Completeness of Full Lambek Calculus for Syntactic Concept Lattices

Christian Wurm

Fakultät für Linguistik und Literaturwissenschaften,
CITEC Universität Bielefeld
cwurm@uni-bielefeld.de

Abstract. Syntactic concept lattices are residuated structures which arise from the distributional analysis of a language. We show that these structures form a complete class of models with respect to the logic $\mathbf{FL}_\bot$; furthermore, its reducts are complete with respect to **FL** and $L1$.

1 Introduction

Syntactic concept lattices arise from the distributional structure of languages. Their main advantage is that they can be constructed on distributional relations which are weaker than strict equivalence. [3] has shown how these lattices can be enriched with a monoid structure to form residuated lattices. This makes it natural to ask whether they are an appropriate model for some substructural logics. Natural candidates are $L1$, the well-studied Lambek calculus (introduced in [11]) with the extension that it allows conclusions from empty premises, and its well-known conservative extensions **FL** and $\mathbf{FL}_\bot$ (see [8]).

We give a proof of completeness of $\mathbf{FL}_\bot$ for the class of residuated concept lattices for any language. Our proof will be constructed on top of well-known completeness results for the class of residuated lattices. We will show that any residuated lattice can be embedded into a syntactic concept lattice. So if an inequation fails in a residuated lattice, it also fails in the image. As corollaries, we get the completeness of syntactic concept lattice reducts for $L1$ and **FL**.

2 Residuated Syntactic Concept Lattices

2.1 Equivalences and Concepts

Syntactic concept lattices originally arose in the structuralist approach to syntax, back when syntacticians tried to capture syntactic structures purely in terms of distributions of strings[1] (see, e.g. [9]). An obvious way to do so is by partitioning strings/substrings into *equivalence classes*: we say that two strings w, v are equivalent in a language $L \subseteq \Sigma^*$, in symbols

[1] Or words, respectively, depending on whether we think of our language as a set of words or a set of strings of words.

G. Morrill and M.-J. Nederhof (Eds.): FG 2012/2013, LNCS 8036, pp. 126–141, 2013.

(1) $w \sim_L^0 v$ iff for all $x \in \Sigma^*$, $wx \in L \Leftrightarrow vx \in L$.

This defines the well-known Nerode-equivalence. We can use a richer equivalence relation, by considering not only left contexts, but also right contexts:

(2) $w \sim_L^1 v$ iff for all $x, y \in \Sigma^*$, $xwy \in L \Leftrightarrow xvy \in L$.

Of course, this can be arbitrarily iterated for tuples of strings. The problem with equivalence classes is that they are too restrictive for many purposes: assume we want to induce our grammar on the basis of a given dataset; then it is quite improbable that we get the equivalence classes we would usually desire as linguists. And even if we have an unlimited supply of examples, it seems unrealistic to describe our grammar on the basis of equivalence classes only: there might be constructions, collocations, idioms which ruin equivalences which we would intuitively consider to be adequate.

Syntactic concepts provide a somewhat less rigid notion of equivalence, which can be conceived of as equivalence restricted to a given set of contexts. This at least partly resolves the difficulties we have described above.

2.2 Syntactic Concepts: Definitions

For a general introduction to lattices, see [6]; for background on residuated lattices, see [8]. Syntactic concept lattices form a particular case of what is well-known as formal concept lattice (or formal concept analysis) in computer science. In linguistics, they have been introduced in [16]. They were brought back to attention and enriched with residuation in [3], [4], as they turn out to be useful representations for language learning. In this section, we follow the presentation given in [3].

Given a language $L \subseteq \Sigma^*$, we define two maps: a map $\triangleright : \wp(\Sigma^*) \to \wp(\Sigma^* \times \Sigma^*)$, and $\triangleleft : \wp(\Sigma^* \times \Sigma^*) \to \wp(\Sigma^*)$, which are defined as follows:

(3) for $M \subseteq \Sigma^*$, $M^{\triangleright} := \{(x, y) : \forall w \in M, xwy \in L\}$;

and dually

(4) for $C \subseteq \Sigma^* \times \Sigma^*$, $C^{\triangleleft} := \{x : \forall (v, w) \in C, vxw \in L\}$.

That is, a set of strings is mapped to the set of contexts, in which all of its elements can occur. The dual function maps a set of contexts to the set of strings, which can occur in all of them. Obviously, $\triangleleft$ and $\triangleright$ are only defined with respect to a given language L, otherwise they are meaningless. As long as it is clear of which language (if any concrete language) we are speaking, we will omit however any reference to it. For a set of contexts C, $C^{\triangleleft}$ can be thought of as an equivalence class with respect to the contexts in C; but not in general: there might be elements in $C^{\triangleleft}$ which can occur in a context $(v, w) \notin C$ (and conversely).

The two compositions of the maps, $\triangleleft\triangleright$ and $\triangleright\triangleleft$, form a closure operator on subsets of $\Sigma^* \times \Sigma^*$ and Σ^*, respectively, that is:

1. $M \subseteq M^{\triangleright\triangleleft}$,
2. $M^{\triangleright\triangleleft} = M^{\triangleright\triangleleft\triangleright\triangleleft}$,
3. $M \subseteq N \Rightarrow M^{\triangleright\triangleleft} \subseteq N^{\triangleright\triangleleft}$,

for $M, N \subseteq \Sigma^*$. The same holds for contexts, where we simply exchange the order of the mappings, and use subsets of $\Sigma^* \times \Sigma^*$. We say a set M is **closed** if $M^{\triangleright\triangleleft} = M$. The closure operator $\triangleright\triangleleft$ gives rise to a lattice $\mathfrak{L}_S := \langle \mathfrak{B}_S, \leq \rangle$, where the elements of $\mathfrak{B}_S$ are the closed sets, and $\leq$ is interpreted as $\subseteq$. The same can be done with the set of closed contexts. Given these two lattices, $\triangleright$ and $\triangleleft$ make up a Galois connection between the two:

1. $M \leq N \Leftrightarrow M^{\triangleleft} \geq N^{\triangleleft}$, and
2. $C \leq D \Leftrightarrow C^{\triangleright} \geq D^{\triangleright}$.

