On commutative and nonassociative syntactic calculi and categorial grammars

Maciej Kandulski
Faculty of Mathematics and Computer Science
Adam Mickiewicz University
ul. Matejki 48/49, 60-769 Poznań, Poland

Abstract

Two axiomatizations of the nonassociative and commutative Lambek syntactic calculus are given and their equivalence is proved. The first axiomatization employs Permutation as the only structural rule, the second one, with no Permutation rule, employs only unidirectional types. It is also shown that in the case of the Ajdukiewicz calculus an analogous equivalence is valid only in the case of a restricted set of formulas. Unidirectional axiomatizations are employed in order to establish the generative power of categorial grammars based on the nonassociative and commutative Lambek calculus with product. Those grammars produce CF-languages of finite degree generated by CF-grammars closed with respect to permutation.

Key words: Lambek calculus, Ajdukiewicz calculus, Structural rules, Categorial grammar, Context-free languages.
Mathematics Subjects Classification: 68Q50, 03D15, 03B65.

1 Introduction

Investigations concerning generative power of categorial grammars (CG's), which date from the paper of Bar-Hillel et al.[2], are currently stimulated by two factors. The first of them is the problem of the generative power of CG's based on the Lambek calculus [18]. This question posed in [2] resulted in a series of papers characterizing classes of languages generated by CG's based on subsystems of the Lambek calculus (see [8,9,11,14,15,25]) and was recently solved by Pentus [22]. On the other hand, in order to describe in a better way some phenomena of natural languages the shape of type reduction systems in CG's was modified so as to obtain more flexible versions of the formalism; the reader will find a detailed description of this and other relevant problems in [6]. Modifications included, among others, a relaxation of the structure of antecedents of reduction formulas. Those changes used to be introduced *via* endowing a type reduction system with structural rules of permutation, contraction and monotonicity performed as it is the case in Gentzen systems **LJ** and **LK** [12]. The introduction of structural rules implies changes in the formal status of antecedents of formulas: if in the case of basic Lambek calculus (associative and noncommutative) antecedents can be viewed as finite sequences of types then for a commutative calculus the proper model for an antecedent is that of a multiset, if we work with a nonassociative calculus, then antecedents are bracketed strings of types (trees) and if a calculus is both commutative and nonassociative then antecedents can be considered as mobiles, see [20]. Generative power of categorial grammars enriched with structural rules of permutation and of permutation and contraction was studied by Buszkowski in [7] and by van Benthem in [3] and [5]. Let us observe, however, that affixing permutation to the Lambek calculus compels one to accept phrases of a language without paying attention to the order of words. Although there is some linguistic evidence in favour of such a general relaxation, see for example [1], but still this global effect caused by permutation seems to be too strong from the point of view of a linguist. As a way out of this situation one can consider nonassociative calculi [19]: by the presence of permutation commutativity is restricted only to the set of direct constituents of a phrase structure. A local, restricted introduction of structural rules can also be performed by means of structural modalities, see [21]. This method is similar to the procedure of retrieving structural rules in Linear Logic, cf. [24]. Commutative syntactic calculi used to be introduced in different ways: as unidirectional systems [4,7], as bidirectional systems augmented with a structural rule of Permutation [21] and also as unidirectional systems with Permutation [6].

In this paper we prove that the bidirectional nonassociative Lambek calculus when enriched with Permutation as the only structural rule (we denote this system NLP) is equivalent to a unidirectional system (without Permutation) NCL which can be obtained from the previous one by unification of the left and right divisions. An analogous result however can not be obtained for the Ajdukiewicz calculus: the equivalence of bidirectional Ajdukiewicz calculus with permutation and unidirectional calculus holds only for a restricted class of formulas with an antecedent consisting of types of the order less than 2 and with a succedent being a primitive type. The problems of equivalence are examined in sections 2 and 3 of the paper. The axiomatizations GNLP and NLP of the nonassociative and commutative Lambek calculus as well as the employed in Lemma 1 axiomatization of an auxiliary system L strongly refer to other formalisms for different versions of Lambek calculi, see [9,18,19,26]. In particular, NLP imitates the formalism of the (product-free and bidirectional) nonassociative Lambek calculus as presented in [9], later transformed in [14] to the version with product. Sections 4 and 5 are devoted to the definite solution of the problem of generative power of categorial grammars based on NCL. In [17] we showed that the class of languages generated by those grammars is included in the class of CF-languages. Here we prove that this class of languages coincides with the class of CF-languages of finite degree generated by CF-grammars closed with respect to permutation ($\overline{\mathrm{CF}}$-languages). To obtain this result, in Section 4 we slightly refine the main theorem proved in [17]. The proof of the eqality of the considered classes of languages is given in Section 5.

2 Bidirectional and unidirectional commutative Lambek calculi

Given a countable but infinite set Pr of *primitive types*, and three binary operations $/$, $\backslash$ and $\cdot$, called *right division, left division* and *product*, respectively, we denote by TP the smallest set which contains Pr and is closed with respect to $/$, $\backslash$ and $\cdot$. The symbol Tp stands for the product-free part of TP. Types of the form x/y or $y\backslash x$ (resp. $x \cdot y$) will be referred to as *functorial* (resp. *product*) types. The number of occurrences of division and product signs in a given type x is called the *complexity* of x and denoted by $\mathrm{c}(x)$. For x being a product-free type, the symbol $\mathrm{o}(x)$ stands for the *order of* x, which is a non-negative integer defined inductively as follows: (i) $\mathrm{o}(x) = 0$, for $x \in \mathrm{Pr}$, (ii) $\mathrm{o}(x/y) = \mathrm{o}(y\backslash x) = \max\{\mathrm{o}(x), \mathrm{o}(y)+1\}$. The set $\mathrm{sub}(x)$ of *subtypes* of a type x is the smallest one such that: (i) $x \in \mathrm{sub}(x)$, (ii) if $y = z/v$ or $y = v\backslash z$ or $y = v \cdot z$, and $y \in \mathrm{sub}(x)$ then $v \in \mathrm{sub}(x)$ and $z \in \mathrm{sub}(x)$. If $\mathrm{A} \subseteq \mathrm{TP}$ then we put $\mathrm{sub}(A) = \bigcup_{x \in A} \mathrm{sub}(x)$. The set BSTP of *bracketed strings of types* is defined inductively in the following way: (i) TP $\subseteq$ BSTP, (ii) if $X \in$ BSTP and $Y \in$ BSTP then $(XY) \in$ BSTP. The symbol BSTp stands for the product-free part of BSTP.

By NA we will denote the nonassociative Ajdukiewicz calculus (with product), i.e. a formal system whose formulas are of the shape $X \to x$, where $X \in$ BSTP and $x \in$ TP, which employs one axiom scheme:

(A0) $\quad x \to x$, where $x \in$ TP,

and the following three rules of inference:

$$\text{(A)} \quad \frac{X \to x/y \qquad Y \to y}{(XY) \to x} \qquad\qquad (\overline{\text{A}}) \quad \frac{X \to y\backslash x \qquad Y \to y}{(YX) \to x}$$

$$\text{(PR)} \quad \frac{X \to x \qquad Y \to y}{(XY) \to x \cdot y} \; .$$

By adding to NA the (structural) rule of permutation:

$$\text{(Perm)} \; \frac{(XY) \to z}{(YX) \to z}, \quad \text{where } X, Y \in \text{BSTP}, \; z \in \text{TP},$$

we obtain the nonassociative Ajdukiewicz calculus (with product) with permutation NAP. In the presence of (Perm) in the system the derivability of a formula in NAP does not depend on the order of the constituents of any bracketed string of types occurring on the left-hand side of this formula. Calculi of syntactic types which neglect the order of types or constituents in their formulas were also introduced as unidirectional systems, cf. [4,6, 7,17]. Below we present a unidirectional version NCA of the nonassociative Ajdukiewicz calculus, to be called the nonassociative and commutative Ajdukiewicz calculus (with product). NCA employs types with product $\cdot$ and only one, right division /, and is formalized by (A0), (A), (PR), and by two additional rules of inference:

$$\text{(A}'\text{)} \quad \frac{X \to x/y \qquad Y \to y}{(YX) \to x} \qquad\qquad \text{(PR}'\text{)} \; \frac{X \to x \qquad Y \to y}{(YX) \to x \cdot y}$$

In this form NCA was introduced in [17]. Product-free versions of the calculi NA, NAP and NCA which employ types from Tp instead of TP, and make no use of axioms and rules involving types with product will be denoted NA°, NAP°, and NCA°, respectively. Calculi NAP and NCA will serve as a basis of two axiomatizations of the nonassociative and commutative Lambek calculus. But before we describe them we present this calculus in another, yet traditional form of a Gentzen sequential system GNLP in which Permutation is the only structural rule.

Let $X[x]$ (resp. $X[Y]$) denote a bracketed string of types X in which, on a certain place, a type x (resp. a bracketed string of types Y) occurrs. By $X[Y:Z]$ we denote a bracketed string of types which arises from $X[Y]$ by replacing a placed occurrence of Y by a bracketed string Z (in particular X, Y and Z can be types). The system GNLP employs the axiom scheme (A0) and the listed below rules of inference:

$$(\mathrm{R1}'_{\mathrm{G}})\ \frac{(X\,x)\to y}{X\to y/x} \qquad (\mathrm{R1}''_{\mathrm{G}})\ \frac{(x\,X)\to y}{X\to x\backslash y}$$

$$(\mathrm{R2}'_{\mathrm{G}})\ \frac{X\to y \quad Y[x]\to z}{Y[x:(x/y\,X)]\to z} \qquad (\mathrm{R2}''_{\mathrm{G}})\ \frac{X\to y \quad Y[x]\to z}{Y[x:(X\,y\backslash x)]\to z}$$

$$(\mathrm{Pr})\ \frac{X[(x\,y)]\to z}{X[x\cdot y]\to z} \qquad (\mathrm{PR})\ \frac{X\to x \quad Y\to y}{(XY)\to x\cdot y}$$

$$(\mathrm{Perm})\ \frac{(XY)\to z}{(YX)\to z}\ , \quad \text{where } X,Y\in \mathrm{BSTP},\ z\in \mathrm{TP}.$$

The above axiomatization differs from the Gentzen–style axiomatization of the nonassociative Lambek calculus NL as, for example presented in [16] only in the presence of the rule (Perm). Using the same procedure as in [18] or [19] one can prove that GNLP is closed with respect to the Cut rule of the form

$$(\mathrm{Cut})\quad \frac{X\to x \quad Y[x]\to y}{Y[x:X]\to y}.$$

It is easily observed that formulas $x/y\to y\backslash x$ and $y\backslash x\to x/y$ are both derivable in GNLP and both derivations essentially make use of (Perm).

