ALGEBRAIC CONSTRUCTION OF PARSING SCHEMATA

Karl-Michael Schneider
Department of General Linguistics
University of Passau
Innstr. 40, 94032 Passau, Germany
schneide@phil.uni-passau.de

Abstract

We propose an algebraic method for the design of tabular parsing algorithms which uses parsing schemata [7]. The parsing strategy is expressed in a tree algebra. A parsing schema is derived from the tree algebra by means of algebraic operations such as homomorphic images, direct products, subalgebras and quotient algebras. The latter yields a tabular interpretation of the parsing strategy. The proposed method allows simpler and more elegant correctness proofs by using general theorems and is not limited to left-right parsing strategies, unlike current automaton-based approaches. Furthermore, it allows to derive parsing schemata for linear indexed grammars (LIG) from parsing schemata for context-free grammars by means of a correctness preserving algebraic transformation. A new bottom-up head corner parsing schema for LIG is constructed to demonstrate the method.

1 Introduction

Linear indexed grammars (LIG) [2] and tree adjoining grammars (TAG) [4] are weakly equivalent grammar formalisms that generate an important subclass of the so-called mildly context-sensitive languages (MCSL). In recent publications (see for example [1, 5] and the papers cited there) the design of parsing algorithms for LIG and TAG is based on an operational model of (formal) language recognition. It consists of the construction of some nondeterministic push-down automaton from a grammar, depending on the parsing strategy, and a tabular interpretation of that automaton. This approach is modular because the tabulation of the automaton is independent of the parsing strategy.

Besides its obvious advantages over a direct construction of parsing algorithms (as in [9]), this approach is still dissatisfying in two respects: First, the tabulation of a LIG automaton is motivated only informally, in terms of a certain non-contextuality of LIG derivations (i.e., parts of LIG derivations do not depend on the bottom parts of the dependent stacks) or in terms of an efficient representation of unbounded LIG stacks. Second, because the usual push-down automata read their input sequentially from left to right, this technique cannot be applied straightforwardly to head-corner strategies, which start the recognition of an input string in the middle.

In this paper we present an algebraic approach to the design of parsing algorithms. By this we mean that a parsing algorithm is derived from an algebraic specification of a parsing strategy by means of algebraic operations such as homomorphic images, direct products, subalgebras and quotient algebras. A parsing strategy is expressed through the operations in an algebra where the objects are partial parse trees (called a *tree algebra*). A second algebra (called *yield algebra*) describes how the input string is processed.

Following [7] we do not construct parsing algorithms but rather *parsing schemata*, i.e., high-level descriptions of tabular parsing algorithms that can be implemented as tabular algorithms in a canonical way [8]. A parsing schema describes the items in the table and the steps that the algorithm performs in order to find all valid items, but leaves the order in which parsing steps are performed unspecified.

Our approach picks up an idea originally proposed but not fully developed[1] by Sikkel [7] whereupon a parsing schema could be regarded as the quotient (with respect to some congruence relation) of a *tree-based parsing schema.* A parse item is seen as a congruence class of a partial parse tree for some part of the input string. The problem is that items that do not denote a valid parse tree for some part of the input string cannot be described in this way because they would denote empty congruence classes. In our approach a parse item is seen as a pair $(a_{\simeq}, \xi)$ where $a_{\simeq}$ is a congruence class of trees and ξ denotes a substring of the input string. In this way all items can be characterized algebraically. This allows us to lift the correctness proof from the level of items to the level of trees.

In this paper we construct a new bottom-up head-corner (buHC) parsing schema for LIG to demonstrate the algebraic approach. The construction proceeds in two steps: In the first step we construct a buHC parsing schema for context-free grammars (CFG) algebraically and give a correctness proof. In the second step an algebraic, correctness preserving transformation is applied to the tree algebra of this parsing schema to obtain a buHC parsing schema for LIG. The transformed tree algebra implements the non-contextuality of LIG derivations into the tree operations and thus makes this notion more precise.

Our approach has a series of advantages over the automaton-based construction of parsing algorithms: It is not limited to parsing strategies that process the input string from left to right; it provides a precise characterization of an item in terms of congruence classes; it allows simpler and more elegant correctness proofs by means of general algebraic theorems; it allows to derive parsing schemata for LIG from parsing schemata for CFG by means of algebraic transformations; and finally, it provides a precise explanation for certain characteristics of LIG parsing algorithms.

The paper may be outlined as follows: In Sect. 2 we define the basic algebraic concepts used in this paper. Sect. 3 presents a short introduction to parsing schemata and describes the general method of constructing parsing schemata algebraically. In Sect. 4 we show the algebraic construction of the buHC parsing schema for CFG, and in Sect. 5 we define an algebraic transformation that yields a buHC parsing schema for LIG. Sect. 6 presents final conclusions.

2 Nondeterministic Algebras

In this section we present generalized versions of standard concepts of Universal Algebra [3] for algebras with nondeterministic operations, called *nondeterministic algebras*, which provide the basis for the algebraic description of parsing schemata. The theorems in this section are given without proof, the proofs can be found in [6]. Although nondeterministic variants of algebras have been defined previously, for example relational systems [3], most concepts of Universal Algebra have been fully developed only for algebras with deterministic operations.

An algebra $\mathcal{A}$ is a pair (A, F) where A is a nonvoid set (the *carrier* of $\mathcal{A}$) and F is a family of finitary operations $f : A^n \to A$. An *n-ary nondeterministic operation* is a set-valued function $f : A^n \to \mathcal{P}(A)$, where $\mathcal{P}(A)$ denotes the powerset of A. We use the notation $f(a_1, \ldots, a_n) \vdash a$ iff $a \in f(a_1, \ldots, a_n)$. If

[1] although it must be pointed out that Sikkel's book is not about the algebraic structure of parsing schemata in the first place, but about relations between different parsing schemata.

f is nullary we write $f \vdash a$ instead of $f() \vdash a$. A *nondeterministic algebra* $\mathcal{A}$ is a pair (A, F) where A is a nonvoid set and F is a family of finitary nondeterministic operations. A *partial algebra* is a nonvoid set A together with a family of finitary partial operations on A, i.e., partial functions $f : A^n \rightharpoonup A$. The *type* of a (partial, nondeterministic) algebra is the function that assigns each operation its arity. We write $\mathcal{A} = (A, F)$ and $\mathcal{B} = (B, F)$ to indicate that $\mathcal{A}$ and $\mathcal{B}$ are algebras of the same type, although the operations can be defined differently in $\mathcal{A}$ and $\mathcal{B}$. To indicate whether an operation $f \in F$ belongs to $\mathcal{A}$ or $\mathcal{B}$ we write $f^{\mathcal{A}}$ and $f^{\mathcal{B}}$, but we omit the superscripts if the algebra is understood.