Furthermore, for $\mathfrak{L}_S$ the lattice of closed subsets of strings, $\mathfrak{L}_C$ the lattice of contexts, it is easy to show that $\mathfrak{L}_S \cong \mathfrak{L}_C^{\partial}$, where by $[-]^{\partial}$ we denote the **dual** of a lattice, that is, the same lattice with its order relation inverted; and by $\cong$ we denote that there is an isomorphism between two structures. Therefore, any statement on the one lattice is by duality a statement on the other. Consequently, we can directly conceive of the two as a single lattice, whose elements are **syntactic concepts**:

Definition 1. *A syntactic concept A is an (ordered) pair, consisting of a closed set of strings, and a closed set of contexts, written $A = \langle S, C \rangle$, such that $S^{\triangleright} = C$ and $C^{\triangleleft} = S$.*

Note also that for any set of strings S and contexts C, $S^{\triangleright} = S^{\triangleright\triangleleft\triangleright}$ and $C^{\triangleleft} = C^{\triangleleft\triangleright\triangleleft}$. Therefore, any set M of strings gives rise to a concept $\langle M^{\triangleright\triangleleft}, M^{\triangleright} \rangle$, and any set of C contexts to a concept $\langle C^{\triangleleft}, C^{\triangleleft\triangleright} \rangle$. Therefore, we denote the concept which is induced by a set M, regardless of whether it is a set of strings or contexts, by $\mathcal{C}(M)$. We speak of the *extent* of a concept A as the set of strings it contains, which we denote by S_A; the *intent* of A is the set of contexts it contains, denoted by C_A. For example, given a language L, we have $S_{\mathcal{C}((\epsilon,\epsilon))} = L$, as all and only the strings in L can occur in L in the context (ϵ, ϵ).

We define the partial order $\leq$ on concepts by

$$\langle S_1, C_1 \rangle \leq \langle S_2, C_2 \rangle \qquad \Longleftrightarrow \qquad S_1 \subseteq S_2; \tag{5}$$

which gives rise to the syntactic concept lattice $\mathfrak{L}$:

Definition 2. *The lattice of concepts of a language L, $\mathfrak{L}(L) = \langle \mathfrak{B}, \wedge, \vee \rangle$, with the partial order $\subseteq$, is called the **syntactic concept lattice**, where $\top = \mathcal{C}(\Sigma^*)$, $\bot = \mathcal{C}(\Sigma^* \times \Sigma^*)$, and for $\langle S_i, C_i \rangle, \langle S_j, C_j \rangle \in \mathfrak{B}$, $\langle S_i, C_i \rangle \wedge \langle S_j, C_j \rangle = \langle S_i \cap S_j, (C_i \cup C_j)^{\triangleleft\triangleright} \rangle$, and $\vee$ as $\langle (S_i \cup S_j)^{\triangleright\triangleleft}, C_i \cap C_j \rangle$.*

It is easy to verify that this forms a complete lattice. Note the close connection between intersection of stringsets and union of context sets, and vice versa. Obviously, $\mathfrak{L} \cong \mathfrak{L}_S$, which we defined before.

2.3 Monoid Structure and Residuation

As we have seen, the set of concepts of a language forms a lattice. In addition, we can also give it the structure of a monoid: for concepts $\langle S_1, C_1\rangle, \langle S_2, C_2\rangle$, we define:

$$\langle S_1, C_1\rangle \circ \langle S_2, C_2\rangle = \langle (S_1S_2)^{\triangleright\triangleleft}, (S_1S_2)^{\triangleright}\rangle, \tag{6}$$

where $S_1S_2 = \{xy : x \in S_1, y \in S_2\}$. Obviously, the result is a concept. $'\circ'$ is associative on concepts:

$$\text{for } X, Y, Z \in \mathfrak{B},\ X \circ (Y \circ Z) = (X \circ Y) \circ Z. \tag{7}$$

This follows from the fact that $[-]^{\triangleright\triangleleft}$ is a *nucleus*, that is, it is a closure operator and in addition it satisfies

$$S^{\triangleright\triangleleft}T^{\triangleright\triangleleft} \subseteq (ST)^{\triangleright\triangleleft}. \tag{8}$$

Using this property and the associativity of string concatenation, the result easily follows. Furthermore, it is easy to see that the neutral element of the monoid is $\mathcal{C}(\epsilon)$. This monoid structure respects the partial order of the lattice, that is:

Lemma 3. *For concepts $X, Y, Z, W \in \mathfrak{B}$, if $X \leq Y$, then $W \circ X \circ Z \leq W \circ Y \circ Z$.*

We can extend the operation $\circ$ to the contexts of concepts:

$$(x, y) \circ (w, z) = (xw, zy). \tag{9}$$

This way, we still have $f \circ (g \circ h) = (f \circ g) \circ h$ for singleton contexts f, g, h. The operation can be extended to sets in the natural way, preserving associativity. For example, $C \circ (\epsilon, S) = \{(x, ay) : (x, y) \in C, a \in S\}$. We will use this as follows:

Definition 4. *Let $X = \langle S_X, C_X\rangle, Y = \langle S_Y, C_Y\rangle$ be concepts. We define the right residual $X/Y := \mathcal{C}(C_1 \circ (\epsilon, S_2))$, and the left residual $Y\backslash X := \mathcal{C}(C_1 \circ (S_2, \epsilon))$.*

For the closed sets of strings S, T, define $S/T := \{w : \text{for all } v \in T, wv \in S\}$. We then have $S_X/S_Y = S_{X/Y}$. So residuals are unique and satisfy the following lemma:

Lemma 5. *For $X, Y, Z \in \mathfrak{B}$, we have $Y \leq X\backslash Z$ iff $X \circ Y \leq Z$ iff $X \leq Z/Y$.*

For a proof, see [3]. This shows that the syntactic concept lattice can be enriched to a residuated lattice (see the definition below, or check the references). Note that every language, whether computable or not, has a syntactic concept lattice. An important question is whether it is finite or not. This question can be answered in the following way.

Proposition 6. *The syntactic concept lattice for a language L is finite if and only if L is regular.*

This is a rather immediate consequence of the Myhill-Nerode theorem. A short note is in order about finite/infinite alphabets. Later on, we will embed residuated monoids/lattices into syntactic concept lattices. If these algebras have an infinite domain, we will need an infinite alphabet for the construction of the corresponding language; and so we will in general need languages over infinite alphabets. In case the domain of the algebra is countable, we can encode its letters with (finite) strings over a finite alphabet; if it is uncountable, however, this does not work anymore. So in the sequel, by a *language* we mean a set of finite strings over some alphabet, regardless of whether it is finite, countable or uncountable.

This is of course unsatisfactory, as for us the notion "language" implies the finiteness of the alphabet. We can yield completeness results for languages over *finite* alphabets in two ways: 1. we use the Lindenbaum-Tarski construction to construct the counter-model, which then results in a countable model, whose domain we can encode in a finite alphabet. 2. there is an even simpler solution using the finite model property of $L1$, **FL** and $\mathbf{FL}_\bot$, which says that for each underivable sequent of the logic, there is a finite model in which it does not hold. We will use this second solution.[2]

In the sequel, we will denote by *SCL* the class of all syntactic concept lattices, that is, the class of all lattices of the form $\mathfrak{L}(L)$ for some language L, without any further requirement regarding L itself except the ones stated above.

