EM-Training for Weighted Aligned Hypergraph Bimorphisms

Frank Drewes
Department of Computing Science
Umeå University
S-901 87 Umeå, Sweden
drewes@cs.umu.se

Kilian Gebhardt and **Heiko Vogler**
Department of Computer Science
Technische Universität Dresden
D-01062 Dresden, Germany
kilian.gebhardt@tu-dresden.de
heiko.vogler@tu-dresden.de

Abstract

We develop the concept of weighted aligned hypergraph bimorphism where the weights may, in particular, represent probabilities. Such a bimorphism consists of an $\mathbb{R}_{\geq 0}$-weighted regular tree grammar, two hypergraph algebras that interpret the generated trees, and a family of alignments between the two interpretations. Semantically, this yields a set of bihypergraphs each consisting of two hypergraphs and an explicit alignment between them; e.g., discontinuous phrase structures and non-projective dependency structures are bihypergraphs. We present an EM-training algorithm which takes a corpus of bihypergraphs and an aligned hypergraph bimorphism as input and generates a sequence of weight assignments which converges to a local maximum or saddle point of the likelihood function of the corpus.

1 Introduction

In natural language processing alignments play an important role. For instance, in machine translation they show up as hidden information when training probabilities of dictionaries (Brown et al., 1993) or when considering pairs of input/output sentences derived by a synchronous grammar (Lewis and Stearns, 1968; Chiang, 2007; Shieber and Schabes, 1990; Nederhof and Vogler, 2012). As another example, in language models for discontinuous phrase structures and non-projective dependency structures they can be used to capture the connection between the words in a natural language sentence and the corresponding nodes of the parse tree or dependency structure of that sentence.

In (Nederhof and Vogler, 2014) the generation of discontinuous phrase structures has been formalized by the new concept of hybrid grammar. Much as in the mentioned synchronous grammars, a hybrid grammar synchronizes the derivations of nonterminals of a string grammar, e.g., a linear context-free rewriting system (LCFRS) (Vijay-Shanker et al., 1987), and of nonterminals of a tree grammar, e.g., regular tree grammar (Brainerd, 1969) or simple definite-clause programs (sDCP) (Deransart and Małuszynski, 1985). Additionally it synchronizes terminal symbols, thereby establishing an explicit alignment between the positions of the string and the nodes of the tree. We note that LCFRS/sDCP hybrid grammars can also generate non-projective dependency structures.

In this paper we focus on the task of training an LCFRS/sDCP hybrid grammar, that is, assigning probabilities to its rules given a corpus of discontinuous phrase structures or non-projective dependency structures. Since the alignments are first class citizens, we develop our approach in the general framework of hypergraphs and hyperedge replacement (HR) (Habel, 1992). We define the concepts of *bihypergraph* (for short: bigraph) and *aligned HR bimorphism*. A bigraph consists of hypergraphs H_1, λ, and H_2, where λ represents the alignment between H_1 and H_2. A bimorphism $\mathcal{B} = (g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$ consists of a regular tree grammar g generating trees over some ranked alphabet Σ, two Σ-algebras $\mathcal{A}_1$ and $\mathcal{A}_2$ which interpret each symbol in Σ as an HR operation (thus evaluating every tree to two hypergraphs), and a Σ-indexed family Λ of alignments between the two interpretations of each $\sigma \in \Sigma$. The semantics of $\mathcal{B}$ is a set of bigraphs.

For instance, each discontinuous phrase structure or non-projective dependency structure can be represented as a bigraph (H_1, λ, H_2) where H_1 and H_2 correspond to the string component and the tree component, respectively. Fig. 1 shows an example of a bigraph representing a non-projective depen-

Proceedings of the ACL Workshop on Statistical NLP and Weighted Automata, pages 60–69,
Berlin, Germany, August 12, 2016.

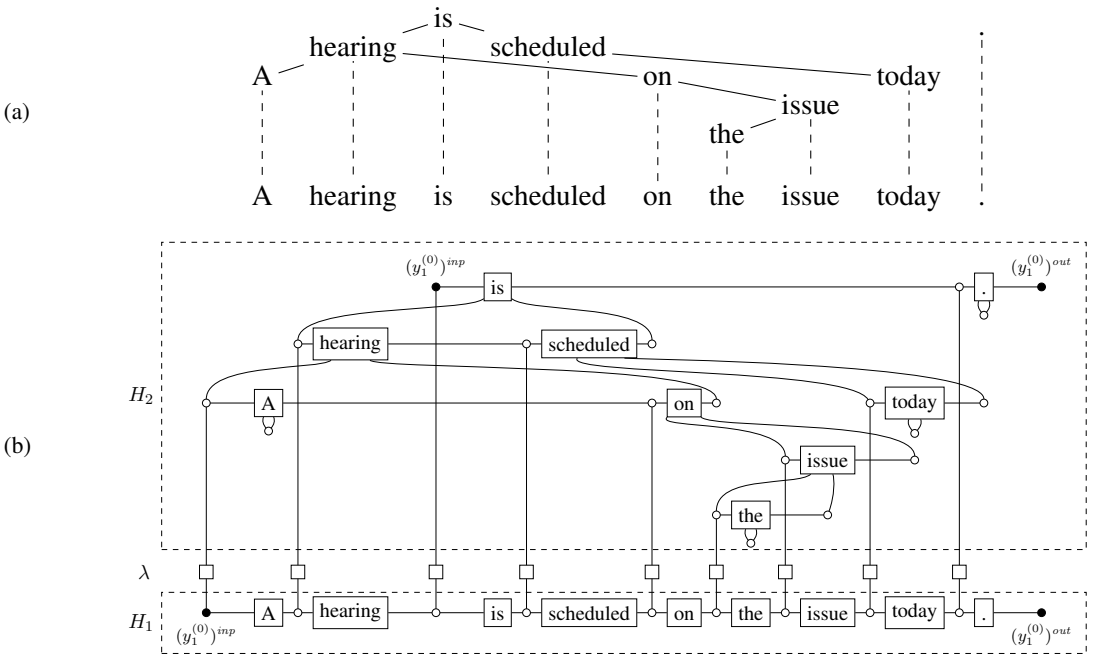


Figure 1: (a) A sentence with non-projective dependencies is represented in (b) by a bigraph (H_1, λ, H_2). Both hypergraphs H_1 and H_2 contain a distinct hyperedge (box) for each word of the sentence. H_1 specifies the linear order on the words. H_2 describes parent-child relationships between the words, where children form a list to whose start and end the parent has a tentacle. The alignment λ establishes a one-to-one correspondence between the (input vertices of the) hyperedges in H_1 and H_2.

dency structure. We present each LCFRS/sDCP hybrid grammar as a particular aligned HR bimorphism; this establishes an initial algebra semantics (Goguen et al., 1977) for hybrid grammars.

The flexibility of aligned HR bimorphisms goes well beyond hybrid grammars as they generalize the synchronous HR grammars of (Jones et al., 2012), making it possible to synchronously generate two graphs connected by explicit alignment structures. Thus, they can for instance model alignments involving directed acyclic graphs like Abstract Meaning Representations (Banarescu et al., 2013) or Millstream systems (Bensch et al., 2014).

Our training algorithm takes as input an aligned HR bimorphism $\mathcal{B} = (g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$ and a corpus c of bigraphs. It is based on the dynamic programming variant (Baker, 1979; Lari and Young, 1990; Prescher, 2001) of the EM-algorithm (Dempster et al., 1977) and thus approximates a local maximum or saddle point of the likelihood function of c.

In order to calculate the significance of each rule of g for the generation of a single bigraph (H_1, λ, H_2) occurring in c, we proceed as usual, constructing the reduct $\mathcal{B} \rhd (H_1, \lambda, H_2)$ which generates the singleton (H_1, λ, H_2) via the same derivation trees as $\mathcal{B}$ and preserves the probabilities. We show that the complexity of constructing the reduct is polynomial in the size of g and (H_1, λ, H_2) if $\mathcal{B}$ is an LCFRS/sDCP hybrid grammar. However, as the algorithm itself is not limited to this situation, we expect it to be useful in other cases as well.