By $\overline{\mathrm{Ax}}$ we denote a system which consists of formulas of the shape $x\to y$, where $x,y\in\mathrm{TP}$, and which admits the following axiom schemata and rules of inference for all $x,y,z\in\mathrm{TP}$:

(A0) $x\to x$

(a0) $x/y\to y\backslash x$ (a0′) $y\backslash x\to x/y$

(a1) $(x/y)\cdot y\to x$ (a1′) $y\cdot(x/y)\to x$

($\overline{\mathrm{a1}}$) $(y\backslash x)\cdot y\to x$ ($\overline{\mathrm{a1}'}$) $y\cdot(y\backslash x)\to x$

(a2) $x\to(x\cdot y)/y$ (a2′) $x\to y\backslash(y\cdot x)$

(a3) $x\to y/(x\backslash y)$ (a3′) $x\to(y/x)\backslash y$

(a4) $x\cdot y\to y\cdot x$

$$(\mathrm{r1})\ \frac{x\to y}{x/z\to y/z} \qquad (\mathrm{r1}')\ \frac{x\to y}{z/y\to z/x}$$

$$(\overline{\mathrm{r1}})\ \frac{x\to y}{z\backslash x\to z\backslash y} \qquad (\overline{\mathrm{r1}'})\ \frac{x\to y}{y\backslash z\to x\backslash z}$$

$$(\mathrm{r2})\ \frac{x\to y}{x\cdot z\to y\cdot z} \qquad (\mathrm{r2}')\ \frac{x\to y}{z\cdot x\to z\cdot y}$$

To simplify the notation we write $x \to y \in \overline{\mathrm{Ax}}$ instead of $\vdash_{\overline{\mathrm{Ax}}} x \to y$. Let us define the nonassociative Lambek calculus (with product) with permutation NLP as the system which arises from NAP by augmenting it with the following compound rule:

$$(\overline{\mathrm{C}}) \qquad \frac{X \to x}{X \to y}, \qquad \text{if } x \to y \in \overline{\mathrm{Ax}} \text{ and } x \neq y.$$

In what follows we are going to prove the equivalence of GNLP and NLP. However, instead of doing it directly we will show that these systems are equivalent to a certain auxiliary system L. This system has (Cut) as the only inference rule and employs (A0), formulas

(a_1)	$(x/y\, y) \to x$	$(\mathrm{a}_{1'})$	$(y\, x/y) \to x$
$(\overline{\mathrm{a}_1})$	$(y\backslash x\, y) \to x$	$(\overline{\mathrm{a}_{1'}})$	$(y\, y\backslash x) \to x$
(AP)	$(x\, y) \to x \cdot y$		

and all theses of $\overline{\mathrm{Ax}}$ as (schemes of) axioms.

For systems which have (Cut) as the unique rule of inference one can introduce the notion of reduction. We present the definition for the calculus L and will refer to it in case of other calculi. For $X, Y \in$ BSTP we say that X *reduces to* Y *in* L if there exists a sequence $X = X_0, X_1, \ldots, X_n = Y$, $X_i \in$ BSTP, $i = 0, 1, \ldots, n$, called *a reduction*, such that for every $k = 1, 2, \ldots, n$ there are $Z, T \in$ BSTP and $t \in$ TP such that $X_{k-1} = Z[T]$, $X_k = Z[T : t]$ and $T \to t$ is a non-(A0) axiom of L.

It is possible to express the notion of derivability in L in terms of reduction. We have the following

Fact 1 *For any formula $X \to x$ there holds the following equivalence:*
$\vdash_{\mathrm{L}} X \to x$ *if and only if X reduces to x in* L.

The proof of this fact is similar to that given in the case of associative Lambek calculi, see [26]. Using induction on the length of reduction one can also prove

Fact 2 *If X reduces to $(Y Z)$ in* L *then there are $X_1, X_2 \in$* BSTP *such that $X = (X_1 X_2)$ and X_1 reduces to Y and X_2 reduces to Z in* L.

Lemma 1 GNLP *and* NLP *are equivalent, i.e. for any formula $X \to x$ there holds the following equivalence:*
$\vdash_{\mathrm{GNLP}} X \to x$ *if and only if* $\vdash_{\mathrm{NLP}} X \to x$.

Proof. To prove the equivalence of GNLP and NLP it is sufficient to show that GNLP is equivalent to L and L is equivalent to NLP.

To prove the first equivalence we proceed as in [27], adjusting slightly the presented method to the calculus with product and with permutation.

In order to prove the equivalence of L and NLP we first show that NLP is a subsystem of L. One can easily check that L is closed with respect to (A), $(\overline{\mathrm{A}})$ and (PR). To prove that L is closed with respect to $(\overline{\mathrm{C}})$ it suffices to observe that every formula from $\overline{\mathrm{Ax}}$ is an axiom in L, thus it is derivable in L. Consequently, if $X \to x$ is derivable in L and $x \to y \in \overline{\mathrm{Ax}}$, then $X \to y$ is derivable in L from the above formulas by a single application of (Cut). Thus L is closed with respect to the rule ($\overline{\mathrm{C}}$). It remains to prove that L is closed with respect to (Perm). Let $\vdash_{\mathrm{L}} (XY) \rightarrow z$. By Fact 1, there exists a reduction $(XY), \ldots, z$ in L. Let i, $1 \leq i < n$, be the smallest number such that $X_i \in$ TP. Thus X_i must be obtained from X_{i-1} by application of one of the axioms (a_1), ($\mathrm{a}_{1'}$), ($\overline{\mathrm{a}_1}$), ($\overline{\mathrm{a}_{1'}}$) or (AP). Moreover, the formula $X_{i-1} \rightarrow X_i$ itself is one of those axioms. If it is an instance of (AP), then we have a reduction $(XY), \ldots, (x\,y), x \cdot y, \ldots, z$. By Fact 2, X reduces to x in L and Y reduces to y in L. We can thus write the reduction $(YX), \ldots, (y\,x), (y\,(x \cdot y)/y), (x \cdot y), \ldots, z$ which, again by Fact 1, gives us derivability of $(YX) \rightarrow z$ in L. If $X_{i-1} \rightarrow X_i$ is an instance of (a_1) (the argument for ($\mathrm{a}_{1'}$), ($\overline{\mathrm{a}_1}$) and ($\overline{\mathrm{a}_{1'}}$) is similar) then the reduction $(XY), \ldots, (x/y\,y), x, \ldots, z$ can be transformed to $(YX), \ldots, (y\,x/y), x, \ldots, z$ (we use ($\mathrm{a}_{1'}$) instead of (a_1) when transforming $(i-1)$th element of the latter reduction to ith).

Now we show that L is a subsystem of NLP. First, let us observe that all formulas from $\overline{\mathrm{Ax}}$ are derivable in NLP, we must only put $X = x$ in the rule ($\overline{\mathrm{C}}$). Axioms (a_1), ($\mathrm{a}_{1'}$), ($\overline{\mathrm{a}_1}$) and ($\overline{\mathrm{a}_{1'}}$) can be easily obtained as conclusions of the rules (A), ($\overline{\mathrm{A}}$) and (Perm). The axiom (AP) is, in turn, a trivial consequence of (PR). Thus it suffices to show that NLP is closed with respect to (Cut), but this can be proved by induction on the length of derivation of $Y[x] \rightarrow y$ in NLP. □

The form of NLP ilustrates the fact that the Lambek calculus can be considered as such an extension of Ajdukiewicz calculus which, in addition to functor-argument analysis of bracketed strings of types provided by the latter system, allows one to transform also individual types. The derivability of formulas $x/y \rightarrow y\backslash x$ and $y\backslash x \rightarrow x/y$ in GNLP (and in NLP) emphasizes a well known property of commutative Lambek calculi that they do not differentiate between right and left division signs. Thus one can ask whether it is possible to present the discussed calculus in the form of a unidirectional system. The answer is positive and we will perform the operation of unification of left and right divisions in NLP calling the resulting system the nonassociative and commutative Lambek calculus NCL.

For every type $x \in$ TP (resp. string $X \in$ BSTP) we define a type $\|x\|$ (resp. a string $\|X\|$) called *the unidirectional version of* x (resp. *of* X) inductively as follows: (i) $\|p\| = p$, if $p \in$ Pr, (ii) $\|x \cdot y\| = \|x\| \cdot \|y\|$, $\|x/y\| = \|y\backslash x\| = \|x\|/\|y\|$, (iii) $\|(XY)\| = (\|X\|\,\|Y\|)$. By *the unidirectional version of a formula* $X \rightarrow x$, to be denoted $\|X \rightarrow x\|$, we mean the formula $\|X\| \rightarrow \|x\|$.

Let Ax denote the unidirectional version of $\overline{\mathrm{Ax}}$, i.e. the system whose axioms are unidirectional versions of axioms of $\overline{\mathrm{Ax}}$ and whose rules of inference are rules of $\overline{\mathrm{Ax}}$ restricted to uniditectional versions of premises and conclusions. Thus types in formulas of Ax contain only product and right division and the system admits the following axioms and inference rules:

(A0) $x \to x$

(A1) $(x/y) \cdot y \to x$ (A1$'$) $y \cdot (x/y) \to x$

(A2) $x \to (x \cdot y)/y$ (A2$'$) $x \to (y \cdot x)/y$

(A3) $x \to y/(y/x)$ (A4) $x \cdot y \to y \cdot x$

(R1) $\dfrac{x \to y}{x/z \to y/z}$ (R1$'$) $\dfrac{x \to y}{z/y \to z/x}$

(R2) $\dfrac{x \to y}{x \cdot z \to y \cdot z}$ (R2$'$) $\dfrac{x \to y}{z \cdot x \to z \cdot y}$

Although some axioms and rules in Ax have the same form as in $\overline{\mathrm{Ax}}$, we give them different names to have the possibility of referring to each of the system without any ambiguity. We adopt a similar convention as for $\overline{\mathrm{Ax}}$ and write $x \to y \in \mathrm{Ax}$ instead of $\vdash_{\mathrm{Ax}} x \to y$.

We define the nonassociative and commutative Lambek calculus NCL as a system which arises from NCA by adding to this calculus the following compound rule:

$$\text{(C)} \qquad \frac{X \to x}{X \to y}, \qquad \text{if } x \to y \in \mathrm{Ax} \text{ and } x \neq y.$$

We adopt a similar convention as in the case of Ajdukiewicz calculi and use symbols NLP° and NCL° to denote product-free parts of NLP and NCL. A formula $X \to y$ is called $\overline{\mathrm{C}}$-*derivable in* NLP (resp. C-*derivable in* NCL) if and only if $X \to y$ is derivable in NLP (resp. in NCL) and every derivation of $X \to y$ employs only (possibly 0 times) the rule ($\overline{\mathrm{C}}$) (resp. (C)).