2.4 The Linguistic Order

Syntactic concepts are related to an order, which will have some importance in the sequel. Given a language $L \subseteq \Sigma^*$, we write $w \leq_L v$ iff $xvy \in L \to xwy \in L$. We call $\leq_L$ the *linguistic order*. Note that this is a pre-order, as from $w \leq_L v$ and $v \leq_L w$ follows $w \sim_L v$, where $\sim_L$ is substitutional equivalence (we denoted this above as $\sim_L^1$), but not equality. We can however think of $\leq_L$ as a partial order if we define it over $[\Sigma^*]_{\sim_L}$, that is, the set of L-equivalence classes rather than the set of strings. As is easy to see, either way $\leq_L$ respects concatenation of strings. This way, a language $L \subseteq \Sigma^*$ defines a preordered monoid $(\Sigma^*, \leq_L, \cdot, \epsilon)$.

We say a set (of strings) W is **downward closed** (with respect to $\leq_L$) if from $w \in W$ and $v \leq_L w$ it follows that $v \in W$.

Lemma 7. *Given a language $L \subseteq \Sigma^*$ and a set of strings $W \subseteq \Sigma^*$, if $W = W^{\triangleright\triangleleft}$, then W is downward closed with respect to $\leq_L$.*

Proof. Assume $W = W^{\triangleright\triangleleft}$, $v \leq_L w$, $w \in W$. We know that for all $(a, b) \in W^{\triangleright}$, $awb \in L$. By $v \leq_L w$ it follows that also $avb \in L$ if $awb \in L$. Consequently, $v \in W^{\triangleright\triangleleft}$, and so $W^{\triangleright\triangleleft}$ is downward closed. □

The converse implication does not hold, that is: not every downward closed set is closed under $[-]^{\triangleright\triangleleft}$. This is because the $[-]^{\triangleright\triangleleft}$-closure considers only the L-contexts which are common to *all* strings in W.

[2] Thanks to an anonymous reviewer for pointing this out to me.

3 Lambek Calculus and Extensions

3.1 The Logics L, $L1$, FL and $\mathbf{FL}_\bot$

The Lambek calculus L was introduced in [11]. $L1$ is a proper extension of L, and $\mathbf{FL}, \mathbf{FL}_\bot$ are each conservative extensions of $L1$ and the preceding one. Let Pr be a set, the set of **primitive types**, and C be a set of **constructors**, which is, depending on the logics we use, $C_L := \{/, \backslash, \bullet\}$, or $C_{\mathbf{FL}} := \{/, \backslash, \bullet, \vee, \wedge\}$. By $Tp_C(Pr)$ we denote the set of types over Pr, which is defined as the smallest set such that:

1. $Pr \subseteq Tp_C(Pr)$.
2. if $\alpha, \beta \in Tp_C(Pr)$, $\star \in C$, then $\alpha \star \beta \in Tp_C(Pr)$.

If there is no danger of confusion regarding the primitive types and constructors, we also simply write Tp for $Tp_C(Pr)$. We now present the inference rules corresponding to these constructors. We call an inference of the form $\Gamma \vdash \alpha$ a **sequent**, for $\Gamma \in Tp^*$, $\alpha \in Tp$, where by Tp^* we denote the set of all (possibly empty) *sequences* over Tp, which are concatenated by $'$,$'$ (keep in mind the difference between *sequents*, which have the form $\Gamma \vdash \alpha$, and *sequences* like Γ, which are in Tp^*).

With one exception, rules of inference in our logics are not given in the form of sequents $\Gamma \vdash \alpha$, but rather as rules to derive new sequents from given ones. In general, uppercase Greek letters range as variables over sequences of types. In the inference rules for L, premises of $' \vdash '$ (that is, left hand sides of sequents) must be non-empty; in $L1$ they can be empty as well; everything else is equal. In **FL** and $\mathbf{FL}_\bot$ we also allow for empty sequents. Lowercase Greek letters range over single types. Below, we present the standard rules of the Lambek calculus L / $L1$, with one axiom schema and several (meta-)rules to derive new sequents from given ones.

$$(ax) \qquad \alpha \vdash \alpha$$

$$(\mathbf{I}-/) \quad \frac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \beta/\alpha} \qquad (\mathbf{I}-\backslash) \quad \frac{\alpha, \Gamma \vdash \beta}{\Gamma \vdash \alpha\backslash\beta}$$

$$(/-\mathbf{I}) \quad \frac{\Delta, \beta, \Theta \vdash \gamma \quad \Gamma \vdash \alpha}{\Delta, \beta/\alpha, \Gamma, \Theta \vdash \gamma} \qquad (\backslash-\mathbf{I}) \quad \frac{\Delta, \beta, \Theta \vdash \gamma \quad \Gamma \vdash \alpha}{\Delta, \Gamma, \alpha\backslash\beta, \Theta \vdash \gamma}$$

$$(\bullet-\mathbf{I}) \quad \frac{\Delta, \alpha, \beta, \Gamma \vdash \gamma}{\Delta, \alpha \bullet \beta, \Gamma \vdash \gamma} \qquad (\mathbf{I}-\bullet) \quad \frac{\Delta \vdash \alpha \quad \Gamma \vdash \beta}{\Delta, \Gamma \vdash \alpha \bullet \beta}$$

These are the standard rules of L / $L1$ (roughly as in [11]). We have rules to introduce either slash and $'\bullet'$ both on the right hand side of $\vdash$ and on the

left hand side of $\vdash$. We will now add two additional connectives, which are well-known from structural logics, namely $\vee$ and $\wedge$. These are not present in $L/L1$, have however been considered as extensions as early as in [12], and have been subsequently studied by [10].

$$(\wedge - \mathbf{I}\ 1) \quad \frac{\Gamma, \alpha, \Delta \vdash \gamma}{\Gamma, \alpha \wedge \beta, \Delta \vdash \gamma} \qquad (\wedge - \mathbf{I}\ 2) \quad \frac{\Gamma, \beta, \Delta \vdash \gamma}{\Gamma, \alpha \wedge \beta, \Delta \vdash \gamma}$$

$$(\mathbf{I} - \wedge) \quad \frac{\Gamma \vdash \alpha \quad \Gamma \vdash \beta}{\Gamma \vdash \alpha \wedge \beta}$$

$$(\vee - \mathbf{I}) \quad \frac{\Gamma, \alpha, \Delta \vdash \gamma \quad \Gamma, \beta, \Delta \vdash \gamma}{\Gamma, \alpha \vee \beta, \Delta \vdash \gamma}$$

$$(\mathbf{I} - \vee\ 1) \quad \frac{\Gamma \vdash \alpha}{\Gamma \vdash \alpha \vee \beta} \qquad (\mathbf{I} - \vee\ 2) \quad \frac{\Gamma \vdash \beta}{\Gamma \vdash \alpha \vee \beta}$$

$$(1 - \mathbf{I}) \quad \frac{\Gamma, \Delta \vdash \alpha}{\Gamma, 1, \Delta \vdash \alpha} \qquad (\mathbf{I} - 1) \quad \vdash 1$$