2 Preliminaries

Basic mathematical notation We denote the set of natural numbers (including 0) by $\mathbb{N}$ and the set $\mathbb{N} \setminus \{0\}$ by $\mathbb{N}_+$. For $n \in \mathbb{N}$, we denote $\{1, \ldots, n\}$ by $[n]$. An alphabet A is a finite set of symbols. We denote the set of all strings over A by A^*, the empty string by ε, and $A^* \setminus \{\varepsilon\}$ by A^+. We denote the length of $s \in A^*$ by $|s|$ and, for each $i \in [|s|]$, the ith item in s by $s(i)$, i.e., s is identified with the function $s\colon [|s|] \to A$ such that $s = s(1) \cdots s(|s|)$. We denote the range $\{s(1), \ldots, s(|s|)\}$ of s by $[s]$. The powerset of a set A is denoted by $\mathcal{P}(A)$. The canonical extension of a function $f\colon A \to B$ to $f\colon \mathcal{P}(A) \to \mathcal{P}(B)$ and to $f\colon A^* \to B^*$ are defined as usual and denoted by f as well. We denote the restriction of $f\colon A \to B$ to $A' \subseteq A$ by $f|_{A'}$.

For an equivalence relation $\sim$ on B we denote the equivalence class of $b \in B$ by $[b]_\sim$ and the quotient of B modulo $\sim$ by $B/\!\sim$. For $f\colon A \to$

B we define the function $f/\sim\colon A \to B/\sim$ by $f/\sim(a) = [f(a)]_\sim$. In particular, for a string $s \in B^*$ we let $s/\sim = [s(1)]_\sim \cdots [s(|s|)]_\sim$.

Terms, regular tree grammars, and algebras A *ranked alphabet* is a pair (Σ, rk) where Σ is an alphabet and $\mathrm{rk}\colon \Sigma \to \mathbb{N}$ is a mapping associating a rank with each symbol of Σ. Often we just write Σ instead of (Σ, rk). We abbreviate $\mathrm{rk}^{-1}(k)$ by Σ_k. In the following let Σ be a ranked alphabet.

Let A be an arbitrary set. We let $\Sigma(A)$ denote the set of strings $\{\sigma(a_1, \ldots, a_k) \mid k \in \mathbb{N}, \sigma \in \Sigma_k, a_1, \ldots, a_k \in A\}$ (where the parentheses and commas are special symbols not in Σ). The *set of well-formed terms over Σ indexed by A*, denoted by $\mathrm{T}_\Sigma(A)$, is defined to be the smallest set T such that $A \subseteq T$ and $\Sigma(T) \subseteq T$. We abbreviate $\mathrm{T}_\Sigma(\emptyset)$ by T_Σ and write σ instead of $\sigma()$ for $\sigma \in \Sigma_0$.

A *regular tree grammar* (RTG)[1] (Gécseg and Steinby, 1984) is a tuple $g = (\Xi, \Sigma, \xi_0, R)$ where Ξ is an alphabet (nonterminals), $\Xi \cap \Sigma = \emptyset$, elements in Σ are called terminals, $\xi_0 \in \Xi$ (initial nonterminal), R is a ranked alphabet (rules); each rule in R_k has the form $\xi \to \sigma(\xi_1, \ldots, \xi_k)$ where $\xi, \xi_1, \ldots, \xi_k \in \Xi$, $\sigma \in \Sigma_k$. We denote the set of all rules with left-hand side ξ by R_ξ for each $\xi \in \Xi$.

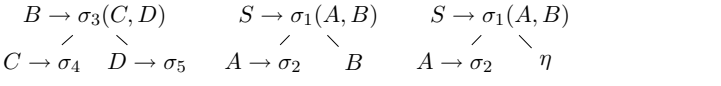
Since RTGs are particular context-free grammars, the concepts of derivation relation and generated language are inherited. The *language of the RTG g* is the set of all well-formed terms in T_Σ generated by g; this language is denoted by $\mathcal{L}(g)$.

We define the Ξ-indexed family $(D_g^\xi \mid \xi \in \Xi)$ of mappings $D_g^\xi\colon \mathrm{T}_\Sigma \to \mathcal{P}(\mathrm{T}_R)$; for each term $t \in \mathrm{T}_\Sigma$, $D_g^\xi(t)$ is the set of t's *derivation trees* in T_R which start with ξ and yield t. Formally, for each $\xi \in \Xi$ and $\sigma(t_1, \ldots, t_k) \in \mathrm{T}_\Sigma$, the set $D_g^\xi(\sigma(t_1, \ldots, t_k))$ contains each term $\varrho(d_1, \ldots, d_k)$ where $\varrho = (\xi \to \sigma(\xi_1, \ldots, \xi_k))$ is in R and $d_i \in D_g^{\xi_i}(t_i)$ for each $i \in [k]$. We define $D_g(t) = \bigcup_{\xi \in \Xi} D_g^\xi(t)$ and $D_g^\xi(\mathrm{T}_\Sigma) = \bigcup_{t \in \mathrm{T}_\Sigma} D_g^\xi(t)$. Finally, $D_g^{\xi_0}(\mathrm{T}_\Sigma, \xi)$ is the set of all $\zeta \in \mathrm{T}_R(\{\xi\})$ such that there is a $\zeta' \in D_g^{\xi_0}(\mathrm{T}_\Sigma)$ which has a subtree whose root is in R_ξ, and ζ is obtained from ζ' by replacing exactly one of these subtrees by ξ.

Example 2.1. Let $\Sigma = \Sigma_0 \cup \Sigma_2$ where $\Sigma_0 = \{\sigma_2, \sigma_4, \sigma_5\}$ and $\Sigma_2 = \{\sigma_1, \sigma_3\}$. Let g be an RTG with initial nonterminal S and the following rules:

$$S \to \sigma_1(A, B) \qquad A \to \sigma_2$$
$$B \to \sigma_3(C, D) \qquad C \to \sigma_4 \qquad D \to \sigma_5$$

We observe that $S \Rightarrow_g^* \sigma_1(\sigma_2, \sigma_3(\sigma_4, \sigma_5))$. Let η, ζ, and ζ' be the following trees (in order):



Then $\zeta \in D_g^S(\mathrm{T}_\Sigma, B)$ because $\zeta' \in D_g^S(\mathrm{T}_\Sigma)$ and the left-hand side of the root of η is B. □

A *Σ-algebra* is a pair $\mathcal{A} = (A, (\sigma_\mathcal{A} \mid \sigma \in \Sigma))$ where A is a set and $\sigma_\mathcal{A}$ is a k-ary operation on A for every $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$. As usual, we will sometimes use $\mathcal{A}$ to refer to its carrier set A or, conversely, denote $\mathcal{A}$ by A (and thus $\sigma_\mathcal{A}$ by σ_A) if there is no risk of confusion. The *Σ-term algebra* is the Σ-algebra T_Σ with $\sigma_{\mathrm{T}_\Sigma}(t_1, \ldots, t_k) = \sigma(t_1, \ldots, t_k)$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$. For each Σ-algebra $\mathcal{A}$ there is exactly one Σ-homomorphism, denoted by $[\![.]\!]_\mathcal{A}$, from the Σ-term algebra to $\mathcal{A}$ (Wechler, 1992).

Hypergraphs and hyperedge replacement In the following let Γ be a finite set of *labels*. A *Γ-hypergraph* is a tuple $H = (V, E, \mathrm{att}, \mathrm{lab}, \mathrm{ports})$, where V is a finite set of *vertices*, E is a finite set of *hyperedges*, $\mathrm{att}\colon E \to V^* \setminus \{\varepsilon\}$ is the *attachment* of hyperedges to vertices, $\mathrm{lab}\colon E \to \Gamma$ is the *labeling* of hyperedges, and $\mathrm{ports} \in V^*$ is a sequence of (not necessarily distinct) *ports*. The set of all Γ-hypergraphs is denoted by $\mathcal{H}_\Gamma$. The vertices in $V \setminus [\mathrm{ports}]$ are also called *internal vertices* and denoted by $\mathrm{int}(H)$.