The *restriction* of a nondeterministic operation $f : A^n \to \mathcal{P}(A)$ to a subset $B \subseteq A$ is the operation $f' : B^n \to \mathcal{P}(B)$ defined by $f'(b_1, \ldots, b_n) = f(b_1, \ldots, b_n) \cap B$. Let $\mathcal{A} = (A, F)$ be a nondeterministic algebra. The smallest subset A' of A that is closed under the operations in F (i.e., if $a_1, \ldots, a_n \in A'$, $n \geq 0$, and $f(a_1, \ldots, a_n) \vdash a$ then $a \in A'$) is denoted with $[\emptyset]_{\mathcal{A}}$. $[\emptyset]_{\mathcal{A}}$ is nonempty only if $\mathcal{A}$ has nullary operations. The elements in $[\emptyset]_{\mathcal{A}}$ are said to be *generated* (by $\mathcal{A}$).

We now present some standard concepts of Universal Algebra for nondeterministic algebras. Let $\mathcal{A} = (A, F)$ and $\mathcal{B} = (B, F)$ be nondeterministic algebras of the same type. $\mathcal{B}$ is called a *relative subalgebra* of $\mathcal{A}$ if $B \subseteq A$ and for every $f \in F$, $f^{\mathcal{B}}$ is the restriction of $f^{\mathcal{A}}$ to B. $\mathcal{B}$ is called a *weak subalgebra* of $\mathcal{A}$ if $B \subseteq A$, and for every $f \in F$, for all $b_1, \ldots, b_n, b \in B$: whenever $f^{\mathcal{B}}(b_1, \ldots, b_n) \vdash b$ then $f^{\mathcal{A}}(b_1, \ldots, b_n) \vdash b$. If $[\emptyset]_{\mathcal{A}}$ is nonempty then the relative subalgebra $([\emptyset]_{\mathcal{A}}, F)$ is called the *generated subalgebra* of $\mathcal{A}$.

The *direct product* of $\mathcal{A}$ and $\mathcal{B}$ is the nondeterministic algebra $\mathcal{A} \times \mathcal{B} = (A \times B, F)$, where $f^{\mathcal{A} \times \mathcal{B}}((a_1, b_1), \ldots, (a_n, b_n)) \vdash (a, b)$ iff $f^{\mathcal{A}}(a_1, \ldots, a_n) \vdash a$ and $f^{\mathcal{B}}(b_1, \ldots, b_n) \vdash b$.

A *homomorphism* of $\mathcal{A}$ into $\mathcal{B}$ is a function $h : A \to B$ satisfying the condition: For all $a_1, \ldots, a_n \in A$, if $f^{\mathcal{A}}(a_1, \ldots, a_n) \vdash a$ then $f^{\mathcal{B}}(ha_1, \ldots, ha_n) \vdash ha$. A homomorphism h of $\mathcal{A}$ into $\mathcal{B}$ is called *strong* if for all $a_1, \ldots, a_n \in A$, for all $b \in B$: whenever $f^{\mathcal{B}}(ha_1, \ldots, ha_n) \vdash b$, then there is some element $a \in A$ such that $f^{\mathcal{A}}(a_1, \ldots, a_n) \vdash a$ and $ha = b$.

A *strong congruence relation* of a $\mathcal{A}$ is an equivalence relation $\simeq$ on A satisfying the condition: if $f(a_1, \ldots, a_n) \vdash a$ and $a_i \simeq a'_i$, for $i = 1, \ldots, n$, then there is some $a' \in A$ such that $a \simeq a'$ and $f(a'_1, \ldots, a'_n) \vdash a'$. The set of all strong congruence relations of $\mathcal{A}$ is denoted with $Cgr(\mathcal{A})$. Strong congruence relations and strong homomorphisms of nondeterministic algebras are related as follows [6]: The *kernel* of a strong homomorphism is a strong congruence relation, and every strong congruence relation is the kernel of some strong homomorphism.

Let $\simeq$ be a strong congruence relation of $\mathcal{A}$. The quotient algebra $\mathcal{A}/\simeq$ is the nondeterministic algebra $(A/\simeq, F)$ where the operations are defined through $f^{\mathcal{A}/\simeq}(a_{1\simeq}, \ldots, a_{n\simeq}) \vdash a_{\simeq}$ iff for some elements $a'_1, \ldots, a'_n, a' \in A$: $a'_i \simeq a_i$ (for all i) and $a' \simeq a$ and $f^{\mathcal{A}}(a'_1, \ldots, a'_n) \vdash a'$.

The following theorem describes the connection between homomorphisms and generated subalgebras of direct products:

Theorem 1. *Let $\mathcal{A}$ be a nondeterministic algebra and $\mathcal{B}$ a partial algebra of the same type and h : $\mathcal{A} \to \mathcal{B}$ a homomorphism. Then* $[\emptyset]_{\mathcal{A}\times\mathcal{B}} = \{(a, b) \in [\emptyset]_{\mathcal{A}} \times [\emptyset]_{\mathcal{B}} \mid ha = b\}$.

The next theorem shows that generated subalgebras and quotient algebras commute:

Theorem 2. *If $\simeq$ is a strong congruence relation of $\mathcal{A}$ then* $[\emptyset]_{\mathcal{A}/\simeq} = [\emptyset]_{\mathcal{A}}/\simeq$.

As a corollary, we also get the following

Theorem 3. *If $h : \mathcal{A} \to \mathcal{B}$ is a strong homomorphism then* $[\emptyset]_{\mathcal{B}} = \{ha \mid a \in [\emptyset]_{\mathcal{A}}\}$.

The last theorem can be interpreted thus: Under a strong homomorphism, computations in a homomorphic algebra are homomorphic images of computations in the original algebra.

3 Algebraic Construction of Parsing Schemata

In this section we present a formal definition of parsing schemata and describe the general scheme of the algebraic construction of parsing schemata. This scheme is completely independent of a particular grammar formalism or parsing strategy. Parsing schemata were proposed by Sikkel as a well-defined level of abstraction for the description and comparison of tabular parsing algorithms [7].

A parsing schema[2] is a deduction system (I, D) consisting of a finite set of (parse) items I and a finite set D of deduction steps written in the form $x_1, \ldots, x_n \vdash x$ (meaning x is deducible from $x_1, \ldots, x_n$) where $n \geq 0$ and $x_1, \ldots, x_n, x \in I$. The *inference relation* $\vdash$ is defined by $X \vdash x$ iff for some $x_1, \ldots, x_n \in X$: $x_1, \ldots, x_n \vdash x \in D$. The *reflexive and transitive inference relation* $\vdash^*$ is defined by $X \vdash^* x$ iff $x \in X$ or there are items $y_1, \ldots, y_m$ such that for all i, $X \cup \{y_1, \ldots, y_i - 1\} \vdash y_i \in D$ and $x = y_m$. If $X \vdash^* x$ we say that x is *deducible* from X. If x is deducible from the void set $\emptyset$ then x is called *valid.*

Let (I, D) be a parsing schema for a grammar G and an input string w. Every item $x \in I$ represents a G-derivation of a particular form of some substring of w. If such a derivation actually exists then x is called *correct.* (I, D) is called correct if valid and correct items coincide. If (I, D) is correct then a string w is in the language of G iff an item representing a G-derivation of w from the start symbol of G is valid.