This gives us the logic **FL**. Note that this slightly deviates from standard terminology, because usually, **FL** has an additional constant 0. In our formulation, 0 and 1 coincide. In order to have logical counterparts of the bounded lattice elements $\top$ and $\bot$, we introduce two logical constants, which are denoted by the same symbol.[3]

$$(\bot - \mathbf{I}) \quad \Gamma, \bot\ \Delta \vdash \alpha \qquad (\mathbf{I} - \top) \quad \Gamma \vdash \top$$

This gives us the calculus $\mathbf{FL}_\bot$. From a logical point of view, all these extensions of L are quite well-behaved: they are conservative, and also allow us to preserve the important result of [11], namely admissibility of the cut-rule in L:

$$(cut) \quad \frac{\Delta, \beta, \Theta \vdash \alpha \quad \Gamma \vdash \beta}{\Delta, \Gamma, \Theta \vdash \alpha}$$

We say that a sequent $\Gamma \vdash \alpha$ is derivable in a calculus L or an extension, if it can be derived by the axiom and the rules of inference; we then write $\Vdash_L \Gamma \vdash \alpha$, $\Vdash_{L1} \Gamma \vdash \alpha$, $\Vdash_{\mathbf{FL}} \Gamma \vdash \alpha$, etc., depending on which calculus we use.

[3] Whereas L and $L1$ are equally powerful in the sense of languages which are recognizable, [10] shows that **FL** is considerably more powerful than L: whereas L only recognizes context-free languages by the classical result of [15], **FL** can recognize any finite intersection of context-free languages. We only briefly mention this, because we have no space to make precise what it means for a calculus to recognize a class of languages.

3.2 The Semantics of $L1$, FL and $\mathbf{FL}_\bot$

The standard model for L is the class of residuated semigroups. We will not consider L in the sequel, as there are some additional problems if we want to interpret L in syntactic concept lattices: there is nothing in L corresponding to the unit element, but we cannot just do away with ϵ in syntactic concept lattices without some complications. The standard model for $L1$ is the class of residuated monoids. These are structures $(M, \cdot, 1, \backslash, /, \leq)$, where $(M, \cdot, 1)$ is a monoid, $(M, \leq)$ is a partial order, and $\cdot, /, \backslash$ satisfy the law of residuation: for $m, n, o \in M$,

$$(10) \qquad m \leq o/n \Leftrightarrow m \cdot n \leq o \Leftrightarrow n \leq m\backslash o.$$

Note that this implies that $\cdot$ respects the order $\leq$. The standard model for **FL** is the class of residuated lattices, and for $\mathbf{FL}_\bot$, the class of bounded residuated lattices. A residuated lattice is an algebraic structure $\langle M, \cdot, \vee, \wedge, \backslash, /, 1\rangle$, where in addition to the previous requirements, $(M, \vee, \wedge)$ is a lattice; the lattice order $\leq$ need not be stated, as it can be induced by $\vee$ or $\wedge$: for $a, b \in M$, $a \leq b$ is a shorthand for $a \vee b = b$. A bounded residuated lattice is a structure $\langle M, \cdot, \vee, \wedge, \backslash, /, 1, \top, \bot\rangle$, where $\langle M, \cdot, \vee, \wedge, \backslash, /, 1\rangle$ is a residuated lattice, $\top$ is the maximal element of the lattice order $\leq$ and $\bot$ is its minimal element.

For a general introduction see [8]. We will give definitions only once for each operator; we can do so because each definition for a given connector is valid for all classes in which it is present.

We call the class of residuated semigroups RS, the class of residuated monoids RM, the class of residuated lattices RL; the class of bounded residuated lattices $RL_\bot$. We now give a semantics for the calculi above. We start with an interpretation σ which interprets elements in Pr, and extend σ to $\overline{\sigma}$ by defining it inductively over our type constructors, which is for now the set $C := \{/, \backslash, \bullet, \vee, \wedge\}$. Assignment goes as follows, for $\alpha, \beta \in Tp_C(Pr)$:

1. $\overline{\sigma}(\alpha) = \sigma(\alpha) \in M$, if $\alpha \in Pr$
2. $\overline{\sigma}(\top) = \top$
3. $\overline{\sigma}(\bot) = \bot$
4. $\overline{\sigma}(1) = 1$
5. $\overline{\sigma}(\alpha \bullet \beta) := \overline{\sigma}(\alpha) \cdot \overline{\sigma}(\beta)$
6. $\overline{\sigma}(\alpha/\beta) := \overline{\sigma}(\alpha)/\overline{\sigma}(\beta)$
7. $\overline{\sigma}(\alpha\backslash\beta) := \overline{\sigma}(\alpha)\backslash\overline{\sigma}(\beta)$
8. $\overline{\sigma}(\alpha \vee \beta) := \overline{\sigma}(\alpha) \vee \overline{\sigma}(\beta)$
9. $\overline{\sigma}(\alpha \wedge \beta) := \overline{\sigma}(\alpha) \wedge \overline{\sigma}(\beta)$

Note that the constructors on the left-hand side and on the right-hand side of the definition look identical (with the exception of $\bullet$ and $\cdot$), but they are not: on the left-hand side, they are type constructors, on the right hand side, they are operators of a residuated lattice. The same holds for the constants $\top, \bot, 1$.

This is how we interpret the types of our logic. What we want to interpret next are the *sequents* of the form $\Gamma \vdash \alpha$. We say that a sequent $R = \gamma_1, ..., \gamma_i \vdash \alpha$

is true in a model $\mathcal{M}$ under assignment σ, in symbols: $(\mathcal{M}, \sigma) \models \gamma_1, ..., \gamma_i \vdash \alpha$, if and only if $\overline{\sigma}(\gamma_1 \bullet ... \bullet \gamma_i) \leq \overline{\sigma}(\alpha)$ holds in $\mathcal{M}$. That is, we interpret the $'$,$'$, which denotes concatenation in sequents, as $\cdot$ in the model, and $\vdash$ as $\leq$. In the sequel, for Γ a sequence of types, we will often write $\sigma(\Gamma)$ as an abbreviation, where we leave the former translation implicit. For the case of theorems, that is, derivable sequents with no antecedent, we have the following convention: $(\mathcal{M}, \sigma) \models \vdash \alpha$ iff $1 \leq \overline{\sigma}(\alpha)$ in $\mathcal{M}$, where 1 is the unit element of $\mathcal{M}$. Note that this case does not arise in L.

