mid F\,\mathtt{b}$ $F\,x \to F\,(x \cdot \mathtt{a})$ $\mid F\,(x \cdot \mathtt{b})$ $\mid x \cdot x$	$S \to \mathtt{a} \cdot \mathtt{a} \mid \mathtt{b} \cdot \mathtt{b} \mid F\,\mathtt{a} \mid F\,\mathtt{b} \mid F\,(\mathtt{a} \cdot \mathtt{a})$ $\mid F\,(\mathtt{a} \cdot \mathtt{b}) \mid F\,(\mathtt{b} \cdot \mathtt{a}) \mid F\,(\mathtt{b} \cdot \mathtt{b})$ $F\,x \to F\,(x \cdot \mathtt{a}) \mid F\,(x \cdot \mathtt{b})$ $\mid F\,(\mathtt{a} \cdot x) \mid F\,(\mathtt{b} \cdot x) \mid x \cdot x$	2.0×10^{-53}	1.0×10^{5}
$L_{w\mathtt{a}w}$	$S \to F\,\mathtt{a} \mid F\,\mathtt{b}$ $F\,x \to F\,(x \cdot \mathtt{a})$ $\mid F\,(x \cdot \mathtt{b})$ $\mid x \cdot \mathtt{a} \cdot x$	$S \to \mathtt{a} \cdot \mathtt{a} \cdot \mathtt{a} \mid \mathtt{b} \cdot \mathtt{a} \cdot \mathtt{b} \mid F\,\mathtt{a} \mid F\,\mathtt{b}$ $\mid F\,(\mathtt{a} \cdot \mathtt{a}) \mid F\,(\mathtt{a} \cdot \mathtt{b})$ $\mid F\,(\mathtt{b} \cdot \mathtt{a}) \mid F\,(\mathtt{b} \cdot \mathtt{b})$ $F\,x \to F\,(x \cdot \mathtt{a}) \mid F\,(x \cdot \mathtt{b})$ $\mid F\,(\mathtt{a} \cdot x) \mid F\,(\mathtt{b} \cdot x) \mid x \cdot \mathtt{a} \cdot x$	2.0×10^{-53}	1.9×10^{5}
$L_{\mathtt{a}ww}$	$S \to F\,\mathtt{a} \mid F\,\mathtt{b}$ $F\,x \to F\,(x \cdot \mathtt{a})$ $\mid F\,(x \cdot \mathtt{b})$ $\mid \mathtt{a} \cdot x \cdot x$	$S \to \mathtt{a} \cdot \mathtt{a} \cdot \mathtt{a} \mid \mathtt{a} \cdot \mathtt{b} \cdot \mathtt{b} \mid F\,\mathtt{a} \mid F\,\mathtt{b}$ $\mid F\,(\mathtt{a} \cdot \mathtt{a}) \mid F\,(\mathtt{a} \cdot \mathtt{b})$ $\mid F\,(\mathtt{b} \cdot \mathtt{a}) \mid F\,(\mathtt{b} \cdot \mathtt{b})$ $F\,x \to F\,(x \cdot \mathtt{a}) \mid F\,(x \cdot \mathtt{b})$ $\mid F\,(\mathtt{a} \cdot x) \mid F\,(\mathtt{b} \cdot x) \mid \mathtt{a} \cdot x \cdot x$	2.0×10^{-53}	1.9×10^{5}
L_{www}	$S \to F\,\mathtt{a} \mid F\,\mathtt{b}$ $F\,x \to F\,(x \cdot \mathtt{a})$ $\mid F\,(x \cdot \mathtt{b})$ $\mid x \cdot x \cdot x$	$S \to \mathtt{a} \cdot \mathtt{a} \cdot \mathtt{a} \mid \mathtt{b} \cdot \mathtt{b} \cdot \mathtt{b} \mid F\,\mathtt{a} \mid F\,\mathtt{b}$ $\mid F\,(\mathtt{a} \cdot \mathtt{a}) \mid F\,(\mathtt{a} \cdot \mathtt{b})$ $\mid F\,(\mathtt{b} \cdot \mathtt{a}) \mid F\,(\mathtt{b} \cdot \mathtt{b})$ $Fx \to F\,(x \cdot \mathtt{a}) \mid F\,(x \cdot \mathtt{b})$ $\mid F\,(\mathtt{a} \cdot x) \mid F\,(\mathtt{b} \cdot x) \mid x \cdot x \cdot x$	2.7×10^{-19}	1.4×10^{5}

We report the result for the language $L_{ww} = \{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^+\}$ in more detail. Here are the probabilities of rewriting rules obtained for one of the successful runs:

$$\begin{array}{lll}
\phi(S \to \mathtt{a} \cdot \mathtt{a}) = 0.026, & \phi(S \to \mathtt{b} \cdot \mathtt{b}) = 0.032, & \phi(S \to F\,\mathtt{a}) = 0.018, \\
\phi(S \to F\,\mathtt{b}) = 0.0045, & \phi(S \to F(\mathtt{a} \cdot \mathtt{a})) = 0.23, & \phi(S \to F(\mathtt{a} \cdot \mathtt{b})) = 0.23, \\
\phi(S \to F(\mathtt{b} \cdot \mathtt{a})) = 0.23, & \phi(S \to F(\mathtt{b} \cdot \mathtt{b})) = 0.23, & \\
\phi(Fx \to F(x \cdot \mathtt{a})) = 0.17, & \phi(Fx \to F(x \cdot \mathtt{b})) = 0.17, & \phi(Fx \to F(\mathtt{a} \cdot x)) = 0.13, \\
\phi(Fx \to F(\mathtt{b} \cdot x)) = 0.13, & \phi(Fx \to x \cdot x) = 0.41. &
\end{array}$$

5. Conclusions

We have developed an inside-outside algorithm for SMGs. The key idea was to consider type derivations (instead of word derivation sequences), which allowed us to naturally extend the inside-outside algorithm for stochastic CFGs. We have implemented the algorithm and applied it to the learning of MGs.