Let G be a grammar. An *(augmented) tree algebra* for G is a nondeterministic algebra $\mathcal{A}_G = (A, F)$ where A is a set of partial derivation trees augmented with some state information (that depends on the parsing strategy) and F is a family of (possibly nondeterministic) tree operations that depend on the grammar G as well as the parsing strategy. We assume that F contains at least one nullary operation. In the sequel we will assume that G is understood and write $\mathcal{A}$ instead of $\mathcal{A}_G$.

Let Σ be an input alphabet. An *(augmented) yield algebra* is a partial algebra $\mathcal{B} = (B, F)$ where B is a set of strings from Σ^* augmented with some state information. A homomorphism $g : \mathcal{A} \to \mathcal{B}$ (where $\mathcal{A}$ is an augmented tree algebra and $\mathcal{B}$ is an augmented yield algebra) that assigns each augmented tree the augmented string of terminal symbols at its leaves is called a *yield homomorphism.*

Let $\mathcal{A}$ be an augmented tree algebra and $\mathcal{B}$ an augmented yield algebra and $g : \mathcal{A} \to \mathcal{B}$ a yield homomorphism. Let $\simeq$ be a strong congruence relation of $\mathcal{A}$. For any string $w \in \Sigma^*$ let $B(w) \subseteq B$ be the set of all augmented substrings of w (the exact definition of *substring* depends on the parsing strategy). Let $\mathcal{B}(w) = (B(w), F)$ be the relative subalgebra of $\mathcal{B}$ with carrier $B(w)$. The nondeterministic algebra $\mathcal{A}/\!\simeq \times \mathcal{B}(w)$, that is, the direct product of the quotient algebra of $\mathcal{A}$ and the relative subalgebra of $\mathcal{B}$ with augmented substrings of w, is called a *parsing algebra* for G, w. The elements of a parsing algebra are pairs $(a_\simeq, b)$ where $a_\simeq$ is a congruence class of an augmented derivation tree and b is an augmented substring of w. By Theorems 1 and 2, $(a_\simeq, b)$ is generated in the parsing algebra iff there is some generated derivation tree a' in $\mathcal{A}$ such that $a'_\simeq = a_\simeq$ and $ga' = b$.

Let $w \in \Sigma^*$ be an input string. An augmented substring of w may be given by a tuple $\xi \in \mathbb{N}^m$ of positions in w. Two different tuples ξ, ζ may determine the same augmented substring of w. A *parse item* is a pair $(a_\simeq, \xi)$ where $a_\simeq$ is an equivalence class of augmented derivation trees and ξ is a tuple of positions. A *parse item algebra* is a nondeterministic algebra $\mathcal{I} = (I, F)$ where I is a set of parse items. A *parse item homomorphism* is a strong homomorphism φ from a parse item algebra into a parsing algebra, such that $\varphi(a_\simeq, \xi) = (a_\simeq, b)$, i.e., φ maps the second component of a parse item to an augmented substring of w. If $\varphi : \mathcal{I} \to \mathcal{A}/\!\simeq \times \mathcal{B}(w)$ is a parse item homomorphism, then

[2]We use a slightly different notation and terminology than that in [7].

by Theorem 3, a parse item $(a_{\simeq}, \xi)$ is generated in $\mathcal{I}$ iff for some $b \in B(w)$, $\varphi(a_{\simeq}, \xi) = (a_{\simeq}, b)$ and $(a_{\simeq}, b)$ is generated in the parsing algebra.

A parsing schema (I, D) is obtained from a finite parse item algebra (I, F) by defining $D = \{x_1, \ldots, x_n \vdash x \mid \exists f \in F : f(x_1, \ldots, x_n) \vdash x\}$. Then by the previous equivalences, $(a_{\simeq}, \xi)$ is deducible in the parsing schema iff for some $a' \in A$, a' is generated in $\mathcal{A}$ and $a'_{\simeq} = a_{\simeq}$ and $ga' \in B(w)$ and $\varphi(a_{\simeq}, \xi) = (a_{\simeq}, ga')$. Note that if φ is a parse item homomorphism of a finite parse item algebra into a parsing algebra, then there are only finitely many congruence classes of *generated* augmented derivation trees.

A nondeterministic algebra $\mathcal{A}$ is called sound (resp. complete) w.r.t. a set $A_0 \subseteq A$ iff $[\emptyset]_{\mathcal{A}} \subseteq A_0$ (resp. $[\emptyset]_{\mathcal{A}} \supseteq A_0$). A nondeterministic algebra is called *correct* iff it is sound and complete (w.r.t. a set A_0). The grammar G defines a subset $A_0 \subseteq A$ of *admissible* augmented derivation trees. Note that A_0 depends on the parsing strategy but not on F. If a parsing schema (I, D) for G, w is constructed as above and the tree algebra is correct w.r.t. admissible derivation trees of G, then a parse item $(a_{\simeq}, \xi)$ is deducible in (I, D) iff $a_{\simeq}$ is the congruence class of some admissible derivation tree a' and ga' is the augmented substring of w denoted by ξ; that is, (I, D) is correct.

By definition, $w \in L(G)$ iff there is a derivation of w from some start symbol of G. An element in A that represents a derivation of a string $w \in \Sigma^*$ from a start symbol is called a *complete (augmented) derivation tree.* An equivalence relation $\simeq$ on A is called *regular* if there are no mixed equivalence classes; that is, if the condition holds: whenever a is complete and $a \simeq a'$ then a' is complete, too. If $\simeq$ is regular then $a_{\simeq}$ is called complete iff a is complete. A parse item $(a_{\simeq}, \xi)$ where $a_{\simeq}$ is complete is called a *final item.* If (I, D) is correct for G, w then $w \in L(G)$ iff there is some final item $(a_{\simeq}, \xi)$ such that $(a_{\simeq}, \xi)$ is deducible in (I, D) and ξ denotes w.

4 Context-Free Bottom-Up Head-Corner Parsing

In this section we present an algebraic description of the bottom-up head-corner (buHC) parsing schema for CFG [7, Schema 11.13], according to the construction scheme described in the previous section. A buHC parser starts the recognition of the right-hand side of a production at a predefined position (the *head* of the production) rather than at the left edge, and proceeds in both directions. In the sequel we will denote terminal symbols with $a, a_1, a_2, \ldots$, nonterminal symbols with $A, B, \ldots$, strings of terminal symbols with u, w and strings of terminal and nonterminal symbols with β, γ, δ. $|\beta|$ denotes the length of β. We borrow a practical notation for trees from [7]: $\langle A \leadsto \beta \rangle$ denotes an arbitrary tree with root symbol A and yield (i.e., sequence of labels on the leaves, from left to right) β (possibly of height 0, in which case $\beta = A$). $\langle A \to \beta \rangle$ denotes the unique tree of height 1 with root symbol A and yield β. A tree of height 1 is called a *local tree.* Expressions of this form can be nested, thus specifying subtrees of larger trees. We also write $\langle \beta \leadsto \gamma \rangle$ for a sequence of trees with root symbols β (from left to right) and concatenated yields γ.