More generally, for a given class of (bounded) residuated lattices (monoids, semigroups) $\mathfrak{C}$, we say that a sequent is *valid* in $\mathfrak{C}$, in symbols, $\mathfrak{C} \models \gamma_1, ..., \gamma_i \vdash \alpha$, if for all $\mathcal{M} \in \mathfrak{C}$ and all assignments σ, $(\mathcal{M}, \sigma) \models \gamma_1, ...\gamma_i \vdash \alpha$.

4 Completeness: Preliminaries

There are a number of completeness results for the logics we have considered here. We will consider the most general ones, which will be important in the sequel.

Theorem 8. *(Buszkowski) For the class of residuated semigroups RS, $RS \models \Gamma \vdash \alpha$ if and only if $\Vdash_L \Gamma \vdash \alpha$. For the class of residuated monoids RM, $RM \models \Gamma \vdash \alpha$ if and only if $\Vdash_{L1} \Gamma \vdash \alpha$.*

Theorem 9. *For the class RL of residuated lattices, $RL \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}} \Gamma \vdash \alpha$. For the class $RL_\bot$ of bounded residuated lattices, $RL_\bot \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}_\bot} \Gamma \vdash \alpha$.*

For reference on theorem 8, see [1], [2]. For theorem 9, see [8]. The proofs for the above completeness theorems usually proceed via the Lindenbaum-Tarski construction: we interpret primitive types as atomic terms modulo mutual derivability, and define $\sigma(\alpha) \leq \sigma(\beta)$ iff $\alpha \vdash \beta$. Then we can perform an induction over constructors to get the same for arbitrary formulas/terms. So there are quite simple completeness proofs for the general case.

What is much harder to obtain is completeness in the finite case, usually referred to as **finite model property**. A logic $\mathcal{L}$ has finite model property if from the fact that a sequent $\Gamma \vdash \alpha$ is not provable in $\mathcal{L}$, it follows that there is a *finite* model $\mathcal{M}$ and an assignment σ such that $(\mathcal{M}, \sigma) \not\models \Gamma \vdash \alpha$.

Theorem 10. *1. L1 has finite model property.*
*2. **FL** has finite model property.*
3. $\boldsymbol{FL}_\bot$ has finite model property.

For the first claim, consider [7]; the second and third has been established by [14]. We want to establish soundness and completeness of the calculi with respect to the class of syntactic concept lattices and their reducts. The latter results are crucial to show that completeness holds also if we restrict ourselves to languages over finite alphabets. First we see that our calculus is sound with respect to the model:

Theorem 11. *(Soundness) If* $\Vdash_{\boldsymbol{FL}_\bot} \Gamma \vdash \alpha$*, then for the class of syntactic concept lattices SCL, we have* $SCL \models \Gamma \vdash \alpha$*.*

This actually follows from soundness direction of theorem 9, because *SCL* is just a particular class of bounded residuated lattices. As $L, L1, \mathbf{FL}$ are fragments of $\mathbf{FL}_\bot$, we get the same result for $L, L1$ and $\mathbf{FL}$, regarding the sequents which contain only the operators which have a counterpart in the logic. What is much harder to obtain is completeness.

Theorem 12. *(Completeness) If* $SCL \models \Gamma \vdash \alpha$*, then* $\Vdash_{\boldsymbol{FL}_\bot} \Gamma \vdash \alpha$*.*

Proof idea: our proof of completeness for the class of syntactic concept lattices is based on the completeness result for the class $RL_\bot$. The idea of the proof is quite simple: starting from the completeness of $\mathbf{FL}_\bot$ for the class of bounded residuated lattices, we show that each residuated lattice can be isomorphically embedded into a syntactic concept lattice. So let $\mathbf{B}$ be a residuated lattice and $\mathfrak{L}$ a syntactic concept lattice, then from the fact that $h : B \to \mathfrak{L}$ is an isomorphic embedding, we can conclude that if $(\mathbf{B}, \sigma) \not\models \Gamma \vdash \alpha$, then $(\mathfrak{L}, h \circ \sigma) \not\models \Gamma \vdash \alpha$. This in turn means that if there is a residuated lattice, where a certain inequation does not hold, then there is a syntactic concept lattice for some language where it does not hold either. This allows us to extend the completeness result from the general class of bounded residuated lattices, which is obtained by contraposition, to the class of residuated concept lattices. The finite model property of the calculi allows us to assume that the countermodels are finite; so we can conclude that $\mathbf{FL}_\bot$ is complete with respect to the class of syntactic concept lattices of languages over finite alphabets.

The next section will be devoted to presenting the embedding and to show why it does the job as required.

5 Proof of the Main Theorem

5.1 Isomorphic Embedding in Syntactic Concept Lattices

Let $\mathbf{B} = (B, \vee, \wedge, /, \backslash, \cdot, 1, \top, \bot)$ be a bounded residuated lattice. We denote the partial order of $\mathbf{B}$ by $\leq_B$. Recall that different terms over B can denote the same element of B. To avoid confusion, we denote this by the equality $=_B$. That means, for terms s, t over B, $s =_{\mathbf{B}} t$ states that s, t denote the same element of B.

Define $\Sigma := \{b, \underline{b} : b \in B\}$. We define a language $L_B \subseteq \Sigma^*$ as the set of strings $L_B := \{b_1 b_2 ... b_n \underline{b} : b_1 \cdot b_2 \cdot ... \cdot b_n \leq_{\mathbf{B}} b\}$. For a string $w = b_1 ... b_n \in B^*$, by $w^\bullet$ we denote the term $b_1 \cdot ... \cdot b_n$.

Note that we now have an ambiguity as to whether a certain $b \in B$ is a letter of Σ or an element of the lattice. We could have generally avoided this ambiguity at the price of complicating notation; but we rather try to avoid the ambiguity in all

particular statements, by using $\leq_B$ or $\leq_L$, etc. Importantly, the non-atomic terms over the lattice $\mathbf{B}$ are *not* part of the alphabet Σ, only the elements of B which these terms denote are in Σ. So we have to take care to not read the terms as syntactic objects of L_B: whereas a term t will not occur in L_B unless it is an atomic term, the element it denotes does occur in L_B for any term t.

We define a map $\gamma : \wp(B^*) \to \wp(\Sigma^*)$ by $\gamma(X) = (X)^{\triangleright\triangleleft}$, where $[-]^{\triangleright\triangleleft}$ is the syntactic concept closure with respect to L_B. As is easy to see, γ is a closure operator, as $[-]^{\triangleright\triangleleft}$ is a closure, and we have $\gamma(b_1)\gamma(b_2) \leq \gamma(b_1 b_2)$, where by the concatenation of two sets we simply mean the concatenation of their elements, that is, $VW := \{ab : a \in V, b \in W\}$.[4] As we have said, a map which has these properties is called *nuclear*.