For the sake of brevity, we shall in the following simply call Γ-hypergraphs and hyperedges *graphs* and *edges*, respectively. We illustrate a graph in figures as follows (cf., e.g., $\mathrm{graph}((\sigma_2)_\mathcal{A})$ in Fig. 2a). A vertex v is illustrated by a circle, which is filled and labeled by i in case that $\mathrm{ports}(i) = v$. An edge e with label γ and $\mathrm{att}(e) = v_1 \ldots v_n$ is depicted as a γ-labeled rectangle with n *tentacles*, lines pointing to $v_1, \ldots, v_n$ which are annotated by $1, \ldots, n$. (We sometimes drop these annotations.)

If we are not interested in the particular set of labels Γ, then we also call a Γ-graph simply *graph* and write $\mathcal{H}$ instead of $\mathcal{H}_\Gamma$. In the following, we will refer to the components of a graph H by indexing them with H unless they are explicitly named.

Let H and H' be graphs. H and H' are *disjoint* if $V_H \cap V_{H'} = \emptyset$ and $E_H \cap E_{H'} = \emptyset$.

[1] in this context we use "tree" and "term" as synonyms

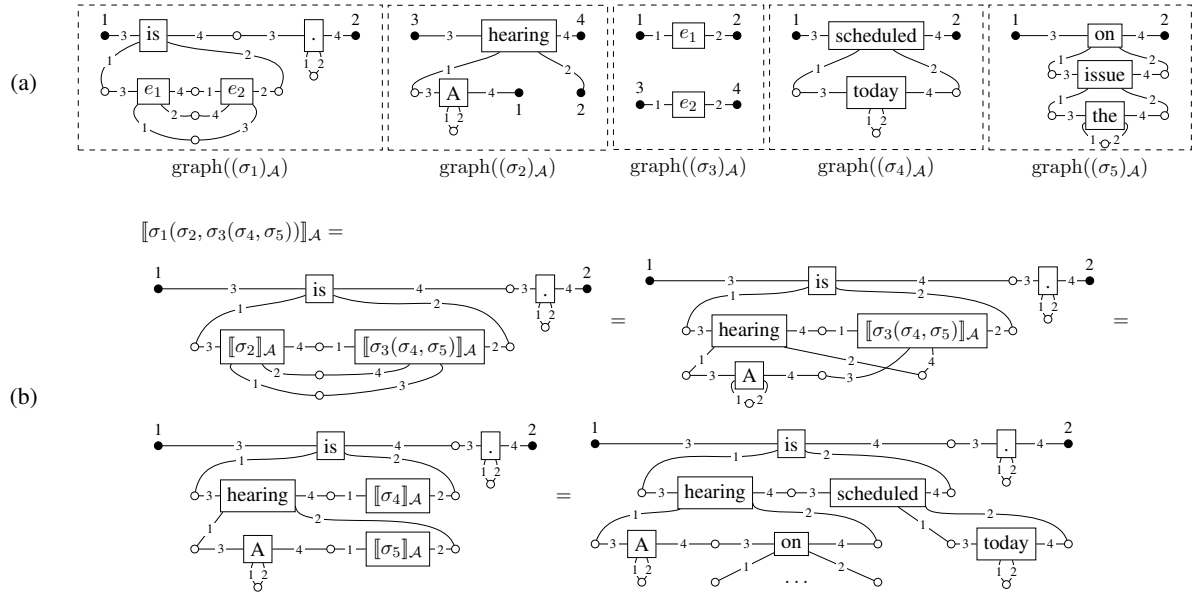


Figure 2: (a) The (Σ, Γ)-HR algebra $\mathcal{A}$ and (b) the evaluation of the term $\sigma_1(\sigma_2, \sigma_3(\sigma_4, \sigma_5))$ in $\mathcal{A}$.

Let $E = E_H \cap E_{H'}$. If $\mathrm{att}_H|_E = \mathrm{att}_{H'}|_E$ and $\mathrm{lab}_H|_E = \mathrm{lab}_{H'}|_E$, then the *union* of the graphs H and H' is the graph $H \cup H' = (V_H \cup V_{H'}, E_H \cup E_{H'}, \mathrm{att}_H \cup \mathrm{att}_{H'}, \mathrm{lab}_H \cup \mathrm{lab}_{H'}, \mathrm{ports}_H \mathrm{ports}_{H'})$.

For every $F \subseteq E_H$ let $H \setminus F = (V_H, E, \mathrm{att}_H|_E, \mathrm{lab}_H|_E, \mathrm{ports}_H)$ where $E = E_H \setminus F$.

Let $k \in \mathbb{N}_+$ and $H, H_1, \ldots, H_k \in \mathcal{H}$ be pairwise disjoint. Let $e_1, \ldots, e_k \in E_H$ be pairwise distinct edges, called *variables*. Let I be the graph $H \setminus \{e_1, \ldots, e_k\} \cup H_1 \cup \cdots \cup H_k$. The *hyperedge replacement* (HR) *of e_1 by H_1, …, and e_k by H_k in H* (Bauderon and Courcelle, 1987; Habel and Kreowski, 1987) yields the graph

$$H[e_1/H_1, \ldots, e_k/H_k] = (V_I/\!\sim, E_I, \mathrm{att}_I/\!\sim, \mathrm{lab}_I, \mathrm{ports}_H/\!\sim)$$

where $\sim$ is the least equivalence relation on V_I such that $\mathrm{att}_H(e_i, j) \sim \mathrm{ports}_{H_i}(j)$ for every $i \in [k]$ and $j \in [\min(|\mathrm{att}_H(e_i)|, |\mathrm{ports}_{H_i}|)]$. In the following, we call $\sim$ the *equivalence relation involved in the HR that yields $H[e_1/H_1, \ldots, e_k/H_k]$*.

We assume that each variable e_i is labeled by a distinguished symbol $\perp \in \Sigma$ and depict e_i by $\boxed{e_i}$ instead of $\boxed{\perp}$. Throughout this paper, we will not distinguish between isomorphic graphs, i.e., graphs that are identical up to a bijective renaming of vertices and edges. However, since hyperedge replacement is defined on concrete graphs and requires that $H, H_1, \ldots, H_k$ are pairwise disjoint, we may choose isomorphic copies of the involved graphs, i.e., rename edges or vertices. To avoid the cumbersome conversion between abstract and concrete graphs, we assume that this renaming is *opaque*. In this sense, we may define an HR operation as a total function from $\mathcal{H}^k$ to $\mathcal{H}$ as follows.

Let H be a graph. For pairwise distinct edges $e_1, \ldots, e_k \in E_H$, the *HR operation* $H_{\langle e_1 \ldots e_k \rangle} \colon \mathcal{H}^k \to \mathcal{H}$ is given by

$$H_{\langle e_1 \ldots e_k \rangle}(H_1, \ldots, H_k) = H[e_1/H_1, \ldots, e_k/H_k]$$

for all graphs $H_1, \ldots, H_k \in \mathcal{H}$.

A *(Σ, Γ)-HR algebra* (Courcelle, 1991) is a Σ-algebra $\mathcal{A} = (\mathcal{H}_\Gamma, (\sigma_\mathcal{A})_{\sigma \in \Sigma})$ where, for every $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$, we have $\sigma_\mathcal{A} = H_{\langle e_1 \ldots e_k \rangle}$ for some $H \in \mathcal{H}_\Gamma$ and pairwise distinct $e_1, \ldots, e_k \in E_H$. Then we denote H by $\mathrm{graph}(\sigma_\mathcal{A})$. An *HR algebra* is a (Σ, Γ)-HR algebra for some Σ and Γ.

An example of a (Σ, Γ)-HR algebra $\mathcal{A}$ and the application of $[\![.]\!]_\mathcal{A}$ to a term are given in Fig. 2.

3 Bigraphs and Aligned HR Bimorphisms

Now we formally introduce our central notions of bigraph and aligned HR bimorphism. A case study with examples follows in the next section. Throughout this section let Δ and Ω be alphabets.