A *headed context-free grammar* G is a tuple (N, Σ, P, S, h) such that (N, Σ, P, S) is a CFG without ε-productions, where N, Σ, P are finite sets of nonterminal symbols, terminal symbols and productions, respectively, S is a start symbol and $h : P \to \mathbb{N}$ is a function that assigns each production $p = A \to \beta$ a position $1 \leq h(p) \leq |\beta|$. The $h(p)$-th symbol in β is called the *head* of p (for simplicity it is assumed that the same production cannot occur twice with different heads). The pair $(p, h(p))$ is called a *headed production.* If $p = A \to \beta X \delta$ and $h(p) = |\beta X|$ then we write $A \to \beta \underline{X} \delta$ for $(p, h(p))$.

A *buHC tree* is a triple (τ, k, l) where $\tau = \langle A \to \beta \langle \gamma \leadsto u \rangle \delta \rangle$ (for some $A, \beta, \gamma, \delta, u$) is a finite,

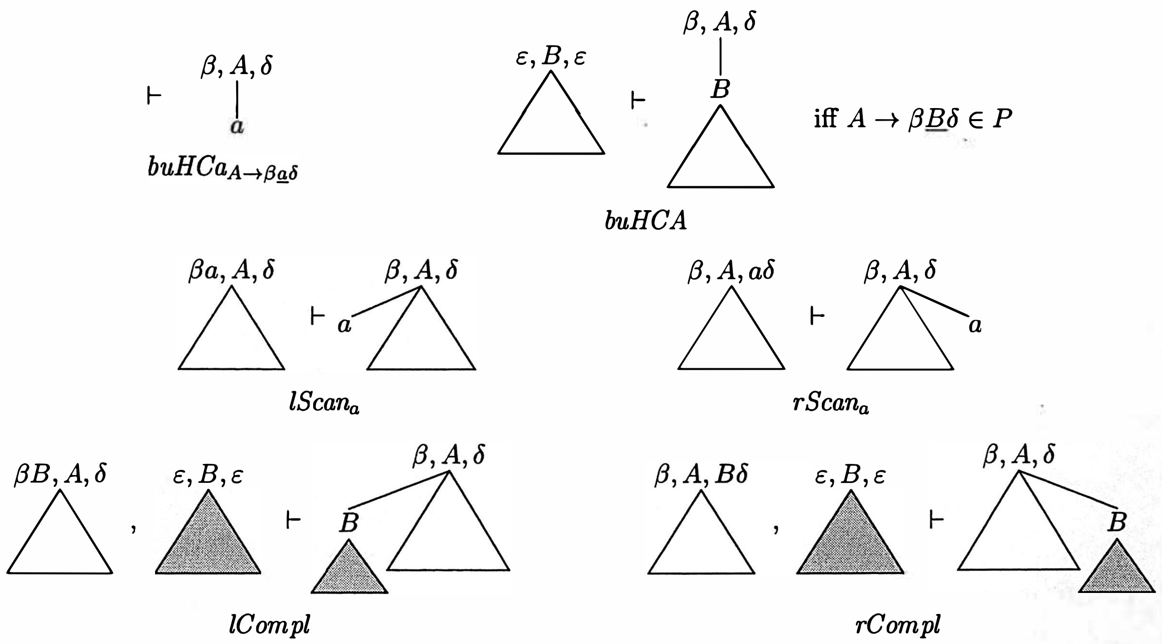


Figure 1: Bottom-up head-corner tree operations.

ordered tree with $k = |\beta|$ and $l = |\beta\gamma|$. k and l are state information; they mark the beginning and end of the recognized part γ of a production. An equivalent representation for (τ, k, l) is the triple (τ', β, δ) where $\tau' = \langle A \to \langle \gamma \rightsquigarrow u \rangle \rangle$ and the subtrees specified by $\langle \gamma \rightsquigarrow u \rangle$ are the same in τ and τ'. We use the second form in graphical representations of buHC trees and write β and δ to the left and right of the root label, respectively.

The *buHC tree operations* are shown in Fig. 1 (the yields of the trees are omitted for simplicity). ε denotes the empty string. The nullary operation $buHCa_{A \to \beta \underline{a} \delta}$ is indexed with a headed production in order to ensure that the corresponding operation in the yield algebra is a partial function. For the same reason, the operations $lScan_a$ and $rScan_a$ are indexed with the symbol a being scanned. The tree rooted by B in *lCompl* and *rCompl* is called *side tree*. We denote the buHC tree algebra with $\mathcal{A}_{buHC}$.

A local tree $\langle A \to \beta \rangle$ is called *admissible* w.r.t. G iff $A \to \beta$ is a production of G. A buHC tree (τ, k, l) where $\tau = \langle A \to \beta \langle \gamma \rightsquigarrow u \rangle \delta \rangle$ is admissible w.r.t. G iff each local tree in τ is admissible w.r.t. G and there is some headed production $(p, h(p))$ with $p = A \to \beta\gamma\delta$ and $k < h(p) \leq l$.

Proposition 1. *$\mathcal{A}_{buHC}$ is correct w.r.t. admissible buHC trees.*

Proof. Soundness is proved by induction on the basis of individual operations. To this end, observe that each operation computes only admissible trees provided that its arguments are admissible. In particular, $buHCa_{A \to \beta \underline{a} \gamma}$ is an admissible tree.

Completeness is proved by induction on the length of computations. Define the function $\lambda : (\tau, k, l) \mapsto (|\tau|, l - k)$ for any buHC tree (τ, k, l) where $|\tau|$ denotes the number of nodes in τ, and define the relation $<_T$ on $\mathbb{N}^2$ by $(m, n) <_T (m', n')$ iff $m < m'$ or $(m = m'$ and $n < n')$. $<_T$ is a well-founded, strict ordering on the λ-values of all buHC trees. Now observe that if (τ, k, l) is an admissible buHC tree then (τ, k, l) is computed by $buHCa_{A \to \beta \underline{a} \gamma}$ or we find admissible buHC trees (τ_i, k_i, l_i) for $1 \leq i \leq j$ such that (τ, k, l) is computed from (τ_i, k_i, l_i) by some j-ary buHC tree operation and $\lambda(\tau_i, k_i, l_i) <_T \lambda(\tau, k, l)$ for all i. Then completeness follows by induction on the values of λ. □

$$
\begin{aligned}
I_{buHC} &= \{[A \to \beta{\bullet}\gamma{\bullet}\delta, i, j] \mid A \to \beta\gamma\delta \in P,\ 0 \le i < j \le |w|\} \\
D^{buHCa} &= \{\vdash [A \to \beta{\bullet} a_i {\bullet}\delta, i-1, i] \mid A \to \beta\underline{a_i}\delta \in P\} \\
D^{buHCA} &= \{[B \to {\bullet}\gamma{\bullet}, i, j] \vdash [A \to \beta{\bullet} B{\bullet}\delta, i, j]\} \mid A \to \beta\underline{B}\delta \in P\} \\
D^{lScan} &= \{[A \to \beta a_i{\bullet}\gamma{\bullet}\delta, i, j] \vdash [A \to \beta{\bullet} a_i\gamma{\bullet}\delta, i-1, j]\} \\
D^{rScan} &= \{[A \to \beta{\bullet}\gamma{\bullet} a_{j+1}\delta, i, j] \vdash [A \to \beta{\bullet}\gamma a_{j+1}{\bullet}\delta, i, j+1]\} \\
D^{lCompl} &= \{[A \to \beta B{\bullet}\gamma{\bullet}\delta, i, j], [B \to {\bullet}\gamma'{\bullet}, k, i] \vdash [A \to \beta{\bullet} B\gamma{\bullet}\delta, k, j]\} \\
D^{rCompl} &= \{[A \to \beta{\bullet}\gamma{\bullet} B\delta, i, j], [B \to {\bullet}\gamma'{\bullet}, j, k] \vdash [A \to \beta{\bullet}\gamma B{\bullet}\delta, i, k]\} \\
D_{buHC}(w) &= D^{buHCa} \cup D^{buHCA} \cup D^{lScan} \cup D^{rScan} \cup D^{lCompl} \cup D^{rCompl}
\end{aligned}
$$

Figure 2: The buHC parsing schema.