A nuclear map gives rise to what is called the *nuclear image* of $\wp(\mathbf{B}^*)$, the lattice $\langle \gamma[\wp(B^*)], \cap, \cup_\gamma, \circ_\gamma, /, \backslash, \gamma(\top), \gamma(\bot)\rangle$, which we will call $Q(\mathbf{B})$. From the fact that γ is nuclear, it follows that $Q(\mathbf{B})$ is a complete residuated lattice (see [8],p.174), where $X \cup_\gamma Y := \gamma(X \cup Y)$, $X \circ_\gamma Y := \gamma(X \cdot Y)$. Furthermore, $Q(\mathbf{B})$ is bounded by $\gamma(\top) = B^*$[5] and $\gamma(\bot) = \{\bot\}$, as $\bot$ is contained in any γ-closed set.

It is easy to see that $Q(\mathbf{B})$ can be isomorphically embedded into the syntactic concept lattice of L_B, which we call $\mathfrak{L}(L_B)$: in fact, $Q(\mathbf{B})$ is isomorphic to the fragment of $\mathfrak{L}(L_B)$ which consists of all concepts from strings in B^*. It is easy to see that in $\mathfrak{L}(L_B)$ these are closed under meet, join and concatenation; but note that this does not follow from general considerations, and is rather a consequence of the particular distributional structure of L_B. Consequently, we have an isomorphic embedding $\mathcal{C} : Q(\mathbf{B}) \to \mathfrak{L}(L_B)$ (as we defined it above), which simply maps closed sets of strings onto their concepts.

What we still need is an appropriate embedding from $\mathbf{B}$ into $Q(\mathbf{B})$. The next lemma will be very helpful to understand what is to follow:

Lemma 13. *Let $\leq_L$ be the linguistic order of L_B, $a, b \in B$. Then $a \leq_B b \iff a \leq_L b$.*

Proof. $\Rightarrow$ Assume $a \leq_B b$. Then if $xby \in L_B$ (we know that $y \neq \epsilon$), then it is easy to see that $xay \in L_B$, by the definition of L_B: each word of L_B corresponds to an inequation which holds in $\mathbf{B}$, and the inequation remains valid under the substitution of a for b. Therefore, $a \leq_L b$.

$\Leftarrow$ Assume $a \leq_L b$. As we have $b \leq_B b$, we have $b\underline{b} \in L_B$. By assumption, we then have $a\underline{b} \in L_B$. This can only be the case if $a \leq_B b$. □

We define a map $h : B \to Q(\mathbf{B})$ (equivalently, $h : B \to \gamma[\wp(B^*)]$), where $h(b) = \{w \in \Sigma^* : w\underline{b} \in L_B\}$. This is clearly a γ-closed set (or put differently: extent of a syntactic concept of L_B), as it equals the closed set $(\epsilon, \underline{b})^\triangleleft$.

In the sequel, we will often use h with terms instead of atoms. Of course, here the same applies as before: h is *not* defined over terms, it only maps the elements denoted by the terms. A crucial lemma is the following:

[4] The latter inequation follows from our above considerations on syntactic concepts.
[5] We assume that $\top \in B$!

Lemma 14. *For $w \in \Sigma^*, b \in B$, the following three are equivalent:*

1. $w \in h(b)$
2. $w^\bullet \leq_B b$
3. $w \leq_L b$

Proof. 2. $\Rightarrow$ 1.: if $w^\bullet \leq_B b$, then $w\underline{b} \in L_B$, and so $w \in h(b)$.

1. $\Rightarrow$ 2.: assume $w \in h(b)$. Then we have $w\underline{b} \in L_B$. But $w\underline{b} \in L_B$ only if $w^\bullet \leq_B b$, so the implication follows.

The biimplication 2. $\Leftrightarrow$ 3. is only a slight generalization of the preceding lemma. We write $w \sim_L v$ as a shorthand for $w \leq_L v$ & $v \leq_L w$ (recall that $\leq_L$ is a pre-order rather than a partial order). So assume that for $a, b, c \in B$, $a \cdot b =_B c$. From this it follows that $x \cdot (a \cdot b) \cdot y \leq_B z$ if and only if $x \cdot c \cdot y \leq_B z$. Consequently, $ab \sim_L c$. This can be extended to arbitrary terms over B and $\cdot$: for any $w \in B^*$, we have $w \sim_L w^\bullet$. Consequently, we have $w^\bullet \leq_B b$ if and only if $w^\bullet \leq_L b$ (lemma 13) if and only if $w \leq_L b$. □

We will now show that h defines a proper isomorphic embedding of $\mathbf{B}$ into the nuclear image $Q(\mathbf{B})$, which forms a fragment of the syntactic concept lattice of L_B.

Theorem 15. *For each $\star \in \{\wedge, \vee, \cdot, /, \backslash\}$, we have $h(a) \star_\gamma h(b) = h(a \star b)$, where $a \star b$ denotes the unique element of B denoted by the term, and $\star_\gamma$ denotes the interpretation of $\star$ in the γ-image of $\wp(B^*)$. This means that h is an isomorphic embedding.*

Proof. We proceed by cases:

Case 1: $\star = \wedge$.

a) $h(a) \cap h(b) \supseteq h(a \wedge b)$. Assume that $c \in h(a \wedge b)$. Then by lemma 14, $c \leq_B (a \wedge b)$. Therefore, $c \leq_B a, b$, and consequently $c \leq_L a, b$. Therefore, $c \in h(a), c \in h(b)$, and thus $c \in h(a) \cap h(b)$.

b) $h(a) \cap h(b) \subseteq h(a \wedge b)$. Assume that $c \in h(a) \cap h(b)$. Then $c \in h(a)$, $c \in h(b)$; consequently, $c \leq_B a, b$ (by lemma 14); consequently, $c \leq a \wedge b$, and therefore $c \in h(a \wedge b)$.

Case 2: $\star = \vee$

a) $h(a) \cup_\gamma h(b) \supseteq h(a \vee b)$: Assume that $c \in h(a \vee b)$. Then $c \leq_L a \vee b$. We now show that $a \vee b \in h(a) \cup_\gamma h(b)$: whenever $wav \in L_B$, $wbv \in L_B$, then $w(a \vee b)v \in L_B$. Consequently, $a \vee b \in \gamma(\{a, b\}) \subseteq h(a) \cup_\gamma h(b)$. As γ-closed sets are downward closed with respect to the linguistic order, we also have $c \in h(a) \cup_\gamma h(b)$.

b) $h(a) \cup_\gamma h(b) \subseteq h(a \vee b)$: By lemma 14, we know that for all $x \in h(a)$, $x \leq_L a$, and for all $y \in h(b)$, $y \leq_L b$. Consequently, by lemma 14 we have $z \leq_L a \vee b$ for all $z \in h(a) \cup_\gamma h(b)$, and so $h(a) \cup h(b) \subseteq h(a \vee b)$. As $h(a \vee b)$ is γ-closed, we must also have $h(a) \cup_\gamma h(b) \subseteq h(a \vee b)$ by order preservation of γ.