Lemma 2 *If* $\vdash_{\mathrm{NLP}} X \to y$ *(resp.* $\vdash_{\mathrm{NCL}} X \to y$*) then* $X \to y$ *is* $\overline{\mathrm{C}}$-*derivable in* NLP *(resp.* C-*derivable in* NCL*) if and only if* $X = x \in \mathrm{TP}$.

Proof. We give the argument for NLP but it can be applied for NCL as well. There are only the rules (A), ($\overline{\mathrm{A}}$) and (PR) which can increase the number of types on the left-hand side of a formula. However, these rules can not be used if a formula is C-derivable, and instances of the rule (C) do not affect left-hand sides of formulas. Thus starting from an axiom, by a number of applications of (C) we get a formula with its left-hand side being a single type. □

A sequence of types $x_1, \ldots, x_n$ is called *a* $\overline{\mathrm{C}}$-*reduction in* NLP (resp. *a* C-*reduction in* NCL) if and only if $n = 1$ or if for every number $1 \leq k < n$ we have $x_k \to x_{k+1} \in \overline{\mathrm{Ax}}$ (resp. Ax) and $x_k \neq x_{k+1}$.

Lemma 3 *A formula* $X \to y$ *is* $\overline{\mathrm{C}}$-*derivable in* NLP *if and only if there exists a* $\overline{\mathrm{C}}$-*reduction* $x_1, \ldots, x_n$ *in* NLP *such that* $X = x_1$ *and* $y = x_n$.

Proof. According to Lemma 2, X must be a type. To prove the lemma we use induction, for ($\Rightarrow$) on the number of instances of ($\overline{\mathrm{C}}$)-rule in a derivation, and for ($\Leftarrow$) on the number of types in $\overline{\mathrm{C}}$-reduction. □

Corollary 1 *If formulas $x \rightarrow y$ and $y \rightarrow z$ are $\overline{\mathrm{C}}$-derivable in* NLP*, then the formula $x \rightarrow z$ is $\overline{\mathrm{C}}$-derivable in* NLP *as well.*

Lemma 4 *If $\|x\| = \|y\|$ and $x, y \in$ TP, then the formulas $x \rightarrow y$ and $y \rightarrow x$ are $\overline{\mathrm{C}}$-derivable.*

Proof. Induction on $\mathrm{c}(x)$. If $x \in \mathrm{Pr}$ then $x = \|x\| = \|y\| = y$, hence $x \rightarrow y$ and $y \rightarrow x$ are instances of (A0). Let $x = x_1/x_2$. The type y must be of one of the forms y_1/y_2 or $y_2\backslash y_1$, and $\|x_1\| = \|y_1\|$ and $\|x_2\| = \|y_2\|$. Consider $y = y_1/y_2$. By inductive assumption $x_1 \rightarrow y_1$ and $y_2 \rightarrow x_2$ as well as $y_1 \rightarrow x_1$ and $x_2 \rightarrow y_2$ are $\overline{\mathrm{C}}$-derivable. By Lemma 3, there exist the following reductions: $x_1, \ldots, y_1$ as well as $y_2, \ldots, x_2$ and $y_1, \ldots, x_1$ as well as $x_2, \ldots, y_2$. But then, the sequences $x_1/x_2, \ldots, y_1/x_2$ as well as $y_1/x_2, \ldots, y_1/y_2$ and $y_1/y_2, \ldots, x_1/y_2$ as well as $x_1/y_2, \ldots, x_1/x_2$, and consequently, by Corollary 1, sequences $x_1/x_2, \ldots, y_1/x_2, \ldots, y_1/y_2$ and $y_1/y_2, \ldots, x_1/y_2, \ldots, x_1/x_2$ are also $\overline{\mathrm{C}}$-reductions. Hence, by Lemma 3 again, formulas $x_1/x_2 \rightarrow y_1/y_2$ and $y_1/y_2 \rightarrow x_1/x_2$ are $\overline{\mathrm{C}}$-derivable. If $y = y_2\backslash y_1$, then to the obtained as above $\overline{\mathrm{C}}$-reductions $x_1/x_2, \ldots, y_1/y_2$ and $y_1/y_2, \ldots, x_1/x_2$ one can add an instance of the axiom (a0) at the end of the first sequence and an instance of the axiom (a0$'$) at the beginning of the second sequence in order to obtain $\overline{\mathrm{C}}$-reductions $x_1/x_2, \ldots, y_2\backslash y_1$ and $y_2\backslash y_1, \ldots, x_1/x_2$. Their existence, by Lemma 3, gives us $\overline{\mathrm{C}}$-derivability of formulas $x_1/x_2 \rightarrow y_2\backslash y_1$ and $y_2\backslash y_1 \rightarrow x_1/x_2$. If $x = x_2\backslash x_1$ then the proof is essentially the same as for $x = x_1/x_2$. Consider $x = x_1 \cdot x_2$. We must have $y = y_1 \cdot y_2$, and $\|x_1\| = \|y_1\|$ and $\|x_2\| = \|y_2\|$. By inductive assumption formulas $x_1 \rightarrow y_1$, and $x_2 \rightarrow y_2$, and $y_1 \rightarrow x_1$, and $y_2 \rightarrow x_2$ are $\overline{\mathrm{C}}$-derivable. We employ Lemma 3 to obtain $\overline{\mathrm{C}}$-reductions $x_1, \ldots, y_1$ as well as $x_2, \ldots, y_2$ and $y_1, \ldots, x_1$ as well as $y_2, \ldots, x_2$. It is easy to observe that $x_1 \cdot x_2, \ldots, y_1 \cdot x_2$ and $y_1 \cdot x_2, \ldots, y_1 \cdot y_2$, and $y_1 \cdot y_2, \ldots, x_1 \cdot y_2$ and $x_1 \cdot y_2, \ldots x_1 \cdot x_2$ are also $\overline{\mathrm{C}}$-reductions. By Corollary 1, $x_1 \cdot x_2, \ldots, y_1 \cdot x_2, \ldots, y_1 \cdot y_2$ and $y_1 \cdot y_2, \ldots, x_1 \cdot y_2, \ldots, x_1 \cdot x_2$ are $\overline{\mathrm{C}}$-reductions as well, thus $x_1 \cdot x_2 \rightarrow y_1 \cdot y_2$ and $y_1 \cdot y_2 \rightarrow x_1 \cdot x_2$ are $\overline{\mathrm{C}}$-derivable. □

Lemma 5 (i) *If $x \rightarrow y \in \overline{\mathrm{Ax}}$, then $\|x\| \rightarrow \|y\| \in$ Ax.*
(ii) *For all $x, y \in$ TP, if $\|x\| \rightarrow \|y\| \in$ Ax, then $x \rightarrow y$ is $\overline{\mathrm{C}}$-derivable in* NLP*.*

Proof. (i) Easy induction on the length of derivation in $\overline{\mathrm{Ax}}$.
(ii) Throughout the proof, in order to make it shorter, we will employ Lemma 3 and Corollary 1 without stating it explicitly. We use induction on the length of derivation in Ax. If $\|x\| \rightarrow \|y\| =$ (A0) then the conclusion follows from Lemma 4. Let $\|x\| \rightarrow \|y\| =$ (A1). Thus $x = (x_1/x_2) \cdot x_3$ or $x = (x_2\backslash x_1) \cdot x_3$ and in both cases $\|x_2\| = \|x_3\|$ and $\|x_1\| = \|y\|$. By Lemma 4, formulas $x_3 \rightarrow x_2$ and $x_1 \rightarrow y$ are $\overline{\mathrm{C}}$-derivable in NLP. We can use $\overline{\mathrm{C}}$-reductions $x_3, \ldots, x_2$ and $x_1, \cdots, y$ to produce a $\overline{\mathrm{C}}$-reduction $(x_1/x_2) \cdot x_3, \ldots, (x_1/x_2) \cdot x_2, x_1, \ldots, y$ whose existence gives us $\overline{\mathrm{C}}$-derivability of $(x_1/x_2) \cdot x_3 \rightarrow y$. In order to obtain $\overline{\mathrm{C}}$-derivability of $(x_2\backslash x_1) \cdot x_3 \rightarrow y$ it suffices to precede the above reduction by the type $(x_2\backslash x_1) \cdot x_3$. If $\|x\| \rightarrow \|y\| =$ (A1$'$) then $x = x_3 \cdot (x_1/x_2)$ or $x = x_3 \cdot (x_2\backslash x_1)$, and $\|x_2\| = \|x_3\|$ and $\|x_1\| = \|y\|$. To obtain $\overline{\mathrm{C}}$-reductions for the formulas $x_3 \cdot (x_1/x_2) \rightarrow y$ and $x_3 \cdot (x_2\backslash x_1) \rightarrow y$ one can precede the two $\overline{\mathrm{C}}$-reductions constructed for the axiom (A1) by, respectively, types $x_3 \cdot (x_1/x_2)$ and $x_3 \cdot (x_2\backslash x_1)$. Let $\|x\| \rightarrow \|y\| =$ (A2). Thus $y = (y_1 \cdot y_2)/y_3$ or $y = y_3\backslash(y_1 \cdot y_2)$ and in both cases $\|y_2\| = \|y_3\|$ and $\|y_1\| = \|x\|$. We employ $\overline{\mathrm{C}}$-reductions $x, \ldots, y_1$ and $y_3, \ldots, y_2$ in construction of a $\overline{\mathrm{C}}$-reduction

$x, \ldots, y_1, (y_1 \cdot y_2)/y_2, \ldots, (y_1 \cdot y_2)/y_3$ which yields $\overline{\mathrm{C}}$-derivability of $x \to (y_1 \cdot y_2)/y_3$. The constructed $\overline{\mathrm{C}}$-reduction, with the type $y_3\backslash(y_1 \cdot y_2)$ added at the very end provides $\overline{\mathrm{C}}$-derivability of $x \to y_3\backslash(y_1 \cdot y_2)$. If $\|x\| \to \|y\| = (\mathrm{A2}')$ then the $\overline{\mathrm{C}}$-reductions for $x \to y$ can be obtained from those constructed for (A2) in an essentially the same way as those for (A1′) were obtained from $\overline{\mathrm{C}}$-reductions for (A1). If $\|x\| \to \|y\| = (\mathrm{A3})$ then we have four possibilities for the type y: it can be of the form $y_1/(y_2/y_3), (y_2/y_3)\backslash y_1, y_1/(y_3\backslash y_2)$ or $(y_3\backslash y_2)\backslash y_1$ where $\|y_3\| = \|x\|$ and $\|y_1\| = \|y_2\|$. We provide the argument for the last case, all other cases are similar. By Lemma 4, formulas $x \to y_3$ and $y_2 \to y_1$ are $\overline{\mathrm{C}}$-derivable. Thus given $\overline{\mathrm{C}}$-reductions $x, \ldots, y_3$ and $y_2, \ldots, y_1$ we can produce a new $\overline{\mathrm{C}}$-reduction $x, \ldots, y_3, (y_2/y_3)\backslash y_2, \ldots, (y_2/y_3)\backslash y_1, (y_3\backslash y_2)\backslash y_1$ which is sufficient to prove $\overline{\mathrm{C}}$-derivability of $x \to y$. If $\|x\| \to \|y\| = (\mathrm{A4})$ then $x = x_1 \cdot x_2$, $y = y_1 \cdot y_2$, and $\|x_1\| = \|y_2\|$ and $\|y_1\| = \|x_2\|$. As $x_2 \to y_1$ and $x_1 \to y_2$ are $\overline{\mathrm{C}}$-derivable we can write a $\overline{\mathrm{C}}$-reduction $x_1 \cdot x_2, x_2 \cdot x_1, \ldots, y_1 \cdot x_1, \ldots, y_1 \cdot y_2$ which proves $\overline{\mathrm{C}}$-derivability of $x_1 \cdot x_2 \to y_1 \cdot y_2$.