A buHC yield is a string in Σ^*. The buHC yield homomorphism is defined as follows: For $(\tau, k, l) = (\tau', \beta, \delta)$, where $\tau' = \langle A \to \langle \gamma \rightsquigarrow u \rangle\rangle$, let $g_{\mathrm{buHC}}(\tau, k, l) = u$. The buHC yield operations are defined as follows: $buHCa_{A \to \beta\underline{a}\delta} = a$, $buHCA(u) = u$, $lScan_a(u) = au$, $rScan_a(u) = ua$, $lCompl(u, v) = vu$, $rCompl(u, v) = uv$. We denote the buHC yield algebra with $\mathcal{B}_{buHC}$. Clearly, g_{buHC} is a homomorphism of $\mathcal{A}_{buHC}$ into $\mathcal{B}_{buHC}$.

Let $\simeq_{\mathrm{buHC}}$ be the equivalence relation defined by $(\tau_1, k_1, l_1) \simeq_{\mathrm{buHC}} (\tau_2, k_2, l_2)$ iff (for some $A, \beta, \gamma, \delta, u_1, u_2$) $\tau_i = \langle A \to \beta\langle\gamma \rightsquigarrow u_i\rangle\delta\rangle$ $(i = 1, 2)$ and $k_1 = k_2 = |\beta|$ and $l_1 = l_2 = |\beta\gamma|$. Then $\simeq_{\mathrm{buHC}}$ is a strong congruence relation of $\mathcal{A}_{buHC}$. We denote the congruence class of (τ, k_i, l_i) with $[A \to \beta{\bullet}\gamma{\bullet}\delta]$. It is obvious that $\simeq_{\mathrm{buHC}}$ partitions the set of admissible buHC trees into finitely many congruence classes, for any headed CFG G. For any $w \in \Sigma^*$ let $sub(w)$ be the set of strings u such that for some v, v': $w = vuv'$ (substrings of w), and let $\mathcal{B}_{buHC}(w)$ be the relative subalgebra of $\mathcal{B}_{buHC}$ with carrier $sub(w)$. Clearly, $\mathcal{B}_{buHC}(w)$ is finite for any w.

Now fix some input string $w = a_1 \ldots a_n$. A buHC item for G, w is a triple $((\tau, k, l)_{\simeq_{\mathrm{buHC}}}, i, j)$, written as $[A \to \beta{\bullet}\gamma{\bullet}\delta, i, j]$, where $(\tau, k, l)_{\simeq_{\mathrm{buHC}}} = [A \to \beta{\bullet}\gamma{\bullet}\delta]$ and $A \to \beta\gamma\delta \in P$ and $0 \le i < j \le n$. A pair (i, j) denotes the nonempty substring of w from position i through j. Let $I_{buHC}(w)$ be the set of all buHC items for G, w. Let φ_{buHC} be the function $[A \to \beta{\bullet}\gamma{\bullet}\delta, i, j] \mapsto ([A \to \beta{\bullet}\gamma{\bullet}\delta], a_{i+1} \ldots a_j)$. The buHC operations can be defined straightforwardly on $I_{buHC}(w)$ such that φ_{buHC} is a strong homomorphism from $I_{buHC}(w)$ into $\mathcal{A}_{buHC}/{\simeq_{\mathrm{buHC}}} \times \mathcal{B}_{buHC}(w)$. Finally, the buHC parsing schema for G, w is obtained by defining the deduction steps as described in Sect. 3. It is shown in Fig. 2.

A buHC tree (τ, k, l) is *complete* iff τ is of the form $\langle S \to \langle\gamma \rightsquigarrow u\rangle\rangle$ and $k = 0$ and $l = |\gamma|$. Thus (τ, k, l) is complete iff $(\tau, k, l)_{\simeq_{\mathrm{buHC}}} = [S \to {\bullet}\gamma{\bullet}]$. The following corollary follows directly from the construction:

Corollary 1. *1.* $\vdash^* [A \to \beta{\bullet}\gamma{\bullet}\delta, i, j]$ *iff there is some admissible buHC tree* (τ, k, l) *with* $\tau = \langle\beta\langle\gamma \rightsquigarrow a_{i+1} \ldots a_j\rangle\delta\rangle$ *and* $k = |\beta|$ *and* $l = |\beta\gamma|$.

2. $w \in L(G)$ *iff for some* γ, $\vdash^* [S \to {\bullet}\gamma{\bullet}, 0, n]$.

5 Bottom-Up Head-Corner Parsing of LIG

A linear indexed grammar (LIG) [2] is an extension of a headed CFG in which the productions are associated with stack operations and where the nonterminal symbols in a derivation are associated

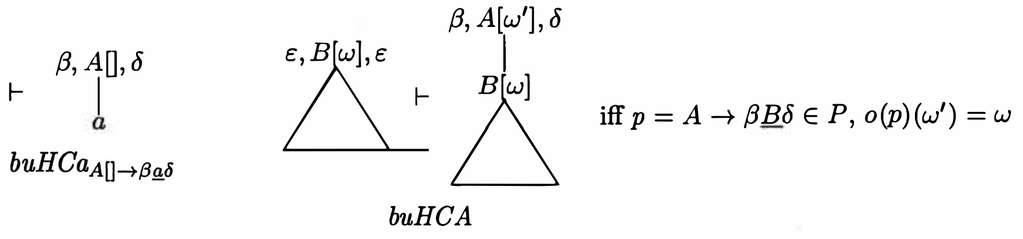

$\vdash$
$\beta, A[], \delta$
a
$buHCa_{A[]\to\beta\underline{a}\delta}$
$\varepsilon, B[\omega], \varepsilon$
$\vdash$
$\beta, A[\omega'], \delta$
$B[\omega]$
iff $p = A \to \beta\underline{B}\delta \in P$, $o(p)(\omega') = \omega$
$buHCA$

Figure 3: buHC-LIG tree operations.

with stacks of symbols. The stacks associated with the head and the left-hand side of a production are related by the stack operation associated with that production while all other descendants have a stack of bounded length. We consider a normal form of LIG where a stack operation either pushes or pops a single symbol, and where the stacks of non-head descendants must be empty.

A LIG in normal form can be represented as a tuple $(N, \Sigma, Q, P, S, h, o)$ where (N, Σ, P, S, h) is a headed CFG, Q is a finite stack alphabet and o is a function that assigns each production $p \in P$ a stack operation push_q or pop_q (where $q \in Q$) if the head of p is a nonterminal symbol, and nop otherwise. Let $q \in Q^*$ be a finite stack. The stack operations are defined as follows: $\text{push}_q(\omega) = \omega q$, $\text{pop}_q(\omega) = \omega'$ if $\omega = \omega' q$, else undefined, $\text{nop}(\omega) = \omega$.

A *headed tree* is a tree such that for each node v that is not a leaf, exactly one child of v is marked and the others are unmarked. The marked child is called the *dependent descendant* of v. A *buHC-LIG tree* is a tuple (τ, k, l) such that τ is a headed tree labeled with pairs $(X, \omega) \in (N \cup \Sigma) \times Q^*$, written as $X[\omega]$, and τ has the form $\langle A[\omega] \to \Lambda\langle\Gamma \rightsquigarrow u\rangle\Delta\rangle$, and $k = |\Lambda|$ and $l = |\Lambda\Gamma|$, where Λ, Γ, Δ denote (finite) sequences of labels in $(N \cup \Sigma) \times Q^*$, and if $X \in \Sigma$ then ω is empty. Instead of $X[]$ we can write X. Let G be a LIG. A local tree $\langle A[\omega] \to \Gamma\rangle$ is admissible w.r.t. G iff there is a production $p = A \to \gamma\underline{X}\gamma' \in P$ such that $\Gamma = \gamma X[\omega']\gamma'$ and $\omega' = o(p)(\omega)$ and the $h(p)$-th child of $A[\omega]$ is marked (note that γ, γ' denote sequences of labels with empty stacks). A buHC-LIG tree (τ, k, l) is admissible w.r.t. G iff every local tree in it is admissible w.r.t. G and, if the m-th child of the root of τ is marked then $k < m \leq l$.

Let (τ, k, l) be a buHC-LIG tree and v a node in τ with a stack of length $n \geq 1$. Consider the unique sequence of dependent descendants beginning at the dependent descendant of v and extending downwards to a leaf. If it exists, the unique node v' closest to v on this sequence with stack length $n - 1$ is called the *dependent stack descendant* of v. This means that on the path from v to v' the stack length does not fall below n except at v'. If v is the root of τ then v' is called the dependent stack descendant of τ. Note that if (τ, k, l) is admissible w.r.t. some LIG then v has a dependent stack descendant.

The buHC-LIG tree operations are obtained from the buHC tree operations by incorporating the stack operations associated with the productions. Fig. 3 shows the *buHCa* and *buHCA* operations. In the resulting trees the nodes labeled with a resp. $B[\omega]$ are marked. The other operations do not mark or unmark nodes. The *Scan* and *Compl* operations do not perform any operations on stacks, however, the *Compl* operations are only defined if the stack on the root of the side tree is empty. The buHC-LIG tree algebra is denoted with $\mathcal{A}_{buHC\text{-}LIG}$. Prop. 1 remains valid if $\mathcal{A}_{buHC}$ is replaced with $\mathcal{A}_{buHC\text{-}LIG}$ and "buHC trees" is replaced with "buHC-LIG trees", if we consider a buHC-LIG tree as a buHC tree over the infinite label domain $(N \cup \Sigma) \times Q^*$ and a LIG production as an abbreviation for an infinite set of context-free productions over this infinite domain. Note that the proof of Prop. 1

does not rely on the finiteness of N and P. However, in $\mathcal{A}_{buHC\text{-}LIG}$ we do not find a strong congruence relation with finitely many congruence classes of admissible buHC-LIG trees:

Proposition 2. *Let $\simeq$ be a strong congruence relation of $\mathcal{A}_{buHC\text{-}LIG}$. If the length of stacks in the derivation trees of G is unbounded, then $[\emptyset]_{\mathcal{A}_{buHC\text{-}LIG}}/{\simeq}$ is infinite.*

Proof. We give an informal proof. Consider the *buHCA* operation and a production with a push$_q$ operation. First, observe that if two admissible buHC-LIG trees are congruent then they must have the same symbol q on top of the stacks at their root nodes, because of the *buHCA* operation. Let $\omega = q_1 \dots q_m$ and $\omega' = q'_1 \dots q'_n$ be the stacks at the root nodes, where $q_m = q'_n$. By the same operation we can conclude that $q_{m-1} = q'_{n-1}$. By induction it follows that any two congruent, admissible buHC-LIG trees must have the same stack at their root nodes. Thus, if the length of stacks is unbounded, there are infinitely many noncongruent buHC-LIG trees. □

Below we will define an algebraic transformation that preserves the correctness of the transformed nondeterministic algebra under certain conditions. Then we proceed as follows: First, we replace the *buHCA* operation in Fig. 3 with two operations $buHCA_{op}$ for $op \in \{\text{push}, \text{pop}\}$, with the additional condition that $o(p) = op_q$ (for some q) in Fig. 3. Obviously, this does not affect the correctness of the tree algebra. Next, we use the transformation to modify the $buHCA_{push}$ operation. The transformed buHC-LIG tree algebra will have new congruence relations with only finitely many congruence classes of admissible buHC-LIG trees.

We first define some additional algebraic concepts. Let $\mathcal{A} = (A, F)$ be a nondeterministic algebra and $R : A^2 \to \mathcal{P}(A)$ a binary operation on A. A subset $A' \subseteq A$ is called *forward closed* w.r.t. R if for all $a_1, a_2 \in A'$: if $R(a_1, a_2) \vdash a$ then $a \in A'$. A' is called *backward closed* w.r.t. R if there is some function $d : A' \to \mathbb{N}$, such that for all $a_1 \in A'$ there is some $a_2 \in A'$ with $d(a_2) < d(a_1)$ and $R(a_1, a_2) \vdash a_1$. We denote with $\mathcal{A}[R] = (A, F \cup \{R\})$ the algebra where R has been adjoined as a new operation. $f \circ R$ denotes functional composition, i.e., $f \circ R(a_1, a_2) \vdash a$ iff for some $a' \in A$: $R(a_1, a_2) \vdash a'$ and $f(a') \vdash a$.

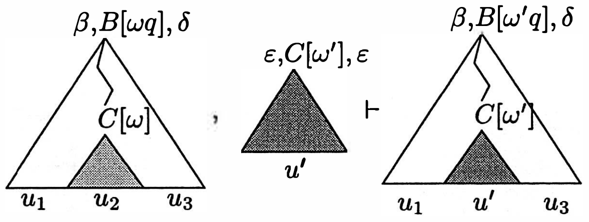
Theorem 4. *Let $\mathcal{A} = (A, F)$ be a nondeterministic algebra and $f \in F$ a unary operation and $R : A^2 \to \mathcal{P}(A)$ and let $\mathcal{A}' = (A, F')$ where $F' = F \setminus \{f\} \cup \{f \circ R\}$.*

1. *If $[\emptyset]_{\mathcal{A}}$ is backward and forward closed w.r.t. R then $[\emptyset]_{\mathcal{A}} = [\emptyset]_{\mathcal{A}'}$.*
2. *$Cgr(\mathcal{A}[R]) = Cgr(\mathcal{A}) \cap Cgr((A, R)) \subseteq Cgr(\mathcal{A}')$.*

Proof. (1) follows from the definition of closure and by induction on the values of d. (2) follows directly from the definitions. □

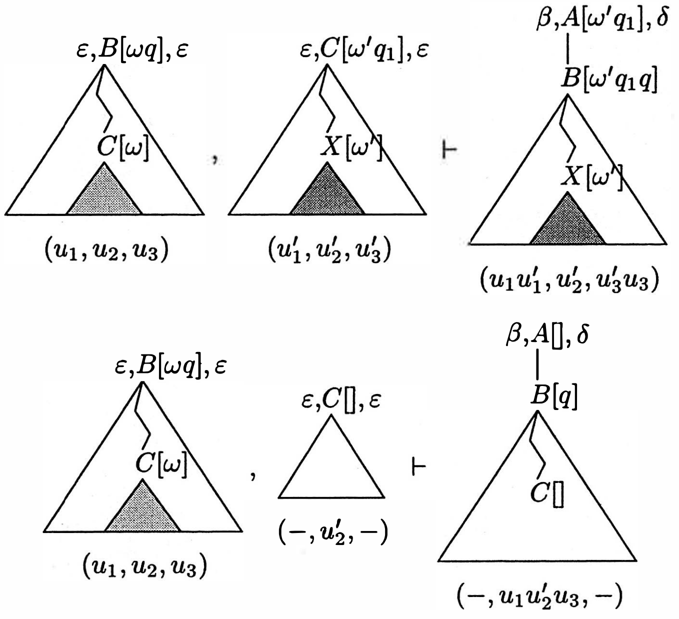
Assume that $\mathcal{A}$ is correct w.r.t. some set of admissible elements A_0, i.e., $[\emptyset]_{\mathcal{A}} = A_0$, let R be a binary operation on A such that A_0 is forward and backward closed w.r.t. R, and define $\mathcal{A}'$ as in Theorem 4. Then by Theorem 4, $\mathcal{A}'$ is also correct w.r.t. A_0. Forward closure of A_0 w.r.t. R preserves the soundness of $\mathcal{A}'$ while backward closure preserves its completeness. The second part of Theorem 4 guarantees that all strong congruence relations of $\mathcal{A}$ are preserved in $\mathcal{A}'$. More importantly, in the example below $\mathcal{A}'$ will have new (interesting) congruence relations.

Define the binary operation R as shown in Fig. 4. The lines indicate sequences of dependent descendants. $C[\omega]$ is the dependent stack descendant of $B[\omega q]$ in the left tree. Note that this implies that the stack ω is not consulted from $B[\omega q]$ to $C[\omega]$. The right tree is obtained by replacing each stack of the form $\omega q_1 \dots q_m$ ($m \geq 0$) on the path from $B[\omega q]$ to $C[\omega]$ in the left tree with a stack of the form $\omega' q_1 \dots q_m$, and then replacing the subtree rooted by $C[\omega']$ with the middle tree. The substitution of stacks exploits the fact that the application of LIG productions on the path from $B[\omega q]$ to $C[\omega]$ does



β,$B[\omega q]$, δ
$C[\omega]$
u_1 u_2 u_3
ε,$C[\omega']$, ε
u'
$\vdash$
β,$B[\omega' q]$, δ
$C[\omega']$
u_1 u' u_3

Figure 4: Substitution in buHC-LIG trees.



ε,$B[\omega q]$, ε
$C[\omega]$
(u_1, u_2, u_3)
ε,$C[\omega' q_1]$, ε
$X[\omega']$
(u'_1, u'_2, u'_3)
$\vdash$
β,$A[\omega' q_1]$, δ
$B[\omega' q_1 q]$
$X[\omega']$
$(u_1 u'_1, u'_2, u'_3 u_3)$
ε,$B[\omega q]$, ε
$C[\omega]$
(u_1, u_2, u_3)
ε,$C[]$, ε
$(-, u'_2, -)$
$\vdash$
β,$A[]$, δ
$B[q]$
$C[]$
$(-, u_1 u'_2 u_3, -)$

Figure 5: $buHCA_{push} \circ R$.

not depend on ω. If (τ, k, l) has no dependent stack descendant then let $R(\tau, k, l)(\tau', k', l') = (\tau, k, l)$ for any buHC-LIG tree (τ', k', l').

Proposition 3. $[\emptyset]_{\mathcal{A}_{buHC\text{-}LIG}}$ *is forward and backward closed w.r.t. R.*

Proof. For forward closure, observe that every node that is not on the path from $B[\omega q]$ to $C[\omega]$ is not a dependent descendent of any node on that path, and hence the stack substitution in the left tree, together with the subtree substitution, does not affect the admissibility of any local tree in the white area.

For backward closure, observe that the right tree in Fig. 4 can be obtained by replacing the subtree rooted by $C[\omega']$ with itself in the right tree (i.e., by doing nothing), and use the number of nodes in a tree as the function d in the definition of backward closure. □

Let $\mathcal{A}'_{buHC\text{-}LIG}$ be the algebra that is obtained from $\mathcal{A}_{buHC\text{-}LIG}$ by replacing the $buHCA_{push}$ operation with $buHCA_{push} \circ R$ (see Fig. 5). By Theorem 4 and Prop. 3, $\mathcal{A}'_{buHC\text{-}LIG}$ is correct w.r.t. admissible buHC-LIG trees. Furthermore, let $\simeq_{\text{buHC-LIG}}$ be the equivalence relation defined as follows: Let (τ_1, k_1, l_1), (τ_2, k_2, l_2) be buHC-LIG trees and for $i = 1, 2$ let $\tau_i = \langle A_i[\omega_i] \to \Lambda_i \langle \Gamma_i \rightsquigarrow u_i \rangle \Delta_i \rangle$. Then $(\tau_1, k_1, l_1) \simeq_{\text{buHC-LIG}} (\tau_2, k_2, l_2)$ iff $A_1 = A_2$ and $\Lambda_1 = \Lambda_2$ and $\Gamma_1 = \Gamma_2$ and $\Delta_1 = \Delta_2$ and $k_1 = k_2$ and $l_1 = l_2$, and if $\omega_1 = \varepsilon$ then $\omega_2 = \varepsilon$, and if $\omega_1 = \omega q$ then $\omega_2 = \omega' q$, and τ_1 has a dependent stack descendant iff τ_2 has a dependent stack descendant, and if $B[]$ is the dependent stack descendant of τ_1 then $B[]$ is the dependent stack descendant of τ_2, and if $B[\overline{\omega} q']$ is the dependent stack descendant

$$\vdash [A \to \beta{\bullet}a_i{\bullet}\delta, -, -, -, i-1, i, -, -] \quad (A \to \beta\underline{a_i}\delta \in P)$$

$$[B \to {\bullet}\gamma{\bullet}, q, C, q'', i, j, r, s], [C \to {\bullet}\gamma'{\bullet}, q_1, X, q_1', r, s, u, v] \vdash [A \to \beta{\bullet}B{\bullet}\delta, q_1, X, q_1', i, j, u, v]$$
$$(p = A \to \beta\underline{B}\delta \in P,\ o(p) = \mathrm{push}_q)$$

$$[B \to {\bullet}\gamma{\bullet}, q, C, q'', i, j, r, s], [C \to {\bullet}\gamma'{\bullet}, -, -, -, r, s, -, -] \vdash [A \to \beta{\bullet}B{\bullet}\delta, -, -, -, i, j, -, -]$$
$$(p = A \to \beta\underline{B}\delta \in P,\ o(p) = \mathrm{push}_q)$$

$$[B \to {\bullet}\gamma{\bullet}, q_1, C, q_1', i, j, r, s] \vdash [A \to \beta{\bullet}B{\bullet}\delta, q, B, q_1, i, j, i, j] \quad (p = A \to \beta\underline{B}\delta \in P,\ o(p) = \mathrm{pop}_q)$$

$$[A \to \beta a_i{\bullet}\gamma{\bullet}\delta, q, B, q', i, j, r, s] \vdash [A \to \beta{\bullet}a_i\gamma{\bullet}\delta, q, B, q', i-1, j, r, s]$$

$$[A \to \beta{\bullet}\gamma{\bullet}a_{j+1}\delta, q, B, q', i, j, r, s] \vdash [A \to \beta{\bullet}\gamma a_{j+1}{\bullet}\delta, q, B, q', i, j+1, r, s]$$

$$[A \to \beta B{\bullet}\gamma{\bullet}\delta, q, C, q', i, j, r, s], [B \to {\bullet}\gamma'{\bullet}, -, -, -, k, i, -, -] \vdash [A \to \beta{\bullet}B\gamma{\bullet}\delta, q, C, q', k, j, r, s]$$

$$[A \to \beta{\bullet}\gamma{\bullet}B\delta, q, C, q', i, j, r, s], [B \to {\bullet}\gamma'{\bullet}, -, -, -, j, k, -, -] \vdash [A \to \beta{\bullet}\gamma B{\bullet}\delta, q, C, q', i, k, r, s]$$

Figure 6: The buHC-LIG deduction steps.

of τ_1 then $B[\overline{\omega}'q']$ is the dependent stack descendant of τ_2, for some $\omega, \omega', \overline{\omega}, \overline{\omega}'$.

Proposition 4. *$\simeq_{buHC\text{-}LIG}$ is a strong congruence relation of $\mathcal{A}'_{buHC\text{-}LIG}$ with only finitely many congruence classes of admissible buHC-LIG trees.*

If (τ, k, l) is as in Fig. 4 (left) then let $g_{\text{buHC-LIG}}(\tau, k, l) = (u_1, u_2, u_3)$, and if $\tau = \langle A[] \to \beta\langle\Gamma \rightsquigarrow u\rangle\delta\rangle$ then let $g_{\text{buHC-LIG}}(\tau, k, l) = (-, u, -)$. Then the buHC-LIG operations can be defined on the buHC-LIG yields in a straightforward way (for example, see Fig. 5 for $buHCA_{push}$), such that $g_{\text{buHC-LIG}}$ is a homomorphism. Using the construction described in Sect. 3 (analogously to Sect. 4) we obtain a (correct!) buHC-LIG parsing schema. The buHC-LIG items are of the form $[A \to \beta{\bullet}\gamma{\bullet}\delta, q, B, q', i, j, r, s]$ where $A \to \beta\gamma\delta$ is a production, q, q' are stack symbols, $0 \le i \le r < s \le j \le |w|$ and q, B, q', r, s are $-$ if a buHC-LIG tree has no dependent stack descendant (then $0 \le i < j \le |w|$). The item homomorphism $\varphi_{\text{buHC-LIG}}$ maps a tuple of positions (i, j, r, s) to $(a_{i+1} \ldots a_r, a_{r+1} \ldots a_s, a_{s+1} \ldots a_j)$, resp. to $(-, a_{i+1} \ldots a_j, -)$ if $r = s = -$, where $w = a_1 \ldots a_n$ is the input string. The deduction steps are shown in Fig. 6.

The transformation defined in Theorem 4 may also be used to account for the form of the steps in the CYK-LIG algorithm in [9, 10]. This algorithm may be seen as the result of a transformation of a CYK tree algebra for CFG in Chomsky normal form using a similar substitution of subtrees as in Fig. 4.

6 Conclusion

We have proposed an algebraic method for the construction of tabular parsing algorithms. A parsing algebra for a grammar G and input string w is a relative subalgebra of a quotient algebra of the direct product of a tree algebra $\mathcal{A}$ (that reflects the parsing strategy) and a yield algebra $\mathcal{B}$ (that describes how the input string is processed) which is homomorphic to $\mathcal{A}$. A parsing schema is the inverse image of a parsing algebra under a strong homomorphism. Correctness of a parsing schema is defined at the level of tree operations. We have demonstrated the construction using a buHC parsing strategy for CFG. Furthermore, we have derived a buHC parsing schema for LIG from the buHC parsing schema for CFG by means of a correctness preserving algebraic transformation.

We have proposed the algebraic construction of tabular parsing algorithms for LIG as an alternative to the automaton-based approach proposed in recent papers [1, 5] because it allows to derive LIG algorithms from CFG algorithms by means of algebraic transformations, allows simpler and more elegant correctness proofs by using general theorems, and is not restricted to left-right parsing strategies. Furthermore, it makes the notion of parse items more precise and thus adds to a better understanding of parsing schemata.

References

[1] Miguel A. Alonso Pardo, Eric de la Clergerie, and David Cabrero Souto. Tabulation of automata for tree adjoining languages. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, pages 127–141, Orlando, Florida, USA, July 1999.

[2] Gerald Gazdar. Applicability of indexed grammars to natural languages. Tech. Rep. CSLI-85-34, Center for Study of Language and Information, Stanford, 1985.

[3] George Grätzer. *Universal Algebra.* Springer Verlag, New York, second edition, 1979.

[4] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3*, chapter 2, pages 69–123. Springer, Berlin, 1997.

[5] Mark-Jan Nederhof. Models of tabulation for TAG parsing. In *Sixth Meeting on Mathematics of Language*, pages 143–158, Orlando, Florida USA, July 1999. University of Central Florida.

[6] Karl-Michael Schneider. *Algebraic Construction of Parsing Schemata.* Doctoral dissertation, University of Passau, 1999.

[7] Klaas Sikkel. *Parsing Schemata.* Proefschrift, Universiteit Twente, CIP-Gegevens Koninklijke Bibliotheek, Den Haag, 1993.

[8] Klaas Sikkel. Parsing schemata and correctness of parsing algorithms. *Theoretical Computer Science*, 199(1–2):87–103, 1998.

[9] K. Vijay-Shanker and David J. Weir. Polynomial parsing of extensions of context-free grammars. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, pages 191–206. Kluwer, Dordrecht, 1991.

[10] K. Vijay-Shanker and David J. Weir. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636, December 1993.