Future work includes an application of the algorithm to data compression based on stochastic grammars (Naganuma et al., 2020). The use of SMGs would allow us to compress data more compactly than stochastic CFGs. We also plan to extend the inside-outside algorithm for higher-order grammars (Damm, 1982; Kobele and Salvati, 2015) (where MGs correspond to order-2 OI grammars). We expect that our type-based approach can naturally be extended to deal with grammars of arbitrary order. In practice, the main bottleneck would be the explosion of the number of types to be considered. We plan to apply techniques for type-based higher-order model checking (Kobayashi, 2013b; Broadbent and Kobayashi, 2013) to cope with the problem.

Acknowledgments

We would like to thank anonymous referees for useful comments. This work was supported by JSPS KAKENHI Grant Number JP15H05706 and JP20H05703.

References

Kazuyuki Asada and Naoki Kobayashi. On word and frontier languages of unsafe higher-order grammars. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 111:1–111:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. ISBN 978-3-95977-013-2. doi: 10.4230/LIPIcs.ICALP.2016.111.

Kazuyuki Asada and Naoki Kobayashi. Pumping lemma for higher-order languages. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 97:1–97:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi: 10.4230/LIPIcs.ICALP.2017.97.

James K Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132, 1979.

Christopher H. Broadbent and Naoki Kobayashi. Saturation-based model checking of higher-order recursion schemes. In *CSL 2013*, volume 23 of *LIPIcs*, pages 129–148, 2013. doi: 10.4230/LIPIcs.CSL.2013.129.

Robert D. Cameron. Source encoding using syntactic information source models. *IEEE Trans. Inf. Theory*, 34(4):843–850, 1988. doi: 10.1109/18.9782.

Werner Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20:95–207, 1982.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1): 1–38, 1977.

L Dodd. Grammatical inference for automatic speech recognition: An application of the inside-outside algorithm to the spelling of english words. In *Proceedings of speech'88, 7th FASE Symposium*, pages 1061–1068, 1988.

Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996. URL https://www.aclweb.org/anthology/C96-1058.

Michael J. Fischer. Grammars with macro-like productions. In *9th Annual Symposium on Switching and Automata Theory (swat 1968)*, pages 131–142, 1968. doi: 10.1109/SWAT.1968.12.

Frederick Jelinek. Markov source modeling of text generation. In *The impact of processing techniques on communications*, pages 569–591. Springer, 1985.

Ryuta Kambe, Naoki Kobayashi, Ryosuke Sato, Ayumi Shinohara, and Ryo Yoshinaka. Inside-outside algorithm for macro grammars, 2021. An extended version, available at https://www.kb.is.s.u-tokyo.ac.jp/~koba/papers/icgi21-long.pdf.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. Rna pseudoknotted structure prediction using stochastic multiple context-free grammar. *IPSJ Digital Courier*, 2:655–664, 2006. doi: 10.2197/ipsjdc.2.655.

Naoki Kobayashi. Pumping by typing. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 398–407. IEEE Computer Society, 2013a. doi: 10.1109/LICS.2013.46.

Naoki Kobayashi. Model checking higher-order programs. *J. ACM*, 60(3), 2013b. doi: 10.1145/2487241.2487246.

Naoki Kobayashi and C.-H. Luke Ong. Complexity of model checking recursion schemes for fragments of the modal mu-calculus. *LMCS*, 7(4), 2011.

Naoki Kobayashi, Kazutaka Matsuda, Ayumi Shinohara, and Kazuya Yaguchi. Functional programs as compressed data. *Higher-Order and Symbolic Computation*, 2013.

Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Inf. Comput.*, 243:205–221, 2015.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007.

John D Lafferty. *A derivation of the inside-outside algorithm from the EM algorithm*. IBM TJ Watson Research Center, 2000.

Hiroaki Naganuma, Diptarama Hendrian, Ryo Yoshinaka, Ayumi Shinohara, and Naoki Kobayashi. Grammar compression with probabilistic context-free grammar. In Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer, editors, *Data Compression Conference, DCC 2020, Snowbird, UT, USA, March 24-27, 2020*, page 386. IEEE, 2020. doi: 10.1109/DCC47342.2020.00093.

C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS 2006*, pages 81–90. IEEE Computer Society Press, 2006. doi: 10.1109/LICS.2006.38.

Steven Ramsay, Robin Neatherway, and C.-H. Luke Ong. An abstraction refinement approach to higher-order model checking. In *POPL 2014*, pages 61–72. ACM, 2014.

Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, 1992.

Jorma Tarhio. On compression of parse trees. In Gonzalo Navarro, editor, *Eighth International Symposium on String Processing and Information Retrieval, SPIRE 2001, Laguna de San Rafael, Chile, November 13-15, 2001*, pages 205–211. IEEE Computer Society, 2001. doi: 10.1109/SPIRE.2001.989759.

C. F. Jeff Wu. On the Convergence Properties of the EM Algorithm. *The Annals of Statistics*, 11(1):95 – 103, 1983. doi: 10.1214/aos/1176346060.