Case 3: $\star = \cdot$.

Recall that we use $a \sim_L b$ as a shorthand for $a \leq_L b$ & $b \leq_L a$, and that for $a, b, c \in B$, from $a \cdot b =_B c$ it follows that $ab \sim_L c$. This means that we may interchange the two arbitrarily in L_B, and likewise when we talk about the map h or any γ-closed sets.

a) $h(a) \circ_\gamma h(b) \supseteq h(a \cdot b)$. Assume that $c \in h(a \cdot b)$. Then $c \leq_L a \cdot b$, and so $c \leq_L ab$. As $h(a) \circ_\gamma h(b) = \gamma(h(a)h(b))$, we have $ab \in h(a) \circ_\gamma h(b)$, and as γ closed sets are downward closed with respect to $\leq_L$, we have $c \in h(a) \circ_\gamma h(b)$.

b) $h(a) \circ_\gamma h(b) \subseteq h(a \cdot b)$. We know that for all $x \in h(a), x \leq_B a$, and for all $y \in h(b), y \leq_B b$ (lemma 14). Consequently, as it holds in any residuated lattice that $w \leq y, x \leq z \Rightarrow wx \leq yz$, we know that for all $x \in h(a), y \in h(b), x \cdot y \leq_B a \cdot b$, and so $xy \leq_L ab \sim_L a \cdot b$. Consequently, we have $h(a)h(b) \subseteq h(a \cdot b)$. As $h(a \cdot b)$ is closed under γ and γ preserves the inclusion order of sets, it follows that $h(a) \circ_\gamma h(b) \subseteq h(a \cdot b)$.

Case 4: $\star = /$

a) $h(a)/h(b) \supseteq h(a/b)$. Assume $c \in h(a/b)$; then $c \leq_L a/b$. We show that $a/b \in h(a)/h(b)$. By definition we have that $h(a)/h(b)$ is the largest element such that $(h(a)/h(b)) \circ_\gamma h(b) \leq_{Q(\mathbf{B})} h(a)$. As for all $d \in h(b)$, $d \leq_L b$, and $a/b \cdot b \leq_L a$, $a \in h(a)$, we have for all $d \in h(b)$, $a/b \cdot d \leq_L a$, and therefore $a/b \cdot d \in h(a)$. It follows that $a/b \in h(a)/h(b)$, and consequently $c \in h(a)/h(b)$.

b) $h(a)/h(b) \subseteq h(a/b)$. We have $b \in h(b)$ as the largest element with respect to $\leq_L$, and a/b is by definition the largest element such that $a/b \cdot b \leq_B a$. Furthermore, as a is the largest element in $h(a)$ with respect to $\leq_L$, we must have for all $x \in h(a)/h(b)$, $x \leq_L a/b$; and therefore, $x \in h(a/b)$ by lemma 14.

Case 5: $\star = \backslash$ is parallel to 4. □

5.2 Back to Completeness

We now return to the proof of the main theorem. Assume that $\nvdash_{FL_\perp} \Gamma \vdash \alpha$. Then by completeness and finite model property of $\mathbf{FL}_\perp$ for bounded residuated lattices, there is a finite bounded residuated lattice $\mathbf{B}$ and assignment σ, such that $(\mathbf{B}, \sigma) \not\models \Gamma \vdash \alpha$, that is, in $\mathbf{B}$ we have $\sigma(\Gamma) \not\leq \sigma(\alpha)$.

We now take the γ-image $Q(\mathbf{B})$ and the embedding h, as we have described them in the previous section. We have to show that $(Q(\mathbf{B}), h \circ \sigma) \not\models \Gamma \vdash \alpha$, that is, we have $h \circ \sigma(\Gamma) \not\subseteq h \circ \sigma(\alpha)$, where by $h \circ \sigma$ we mean function composition, such that we have $h \circ \sigma(\gamma_1 \bullet ... \bullet \gamma_n) = h(\sigma(\gamma_1)) \cdot ... \cdot h(\sigma(\gamma_n))$.

Lemma 16. *Let $\boldsymbol{B}$ be a (bounded) residuated lattice, s, t terms over B, and $Q(\boldsymbol{B})$, h as defined above. Then $s \leq_B t$ if and only if $h(s) \subseteq h(t)$.*

Actually, for the completeness theorem we would only need the only-if direction; also, we might obtain this as a corollary to theorem 15. But as the proof is quite simple, we show both directions for "completeness".

Proof. *If*: Assume $s \leq_B t$. Then if $u \in h(s)$, we have $u \leq_B s$ (lemma 14). By transitivity, $u \leq_B t$, and therefore $u \in h(t)$. Therefore, $h(s) \subseteq h(t)$.

Only if: Assume $h(s) \subseteq h(t)$. As for all $u \in B$, $u\underline{u} \in L_B$, we have $s \in h(s)$. Therefore, $s \in h(t)$. And so, by lemma 14, $s \leq_B t$. □

To complete the proof of the main theorem, there is one step missing, which is quite straightforward: define L_B for $\mathbf{B}$ as above. Every residuated lattice $\mathbf{B}$ can be embedded into the γ-image $Q(\mathbf{B})$ by a map h, such that if $s \not\leq t$ in $\mathbf{B}$, then $h(s) \not\leq h(t)$ in $Q(\mathbf{B})$. We now have an isomorphic embedding $\mathcal{C} : Q(\mathbf{B}) \to \mathfrak{L}(L_B)$ of $Q(\mathbf{B})$ into the syntactic concept lattice of L_B, which maps the γ closed sets onto their concepts. We thus have the implication: if $s \not\leq t$ in $\mathbf{B}$, then there is a language L_B such that $\mathcal{C} \circ h(s) \not\subseteq \mathcal{C} \circ h(t)$, which is to say $\mathcal{C} \circ h(s) \not\leq_{\mathfrak{L}(L_B)} \mathcal{C} \circ h(t)$. Consequently, if $\nvdash_{FL_\bot} \Gamma \vdash \alpha$, then there is a language L_B over a finite alphabet, such that $(\mathfrak{L}(L_B), \mathcal{C} \circ h \circ \sigma) \not\models \Gamma \vdash \alpha$. This completes the proof of the main theorem.

6 Corollaries

An important feature of our proof is the following: let the logic $\mathcal{L}$ be a fragment of $\mathbf{FL}_\bot$, such that $\mathbf{FL}_\bot$ is a conservative extension of $\mathcal{L}$. We know that there exist such fragments, as $\mathbf{FL}$ is a conservative extension of $L1$, and $\mathbf{FL}_\bot$ is a conservative extension of $\mathbf{FL}$. L (and its fragments) do not satisfy this requirement, and pose some difficulties, which we hope to address in further work.