Now we take into consideration the cases when $\|x\| \to \|y\|$ arises by an application of derivation rules of Ax. If $\|x\| \to \|y\|$ arises by (R1) then x and y must be of one of the following forms: $x = x_1/z_1$, $y = y_1/z_2$ or $x = z_1\backslash x_1$, $y = y_1/z_2$ or $x = x_1/z_1$, $y = z_2\backslash y_2$ or $x = z_1\backslash x_1$, $y = z_2\backslash y_1$ where $\|x_1\| \to \|y_1\| \in \mathrm{Ax}$ and $\|z_1\| = \|z_2\|$. By our inductive assumption $x_1 \to y_1$ is $\overline{\mathrm{C}}$-derivable in NLP. If $x = x_1/z_1$ and $y = y_1/z_2$ we employ $\overline{\mathrm{C}}$-reductions $x_1, \ldots, y_1$ and $z_2, \ldots, z_1$, in order to construct a $\overline{\mathrm{C}}$-reduction $x_1/z_1, \ldots, y_1/z_1, \ldots, y_1/z_2$ which gives us $\overline{\mathrm{C}}$-derivability of $x_1/z_1 \to y_1/z_2$. For $x = z_1\backslash x_1, y = y_1/z_2$ the appropriate $\overline{\mathrm{C}}$-reduction arises from that constructed above by preceding it by the type $z_1\backslash x_1$. Other cases can be treated in a similar manner. Consequently, the formula $x \to y$ is $\overline{\mathrm{C}}$-derivable. If $\|x\| \to \|y\|$ arises by (R1′), then the proof is similar to that presented above. Let $\|x\| \to \|y\|$ be a conclusion of (R2). Thus we must have $x = x_1 \cdot z_1$, $y = y_1 \cdot z_2$, $\|z_1\| = \|z_2\|$ and $\|x_1\| \to \|y_1\| \in \mathrm{Ax}$. As by the inductive assumption $x_1 \to y_1$ is $\overline{\mathrm{C}}$-derivable, we have a $\overline{\mathrm{C}}$-reduction $x_1, \ldots, y_1$. There exists also a $\overline{\mathrm{C}}$-reduction $z_1, \ldots, z_2$. This is sufficient to construct a $\overline{\mathrm{C}}$-reduction $x_1 \cdot z_1, \ldots, y_1 \cdot z_1, \ldots, y_1 \cdot z_2$. Consequently, the formula $x \to y$ is $\overline{\mathrm{C}}$-derivable in NLP. A similar argument is valid for the rule (R2′). □

The equivalence of NLP and NCL is established by the following

Theorem 1 *If the formula $Y \to y$ is a unidirectional version of a formula $X \to x$, then $\vdash_{\mathrm{NLP}} X \to x$ if and only if $\vdash_{\mathrm{NCL}} Y \to y$.*

Proof. The 'if'–part of the equivalence can be proved by induction on the length of derivation of $X \to x$ in NLP. For axioms there is nothing to prove. For formulas $X \to x$ arising by (A), $(\overline{\mathrm{A}})$ or (PR) one obtains the conclusion immediately (one must use (A), (A′) or (PR) in NCL). If $X \to x$ arises by $(\overline{\mathrm{C}})$, then there must be formulas $X \to z$ and $z \to x$ such that $\vdash_{\mathrm{NLP}} X \to z$ and $z \to x \in \overline{\mathrm{Ax}}$. By inductive assumption $\|X\| \to \|z\|$ is derivable in NCL, and by Lemma 5(i), $\|z\| \to \|x\| \in \mathrm{Ax}$. Consequently, $\|X\| \to \|x\|$ is derivable in NCL from $\|X\| \to \|z\|$ and $\|z\| \to \|x\|$ *via* (C)-rule. The only case which needs some comment is that where $X \to x$ is the result of an application of (Perm). Then $X = (X_1 X_2)$ and, by inductive assumption $(\|X_2\|\,\|X_1\|) \to \|x\|$ is derivable in NCL. We have to show that it is the case for the formula $(\|X_1\|\,\|X_2\|) \to \|x\|$ as well. In a derivation of $(\|X_2\|\,\|X_1\|) \to \|x\|$ one of the rules (A), (A′), (PR) or (PR′) must be used at last once. Let us consider the last occurrence of a rule from this group in the derivation. This instance of the rule introduces the bracketed string $(\|X_2\|\,\|X_1\|)$ into the antecedens of the formula. As a result, in this step we get a formula $(\|X_2\|\,\|X_1\|) \to z$ which is then transformed to $(\|X_2\|\,\|X_1\|) \to \|x\|$ *via* a number (possibly zero) of applications of rule (C). Now, if the last afore mentioned rule is (A) (resp. (A′), (PR), (PR′)) then in order to obtain a derivation of $(\|X_1\|\,\|X_2\|) \to \|x\|$ we replace it by an instance of (A′) (resp. (A), (PR′), (PR)) and execute the same transformation by means of (C)-rule as in the transition from $(\|X_2\|\,\|X_1\|) \to z$ to $(\|X_2\|\,\|X_1\|) \to x$.

Now we prove the 'only if'–part of the thesis. Assume that $Y \to y$ is derivable in NCL and let $X \to x$ be any formula such that $Y \to y$ is its unidirectional version. If $Y \to y =$ (A0) then $Y = y$ and $\|X\| = y = \|x\|$. Thus, by Lemma 4, $X \to x$ is $\overline{\mathrm{C}}$-derivable in NLP, hence it is derivable in NLP. Let $Y \to y$ arises by (A). We have $Y = (Y_1 Y_2)$ and for some y_1 the formulas $Y_1 \to y/y_1$ and $Y_2 \to y_1$ are derivable in NCL. Any X such that $\|X\| = Y$ must be of the form $X = (X_1 X_2)$ such that $\|X_1\| = Y_1$ and $\|X_2\| = Y_2$. Let us take a type x such that $\|x\| = y$. As $\|y_1\| = y_1$ we can apply the inductive assumption to formulas $Y_1 \to \|x\|/\|y_1\|$ and $Y_2 \to \|y_1\|$. Thus $X_1 \to x/y_1$ and $Y_2 \to y_1$ are derivable in NLP, and so is the case for $(X_1 X_2) \to x$ as we can apply (A) in NLP to the latter formulas. If $Y \to y$ arises by (A′), then the argument is similar to that presented above: We apply the inductive assumption to formulas $Y_1 \to \|x\|/\|y_1\|$ and $Y_2 \to \|y_1\|$ in order to obtain formulas $X_1 \to y_1\backslash x$ and $X_2 \to y_1$ derivable in NLP, and then we employ the rule $(\overline{\mathrm{A}})$ to produce the formula $(X_2 X_1) \to x$. Let $Y \to y$ be a conclusion of the rule (PR). Thus $Y = (Y_1 Y_2)$, $y = y_1 \cdot y_2$, and if $Y \to y$ is a unidirectional version of some $X \to x$, the following equalities hold: $X = (X_1 X_2)$, $x = x_1 \cdot x_2$, $\|X_i\| = Y_i$, $\|x_i\| = y_i$, $i = 1, 2$. As, according to our inductive assumption, from the derivability of $Y_i \to y_i$, $i = 1, 2$, in NCL follows the derivability of $X_i \to x_i$, $i = 1, 2$, in NLP, it suffices to apply (PR) to the latter formulas to obtain the derivation of $X \to x$ in NLP. If $Y \to y$ arises by (PR′) then we proceed as in the previous case and get an in NLP derivable formula $(X_1 X_2) \to x_1 \cdot x_2$. Now it suffices to apply the rule (Perm). In the last case which we must take into consideration the formula $Y \to y$ arises in NCL by (C)-rule. Thus there must be a type z such that $z \to y \in \mathrm{Ax}$ and $Y \to y$ is derivable in NCL. By the inductive assumption, for any X such that $\|X\| = Y$ and any u such that $\|u\| = z$ the formula $X \to u$ is derivable in NLP. Let x be any type such that $\|x\| = y$. Consequently $\|u\| \to \|x\| \in \mathrm{Ax}$ and by Lemma 5(ii), $u \to x$ is $\overline{\mathrm{C}}$-derivable in NLP. By Lemma 3, there exists a $\overline{\mathrm{C}}$-reduction $u, \ldots, x$ in NLP. Thus by the definition of $\overline{\mathrm{C}}$-reduction, the formula $X \to x$ can be obtained from $X \to u$ by a number of applications of the $\overline{\mathrm{C}}$-rule. Hence $X \to x$ is derivable in NLP. □

3 The nonassociative and commutative Ajdukiewicz calculus

In general, Ajdukiewicz calculi NAP and NCA do not have a property which was proved for NLP and NCL in Theorem 1. As counterexamples one can take for instance a pair of formulas $X \to x = u/v \to v\backslash u$ and $Y \to y = u/v \to u/v$ or another pair $X \to x = (z/(u/v))(v\backslash u) \to z$ and $Y \to y = (z/(u/v))(u/v) \to z$ where $u, v, z \in \mathrm{Pr}$. Obviously, $Y \to y$ is a unidirectional version of $X \to x$ and $\vdash_{\mathrm{NCA}} Y \to y$ whereas it is not the case that $\vdash_{\mathrm{NAP}} X \to x$. However, a result similar to that presented in Theorem 1 can be proved for product-free versions of the calculi NAP and NCA in the case of a restricted class of formulas.

Let $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ (resp. $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$) denote a system formalized by (A0), (a_1), ($\mathrm{a}_{1'}$), ($\overline{\mathrm{a}_1}$), ($\overline{\mathrm{a}_{1'}}$) (resp. (A0), ($\mathrm{a}_1$) ($\mathrm{a}_{1'}$)) and (Cut) which employs types and bracketed strings of types from Tp and BSTp. By induction on the length of derivation one proves

Lemma 6 *Systems* $\mathrm{NAP}^{\mathrm{o}}$ *and* $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ *(resp.* $\mathrm{NCA}^{\mathrm{o}}$ *and* $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$*) are equivalent, i.e. for every formula* $X \to x$*, where* $X \in \mathrm{BSTp}$*,* $x \in \mathrm{Tp}$*, there holds the equivalence:* $\vdash_{\mathrm{NAP}^{\mathrm{o}}} X \to x$ *if and only if* $\vdash_{\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}} X \to x$ *(resp.* $\vdash_{\mathrm{NCA}^{\mathrm{o}}} X \to x$ *if and only if* $\vdash_{\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}} X \to x$*).*

We introduce the notion of *reduction* in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ and $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$ in a similar way as we did it for the calculus L. Moreover, similar as for L a characterization of derivability in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ and $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$ in terms of reduction can be given. We have

Fact 3 *A formula* $X \to x$ *is derivable in* $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ *(resp.* $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$*) if and only if* X *reduces to* x *in* $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ *(resp.* $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$*)*

The proof of this fact is similar to that presented in [26] or [14].

In Section 2 the definition of order of a type was given. It is easily observed that the order of a compound type does not depend whether we have left or right divisions in this type. Consequently, we have $\mathrm{o}(x)=\mathrm{o}(\|x\|)$.

Lemma 7 *If* $x \in Tp$ *and* $\mathrm{o}(x) \le 1$*, then* $\mathrm{o}(y) \le 1$ *for every subtype* y *of* x*.*

Proof. We follow the inductive definition of the set of subtypes of x. For x considered as a subtype of x there is nothing to show. Assume that for some subtype y of x we have $\mathrm{o}(y) \le 1$, and let $y = z/v$ (if $y = v\backslash z$, the argument is similar). Thus $\mathrm{o}(y) = \max\{\mathrm{o}(z),\mathrm{o}(v)+1\} \le 1$. Consequently we have $\mathrm{o}(v) = 0$ and $\mathrm{o}(z) \le 1$. □

Corollary 2 *Every non-primitive, product-free type of order* ≤ 1 *is of the form* y/p *or* $p\backslash y$ *where* $p \in \mathrm{Pr}$ *and* $\mathrm{o}(y) \le 1$*.*

Lemma 8 (i) *If* $X \in \mathrm{BSTp}$*,* $\|X\| = Y$ *and* $Y = Y[Z]$ *then there exists* $U \in \mathrm{BSTp}$ *such that* $X = X[U]$ *and* $\|U\| = Z$*.*
(ii) *If* $X, Y, Z \in \mathrm{BSTp}$*, then* $\|X[Y : Z]\| = \|X\|[\|Y\| : \|Z\|]$*.*

Proof. We use induction on construction of a string X which contains a placed occurrence of Y. □

Theorem 2 *If the formula* $Y \to y$ *is a unidirectional version of a formula* $X \to p$ *such that* $p \in \mathrm{Pr}$ *and all types in* X *are product-free and of order* ≤ 1 *then* $\vdash_{\mathrm{NAP}^{\mathrm{o}}} X \to p$ *if and only if* $\vdash_{\mathrm{NCA}^{\mathrm{o}}} Y \to y$*.*

Proof. The proof of the 'if'–part of the conclusion is in fact an inessential modification of the first part of the proof of Theorem 1. In order to prove the 'only if'–part of the conclusion let us first notice that by Fact 3, instead of derivability in $\mathrm{NAP}^{\mathrm{o}}$ and $\mathrm{NCA}^{\mathrm{o}}$ one can consider derivability in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ and $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$. Moreover, as $Y \to y = \|X\| \to \|p\|$ and $p \in \mathrm{Pr}$, thus $y = p = \|p\|$. As a result, having established the value of the succedents of both formulas, instead of discussing unidirectional versions of formulas we will only consider unidirectional versions of their antecedents. We employ induction on the length of a reduction of Y to p in $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$. If $Y \to p = (\mathrm{A0})$ then $Y = p$. Consequently, $X = p$ and $X \to p$, as an instance of (A0), is derivable in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$. If the length of a reduction of Y to p is 1, then $Y = (p/q\,q)$ or $Y = (q\,p/q)$. By Corollary 2, we have $q \in \mathrm{Pr}$. Thus any $X \in \mathrm{BSTp}$ such that $\|X\| = Y$ must be of one of the following forms: $(p/q\,q)$, $(q\backslash p\,q)$, $(q\,p/q)$ or $(q\,q\backslash p)$, where $p, q \in \mathrm{Pr}$. Obviously, $X \to p$ as axioms are derivable in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$. Let $Y \to p$ be derivable in $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$ and let $Y = Y_1, \ldots, Y_n = p$, $n > 2$ be a reduction of Y to p. According to the definition of reduction, $Y_1 = Y_1[Z]$, where $Z = (x/q\,q)$ or $Z = (q\,x/q)$, and $Y_2 = Y_1[Z : x]$. By Corollary 2, $q \in \mathrm{Pr}$, and by Lemma 7, $\mathrm{o}(x) \leq 1$. Let $X \in \mathrm{BSTp}$ be such that $\|X\| = Y_1$. By Lemma 8(i), $X = X[U]$ and $\|U\| = Z$. Thus U is of the form $(u/q\,q)$, or $(q\backslash u\,u)$, or $(q\,u/q)$, or $(q\,q\backslash u)$, for some $u \in \mathrm{Tp}$ such that $\|u\| = x$. Let us consider $X[U : u]$. According to Lemma 8(ii) we have $\|X[U : u]\| = \|X\|[\|U\| : \|u\|] = Y_1[Z : x] = Y_2$. Consequently, as $Y_2 \to p$ is derivable in $\mathrm{NCA}^{\mathrm{o}}_{\mathrm{c}}$, by the inductive assumption we get derivability of $X[U : u] \to p$. Thus, there exists a reduction $X[U : u], \ldots, p$ in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$. But $U \to u$ is an instance of a non-(A0) axiom in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$, hence the sequence $X[U], X[U : u], \ldots, p$ is a reduction in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$ as well. By Fact 3, the formula $X[U] \to p$, i.e. the formula $X \to p$ is derivable in $\mathrm{NAP}^{\mathrm{o}}_{\mathrm{c}}$. □

4 The inclusion of the class of NCL-languages in the class of $\overline{\mathrm{CF}}$-languages

In this section we slightly refine the result from [17] about the relation between the class of languages generated by categorial grammars based on the calculus NCL and the class of CF-languages. The argument presented in both remaining parts of the paper essentially makes use of concepts and constructions included in former papers concerning the subject. However, to make this work self–contained, we briefly summarize in this section all relevant facts and provide the reader with necessary references.

Let V be a finite vocabulary. The set $\mathrm{BS}(V)$ of *phrase structures over* V is defined as the smallest set such that: (i) $V \subseteq \mathrm{BS}(V)$, (ii) if $A_1, \ldots, A_n \in \mathrm{BS}(V)$, then $(A_1 \ldots A_n) \in \mathrm{BS}(V)$. We will denote by $|A|$ the sequence arising from a phrase structure A by deleting all brackets. Any subset L of $\mathrm{BS}(V)$ is called a *phrase language over* V.

The set $\mathrm{BS}(V)$ provided with operations $f_n(A_1, \ldots, A_n) = (A_1 \ldots A_n)$, $n = 2, 3, 4, \ldots$ can be considered as an absolutely free algebra over the set of generators V. The largest congruence with respect to inclusion on a phrase language L, denoted by INT_L, is called the *intersubstitutability relation for* L, see [10,15]. We refer to the index of the relation INT_L as to the *index of* L and denote this number by $\mathrm{ind}(L)$.

Given a phrase structure $A = (A_1 \ldots A_n)$ we call $A_1, \ldots, A_n$ as to the *direct substructures of* A. The set $\mathrm{sub}(A)$ of *substructures of* A is defined in a natural way: (i) $A \in \mathrm{sub}(A)$, (ii) if $B \in \mathrm{sub}(A)$ and C is a direct substructure of A, then $C \in \mathrm{sub}(A)$. The *size of* $A \in \mathrm{BS}(V)$, denoted by $\mathrm{s}(A)$, is the maximum number of direct substructures in any element of $\mathrm{sub}(A)$. For $L \subseteq \mathrm{BS}(V)$ we put $\mathrm{s}(L) = \sup\{\mathrm{s}(A) : A \in L\}$ and call $\mathrm{s}(L)$ the *size of* L.

By a *path in* $A \in \mathrm{BS}(V)$ we mean a sequence $A_0, A_1, \ldots, A_n$ of substructures of A such that for all $1 \leq i \leq n$, A_i is a direct substructure of A_{i-1}. The *external degree* *of* $A \in \mathrm{BS}(V)$, denoted by $\deg^{\mathrm{e}}(A)$, is defined to be the minimal length of paths in A whose initial term is A itself and whose final term is an element from V. For $A \in \mathrm{BS}(V)$ we put

$$\deg(A) = \max\{\deg^{\mathrm{e}}(B) : B \in \mathrm{sub}(A)\}$$

and for $L \subseteq \mathrm{BS}(V)$ let

$$\deg(L) = \sup\{\deg(A) : A \in L\}$$

and we call those numbers the *degree of* A and the *degree of* L, respectively.

Any calculus of syntactic types described in the paper can play the role of a *type reduction system* in a categorial grammar. A *categorial grammar over a (nonassociative) type reduction system* TRS is an ordered quadruple $G = \langle V_G, I_G, s_G, \mathrm{TRS}\rangle$, where V_G is an nonempty *vocabulary of* G, s_G is a *distinguished primitive type* understood as a type of properly built sentences, $I_G \subseteq V_G \times \mathrm{TP}$ is a finite relation called the *initial type assignment of* G and TRS is a certain nonassociative calculus of syntactic types. In a natural way I_G can inductively be extended to a relation $F_G \subseteq \mathrm{BS}(V_G)\times$ BSTP: (i) $I_G \subseteq F_G$, (ii) if $(A_1, X_1) \in F_G$ and $(A_2, X_2) \in F_G$, then $((A_1A_2), (X_1X_2)) \in F_G$. By the *string* (resp. *phrase*) *language* $\mathrm{L}(G)$ (resp. $\mathrm{BL}(G)$) *generated by a categorial grammar* G *over* TRS we mean the set

$$\mathrm{L}(G) = \{|A| : (\exists X \in \mathrm{BSTP})((A, X) \in F_G \;\&\; \vdash_{\mathrm{TRS}} X \to s_G)\}$$

(resp. $\mathrm{BL}(G) = \{ A : (\exists X \in \mathrm{BSTP})((A, X) \in F_G \;\&\; \vdash_{\mathrm{TRS}} X \to s_G)\}$).

As $\mathrm{L}(G) = \{|A| : A \in \mathrm{BL}(G)\}$ thus, if two CG's generate the same phrase languages, then they also generate the same string languages; the converse implication however does not hold. If there is no special reason we call string languages generated by CG's simply languages. A categorial grammar in which TRS = NA (resp. NCA, NAP, NCL, NLP) will be called to as an NA- (resp. NCA-, NAP-, NCL-, NLP-)*grammar*. We adopt this convention for product-free calculi as well. It is easily observed that all phrase languages generated by NA-, NCA-, NAP-, NCL- or NLP-grammars are of size ≤ 2.

The following theorem establishes the equivalence of NA- and NA^{o}-grammars within the scope of phrase languages, thus within the scope of string languages as well (see [15] where this result is given in a stronger form):

Theorem 3 NA-*grammars and* NA^{o}-*grammars generate the same class of phrase languages.*

By the *order of a product-free categorial grammar* G we mean the number $\mathrm{o}(G)$ such that $\mathrm{o}(G) = \sup\{\mathrm{o}(x) : (\exists v \in V_G)((v, x) \in I_G)\}$. The following theorem was proved in [10] (in a stronger case of functorial languages):

Theorem 4 *Any phrase language generated by an* NA^{o}-*grammar is also generated by an* NA^{o}-*grammar* G *such that* $\mathrm{o}(G) \leq 1$.

Lemma 9 $\mathrm{NAP}^{\mathrm{o}}$-*grammars of the order* ≤ 1 *generate the same class of phrase languages as* $\mathrm{NCA}^{\mathrm{o}}$-*grammars of the order* ≤ 1.

Proof. Let $G = \langle V_G, I_G, s_G, \mathrm{NAP}^{\mathrm{o}}\rangle$ be an NAP$^{\mathrm{o}}$-grammar and let $\mathrm{o}(G) \leq 1$. We define an NCA$^{\mathrm{o}}$-grammar G_1 in the following way: $V_{G_1} = V_G$, $s_{G_1} = s_G$, and for $a \in V_G$, $x \in \mathrm{TP}$, $(a, x) \in I_G$ if and only if $(a, \|x\|) \in I_{G_1}$. It is clear that $\mathrm{o}(G) = \mathrm{o}(G_1) \leq 1$. By induction on the complexity of A one can easily prove that $(A, X) \in F_G$ if and only if $(A, \|X\|) \in F_{G_1}$. Assume $A \in \mathrm{BL}(G)$. Thus there exists $X \in \mathrm{BSTp}$ such that $(A, X) \in F_G$ and $\vdash_{\mathrm{NAP}^{\mathrm{o}}} X \to s_G$. Consequently $(A, \|X\|) \in F_{G_1}$. Formulas $X \to s_G$ and $\|X\| \to s_{G_1}$ fulfil the assumptions of Theorem 2, hence $\vdash_{\mathrm{NCA}^{\mathrm{o}}} \|X\| \to s_{G_1}$. It proves that $A \in \mathrm{BL}(G_1)$. To prove that $\mathrm{BL}(G_1) \subseteq \mathrm{BL}(G)$ assume $A \in \mathrm{BL}(G_1)$. There exists then $X \in \mathrm{BSTp}$, consisting of unidirectional types, such that $(A, X) \in F_{G_1}$ and $\vdash_{\mathrm{NCA}^{\mathrm{o}}} X \to s_{G_1}$. Thus one can find $Y \in \mathrm{BSTp}$ such that $(A, Y) \in F_G$ and $\|Y\| = X$. Employing Theorem 2 once more we conclude that $\vdash_{\mathrm{NAP}^{\mathrm{o}}} Y \to s_G$, thus $A \in \mathrm{BL}(G)$.

Conversely, let $\mathrm{L} = \mathrm{L}(G)$ for some NCA$^{\mathrm{o}}$-grammar $G = \langle V_G, I_G, s_G, \mathrm{NCA}^{\mathrm{o}}\rangle$ of order ≤ 1, and let G_1 be an NAP$^{\mathrm{o}}$-grammar whose components are described as follows: $V_{G_1} = V_G$, $s_{G_1} = s_G$, and for all $a \in V_{G_1}$ and $x, y \in \mathrm{Tp}$, $(a, y) \in I_{G_1}$ if and only if $(a, x) \in I_G$ and $\|y\| = x$. As there exists only a finite number of types y such that for a given type x the equality $\|y\| = x$ holds, I_{G_1} is still a finite relation. Proceeding essentially in the same way as previously we show that $\mathrm{BL}(G) = \mathrm{BL}(G_1)$. □

The next theorem gives characterization of phrase languages generated by NA$^{\mathrm{o}}$-grammars (see [9]):

Theorem 5 *A phrase language L such that $\mathrm{s}(L) \leq 2$ is generated by an* NA$^{\mathrm{o}}$*-grammar if and only if both* $\mathrm{ind}(L)$ *and* $\deg(L)$ *are finite.*

We admit a standard definition of a (λ-free) CF-grammar as an ordered quadruple $\langle V, U, s, P\rangle$ in which symbols V, U, s, P denote, respectively, the set of *terminals*, the set of *nonterminals*, the *initial symbol* and the *set of production rules*. We adopt the notation $a \mapsto b_1 \ldots b_n$ for elements of P. A production rule $a \mapsto b_1 \ldots b_n$ is called a *permutation variant of* $a \mapsto c_1 \ldots c_n$ if the sequence $b_1 \ldots b_n$ is a permutation of the sequence $c_1 \ldots c_n$. In case of a binary rule, i.e. when $n = 2$ we use the term 'transposition variant' instead of 'permutation variant'. A set of production rules is *closed with respect to permutations* if together with a certain rule it also contains all permutation variants of this rule. A CF-grammar (resp. CF-language) is called *closed with respect to permutations* if is closed with respect to permutations its set of production rules (resp. a CF-grammar generating this language). The definitions of *closed with respect to transposition* for grammars and for languages are similar.

Every CF-grammar induces in a natural way a bracketing on the elements of the generated (string) language. Thus, together with a (string) language $\mathrm{L}(\mathcal{G}) \subseteq V^*$, a CF-grammar $\mathcal{G}$ generates also a *phrase language* $\mathrm{BL}(\mathcal{G}) \subseteq \mathrm{BS}(V)$ such that $\mathrm{L}(\mathcal{G}) = \{|A| : A \in \mathrm{BL}(\mathcal{G})\}$. The following theorem provides a necessary and sufficient condition for a phrase language L to be generated by a CF-grammar (see [23]):

Theorem 6 *A phrase language L is generated by a* CF*-grammar if and only if* $\mathrm{s}(L)$ *and* $\mathrm{ind}(L)$ *are finite.*

The following definition is essential for our further considerations: A CF-grammar $\mathcal{G}$ is called a $\overline{\mathrm{CF}}$*-grammar* if (i) $\mathcal{G}$ is closed with respect to permutation and (ii) $\deg(\mathrm{BL}(\mathcal{G}))$ is finite.

Let $x \to y \in \mathrm{Ax}$. We call $x \to y$ an E-*formula* (resp. an R-*formula* or an O-*formula*) if $c(x) < c(y)$ (resp. $c(x) > c(y)$ or $c(x) = c(y)$). An instance of the (C)-rule employing an E- (resp. R-, O-) formula is called an E- (resp. R-, O-) *instance* of this rule. We call a derivation D of $X \to y$ in NCL *seminormal* if all E-instances of (C) follow R-instances of (C) as well as the rules (A), (A$'$), (PR) and (PR$'$). We call a derivation D of $X \to y$ in NCL *normal* if it is seminormal and additionally if all R-instances of (C) precede (A), (A$'$), (PR) and (PR$'$) and no O-instance of (C) is placed between (A), (A$'$), (PR) and (PR$'$).

In [17] the following theorem was proved:

Theorem 7 *If $\vdash_{\mathrm{NCL}} X \to x$, then any derivation D of $X \to x$ can be transformed effectively to a normal form.*

The existence of normal form for derivations in NCL provided by Theorem 7 enables us to construct for every NCL-grammar an NCA-grammar generating the same language. The reader can find details of this construction in [9] or in [14]. Here we only formulate this result in the form of the following lemma:

Lemma 10 *For every* NCL-*grammar G one can construct an* NCA-*grammar G_1 such that* $\mathrm{L}(G) = \mathrm{L}(G_1)$.

Let $G = \langle V_G, I_G, s_G, \mathrm{NCA}\rangle$ be an NCA-grammar. We define a CF-grammar $\mathcal{G}$ in the following way: $V_{\mathcal{G}} = V_G, s_{\mathcal{G}} = s_G$ and $U_{\mathcal{G}} = \mathrm{sub}(\{x \in \mathrm{TP}\colon (\exists v \in V_G)((v, x) \in I_G\})$. The set $P_{\mathcal{G}}$ of production rules contains all rules of one of the shapes: (i) $x \mapsto v$ where $x \in \mathrm{TP}$, $v \in V_G$, and $(v, x) \in I_G$, or (ii) $x \mapsto x/y\ y$, $x \mapsto y\ x/y$, $x \cdot y \mapsto x\ y$, $x \cdot y \mapsto y\ x$, for all $x, y \in U_{\mathcal{G}}$. For the grammar $\mathcal{G}$ defined as above we have:

Lemma 11 $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(G)$.

For details of the described construction and the proof of the lemma see the previously mentioned references.

The following lemma is a straightforward consequence of Lemma 11 and the presented construction:

Lemma 12 *For every* NCA-*grammar G one can find a* CF-*grammar $\mathcal{G}$ such that $\mathcal{G}$ is closed with respect to transpositions and* $\mathrm{BL}(G) = \mathrm{BL}(\mathcal{G})$.

Lemma 13 *Let G be an* NCA-*grammar. If $\mathcal{G}$ is a* CF-*grammar constructed as above, then* $\deg(\mathrm{BL}(\mathcal{G})) < \aleph_0$.

Proof. By Lemma 11, $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(G)$, thus it is sufficient to show the finiteness of $\deg(\mathrm{BL}(G))$. Let $G = \langle V_G, I_G, s_G, \mathrm{NCA}\rangle$ and let $\overline{\mathrm{NCA}}$ denote the calculus obtained from NCA by dropping the rules (A$'$) and (PR$'$), thus employing only (A) and (PR). We put $\overline{G} = \langle V_G, I_G, s_G, \overline{\mathrm{NCA}}\rangle$. Obviousely $\mathrm{BL}(\overline{G}) \subseteq \mathrm{BL}(G)$. Observe that $\mathrm{BL}(G)$ arises from $\mathrm{BL}(\overline{G})$ by adding all phrase structures obtained by a finite number of transpositions of direct substructures in substructures of elements of $\mathrm{BL}(\overline{G})$. However, the degree of any phrase structure B obtained by transpositions from a given structure A is the same as the degree of the structure A itself. Thus $\deg(\mathrm{BL}(\overline{G})) = \deg(\mathrm{BL}(G))$ and we will show that $\deg(\mathrm{BL}(\overline{G}))$ is finite. For this we prove that for some phrase language L_0 such that $\mathrm{BL}(\overline{G}) \subseteq L_0$ we have $\deg(L_0) < \aleph_0$. We put the calculus NA instead of $\overline{\mathrm{NCA}}$ in the definition of $\overline{G}$ and denote the obtained grammar by G_0. Let $L_0 = \mathrm{BL}(G_0)$. Every in $\overline{\mathrm{NCA}}$ derivable formula is also derivable in NA, thus we get the inclusion $\mathrm{BL}(\overline{G}) \subseteq L_0$. By Theorem 3, the language L_0 being generated by an NA-grammar is also generated by an NA$^{\mathrm{o}}$-grammar. Consequently, as $\mathrm{s}(L_0) \leq 2$ thus according to Theorem 5, the number $\deg(L_0)$ is finite. Thus we have $\deg(\mathrm{BL}(\mathcal{G})) = \deg(\mathrm{BL}(G)) = \deg(\mathrm{BL}(\overline{G})) \leq \deg(L_0) < \aleph_0$. □

Lemmas 2, 4 and 5 give

Theorem 8 *For every* NCL-*grammar G one finds an* $\overline{\mathrm{CF}}$-*grammar $\mathcal{G}$ such that* $\mathrm{L}(G) = \mathrm{L}(\mathcal{G})$, *i.e. the class of languages generated by* NCL-*grammars is included in the class of languages generated by* $\overline{\mathrm{CF}}$-*grammars.*

5 The equivalence of $\overline{\mathrm{CF}}$-grammars and NCL-grammars.

Lemma 14 *For every* CF-*grammar $\mathcal{G}$ one can find a* CF-*grammar $\mathcal{G}'$ such that* $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(\mathcal{G}')$ *and all the production rules in the grammar $\mathcal{G}'$ are of one of the following forms: $a \mapsto b_1 \dots b_n$, $n \geq 2$, or $a \mapsto v$, where $a, b_1, \dots, b_n$ are nonterminals and v is a terminal in $\mathcal{G}'$.*

Proof. This lemma usually constitutes a part of the proof of the Chomsky normal form theorem, see for example [13]. The equality $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(\mathcal{G}')$ is a straihgtforward consequence of the fact that the employed in the proof procedures of getting rid of unit productions (i.e. of productions of the form $a \mapsto b$) as well as of productions containing terminals on right-hand sides do not affect the phrase structure of elements of the generated language. □

Lemma 15 *Let $\mathcal{G} = \langle V_{\mathcal{G}}, U_{\mathcal{G}}, s_{\mathcal{G}}, P_{\mathcal{G}} \rangle$ be a* $\overline{\mathrm{CF}}$-*grammar. Then there exists a* CF-*grammar $\mathcal{G}'$ in Chomsky normal form which is closed with respect to transposition such that* $\mathrm{L}(\mathcal{G}) = \mathrm{L}(\mathcal{G}')$ *and* $\deg(\mathrm{BL}(\mathcal{G}')) < \aleph_0$.

Proof. According to Lemma 14 we can assume that $P_{\mathcal{G}} = \overline{\overline{P_{\mathcal{G}}}} \cup \overline{P_{\mathcal{G}}}$, where $\overline{\overline{P_{\mathcal{G}}}}$ consists of productions of the form $a \mapsto b_1 \dots b_n, n \geq 2$ and $\overline{P_{\mathcal{G}}}$ consists of productions of the form $a \mapsto v$, where $a, b_1, \dots, b_n \in U_{\mathcal{G}}$ and $v \in V_{\mathcal{G}}$. We show that every set of production rules which comprises all permutation variants of a given production rule can be replaced by a set of binary rules which is closed with respect to transpositions. Let $R = a \mapsto b_1 \dots b_n \in \overline{\overline{P_{\mathcal{G}}}}$. If $n = 2$, then there is nothing to show as, according to our assumptions, both $a \mapsto b_1 b_2$ and $a \mapsto b_2 b_1$ are in $\overline{\overline{P_{\mathcal{G}}}}$ and they constitute the desired set of binary rules. For $n \geq 3$ we replace R by a set F_R^0 consisting of the rules $a \mapsto b_1 c_1$, $c_1 \mapsto b_2 c_2, \dots, c_{n-2} \mapsto b_{n-1} b_n$ as we usually do in the construction of the Chomsky normal form for a CF-grammar ($c_1, \dots, c_{n-2}$ are new nonterminals). Then we add to F_R^0 all transposition variants of its elements and denote the set obtained in this way by F_R. Observe that for generation of a language over $V_{\mathcal{G}}$ only those strings derivable from a are essential which consist of nonterminals $b_1, \ldots, b_n$ but not $c_1, \ldots, c_{n-2}$. However, due to the form of the rules in F_R, every string derivable from a by means of those rules, which consists exclusively of nonterminals $b_1, \ldots, b_n$ must contain all of them, additionally, together with a sequence $b_1 \ldots b_n$, some of its permutations can also be derived from a by means of productions from F_R. Thus, F_R is a substitute for the rule R as well as for some of its permutation variants. The described procedure can be performed for all permutation variants of R (all new nonterminals must differ one from another in order to avoid an interaction of rules). The set of all binary rules obtained in this way for R and all its permutation variants will be denoted by $F_{Perm(R)}$. This set produces the same strings as the rule R and its permutation variants and no other strings. $F_{Perm(R)}$ is also closed with respect to transposition. We define $\overline{P_{\mathcal{G}'}} = \overline{P_{\mathcal{G}}}$, $\overline{\overline{P_{\mathcal{G}'}}} = \bigcup\{F_{Perm(R)} : R \in \overline{\overline{P_{\mathcal{G}}}}\}$, $P_{\mathcal{G}'} = \overline{\overline{P_{\mathcal{G}'}}} \cup \overline{P_{\mathcal{G}'}}$ $s_{\mathcal{G}'} = s_{\mathcal{G}}$, $V_{\mathcal{G}'} = V_{\mathcal{G}}$, and let $U_{\mathcal{G}'}$ consist of all nonterminals from $U_{\mathcal{G}}$ as well as of all new nonterminals introduced in the process of constructing the sets $F_{Perm(R)}$ for all R's. We put $\mathcal{G}' = \langle V_{\mathcal{G}'}, U_{\mathcal{G}'}, s_{\mathcal{G}'}, P_{\mathcal{G}'}\rangle$. The set $\overline{\overline{P_{\mathcal{G}'}}}$ produces precisely the same strings over $U_{\mathcal{G}'}$ as $\overline{\overline{P_{\mathcal{G}}}}$ over $U_{\mathcal{G}}$ and consequently, as $\overline{P_{\mathcal{G}'}} = \overline{P_{\mathcal{G}}}$, we have $\mathrm{L}(\mathcal{G}') = \mathrm{L}(\mathcal{G})$.

Now we show that $\deg(\mathrm{BL}(\mathcal{G}')) < \aleph_0$. The replacement of a rule $a \mapsto b_1 \ldots b_n$, $n \geq 3$ by a set of binary rules introduces a (binary) phrase structure on $b_1 \ldots b_n$ and consequently makes the phrase structure of elements of $\mathrm{BL}(\mathcal{G}')$ finer than that we have in $\mathrm{BL}(\mathcal{G})$. As a result, the length of paths leading from any substructure of an element of $\mathrm{BL}(\mathcal{G}')$ to an atom (terminal) can increase. However, for any $A \in \mathrm{BL}(\mathcal{G})$ and $A' \in \mathrm{BL}(\mathcal{G}')$ such that $|A| = |A'|$ we have $\deg(A') \leq \deg(A)\cdot(\mathrm{s}(\mathrm{BL}(G)) - 1) < \deg(A)\cdot \mathrm{s}(\mathrm{BL}(\mathcal{G}))$. But $\mathrm{s}(\mathrm{BL}(\mathcal{G}))$ is finite (it is the maximal length of strings on the right-hand sides of production rules from $P_{\mathcal{G}}$) and $\deg(\mathrm{BL}(\mathcal{G}))$ is finite as well ($\mathcal{G}$ is a $\overline{\mathrm{CF}}$-grammar). Therefore $\deg(\mathrm{BL}(\mathcal{G})) = \sup\{\deg(A') : A' \in \mathrm{BL}(\mathcal{G}')\} \leq \sup\{\deg(A)\cdot\mathrm{s}(\mathrm{BL}(\mathcal{G})) : A \in \mathrm{BL}(\mathcal{G}) \,\&|A| = |A'|\} = \deg(\mathrm{BL}(\mathcal{G}))\cdot\mathrm{s}(\mathrm{BL}(\mathcal{G})) < \aleph_0$. □

Lemma 16 *If $\mathcal{G}$ is a $\overline{\mathrm{CF}}$-grammar in Chomsky normal form, then* $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(G)$ *for some* $\mathrm{NCA}^{\mathrm{o}}$*-grammar G of order* ≤ 1.

Proof. By Theorem 6, $\mathrm{ind}(\mathrm{BL}(\mathcal{G})) < \aleph_0$ and $\mathrm{s}(\mathrm{BL}(\mathcal{G})) < \aleph_0$ (the second inequality is not important because in our case we have $\mathrm{s}(\mathrm{BL}(\mathcal{G})) \leq 2$). Since $\deg(\mathrm{BL}(\mathcal{G})) < \aleph_0$, by Theorem 5 we conclude that $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(G_0)$, for some NA^{o}-grammar $G_0 = \langle V_{G_0}, I_{G_0}, s_{G_0}, \mathrm{NA}^{\mathrm{o}}\rangle$. According to Theorem 4, we can assume that $\mathrm{o}(G_0) \leq 1$. Let $G_1 = \langle V_{G_0}, I_{G_0}, s_{G_0}, \mathrm{NAP}^{\mathrm{o}}\rangle$. As $\mathrm{NAP}^{\mathrm{o}}$ is stronger than NA^{o}, we immediately have $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(G_0) \subseteq \mathrm{BL}(G_1)$. But we also have $\mathrm{BL}(G_1) \subseteq \mathrm{BL}(\mathcal{G})$: this follows from the fact that $\mathcal{G}$ is closed with respect to transposition, thus the language $\mathrm{BL}(\mathcal{G})$ (and $\mathrm{BL}(G_0)$) is closed with respect to transpositions of its substructures. As a result, adding the rule (Perm) to NA^{o}, i.e. employing $\mathrm{NAP}^{\mathrm{o}}$as a type reduction system instead of NA^{o}, does not lead us beyond the language generated by G_0. Now, as $\mathrm{o}(G_1) \leq 1$, we apply Lemma 9 and get an $\mathrm{NCA}^{\mathrm{o}}$-grammar G of the order ≤ 1 such that $\mathrm{BL}(G) = \mathrm{BL}(G_1)$. Consequently $\mathrm{BL}(\mathcal{G}) = \mathrm{BL}(G)$. □

Lemma 17 *For every* $\mathrm{NCA}^{\mathrm{o}}$*-grammar* G_0 *of the order* ≤ 1 *there exists an* NCL*-grammar* G_1 *such that* $\mathrm{BL}(G) = \mathrm{BL}(G_1)$.

Proof. We adopt a standard argument presented for example in [9] or in [15] but suited here to the case of commutative calculi. The axiomatization of $\mathrm{NCA}^{\mathrm{o}}$ we use consists of the axiom scheme (A0) and the rules (A) and (A$'$). Given an NCA$^{\text{o}}$-grammar $G = \langle V_G, I_G, s_G, \text{NCA}^{\text{o}}\rangle$ we put $G_1 = \langle V_G, I_G, s_G, \text{NCL}\rangle$ and claim that $\text{BL}(G) = \text{BL}(G_1)$. In order to obtain this equality it is sufficient to prove that $\vdash_{\text{NCA}^{\text{o}}} X \rightarrow s_G$ if and only if $\vdash_{\text{NCL}} X \rightarrow s_G$, unless all types in X are product-free and of order ≤ 1. It is obvious that every formula derivable in NCA$^{\text{o}}$ is also derivable in NCL because NCA$^{\text{o}}$ is a subsystem of NCL. To prove the converse implication let us assume $\vdash_{\text{NCL}} X \rightarrow s_G$. By Theorem 7, the formula $X \rightarrow s_G$ possesses a normal derivation D in NCL. Since $s_G \in \text{Pr}$, no E-instances of (C)-rule occur in D. For every type x in X any R-instance of (C)-rule would employ such a formula $x \rightarrow y$ from Ax that $c(x) > c(y)$. This formula can not be obtained by means of the rules (R2) or (R2$'$) because of the presence of the product sign $\cdot$ in their conclusions. For axioms (A1), (A1$'$), (A2), (A2$'$) as well as for formulas in Ax which arise from them by an application of rules (R1) or (R1$'$), one sees that this side of a formula which is of greater complexity contains also a product sign. The formula $x \rightarrow y$ can not be obtained from (A4) by any rule from Ax as well, otherwise we would have $c(x) = c(y)$. The only remaining possibility of constructing $x \rightarrow y$ is that using (A3) and the rules (R1) or (R1$'$). For any $z, t \in \text{Tp}$ we have however $\text{o}(z/(z/t)) = \max\{\text{o}(z), \text{o}(t) + 2\}$, and thus $\text{o}(z/(z/t)) \geq 2$. Consequently, this side of the formula $x \rightarrow y$ which is of greater complexity would have the order ≥ 2, but for $x \rightarrow y$ being an R-formula this is impossible, as $\text{o}(G) \leq 1$. We conclude that no R-formulas are employed in D, thus D is a derivation in NCA. But the rules (PR) and (PR$'$) can not be applied in D, otherwise for some types z, t a product type $z \cdot y$ would be a subtype of a type in X. As a result, D is a derivation in NCA$^{\text{o}}$ and $\vdash_{\text{NCA}^{\text{o}}} X \rightarrow s_G$. □

Theorem 9 *For any* $\overline{\text{CF}}$*-grammar* $\mathcal{G}$ *there exists an* NCL*-grammar* G *such that* $\text{L}(G) = \text{L}(\mathcal{G})$.

Proof. We conclude from Lemma 15 that for the grammar $\mathcal{G}$ one can construct a CF-grammar $\mathcal{G}'$ in Chomsky normal form, closed with respect to transposition and such that $\text{L}(\mathcal{G}) = \text{L}(\mathcal{G}')$ and $\deg(\text{BL}(\mathcal{G}')) < \aleph_0$. Employing Lemma 16 we find for $\mathcal{G}'$ an NCA$^{\text{o}}$-grammar G_1 such that $\text{o}(G_1) \leq 1$ and $\text{BL}(\mathcal{G}') = \text{BL}(G_1)$. By Lemma 17, we can find for G_1 an NCL-grammar G such that $\text{BL}(G_1) = \text{BL}(G)$. Accordingly, as $\text{L}(\mathcal{G}') = \text{L}(G_1) = \text{L}(G)$, G is the grammar fulfilling the thesis. □

Theorem 10 NLC*-grammars and* $\overline{\text{CF}}$*-grammars generate the same class of (string) languages.*

Proof. This is a consequence of Theorem 8 and Theorem 9. □

Note. It is not known however, whether the finiteness of the degree in the definition of $\overline{\text{CF}}$-grammars is an essential restriction. So far we do not know if the question whether for every closed with respect to permutation CF-grammar $\mathcal{G}$ one can construct a closed with respect to permutation CF-grammar $\mathcal{G}'$ such that $\text{L}(\mathcal{G}) = \text{L}(\mathcal{G}')$ and $\deg(\text{BL}(\mathcal{G}')) < \aleph_0$ has a positive or negative answer.

References

[1] Bach, E., Some generalizations of categorial grammars. In: Varieties of Formal Semantics (Landman, F. and F. Veltman, eds.) Foris, Dordrecht 1984, 1–23.

[2] Bar-Hillel, Y., C. Gaifman and E. Shamir, On categorial and phrase structure grammars. Bulletin of the Research Council of Israel F.**9** (1960), 1–16.

[3] Benthem, J. van, The Lambek calculus. In: Categorial Grammars and Natural Language Structures. Studies in Linguistics and Philosophy (Oehrle, R., E. Bach and D. Wheeler, eds.), D. Reidel, Dordrecht 1988, 35–68.

[4] Benthem, J. van, The semantics of variety in categorial grammar. In: Categorial grammar (W. Buszkowski, W. Marciszewski and J. van Benthem, eds.), J. Benjamins, Amsterdam 1988, 37–56.

[5] Benthem, J. van, Semantic type change and syntactic recognition. In: Properties, Types and Meaning, vol I: Foundational Issues, vol II: Semantic Issues (Chierchia, G., B. Partee and R. Turner, eds.), Kluwer, Dordrecht 1989, 231–249.

[6] Benthem, J. van, Language in Action. Categories, Lambdas and Dynamic Logic. Studies in Logic and the Foundations of Mathematics, North–Holland, Amsterdam 1991.

[7] Buszkowski, W., A note on the Lambek-van Benthem calculus. Bulletin of the Section of Logic **131** (1984), 31–37.

[8] Buszkowski, W., The equivalence of unidirectional Lambek categorial grammars and context-free grammars. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **31** (1985), 369–384.

[9] Buszkowski, W., Generative capacity of nonassociative Lambek calculus. Bulletin of the Polish Academy of Sciences: Mathematics **34** (1986), 507–516.

[10] Buszkowski, W., Typed functorial languages. Bulletin of the Polish Academy of Sciences: Mathematics **34** (1986), 495–505.

[11] Buszkowski, W., On generative capacity of the Lambek calculus. In: Logics in AI (Eijck, J. van, ed.), Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin–Heidelberg–New York 1991, 139–152.

[12] Gentzen, G., Untersuchungen über das logische Schliessen I–II. Mathematische Zeitschrift **39** (1934), 176–210, 405–431.

[13] Hopcroft, J. and J. Ullman, Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company, Reading, Massachusetts 1979.

[14] Kandulski, M., The equivalence of nonassociative Lambek categorial grammars and context-free grammars. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **34** (1988), 41–52.

[15] Kandulski, M., Phrase structure languages generated by categorial grammars with product. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **34** (1988), 373–383.

[16] Kandulski, M., The non-associative Lambek calculus. In: Categorial Grammar (W. Buszkowski, W. Marciszewski and J. van Benthem, eds.), Benjamins, Amsterdam 1988, 141–152.

[17] Kandulski, M., Normal form of derivations in the nonassociative and commutative Lambek calculus with product. Mathematical Logic Quarterly **39** (1993), 103–114.

[18] Lambek, J., The mathematics of sentence structure. American Mathematical Monthly **65** (1958), 154–170.

[19] Lambek, J., On the calculus of syntactic types. In: Structure of Language and Its Mathematical Aspects (Jacobson, R., ed.), Amer. Math. Soc., Providence, R.I., 1961, 166–178.

[20] Moortgat, M., Labelled Deductive Systems for categorial theorem proving. OTS Working Papers, OTS-WP-CL-92-003, Research Institute for Language and Speech, Rijksuniversiteit Utrecht, 1992.

[21] Moortgat, M. and G. Morrill, Heads and phrases. Type calculus for dependency and constituent structure. ms., OTS Utrecht, (to appear in Journal of Logic, Language and Information).

[22] Pentus, M., Lambek grammars are context-free. Unpublished manuscript, 1992.

[23] Thatcher, J.W., Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. Journal Comput. Systems Sci. **1** (1967), 317–322.

[24] Troelstra, A.S., Lectures on Linear Logic. CSLI Lecture Notes, vol.29, Center for the Study of Language and Information, Stanford University 1992.

[25] Zielonka, W., A direct proof of the equivalence of free categorial grammars and simple phrase structure grammars. Studia Logica **37** (1978), 41–57.

[26] Zielonka, W., Axiomatizability of Ajdukiewicz-Lambek calculus by means of cancellation schemes. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **27** (1981), 215–224.

[27] Zielonka, W., A simple and general method of solving the finite axiomatizability problems for Lambek's syntactic calculi. Studia Logica **48** (1989), 35–39.