Definition 3.1. A *bigraph of type* (Δ, Ω) is a triple $B = (H_1, \lambda, H_2)$ where $H_1 \in \mathcal{H}_\Delta$ and $H_2 \in \mathcal{H}_\Omega$ are disjoint, and λ is an *alignment of H_1 and H_2*, i.e., a graph with $V_\lambda = V_{H_1} \cup V_{H_2}$, $E_\lambda \cap E_{H_1} = E_\lambda \cap E_{H_2} = \emptyset$, $\mathrm{att}(e) \in (V_{H_1})^+ \cdot (V_{H_2})^+$ for each $e \in E_\lambda$, and $\mathrm{ports}_\lambda = \varepsilon$. □

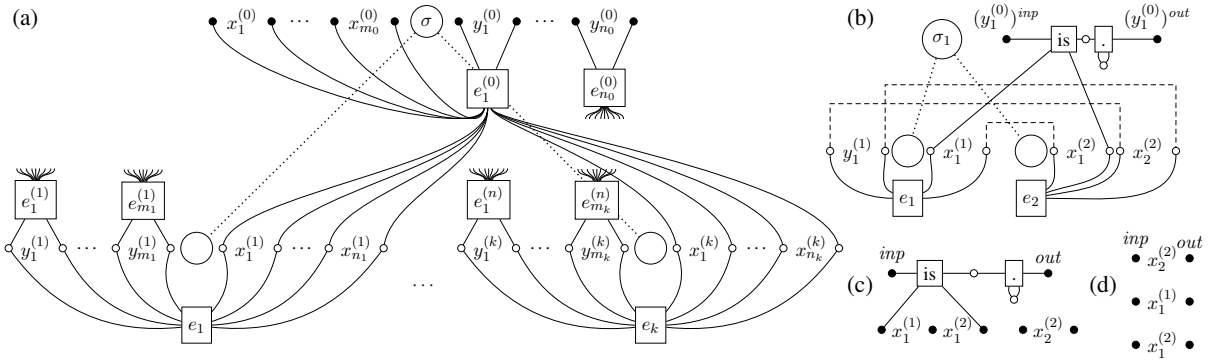


Figure 3: The graphs (a) H^{σ}_{IO}, (b) H^{σ_1}, (c) $(\!|s_1^{(0)}|\!)$, and (d) $(\!|s_1^{(1)}|\!)$. The large circles and the dotted lines in (a) and (b) visualize the underlying term structure; e.g., in (b) σ_1 has two children because $\sigma_1 \in \Sigma_2$.

Definition 3.2. An *aligned HR bimorphism of type* (Σ, Δ, Ω) is a tuple $\mathcal{B} = (g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$, where g is an RTG over Σ and $\mathcal{A}_1, \mathcal{A}_2$ are a (Σ, Δ)-HR algebra and a (Σ, Ω)-HR algebra, resp., such that $\mathrm{graph}(\sigma_{\mathcal{A}_1})$ and $\mathrm{graph}(\sigma_{\mathcal{A}_2})$ are disjoint for each $\sigma \in \Sigma$. Further, Λ is a Σ-indexed family $(\Lambda_\sigma \mid \sigma \in \Sigma)$, each Λ_σ being an alignment of $\mathrm{graph}(\sigma_{\mathcal{A}_1})$ and $\mathrm{graph}(\sigma_{\mathcal{A}_2})$. □

In the following, for each term $t \in \mathrm{T}_\Sigma$, we assume (w.l.o.g.) that $[\![t]\!]_{\mathcal{A}_1}$ and $[\![t]\!]_{\mathcal{A}_2}$ are disjoint.

Definition 3.3. Let $\mathcal{B} = (g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$ be an aligned HR bimorphism. The *semantics of* $\mathcal{B}$, denoted by $\mathcal{L}(\mathcal{B})$, is the set of bigraphs defined as follows.

First, let the *$\mathcal{B}$-alignment* be the T_Σ-indexed family $\Lambda_\mathcal{B} = (\Lambda_\mathcal{B}(t) \mid t \in \mathrm{T}_\Sigma)$ where $\Lambda_\mathcal{B}(t)$ is the alignment of $[\![t]\!]_{\mathcal{A}_1}$ and $[\![t]\!]_{\mathcal{A}_2}$ defined inductively on t as follows. Let $t = \sigma(t_1, \ldots, t_k) \in \mathrm{T}_\Sigma$. For $j \in [2]$, suppose that $\sim_j$ is the equivalence relation involved in the hyperedge replacement that yields $\mathrm{graph}(\sigma_{\mathcal{A}_j})[e_1/[\![t_1]\!]_{\mathcal{A}_j}, \ldots, e_k/[\![t_k]\!]_{\mathcal{A}_j}]$. We assume that $\Lambda_\sigma, \Lambda_\mathcal{B}(t_1), \ldots, \Lambda_\mathcal{B}(t_k)$ have pairwise disjoint sets of edges. Then we define the $\mathcal{B}$-alignment of $[\![t]\!]_{\mathcal{A}_1}$ and $[\![t]\!]_{\mathcal{A}_2}$ to be the graph

$$\Lambda_\mathcal{B}(t) = (\Lambda_\sigma \cup \Lambda_\mathcal{B}(t_1) \cup \cdots \cup \Lambda_\mathcal{B}(t_k))/(\sim_1 \cup \sim_2)$$

and we let $[\![t]\!]_\mathcal{B} = ([\![t]\!]_{\mathcal{A}_1}, \Lambda_\mathcal{B}(t), [\![t]\!]_{\mathcal{A}_2})$. Finally, we define $\mathcal{L}(\mathcal{B}) = \{[\![t]\!]_\mathcal{B} \mid t \in \mathcal{L}(g)\}$. □

4 Case Study: Hybrid Grammars

We show how an LCFRS/sDCP hybrid grammar (Nederhof and Vogler, 2014) can be represented as an aligned HR bimorphism. These grammars deal with sequence terms; hence, we first recall their definition and show how to view sequence terms as particular graphs.

4.1 A Graph View on Sequence Terms

Let Γ be a ranked alphabet and Y be a set disjoint from Γ. The sets of *terms* and *sequence-terms* (*s-terms*) over Γ indexed by Y (Seki and Kato, 2008) are denoted by $\mathcal{T}_\Gamma(Y)$ and $\mathcal{T}^*_\Gamma(Y)$, respectively, and defined inductively as follows:

1. $Y \subseteq \mathcal{T}_\Gamma(Y)$,
2. if $k \in \mathbb{N}$, $\gamma \in \Gamma_k$ and $s_i \in \mathcal{T}^*_\Gamma(Y)$ for each $i \in [k]$, then $\gamma(s_1, \ldots, s_k) \in \mathcal{T}_\Gamma(Y)$, and
3. if $n \in \mathbb{N}$ and $t_i \in \mathcal{T}_\Gamma(Y)$ for each $i \in [n]$, then $\langle t_1, \ldots, t_n \rangle \in \mathcal{T}^*_\Gamma(Y)$.

Let $s \in \mathcal{T}^*_\Gamma(Y)$. We say that s is *linear* if every $y \in Y$ occurs at most once in s. *In the following we only consider linear s-terms.* We note that, if $\Gamma = \Gamma_0$, then s is essentially a string over Γ_0 and Y. If $\Gamma = \Gamma_1$, then s corresponds to a sequence of ordinary (unranked) terms over Γ_1 indexed by Y.

Every linear s-term s can be represented as a graph $(\!|s|\!)$: it has two distinct ports inp and out, representing the start and end of s, resp. For each variable $y \in Y$, $(\!|s|\!)$ has two distinct ports y^{inp} and y^{out}. For each occurrence of a symbol $\gamma \in \Gamma_k$ in s, there is a γ-labeled edge with $2k + 2$ tentacles in $(\!|s|\!)$. The $(2i-1)$-th and $2i$-th tentacle ($i \in [k]$) point to the start and end vertex, respectively, of the i-th child sequence of γ. The last two tentacles point towards the predecessor and the successor of γ, respectively: this may be a vertex separating two symbols, the start or end vertex of a (sub-)sequence, or the port realizing y^{inp} or y^{out} for some $y \in Y$.