The algebraic notion corresponding to a fragment in logic is the notion of a reduct. A reduct of an algebra is the same algebra with only a proper subset of connectives; the notion extends easily to classes. So let *RED* be a certain class of reducts of $RL_\bot$, such that for $\mathcal{L}$ a fragment of $\mathbf{FL}_\bot$, $\mathcal{L}$ is complete with respect to *RED*. Then our proof of completeness regarding the class *SCL* of syntactic concept lattices can be easily adapted to give a proof of the completeness of a class of reducts of *SCL* with respect to $\mathcal{L}$, which corresponds to *RED*. The reason is that the crucial step, which is the embedding in theorem 15, is equally valid for any subset of the operators $\{\vee, \wedge, \cdot, /, \backslash\}$. So the question whether a reduct of *SCL* is complete with respect to a fragment $\mathcal{L}$ of $\mathbf{FL}_\bot$ reduces to the question whether there is a strongly complete algebraic semantics for $\mathcal{L}$, in the sense we have specified above.

Now, let SCL_{L1} be the class of SCL reducts with $\{\circ, /, \backslash\}$, which specify a unit, and SCL_{FL} be the class of SCL reducts with operators $\{\circ, /, \backslash, \vee, \wedge\}$, that is, without the constants $\top$ and $\bot$.

We get the following corollaries:

Corollary 17. *The following biimplications hold:*

1. $SCL_{L1} \models \Gamma \vdash \alpha$ *if and only if* $\Vdash_{L1} \Gamma \vdash \alpha$;
2. $SCL_{FL} \models \Gamma \vdash \alpha$ *if and only if* $\Vdash_{FL} \Gamma \vdash \alpha$.

The soundness part follows as a corollary from the soundness theorem 11; the completeness direction can be shown by a simple modification of theorem 15 in the completeness proof. By finite model property, we can again conclude that the same holds if we restrict ourselves to syntactic concept lattices of languages over finite alphabets.

7 Conclusion and Further Work

We have shown strong completeness of the class of syntactic concept lattices for languages over finite alphabets with respect to $\mathbf{FL}_\bot$ and some of its fragments. Our main conclusion is that $\mathbf{FL}_\bot$, $\mathbf{FL}$ and $L1$ are the logics of syntactic concept lattices. Apart from intrinsic mathematical interest, we think that the result presented so far is mainly of preliminary importance for formal linguistic theory. The reason is the following: the main purpose for using syntactic concept lattices in a "post-structuralist" setting is *learning*; so one major interest is to get a finite axiomatization for an infinite lattice. We hope our results will find some application in this task, which is however beyond the scope of this paper.

A further interesting open question is the following[6]: syntactic concept lattices can be extended in a very natural way (see for example [5],[13]). As extents of syntactic concepts, we take subsets of $\Sigma^* \times \Sigma^*$ (instead of subsets of Σ^*), and as intents, we take subsets of $\Sigma^* \times \Sigma^* \times \Sigma^*$ (instead of $\Sigma^* \times \Sigma^*$). Given a language $L \subseteq \Sigma^*$, and a set of pairs of strings $M \subseteq \Sigma^* \times \Sigma^*$, we put $M^\triangleright := \{(x, y, z) : \forall (a, b) \in M, xaybz \in L\}$; $[-]^\triangleleft$ is defined inversely. Obviously, this can be easily generalized to sets of arbitrary $n, n+1$ tuples. Denote the class of (generalized) syntactic concept lattices, where extents are sets of n-tuples, intents sets of $n+1$-tuples, by SCL_n. Now the question is: can our completeness result be generalized from SCL (that is, SCL_1) to SCL_n for any $n \in \mathbb{N}$?

We think the odds are good: all we need is a family of mappings $h_n : n \in \mathbb{N}$, where every $h_n : SCL_1 \to SCL_{n+1}$ is an appropriate embedding of residuated lattices. It might well be that a quite simple mapping will do the job, but we have not found a simple way of verification for the general case. Therefore, we leave this question open for further research.

Acknowledgements. I would like to thank the three anonymous reviewers for many good comments and questions, and Professor Wojciech Buszkowski, who gave me invaluable support in writing this paper.

References

1. Buszkowski, W.: Completeness results for Lambek syntactic calculus. Mathematical Logic Quarterly 32(1-5), 13–28 (1986)
2. Buszkowski, W.: Algebraic structures in categorial grammar. Theor. Comput. Sci. 1998(1-2), 5–24 (1998)
3. Clark, A.: A learnable representation for syntax using residuated lattices. In: de Groote, P., Egg, M., Kallmeyer, L. (eds.) FG 2009. LNCS (LNAI), vol. 5591, pp. 183–198. Springer, Heidelberg (2011)
4. Clark, A.: Learning context free grammars with the syntactic concept lattice. In: Sempere, J.M., García, P. (eds.) ICGI 2010. LNCS, vol. 6339, pp. 38–51. Springer, Heidelberg (2010)
5. Clark, A.: Logical grammars, logical theories. In: Béchet, D., Dikovsky, A. (eds.) LACL 2012. LNCS, vol. 7351, pp. 1–20. Springer, Heidelberg (2012)

[6] This question has been raised by one of the reviewers.

6. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (1991)
7. Farulewski, M.: On Finite Models of the Lambek Calculus. Studia Logica 80(1), 63–74 (2005)
8. Galatos, N., Jipsen, P., Kowalski, T., Ono, H.: Residuated Lattices: An Algebraic Glimpse at Substructural Logics. Elsevier (2007)
9. Harris, Z.S.: Structural Linguistics. The University of Chicago Press (1963)
10. Kanazawa, M.: The Lambek calculus enriched with additional connectives. Journal of Logic, Language, and Information 1, 141–171 (1992)
11. Lambek, J.: The Mathematics of Sentence Structure. The American Mathematical Monthly 65, 154–169 (1958)
12. Lambek, J.: On the calculus of syntactic types. In: Jakobson, R. (ed.) Structure of Language and its Mathematical Aspects, pp. 166–178. American Mathematical Society, Providence (1961)
13. Morrill, G., Valentín, O., Fadda, M.: The displacement calculus. Journal of Logic, Language and Information 20(1), 1–48 (2011)
14. Okada, M., Terui, K.: The finite model property for various fragments of intuitionistic linear logic. J. Symb. Log. 64(2), 790–802 (1999)
15. Pentus, M.: Lambek grammars are context free. In: Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science, Los Alamitos, California, pp. 429–433. IEEE Computer Society Press (1993)
16. Sestier, A.: Contributions à une théorie ensembliste des classifications linguistiques (Contributions to a set–theoretical theory of classifications). In: Actes du Ier Congrès de l'AFCAL, Grenoble, pp. 293–305 (1960)