For instance, the s-term

$$s_1^{(0)} = \langle \mathrm{is}(\langle x_1^{(1)}, x_1^{(2)} \rangle), .(\langle\rangle) \rangle$$

in $\mathcal{T}^*_\Gamma(\{x_1^{(1)}, x_1^{(2)}, x_2^{(2)}\})$ is represented by $(\!|s_1^{(0)}|\!)$ in Fig. 3c. The ports $(x_2^{(2)})^{inp}$ and $(x_2^{(2)})^{out}$ for $x_2^{(2)}$

are depicted as filled circles to the left and the right of $x_2^{(2)}$, and similarly for the other variables. Note that $(x_1^{(1)})^{out}$ and $(x_1^{(2)})^{inp}$ coincide because $x_1^{(1)}$ is succeeded by $x_1^{(2)}$ in $s_1^{(0)}$.

4.2 LCFRS, sDCP, and LCFRS/sDCP Hybrid Grammars

Here we formalize LCFRS/sDCP hybrid grammars as particular aligned HR bimorphisms, where the algebras $\mathcal{A}_1$ and $\mathcal{A}_2$ are an LCFRS algebra and an sDCP algebra, resp. Since both LCFRS and sDCP can be viewed as particular types of attribute grammars (AG), we first define the concept of AG algebra and, in a second step, instantiate it to LCFRS algebra and sDCP algebra.

For each $\sigma \in \Sigma_k$, let $\mathrm{syn}^\sigma_\mathcal{A} = (n_0, \ldots, n_k) \in {\mathbb{N}_+}^{k+1}$ and $\mathrm{inh}^\sigma_\mathcal{A} = (m_0, \ldots, m_k) \in \mathbb{N}^{k+1}$ be tuples defining the sets $I = \{y_j^{(0)} \mid j \in [n_0]\} \cup \{y_j^{(i)} \mid i \in [k], j \in [m_i]\}$ and $O = \{x_r^{(0)} \mid r \in [m_0]\} \cup \{x_r^{(\ell)} \mid \ell \in [k], r \in [n_\ell]\}$ of *inside* and *outside attributes*, resp. (The abbreviations stem from the AG notions *synthesized attributes* and *inherited attributes*.) The definition of an AG algebra $\mathcal{A}$ follows the two-phase approach in (Engelfriet and Heyker, 1992). In the first phase, for each symbol σ in Σ_k, we define a graph H^σ_{IO} *of type* $\mathrm{syn}^\sigma_\mathcal{A}, \mathrm{inh}^\sigma_\mathcal{A}$ as shown in Fig. 3a: there is a pair of vertices for each inside attribute $y_j^{(i)}$ and each outside attribute $x_r^{(\ell)}$. For each inside attribute $y_j^{(i)}$ there is a edge $e_j^{(i)}$ which connects $y_j^{(i)}$ with all outside attributes. For the edge $e_1^{(0)}$ the tentacles are shown completely in Fig. 3a; for the other edges the tentacles to outside attributes are abridged for clarity. The edges $e_1, \ldots, e_k$ correspond to the k successors of σ.

In the second phase, we choose an I-indexed family of s-terms $(s_j^{(i)} \in \mathcal{T}^*_\Gamma(O) \mid y_j^{(i)} \in I)$ such that each $x_r^{(\ell)}$ in O occurs exactly once in all $s_j^{(i)}$ together (single syntactic use restriction). Then we replace each edge $e_j^{(i)}$ by the graph $(\!|s_j^{(i)}|\!)$; this specifies a particular information flow. Formally, we set $\sigma_\mathcal{A} = H^\sigma_{\langle e_1 \ldots e_k \rangle}$ where

$$H^\sigma = H^\sigma_{IO}[e_j^{(i)}/(\!|s_j^{(i)}|\!) \mid y_j^{(i)} \in I] \ .$$

For instance, for $\sigma_1 \in \Sigma_2$ we have $\mathrm{syn}^{\sigma_1}_\mathcal{A} = (1,1,2)$ and $\mathrm{inh}^{\sigma_1}_\mathcal{A} = (0,1,0)$, and so we construct $H^{\sigma_1}_{IO}$ accordingly. Next, we choose $(\!|s_1^{(0)}|\!)$ and $(\!|s_1^{(1)}|\!)$ as depicted in Fig. 3c and 3d, respectively. Then $H^{\sigma_1} = H^{\sigma_1}_{IO}[e_1^{(0)}/(\!|s_1^{(0)}|\!), e_1^{(1)}/(\!|s_1^{(1)}|\!)]$ is the graph in Fig. 3b where dashed lines indicate the identification of vertices. Note that H^{σ_1} equals $\mathrm{graph}((\sigma_1)_\mathcal{A})$ in Fig. 2a.

A (Σ, Γ)-HR algebra $\mathcal{A}$ is a (Σ, Γ)-*attribute grammar algebra* ((Σ, Γ)-AG algebra), if each symbol σ in Σ is interpreted as described above. For instance, $\mathcal{A}$ of Fig. 2a is a (Σ, Γ)-AG algebra.

We observe that (Σ, Γ)-AG algebras have the following property: for every edge $e \in E_{H^\sigma}$, if $\mathrm{lab}_{H^\sigma}(e) \in \Gamma_k$ then e has $2k+2$ tentacles. We call the vertex $\mathrm{att}_{H_\sigma}(e)(k+1)$ the *input vertex of* e and denote it by $\mathrm{inp}(e)$. Note that no two terminal edges in H^σ have the same input vertex. This *single-input property* will be crucial for an efficient representation of subgraphs during the reduct construction (cf. Sec. 5.2).

Next we instantiate the concept of AG-algebra to LCFRS algebras and to sDCP algebras. An LCFRS does not have inherited attributes:

Definition 4.1. Let $\Delta = \Delta_0$ be a ranked alphabet. A (Σ, Δ)-AG algebra $\mathcal{A}$ is a (Σ, Δ)-*LCFRS algebra*, if $\mathrm{inh}^\sigma_\mathcal{A} = (0, \ldots, 0)$ for all $\sigma \in \Sigma$. □

Definition 4.2. Let $\Omega = \Omega_1$ be a ranked alphabet and let $\mathcal{A}$ be a (Σ, Ω)-AG algebra. We say that $\mathcal{A}$ is a (Σ, Ω)-*sDCP algebra*. □

Then the graph view on an LCFRS/sDCP hybrid grammar is an HR bimorphism $\mathcal{B} = (g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$, where

- $g = (\Xi, \Sigma, \xi_0, R)$ is an RTG,
- $\mathcal{A}_1$ is a (Σ, Δ)-LCFRS algebra,
- $\mathcal{A}_2$ is a (Σ, Ω)-sDCP algebra, $\Delta = \Omega$ (regarding Δ and Ω as sets of symbols), and
- there are functions $\mathrm{fan}\colon \Xi \to \mathbb{N}_+$, $\mathrm{inh}\colon \Xi \to \mathbb{N}$, and $\mathrm{syn}\colon \Xi \to \mathbb{N}_+$ such that $\mathrm{fan}(\xi_0) = 1$, $\mathrm{inh}(\xi_0) = 0$, $\mathrm{syn}(\xi_0) = 1$, and for every $(\xi \to \sigma(\xi_1, \ldots, \xi_k)) \in R$, it holds that
  - $(\mathrm{fan}(\xi), \mathrm{fan}(\xi_1), \ldots, \mathrm{fan}(\xi_k)) = \mathrm{syn}^\sigma_{\mathcal{A}_1}$ and
  - $(\mathrm{inh}(\xi), \mathrm{inh}(\xi_1), \ldots, \mathrm{inh}(\xi_k)) = \mathrm{inh}^\sigma_{\mathcal{A}_2}$ and $(\mathrm{syn}(\xi), \mathrm{syn}(\xi_1), \ldots, \mathrm{syn}(\xi_k)) = \mathrm{syn}^\sigma_{\mathcal{A}_2}$.

Moreover, we require the following: Let $\sigma \in \Sigma$ and $H_j = \mathrm{graph}(\sigma_{\mathcal{A}_j})$ for $j \in [2]$. For each $e \in E_{\Lambda_\sigma}$ we have $\mathrm{att}_{\Lambda_\sigma}(e) = \mathrm{inp}(e_1)\,\mathrm{inp}(e_2)$ where $e_1 \in E_{H_1}$, $e_2 \in E_{H_2}$, and $\mathrm{lab}_{H_1}(e_1) = \mathrm{lab}_{H_2}(e_2)$.

Example 4.3. Let g be as in Ex. 2.1 and consider the LCFRS/sDCP hybrid grammar $\mathcal{B} = (g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$, where $\mathcal{A}_1$, Λ, and $\mathcal{A}_2$ are as specified in Fig. 4. Then the bigraph in Fig. 1b equals $[\![\sigma_1(\sigma_2, \sigma_3(\sigma_4, \sigma_5))]\!]_\mathcal{B}$ and is thus in $\mathcal{L}(\mathcal{B})$. □

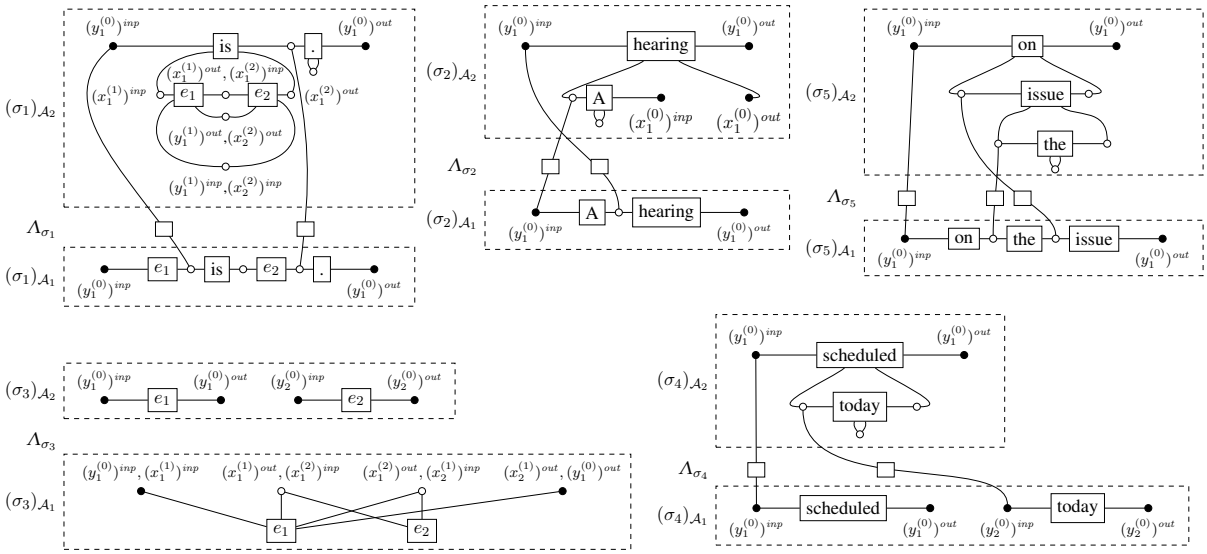


Figure 4: The interpretation of $\sigma_1, \ldots, \sigma_5$ in $\mathcal{A}_1$, Λ, and $\mathcal{A}_2$.

5 EM Training

We present a training algorithm which takes as input a weighted aligned HR bimorphism and a finite, non-empty corpus c of bigraphs. It is essentially the same as the training algorithm for probabilistic context-free grammars (PCFG) (Baker, 1979; Lari and Young, 1990; Nederhof and Satta, 2008). As shown in (Prescher, 2001), this algorithm is a dynamic programming variant of the EM-algorithm (Dempster et al., 1977). Thus, our algorithm generates a sequence of probability assignments which converges to a probability assignment $\hat{p}$; the likelihood of c under $\hat{p}$ is a local maximum or saddle point of the likelihood function of c.

5.1 Weighted Aligned HR Bimorphisms

We define weighted RTG in a similar way as PCFG was defined in (Nederhof and Satta, 2006).

A *weighted regular tree grammar (WRTG)* is a pair (g, p) where $g = (\Xi, \Sigma, \xi_0, R)$ is an RTG and $p\colon R \to \mathbb{R}_{\geq 0}$ is the *weight assignment*. A weight assignment p is a *probability assignment* if $\sum_{\rho \in R_\xi} p(\rho) = 1$ for each $\xi \in \Xi$. We extend p to the mapping $p'\colon D_g(\mathrm{T}_\Sigma) \to \mathbb{R}_{\geq 0}$ on derivations as follows: for each $d = \varrho(d_1, \ldots, d_k)$ in $D_g(\mathrm{T}_\Sigma)$ we define $p'(d) = p(\varrho) \cdot \prod_{i=1}^{k} p'(d_i)$. For each $t \in T_\Sigma$ we define $p''(t) = \sum_{d \in D_g^\xi(t)} p'(d)$. We define the mappings $\mathrm{in}\colon \Xi \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ (*inside weight*) and $\mathrm{out}\colon \Xi \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ (*outside weight*) for each $\xi \in \Xi$ by

$$\mathrm{in}(\xi) = \sum_{d \in D_g^\xi(\mathrm{T}_\Sigma)} p'(d) \qquad \mathrm{out}(\xi) = \sum_{d \in D_g^{\xi_0}(\mathrm{T}_\Sigma, \xi)} p'''(d)$$

where $p'''\colon D_g^{\xi_0}(\mathrm{T}_\Sigma, \xi) \to \mathbb{R}_{\geq 0}$ is defined in the same way as p', with the addition that $p'''(\xi) = 1$. As usual, we will drop the primes from p', p'', and p'''.

Definition 5.1. A *weighted aligned HR bimorphism* is a pair $(\mathcal{B}, p) = ((g, \mathcal{A}_1, \Lambda, \mathcal{A}_2), p)$ where $(g, \mathcal{A}_1, \Lambda, \mathcal{A}_2)$ is an aligned HR bimorphism and (g, p) is a WRTG. □

5.2 Reduct Construction

Given a weighted aligned HR bimorphism $(\mathcal{B}, p) = ((g, \mathcal{A}_1, \Lambda, \mathcal{A}_2), p)$ and a bigraph (H_1, λ, H_2), we restrict g to an RTG g' such that only trees $t \in \mathcal{L}(g)$ satisfying $[\![t]\!]_\mathcal{B} = (H_1, \lambda, H_2)$ are in $\mathcal{L}(g')$. Also, we show that if $\mathcal{B}$ is an LCFRS/sDCP hybrid grammar, then g' can be constructed in time polynomial in the size of $\mathcal{B}$ and (H_1, λ, H_2).

Definition 5.2. Let $m \in \mathbb{N}$ and $H \in \mathcal{H}$. A graph H' is an *m-subgraph of H* if $\mathrm{int}(H') \subseteq \mathrm{int}(H)$, $E_{H'} \subseteq E_H$, $\mathrm{lab}_{H'} = \mathrm{lab}_H|_{E_{H'}}$, and $|\mathrm{ports}_{H'}| \leq m$. Moreover, we require that there is a mapping $\varphi\colon V_{H'} \to V_H$, called *vertex mapping*, such that $\varphi(\mathrm{att}_{H'}(e)) = \mathrm{att}_H(e)$ for each $e \in E_H$, $\varphi(v) = v$ for each $v \in \mathrm{int}(H')$, and $\varphi(\mathrm{int}(H')) \cap \varphi([\mathrm{ports}_{H'}]) = \emptyset$. Moreover, for each $v \in \mathrm{int}(H')$, if $v \in [\mathrm{att}_H(e)]$ for some $e \in E_H$, then $e \in E_{H'}$. The set of all *m-subgraphs of H* is denoted by $\mathcal{H}_S^m(H)$. □

For instance, $\mathrm{graph}((\sigma_4)_\mathcal{A})$ in Fig. 2a is a 2-subgraph of the last graph in Fig. 2b. If a graph H is the result of applying an HR operation to graphs $H_1, \ldots, H_k$, then each H_i is a $|\mathrm{ports}_{H_i}|$-subgraph of H. (For this, the mapping φ in Definition 5.2 is

needed, because some of the ports of H_i may be identified with each other in H.) Hence, for the reduct we consider only m-subgraphs of H, where m is the maximal port length of HR operations in $\mathcal{A}_1$ or $\mathcal{A}_2$. We observe that $\mathcal{H}^m_{\mathrm{S}}(H)$ is finite because we identify isomorphic graphs.

Definition 5.3. Let $(\mathcal{B}, p) = ((g, \mathcal{A}_1, \Lambda, \mathcal{A}_2), p)$ be a weighted aligned HR bimorphism with $g = (\Xi, \Sigma, \xi_0, R)$ and let (H_1, λ, H_2) be a bigraph.

We define $(\mathcal{B}, p) \triangleright (H_1, \lambda, H_2)$, the *reduct of* $(\mathcal{B}, p)$ *with respect to* (H_1, λ, H_2), to be the weighted aligned HR bimorphism $((g', \mathcal{A}_1, \Lambda, \mathcal{A}_2), p')$ where g' and p' are defined as follows.

If $[\![.]\!]_{\mathcal{B}}{}^{-1}(H_1, \lambda, H_2) = \emptyset$, then $g' = (\{\xi'_0\}, \Sigma, \xi'_0, \emptyset)$ and $p' = \emptyset$. Otherwise, let $m \in \mathbb{N}$ be the maximum of all $|\mathrm{ports}_{\mathrm{graph}(\sigma_{\mathcal{A}_1})}|$ and $|\mathrm{ports}_{\mathrm{graph}(\sigma_{\mathcal{A}_2})}|$ where $\sigma \in \Sigma$. Now, we construct $g' = (\Xi', \Sigma, \xi'_0, R')$ where we abbreviate $\mathcal{H}^m_{\mathrm{S}}(H_1) \times \mathcal{H}^m_{\mathrm{S}}(\lambda) \times \mathcal{H}^m_{\mathrm{S}}(H_2)$ by $\mathcal{H}^m_{\mathrm{S}}(H_1, \lambda, H_2)$:

- $\Xi' = \Xi \times (\mathcal{H}^m_{\mathrm{S}}(H_1, \lambda, H_2) \cap [\![\mathrm{T}_\Sigma]\!]_{\mathcal{B}})$,
- $\xi'_0 = (\xi_0, H_1, \lambda, H_2)$, and
- for every rule $\varrho = (\xi \to \sigma(\xi_1, \ldots, \xi_k)) \in R$ and $(s, \eta, r), (s_1, \eta_1, r_1), \ldots, (s_k, \eta_k, r_k)$ in $\mathcal{H}^m_{\mathrm{S}}(H_1, \lambda, H_2) \cap [\![\mathrm{T}_\Sigma]\!]_{\mathcal{B}}$ we have

$$\varrho' = \big((\xi, s, \eta, r) \to \sigma((\xi_1, s_1, \eta_1, r_1), \ldots, (\xi_k, s_k, \eta_k, r_k))\big) \in R'$$

  if $s = \sigma_{\mathcal{A}_1}(s_1, \ldots, s_k)$, $\eta = \Lambda_\sigma \cup \eta_1 \cup \ldots \cup \eta_k$, and $r = \sigma_{\mathcal{A}_2}(r_1, \ldots, r_k)$.

We define $p'(\varrho') = p(\varrho)$. □

Theorem 5.4. In Def. 5.3 the following hold:

1. $\mathcal{L}(g') = [\![.]\!]_{\mathcal{B}}{}^{-1}(H_1, \lambda, H_2) \cap \mathcal{L}(g)$.
2. There is a deterministic tree relabeling ϕ from $D_{g'}$ to D_g such that for all $t \in \mathcal{L}(g')$ and $d' \in D_{g'}(t)$, $\phi|_{D_{g'}(t)}$ is a bijection between $D_{g'}(t)$ and $D_g(t)$, and $p'(d') = p(\phi(d'))$.

Proof. If $[\![.]\!]_{\mathcal{B}}{}^{-1}(H_1, \lambda, H_2) = \emptyset$, then $\mathcal{L}(g') = \emptyset$ by construction, and thus, both statements of the theorem hold. Otherwise, the first statement follows from the following claim.

Claim (*) For every $n \in \mathbb{N}$, $\xi \in \Xi$, $t \in \mathrm{T}_\Sigma$, and $(s, \eta, r) \in (\mathcal{H}^m_{\mathrm{S}}(H_1, \lambda, H_2)) \cap [\![\mathrm{T}_\Sigma]\!]_{\mathcal{B}}$ it holds that $(\xi, s, \eta, r) \Rightarrow^n_{g'} t$ iff $\xi \Rightarrow^n_g t$ and $[\![t]\!]_{\mathcal{A}_1} = s$ and $\Lambda_{\mathcal{B}}(t) = \eta$ and $[\![t]\!]_{\mathcal{A}_2} = r$.

For the proof of the second statement we define $\phi((\xi, s, \eta, r)) = \xi$ for each $(\xi, s, \eta, r) \in \Xi'$, and extend ϕ in the canonical way to derivation trees. Then the statement is an immediate consequence of the constructions of R' and ϕ and Claim (*). ■

Complexity We determine the complexity of the reduct construction for the special case of LCFRS/sDCP hybrid grammars. We assume that the maximal length m of ports in the HR operations is fixed and not part of the input. In preparation, we determine an upper bound on $|\Xi'|$.

Let $H \in \mathcal{H}$ be such that from each vertex there is an (undirected) path to a port. Given H each $H' \in \mathcal{H}^m_{\mathrm{S}}(H) \cap [\![\mathrm{T}_\Sigma]\!]_{\mathcal{A}}$ is uniquely determined by its *boundary representation* (Lautemann, 1990; Chiang et al., 2013; Groschwitz et al., 2015), which consists of (a) the pair $(\mathrm{ports}_{H'}, \varphi(\mathrm{ports}_{H'}))$, (b) the set of *boundary edges* $E^B_{H'}$ *of* H' consisting of all $e \in E_{H'}$ such that $\mathrm{att}_{H'}(e) \cap [\mathrm{ports}_{H'}] \neq \emptyset$, and (c) a function $\mathrm{att}'\colon E^B_{H'} \to ([\mathrm{ports}_{H'}] \cup \{\bot\})^*$ such that $\mathrm{att}'(e)(i) = \mathrm{att}_{H'}(e)(i)$ if $\mathrm{att}_{H'}(e)(i) \in [\mathrm{ports}_{H'}]$, and $\bot$ otherwise. Note that (c) is necessary because $\varphi|_{[\mathrm{ports}_{H'}]}$ might not be injective.

Now, for an arbitrary (Σ, Γ)-AG algebra $\mathcal{A}$ and $H \in (\!|\mathcal{T}^*_\Gamma(\emptyset)|\!)$ the following holds. Due to the single-input property of H and of the involved HR operations, for each m-subgraph H' the information (b) and (c) can be inferred from (a). There are $S_{m,k}$ port sequences of length m with k distinct vertices, where $S_{m,k}$ is the *Stirling number of the second kind*. For each port we choose a vertex in V_H. Thus, we obtain that $|\mathcal{H}^m_{\mathrm{S}}(H) \cap [\![\mathrm{T}_\Sigma]\!]_{\mathcal{A}}| \leq \sum_{k=0}^m S_{m,k} \cdot |V_H|^k \leq m^m \cdot |V_H|^m$.

Next, we analyze the $\mathcal{B}$-alignments of an LCFRS/sDCP hybrid grammar $\mathcal{B}$. Let $(s, \eta, r) \in \mathcal{H}^m_{\mathrm{S}}(H_1, \lambda, H_2)$ and $t \in \mathrm{T}_\Sigma$ with $[\![t]\!]_{\mathcal{B}} = (s, \eta, r)$. Then $V_\eta = V_s \cup V_r$. Let $e \in E_\lambda$ and $e_1 \in E_{H_1}$ and $e_2 \in E_{H_2}$ with $\mathrm{att}_\lambda(e) = \mathrm{inp}(e_1)\,\mathrm{inp}(e_2)$. (i) Assume that there is $t' \in \mathrm{T}_\Sigma$ with $[\![t']\!]_{\mathcal{B}} = (H_1, \lambda, H_2)$ and t is a subtree of t'. Then e_1 and e_2 are unique by the single input property. Hence, $e_1 \in E_s$ iff $e_2 \in E_r$ iff $e \in E_\eta$ because t is a subtree of t' and by the structure of Λ_σ. (ii) If there is no such t', then the equivalences under (i) may be violated, in which case (s, η, r) can safely be pruned.

Thus, for each LCFRS/sDCP hybrid grammar $\mathcal{B}$ and each (H_1, λ, H_2) with $H_1 \in (\!|\mathcal{T}^*_\Delta(\emptyset)|\!)$ and $H_2 \in (\!|\mathcal{T}^*_\Omega(\emptyset)|\!)$ we obtain the upper bound $|\Xi| \cdot m^{2m} \cdot |V_{H_1}|^m \cdot |V_{H_2}|^m$ on $|\Xi'|$. This bound can be refined to $|\Xi| \cdot m^{2m} \cdot |V_{H_1}|^{2 \cdot \mathrm{fan}^*} \cdot |V_{H_2}|^{2 \cdot (\mathrm{syn}^* + \mathrm{inh}^*)}$, where $f^* = \max_{\xi \in \Xi} f(\xi)$ for $f \in \{\mathrm{fan}, \mathrm{syn}, \mathrm{inh}\}$. Constructing Ξ' and R' simultaneously with a deductive parsing algorithm (Shieber et al., 1995) has

a worst-case time complexity in $\mathcal{O}(|\Xi'|^k)$ where k is the maximum rank of Σ.

5.3 EM Training Algorithm

In the first step of our training algorithm, a corpus $c' \colon R \to \mathbb{R}_{\geq 0}$ is computed as follows. After initialization (line 3), each bigraph B occurring in c is considered (line 4), the reduct $(\mathcal{B}, p_i) \rhd B$ is built (line 5), the inside/outside weights of the new WRTG (g', p') are calculated (line 6), and according to these weights and the current weight assignment p_i the count $c'(\varrho)$ of each rule is incremented (lines 8–9). In the second step, the corpus c' is normalized (lines 10–14) and the result is the next probability assignment p_{i+1} (line 15).

Algorithm 5.1 EM-training algorithms for weighted aligned HR bimorphisms.

Input: weighted aligned HR bimorphism $(\mathcal{B}, p_0) = ((g, \mathcal{A}_1, \Lambda, \mathcal{A}_2), p_0)$ with $g = (\Xi, \Sigma, \xi_0, R)$, and a finite, non-empty corpus c of bigraphs.

Output: sequence $p_1, p_2, p_3, \ldots$ of improved probability assignments for R.

1: $i \leftarrow 0$
2: **while** true **do**
3: initialize counts $c' : R \to \mathbb{R}_{\geq 0}$: $c'(\varrho) \leftarrow 0$
4: **for** $B = (H_1, \lambda, H_2)$ s.t. $c(B) > 0$ **do**
5: $((g', \mathcal{A}_1, \Lambda, \mathcal{A}_2), p') \leftarrow (\mathcal{B}, p_i) \rhd B$ with RTG $g' = (\Xi', \Sigma, \xi_0', R')$ and det. tree rel. ϕ // cf. Thm 5.4
6: compute out and in for the WRTG (g', p')
7: **if** $\mathrm{in}(\xi_0') \neq 0$ **then**
8: **for** $\varrho = (\xi \to \sigma(\xi_1, \ldots, \xi_k)) \in R$ **do**
9: $c'(\varrho) \leftarrow c'(\varrho) + c(B) \cdot \mathrm{in}(\xi_0')^{-1} \cdot \sum_{\substack{\varrho' \in R' : \\ \phi(\varrho') = \varrho \,\wedge\, \varrho' = (\xi' \to \sigma(\xi_1', \ldots, \xi_k'))}} \mathrm{out}(\xi') \cdot p_i(\varrho) \cdot \prod_{j=1}^{k} \mathrm{in}(\xi_j')$
10: **for** $\xi \in \Xi$ **do**
11: $s \leftarrow \sum_{\varrho \in R_\xi} c'(\varrho)$
12: **for** $\varrho \in R_\xi$ **do**
13: **if** s = 0 **then** $p_{i+1}(\varrho) \leftarrow p_i(\varrho) \cdot |R_\xi|^{-1}$
14: **else** $p_{i+1}(\varrho) \leftarrow s^{-1} \cdot c'(\varrho)$
15: output p_{i+1} and $i \leftarrow i + 1$

Acknowledgment

We thank the referees for their careful reading of the manuscript.

References

James K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. 7th Linguistic Annotation Workshop, ACL 2013 Workshop*.

Michel Bauderon and Bruno Courcelle. 1987. Graph expressions and graph rewriting. *Mathematical Systems Theory*, 20:83–127.

Suna Bensch, Frank Drewes, Helmut Jürgensen, and Brink van der Merwe. 2014. Graph transformation for incremental natural language analysis. *Theoretical Computer Science*, 531:1–25.

Walter S. Brainerd. 1969. Tree generating regular systems. *Inform. and Control*, 14:217–231.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 924–932.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Bruno Courcelle. 1991. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theoretical Computer Science*, 80:153–202.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Pierre Deransart and Jan Małuszynski. 1985. Relating logic programs and attribute grammars. *J. Logic Programming*, 2:119–155.

Joost Engelfriet and Linda Heyker. 1992. Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29(2):161–210.

Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest. (See also arXiv:1509.06233, 2015).

Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. 1977. Initial algebra semantics and continuous algebras. *J. ACM*, 24:68–95.

Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. 2015. Graph parsing with s-graph grammars. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1481–1490.

Annegret Habel and Hans-Jörg Kreowski. 1987. May we introduce to you: Hyperedge replacement. In *Proc. of the Third Intl. Workshop on Graph Grammars and Their Application to Computer Science*, pages 15–26.

Annegret Habel. 1992. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In M. Kay and C. Boitet, editors, *Proc. 24th Intl. Conf. on Computational Linguistics (COLING 2012): Technical Papers*, pages 1359–1376.

Karim Lari and Steve J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Clemens Lautemann. 1990. The complexity of graph languages generated by hyperedge replacement. *Acta Inf.*, 27(5):399–421.

Philip M. Lewis and Richard E. Stearns. 1968. Syntax-directed transduction. *J. ACM*, 15(3):465–488.

Mark-Jan Nederhof and Giorgio Satta. 2006. Probabilistic parsing strategies. *J. ACM*, 53(3):406–436.

Mark-Jan Nederhof and Giorgio Satta. 2008. Probabilistic parsing. In Gemma Bel-Enguix, M. Dolores Jiménez-López, and Carlos Martín-Vide, editors, *New Developments in Formal Languages and Applications*, volume 113 of *Studies in Computational Intelligence*, pages 229–258. Springer.

Mark-Jan Nederhof and Heiko Vogler. 2012. Synchronous context-free tree grammars. In *Proc. of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 55–63.

Mark-Jan Nederhof and Heiko Vogler. 2014. Hybrid grammars for discontinuous parsing. In *Proc. of 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1370–1381.

Detlef Prescher. 2001. Inside-outside estimation meets dynamic EM. In *Proc. of the 7th International Workshop on Parsing Technologies*, pages 241–244.

Hiroyuki Seki and Yuki Kato. 2008. On the generative power of multiple context-free grammars and macro grammars. *IEICE - Transactions on Information and Systems*, E91-D(2):209–221.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. of the 13th International Conference on Computational Linguistics*, volume 3, pages 253–258.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *J. Logic Programming*, 24(1–2):3 – 36.

Krishnamurti Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Wolfgang Wechler. 1992. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer.