Automatic Concepts and Automata-theoretic Semantics for the Full Lambek Calculus

Christian Wurm
cwurm@phil.uni-duesseldorf.de
Universität Düsseldorf

March 22, 2017

Abstract

We introduce a new semantics for the (full) Lambek calculus, which is based on an automata-theoretic construction. This automata-theoretic semantics combines languages and relations via closure operators which are based on automaton transitions. We establish the strong completeness of this semantics for the full Lambek calculus via an isomorphism theorem for the syntactic concepts lattice of a language and a construction for the universal automaton recognizing the same language. Automata-theoretic semantics is interesting because it connects two important semantics of the Lambek calculus, namely the relational and the language-theoretic. At the same time, it establishes a strong relation between two canonical constructions over a given language, namely its syntactic concept lattice and its universal automaton.

1 Introduction

The main contributions of this article are the following: we extend some established completeness results for the Full Lambek calculus $\mathbf{FL}_\perp$ and its fragments for syntactic concept lattices (SCL) to regular languages and finite algebras. (Throughout this article, we use completeness in the sense of *strong* completeness with respect to the *internal* consequence relation, that is, the semantics models the set of sequents which are derivable in the logical calculus.) We then present a new kind of semantics we call "automata-theoretic", where closure operators relate strings with transitions they induce in an automaton (we call the resulting structure *automatic concept lattice*). We prove its completeness for $\mathbf{FL}_\perp$ by showing that the syntactic concept lattice of a language is isomorphic to the automatic concept lattice of the universal automaton recognizing the same language (for the universal automaton, consider [**?**]; completeness for syntactic concept lattices has been established in [**?**]).

The semantics of binary relations and composition is usually associated with a "dynamic" interpretation of formulas as computations in programs (see

[**?**]), whereas the language-semantics of stringsets and concatenation is a more "static" interpretation of formulas. Completeness (in our sense) of relational semantics has been shown by Brown&Gurr in [**?**], who use relational quantales and prove results for a wide variety of substructural logics (or put differently, non-commutative linear logics). An even stronger result has been obtained by Pentus in [**?**], who also proved completeness of language (L-)models for the Lambek calculus. Automata-theoretic semantics shows how we can link language and relation models via a Galois connection.

The article is structured as follows: section 2 presents established results on the full Lambek calculus and its semantics; section 3 strengthens these results to regular languages, and section 4 introduces the automata-theoretic semantics and proves (among other results) its completeness.

2 The Logics L, L1, FL, FL$_\perp$ and their Models

2.1 The Logics L, L1, FL and FL$_\perp$

The Lambek calculus **L** was introduced in [**?**]. **L1** is a proper extension of **L**, and **FL**, **FL**$_\perp$ 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, $C_{\mathbf{L}} := \{/, \backslash, \bullet\}$, 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. $Pr \subseteq Tp_C(Pr)$, and if $\alpha, \beta \in Tp_C(Pr)$, $\star \in C$, then $\alpha \star \beta \in Tp_C(Pr)$. As there is usually 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 ','.

In general, uppercase Greek letters range as variables over sequences of types, lowercase Greek letters range over single 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; besides this, the calculi are identical. In **FL** and **FL**$_\perp$ we also allow for empty sequents. Below, we present the standard rules of the Lambek calculus **L** (and **L1**).

(ax) $\quad \alpha \vdash \alpha$

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

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

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

These are the standard rules of **L** and **L1** (roughly as in [**?**]). We now add the two additional connectives $\vee$ and $\wedge$. These are not present in **L**/**L1**, have however been considered as extensions as early as in [**?**], and have been subsequently studied by [**?**].

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

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

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

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

This gives us the logic **FL**. 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 for the bounded lattice elements $\top$ and $\bot$, we introduce two logical constants, which are denoted by the same symbol.

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

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 [**?**], namely admissibility of the cut-rule:

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

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

2.2 Interpretations of L1, FL and $\mathbf{FL}_\bot$

The standard model for **L1** is the class of residuated monoids, which are structures $(M, \cdot, \backslash, /, 1, \leq)$ such that $(M, \cdot, 1)$ is a monoid, $(M, \leq)$ is a partial order, and $\cdot, /, \backslash$ satisfy the law of residuation: for $m, n, o \in M$,

$$m \leq o/n \Leftrightarrow m \cdot n \leq o \Leftrightarrow n \leq m\backslash o.$$

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 (for background on residuated lattices, see [**?**]). A residuated lattice is a structure $(M, \cdot, \vee, \wedge, \backslash, /, 1)$, 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 $(M, \cdot, \vee, \wedge, \backslash, /, 1, \top, \bot)$, where $(M, \cdot, \vee, \wedge, \backslash, /, 1)$ is a residuated lattice, $\top$ is the maximal element of the lattice order $\leq$ and $\bot$ is its minimal element.

We call 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 $\sigma : Pr \to M$ which interprets elements in Pr as elements of the algebra, and extend σ to $\overline{\sigma}$ by defining it appropriately for $1, \top, \bot$, and extending it inductively over our type constructors $C := \{/, \backslash, \bullet, \vee, \wedge\}$ by

1. $\overline{\sigma}(\alpha) = \sigma(\alpha) \in M$, if $\alpha \in Pr$

2. $\overline{\sigma}(\top) = \top$

2' $\overline{\sigma}(\top)$ is an arbitrary $m \in M$ such that for all $\alpha \in Tp_C(Pr)$, $\overline{\sigma}(\alpha) \leq m$.

3. $\overline{\sigma}(\bot) = \bot$

3' $\overline{\sigma}(\bot)$ is an arbitrary $m \in M$ such that for all $\alpha \in Tp_C(Pr)$, $m \leq \overline{\sigma}(\alpha)$.

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 there are two alternative interpretations for $\top, \bot$: one which interprets them as the upper/lower bound of the lattice, which is the standard interpretation, and one which just interprets them as arbitrary elements which only have to be larger/smaller than the interpretation of any other formula. The latter will be called the **non-standard** interpretation and play some role in the sequel, but only for technical reasons.

What we interpret next is the *sequents* of the form $\Gamma \vdash \alpha$. We say that a sequent $\gamma_1, ..., \gamma_i \vdash \alpha$ 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 $\leq$. For derivable sequents with no antecedent, we have the following convention: $\mathcal{M}, \sigma \models \; \vdash \alpha$, iff $1 \leq \overline{\sigma}(\alpha)$, where 1 is the unit element of $\mathcal{M}$ (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 interpretations σ, $\mathcal{M}, \sigma \models \gamma_1, ...\gamma_i \vdash \alpha$ (here we have to distinguish between standard and non-standard interpretations).

2.3 Syntactic Concepts and Galois Connections

We now present a language-theoretic semantics for $\mathbf{FL}_\perp$ which is based on closure operators, namely syntactic concepts. Syntactic concept lattices form a particular case of what is well-known as formal concept lattice (or formal concept analysis, FCA) in computer science (see [**?**]). In linguistics, they have been introduced by Sestier in [**?**]. They were brought back to attention and enriched with residuation by Clark in [**?**] (see also [**?**]), as they turn out to be useful representations for language learning (see [**?**],[**?**]).

Let $\wp(-)$ denote the powerset. Given a language $L \subseteq \Sigma^*$, we define two maps: a map $\rhd : \wp(\Sigma^*) \to \wp((\Sigma^*)^2)$, and $\lhd : \wp((\Sigma^*)^2) \to \wp(\Sigma^*)$, which are defined as follows:

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

(2) for $C \subseteq (\Sigma^*)^2$, $C^\lhd := \{w : \forall (x, y) \in C, xwy \in L\}$

So a set of strings which is mapped to the 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. This results in a Galois connection between the two $\subseteq$-ordered structures of closed sets and contexts, see [**?**],[**?**]. For extension of these maps to larger tuples, consider [**?**]). Importantly, all these are special applications of the general theory of Galois connections; for background, see [**?**]. Obviously, $[-]^\lhd$ and $[-]^\rhd$ are only defined with respect to a given language L, otherwise they are meaningless. As long as it is clear about which language (if any particular language) we are speaking, we will omit however any reference to it, to keep notation perspicuous. Regardless of the underlying objects, the two compositions of the maps, $[-]^{\lhd\rhd}$ and $[-]^{\rhd\lhd}$, form **closure operators**. Note also that for any set of strings M and contexts C, $M^\rhd = M^{\rhd\lhd\rhd}$ and $C^\lhd = C^{\lhd\rhd\lhd}$. A set M is **closed**, if $M^{\rhd\lhd} = M$ etc. The closure operator $\rhd\lhd$ gives rise to a lattice $(\mathcal{B}_L, \leq)$, where the elements of $\mathcal{B}_L$ are the sets $M \subseteq \Sigma^*$ such that $M = M^{\rhd\lhd}$, and $\leq$ is interpreted as $\subseteq$. The same can be done with the set of closed contexts. Given these two lattices, $[-]^\rhd$ and $[-]^\lhd$ form a Galois connection between the two (see [**?**] for more background), that is:

1. $M \leq N \Leftrightarrow M^\rhd \geq N^\rhd$, and
2. $C \leq D \Leftrightarrow C^\lhd \geq D^\lhd$.

A **syntactic concept** is usually defined to be an ordered pair, consisting of a closed set of strings, and a closed set of contexts, so it has the form (S, C), such that $S^\rhd = C$ and $C^\lhd = S$; $S^\rhd$ is the set of all contexts in which all strings in S can occur; inversely for $C^\lhd$. For our purposes, we mostly need to consider only the left component, so we suppress the contexts and only consider the stringsets of the form $M^{\triangleright\triangleleft}$. An exception to this convention is section 4, where we will make use of concepts as pairs (M, C) with $M = C^{\triangleleft}$, $C = M^{\triangleright}$, as it will increase readability in this case. For all operations we define below, it can be easily seen that the resulting structures are isomorphic. So when we refer to a concept, we only mean a $[-]^{\triangleright\triangleleft}$ closed set of strings (with the exception of section 4), the concept in the classical sense being easily reconstructible.

Definition 1 *For $[-]^{\triangleright\triangleleft}$ defined with respect to $L \subseteq \Sigma^*$, let $\mathcal{B}_L$ denote the set of $[-]^{\triangleright\triangleleft}$-closed subsets of Σ^*. This set forms a bounded lattice $(\mathcal{B}_L, \wedge, \vee, \top, \bot)$, where $\top = \Sigma^*$, $\bot = \emptyset^{\triangleright\triangleleft}$, and for $M, N \in \mathcal{B}_L$, $M \wedge N = M \cap N$, $M \vee N = (M \cup N)^{\triangleright\triangleleft}$.*

It is also easy to verify that this forms a complete lattice, as infinite joins are defined by (closure of) infinite unions, infinite meets by infinite intersections.

2.4 Monoid Structure and Residuation for Syntactic Concepts

The set of concepts of a language forms a lattice. In addition, we can also give it the structure of a monoid: for concepts M, N, we define $M \circ N := (M \cdot N)^{\triangleright\triangleleft}$, where $M \cdot N = \{wv : w \in M, v \in N\}$. We usually write MN for $M \cdot N$, if M, N are sets of strings. '$\circ$' is associative on concepts: for $M, N, O \in \mathcal{B}_L$, $M \circ (N \circ O) = (M \circ N) \circ O$. This follows from the associativity of $\cdot$-concatenation and the fact that $[-]^{\triangleright\triangleleft}$ is a **nucleus**, that is, it is a closure operator and in addition it satisfies $M^{\triangleright\triangleleft}N^{\triangleright\triangleleft} \subseteq (MN)^{\triangleright\triangleleft}$.

It is easy to see that the neutral element of '$\circ$' is $\{\epsilon\}^{\triangleright\triangleleft}$ (which need not be $\{\epsilon\}$). The monoid operation respects the partial order of the lattice, that is, for $X, Y, Z, W \in \mathcal{B}_L$, if $X \leq Y$, then $W \circ X \circ Z \leq W \circ Y \circ Z$. A stronger property is the following: $\circ$ distributes over infinite joins, that is, we have

$$\bigvee_{Z \in \mathbf{Z}} (X \circ Z \circ Y) = X \circ \bigvee \mathbf{Z} \circ Y$$

Here $\leq$ follows algebraically ($\circ$ respects the order $\subseteq$), and $\geq$ follows from the fact that 1. $\bigcup$ distributes over $\cdot$ (infinite unions distribute over concatenation), and 2. $[-]^{\triangleright\triangleleft}$ is a nucleus. We can thus also conceive of syntactic concepts with $\bigvee, \circ, 1$ as *quantales*, and in quantales we can easily define residuals as follows:

Definition 2 *Let X, Y be concepts. We define the right residual $X/Y := \bigvee\{Z : Z \circ Y \leq X\}$, the left residual $Y\backslash X := \bigvee\{Z : Y \circ Z \leq X\}$.*

Note that this is an entirely abstract definition which does not make reference to any underlying structure. It works because of the well-known fact that for any complete lattice with a monoid operation distributing over infinite joins, residuals defined as above, we have $Y \leq X\backslash Z$ iff $X \circ Y \leq Z$ iff $X \leq Z/Y$.

Definition 3 *The **syntactic concept lattice** of a language L is defined as $SCL(L) := (\mathcal{B}_L, \circ, \wedge, \vee, /, \backslash, 1, \top, \bot)$, where $\mathcal{B}_L, \wedge, \vee, \top, \bot$ are defined as in definition 1, $1 = \{\epsilon\}^{\triangleright\triangleleft}$, and $\circ, /, \backslash$ are as defined above.*

The syntactic concept lattice thus is a residuated lattice (see [**?**]). We will denote by SCL the class of all lattices of the form $SCL(L)$ for some language L, without any further requirement regarding L. We can apply the definition of interpretations to SCL, so it is clear how $\mathbf{FL}_\perp$ is interpreted in SCL.

The algebraic notion corresponding to the notion of 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 easily extends to classes. We let $SCL_{\boldsymbol{FL}}$ be the class of SCL reducts with operators $\{\circ, /, \backslash, \vee, \wedge\}$ without the constants $\top$ and $\perp$, and $SCL_{\mathbf{L1}}$ be the class of SCL reducts with $\{\circ, /, \backslash\}$, which all specify a unit. So it is clear how the logical fragments $\mathbf{FL}, \mathbf{L1}$ are interpreted in the reducts appropriate reducts.

2.5 Completeness: Previous Results

There are a number of completeness results for the logics we have considered here. We quickly present the ones which will be important in the sequel.

Theorem 4 *1. $RM \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{L1}} \Gamma \vdash \alpha$*

2. $RL \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}} \Gamma \vdash \alpha$

3. $RL_\perp \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}_\perp} \Gamma \vdash \alpha$

For reference on theorem 4, see [**?**], [**?**], [**?**]. These completeness results can actually be strengthened to the *finite model property.* A logic, equipped with a class of models and interpretations, is said to have finite model property if it is complete in the finite, that is, theorem 4 remains valid if we restrict ourselves to finite models.

Theorem 5 *1. **L1** has finite model property*

*2. **FL** has finite model property*

3. $\boldsymbol{FL}_\perp$ has finite model property

For the first and second claim, consider [**?**]; the third and forth has been established in [**?**]. Theorem 5 is crucial to show that completeness for syntactic concept lattices and their reducts also holds if we restrict ourselves to languages over finite alphabets. The following results have been proved in [**?**].

Theorem 6 *1. $SCL \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}_\perp} \Gamma \vdash \alpha$*

2. $SCL_{\boldsymbol{FL}} \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}} \Gamma \vdash \alpha$

3. $SCL_{\boldsymbol{L1}} \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{L1}} \Gamma \vdash \alpha$

L requires some additional considerations, as **L1** is *not* a conservative extension of it. The soundness directions follow *a fortiori* from theorem 4. We will now strengthen the completeness result to syntactic concept lattices over regular languages, for which we have to provide a sketch of the original completeness proof.

3 Regular Languages and $SCL(REG)$

Let $\mathbf{B} = (B, \cdot, \vee, \wedge, /, \backslash, 1, \top, \bot)$ be a bounded residuated lattice. We denote the partial order of $\mathbf{B}$ by $\leq_B$, equality by $=_B$. Define $\Sigma' := \{\underline{b} : b \in B\}$, and put $\Sigma = B \cup \Sigma'$. Let $SI^*_B := \{b_1 ... b_i : b_1 \cdot ... \cdot b_i \leq_B 1\}$ be the set of sub-identity words over B. We define the language $L_B \subseteq \Sigma^*$ as the set of strings

$$L_B := \{b_1 b_2 ... b_n \underline{b} w : b_1 \cdot b_2 \cdot ... \cdot b_n \leq_{\mathbf{B}} b,\, w \in SI^*_B\}.$$

For a string $w = b_1 ... b_n \in B^*$, by $w^\bullet$ we denote the term $b_1 \cdot ... \cdot b_n$; we put $\epsilon^\bullet = 1$. By $w \sim_L v$, we mean that $xwy \in L$ iff $xvy \in L$. It is easy to prove that $w^\bullet =_B v^\bullet$ iff $w \sim_{L_B} v$ (see [**?**]).

Proposition 7 *For every bounded residuated lattice* $\mathbf{B}$*, there is an faithful embedding* $\psi : \mathbf{B} \to SCL(L_B)$*, such that*

1. $\psi(\top) = B^*$, $\psi(\bot) = \{\bot\}^{\rhd\lhd}$.
2. $\psi(1) = \{\epsilon\}^{\rhd\lhd}$.

For proof of this fact, consider [**?**],[**?**]. Note that $\psi(\bot)$ is not the $\bot$-element of $SCL(L_B)$, as we can never substitute $\underline{b}$ with $\bot$. Note also that $\psi(\top)$ is not maximal in $SCL(L_B)$, as $\psi(\top) = B^* \subsetneq \Sigma^*$.

From here, it is easy to complete the proof of theorem 6: just use the faithful embedding to perform the usual contraposition, where from $\not\Vdash_{FL_\bot} \Gamma \vdash \alpha$ and algebraic completeness then follows $SCL \not\models \Gamma \vdash \alpha$. This completes the proof of theorem 6.1. Note however that the resulting interpretation is non-standard: we have $\psi(\top) = B^* \neq \Sigma^*$, and $\bot\ (\bot) = \{\bot\}^{\rhd\lhd} \neq \emptyset^{\rhd\lhd}$.

An important feature of our proof is that it works for all reducts of bounded residuated lattices (a reduct is the same algebra with a proper subset of connectives); and hence it allows to prove completeness for all logics $\mathcal{L}$ for which $\mathbf{FL}_\bot$ is a conservative extension (this holds for **L1** and **FL**; **L** and its fragments do not satisfy this requirement). Hence we also have a proof for the other parts of theorem 6.

By REG we denote the class of regular languages. Given an equivalence relation $\sim$ over Σ^*, we put $[w]_\sim = \{v : w \sim v\}$, and $\Sigma^*_\sim = \{[w]_\sim : w \in \Sigma^*\}$. So $\Sigma^*_{\sim_L}$ denotes the set of $\sim_L$-congruence classes over Σ^*. Recall that a language $L \subseteq \Sigma^*$ is regular if and only if $\Sigma^*_{\sim_L} = \{[w]_{\sim_L} : w \in \Sigma^*\}$ is finite. The next lemma follows easily (see also [**?**]):

Lemma 8 *$SCL(L)$ is finite if and only if L is regular.*

Let $\mathbf{B}$ be an arbitrary algebra equipped with a semigroup operation and a partial order respecting it, so we can define L_B as above (this covers all algebras we consider in this paper). Then for $w, v \in B^*$, we have $w \sim_L v$ iff $w^\bullet =_{\mathbf{B}} v^\bullet$. But recall that $B \subsetneq \Sigma$ in this case!

Lemma 9 ***B*** *is a finite algebra if and only if L_B is a regular language.*

Proof. $\Leftarrow$ Contraposition: if $\mathbf{B}$ is infinite, there is an infinite sequence of $=_B$-distinct objects $(w_1)^\bullet, (w_2)^\bullet, ... \in B$, so there are $w_1, w_2, ...$ which are not $\sim_L$-equivalent.

$\Rightarrow$ We construct $L'_B = \bigcup_{b\in B}\{w\underline{b} : w^\bullet \leq_B b\}$. Assume a language $\{w\underline{b} : w^\bullet \leq_B b\}$ is not regular. Then $\Sigma^*_{\sim_L}$ is infinite, and there is an infinite sequence of words $w_1, w_2, ...$, such that if $i \neq j$, then $w_i \not\sim_L w_j$. So there is an infinite sequence of objects $(w_1)^\bullet, (w_2)^\bullet, ... \in B$, such that if $i \neq j$, then $(w_i)^\bullet \neq_B (w_j)^\bullet$. Thus $\mathbf{B}$ is infinite – contradiction. Hence $\{w\underline{b} : w^\bullet \leq_B b\}$ is regular, and as B is finite, L'_B is a finite union of regular languages, which is still regular. Finally, SI^*_B is regular for the same reason as above, and so $L_B = L'_B \cdot SI^*_B$ is also regular. $\square$

Let C be a class of languages; then by $SCL(C)$ we denote the class of structures $SCL(L) : L \in C$. So $SCL(REG)$ equals the class of finite syntactic concept lattices. As we have said, a finite algebra $\mathbf{B}$ entails a language L_B over a finite alphabet; the last lemma shows us that it also entails that L_B is regular. Moreover, as $\mathbf{L1}, \mathbf{FL}, \mathbf{FL}_\perp$ have the finite model property, for completeness it is sufficient to consider only finite algebras, and consequently we can strengthen theorem 6 to the following:

Corollary 10 *1. $SCL(REG) \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\mathbf{FL}_\perp} \Gamma \vdash \alpha$.*

2. $SCL_{\mathbf{FL}}(REG) \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\mathbf{FL}} \Gamma \vdash \alpha$.

3. $SCL_{\mathbf{L1}}(REG) \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\mathbf{L1}} \Gamma \vdash \alpha$.

4 Automata-theoretic Semantics

4.1 Automata-theoretic Preliminaries

We now introduce a new class of bounded residuated lattices, the **automatic concept lattices**. It is very similar to SCL in that it is based on a Galois connection which, provided the certain conditions, gives rise to a nucleus. As we will learn from the main result of this section, the isomorphism theorem, if we consider structures only up to isomorphism, then automatic concept lattices form a proper generalization of syntactic concept lattices (in fact, in general they are not even residuated lattices[1]).

One can present automata in many different ways, the most standard one being probably the following: an automaton as *state-transition system* is a tuple $\mathfrak{A} = (\Sigma, Q, \delta, F, I)$, where Σ is a finite input alphabet, Q a set of states, $\delta \subseteq Q \times \Sigma \times Q$ a transition relation, $F \subseteq Q$ a set of accepting states, $I \subseteq Q$ the set of initial states. This notation of automata is somewhat clumsy in connection with the techniques we use later on, so we will choose a slightly different presentation which we call **relational**. This is a notional change we adopt for convenience. We define a **semi-automaton** as a tuple $\langle \Sigma, \phi \rangle$, where ϕ is a map $\phi : \Sigma \to \wp(Q \times Q)$, mapping letters in Σ onto relations over Q,

[1]Thanks to an anonymous reviewer for pointing this out!

where we use Q is an arbitrary (finite or infinite) carrier set. It is extended to strings by interpreting concatenation as **relation composition** '$;$', where $R;R' = \{(x,y) : (x,z) \in R, (z,y) \in R'\}$. So we have $\phi(aw) = \phi(a);\phi(w)$, and ϕ is a homomorphism from the free monoid Σ^* into a relation monoid over Q, and a word $w \in \Sigma^*$ then induces a relation $\phi(w) \subseteq Q \times Q$. Defining ϕ as a homomorphism, we should take care of $\phi(\epsilon)$, which we simply define by $\phi(\epsilon) = \mathsf{id}_Q := \{(q,q) : q \in Q\}$.[2] To get a full automaton, we still need an *accepting relation*. One usually specifies a set of initial and accepting states, yielding an accepting relation $I \times F$. As for us, acceptance will only play a minor role, we will take a slightly more general convention and assume that automata specify an **accepting relation** $F_R \subseteq Q \times Q$. Thus a full **automaton** is a tuple $\langle \Sigma, \phi, F_R \rangle$. We define the language recognized by an automaton $\mathcal{A} = \langle \Sigma, \phi, F_R \rangle$ by $L(\mathcal{A}) := \{w \in \Sigma^* : \phi(w) \cap F_R \neq \emptyset\}$.

4.2 Automatic Concepts

In what is to follow, we will take the "canonical view" on formal concepts, that is: concepts are not simply $[-]^{\triangleright\triangleleft}$-closed sets, but pairs (M,C) such that $M^{\triangleright} = C$, $C^{\triangleleft} = M$ (this entails that both are closed). Henceforth, we will use the maps $[-]^{\triangleright}, [-]^{\triangleleft}$ for syntactic concepts only. Given a semi-automaton $\langle \Sigma, \phi \rangle$, $M \subseteq \Sigma^*$, $R \subseteq Q \times Q$, we define the two polar maps

$$M^{\blacktriangleright} = \bigcap_{w \in M} \phi(w) \tag{3}$$

$$R^{\blacktriangleleft} = \{w : \phi(w) \supseteq R\} \tag{4}$$

It is easy to see that these maps establish a Galois connection and their compositions $[-]^{\blacktriangleright\blacktriangleleft}$, $[-]^{\blacktriangleleft\blacktriangleright}$ are closure operators. An **automatic concept** is then a pair (M,R) with $M^{\blacktriangleright} = R$, $R^{\blacktriangleleft} = M$ (of course, the underlying (semi-)automaton is understood as given). We denote the set of automatic concepts, given an automaton $\mathcal{A}$, by $\mathfrak{A}_{\mathcal{A}}$. Importantly, the map $[-]^{\blacktriangleright\blacktriangleleft}$ does **not** form a nucleus on Σ^*, and in general, $[-]^{\blacktriangleright\blacktriangleleft}$-closed concatenation does not distribute over infinite joins. Consequently, we cannot simply define a residuated lattice of concepts in the usual fashion. Rather, we have to restrict our attention to a certain class of automata.

Definition 11 *A (semi-)automaton $\langle \Sigma, \phi(, F_R) \rangle$ is **nuclear**, if for all $M, N \subseteq \Sigma^*$, $(\bigcap_{w \in M} \phi(w));(\bigcap_{v \in N} \phi(w)) = \bigcap_{wv \in MN} \phi(wv)$.*

Note that $\subseteq$ always holds. The equality ensures that $[-]^{\blacktriangleright\blacktriangleleft}$ is a nucleus on Σ^*, because if $w \in M^{\blacktriangleright\blacktriangleleft}$, $v \in N^{\blacktriangleright\blacktriangleleft}$, then $\phi(w) \supseteq M^{\blacktriangleright}, \phi(v) \supseteq N^{\blacktriangleright}$. Hence $\phi(wv) = \phi(w);\phi(v) \supseteq M^{\blacktriangleright};N^{\blacktriangleright} = (MN)^{\blacktriangleright}$, and hence $wv \in M^{\blacktriangleright\blacktriangleleft}$. So being nuclear boils down to composition distributing over (infinite) intersections of closed sets. We will later see that for every automaton there is a nuclear automaton recognizing the same language.

[2]But in principle, nothing prevents us from having $(x,y) \in \phi(\epsilon)$ with $x \neq y$ – we just have to make sure that for all $a \in \Sigma$, we have $\phi(\epsilon);\phi(a) = \phi(a) = \phi(a);\phi(\epsilon)$.

We define $(M,R)\wedge(N,S)=(M\cap N,(R\cup S)^{\blacktriangleleft\blacktriangleright})$, $(M,R)\vee(N,S)=((M\cup N)^{\blacktriangleright\blacktriangleleft},R\cap S)$, and $(M,R)\circ(N,S)=((MN)^{\blacktriangleright\blacktriangleleft},(MN)^{\blacktriangleright})$. It is easy to see that $\wedge,\vee$ can be extended to the infinitary operators $\bigwedge,\bigvee$ (as they are based on sets). Moreover, in case the underlying automaton is nuclear, $\circ$ distributes over infinite joins (because it is a nuclear operation), so the residuals are easily defined in the usual fashion by $M/N=\bigvee\{X:X\circ N\leq M\}$, $N\backslash M=\bigvee\{X:N\circ X\leq M\}$. We put $\top=(\Sigma^*,(\Sigma^*)^{\blacktriangleright})$, $\bot=(\emptyset^{\blacktriangleright\blacktriangleleft},\emptyset^{\blacktriangleright})$, where by convention we put $\emptyset^{\blacktriangleright}=\bigcup_{w\in\Sigma^*}\phi(w)$. Finally, we put $1=(\{\epsilon\}^{\blacktriangleright\blacktriangleleft},\phi(\epsilon))$ (recall that $\phi(\epsilon)=\mathsf{id}_C$ by definition). So given a nuclear automaton $\mathcal{A}$, we have the complete bounded residuated lattice $(\mathfrak{A}_{\mathcal{A}},\circ,\wedge,\vee,/,\backslash,1,\top,\bot)$, which is the **automatic concept lattice** of $\mathcal{A}$, for short $ACL(\mathcal{A})$. As is easy to see, acceptance does not play a role for the automatic concept lattice, so it is sufficient to refer to semi-automata. By ACL we denote the class of all $ACL(\mathcal{A})$ for $\mathcal{A}$ an arbitrary nuclear (semi-)automaton, and we define the reducts $ACL_{\mathbf{FL}},ACL_{\mathbf{L1}}$ in the same way we did for SCL.

We will refer to the straightforward interpretation of $\mathbf{FL}_{\bot}$ and its fragments into automatic concept lattices as **automata-theoretic semantics**, and write $ACL\models\Gamma\vdash\alpha$ in the usual sense that for all nuclear semi-automata $\mathcal{A}$, interpretations σ into $ACL(\mathcal{A})$, we have $\overline{\sigma}(\Gamma)\leq_{ACL(\mathcal{A})}\overline{\sigma}(\alpha)$; same for reducts $ACL_{\mathbf{FL}},ACL_{\mathbf{L1}}$ etc.

For $ACL(\langle\phi,\Sigma,F_R\rangle)$, F_R is irrelevant. Still, F_R is useful because it links automata to languages, which in turn is necessary to establish the relation between ACL and SCL. For what is to follow, the phrase "automaton recognizing L" could be exchanged with "semi-automaton $\langle\phi,\Sigma\rangle$ for which there is F_R such that $L(\langle\phi,\Sigma,F_R\rangle)=L$", which however is clumsy to repeat. As automata are related to languages, there should be thus a relation between $ACL(\mathcal{A})$ and $SCL(L)$, provided that $L(\mathcal{A})=L$. In particular, one knows that in this case, if $w\not\sim_L v$, then $\phi(w)\neq\phi(v)$ – otherwise, the automaton could not distinguish acceptance of words containing the two substrings. The inverse direction is obviously incorrect, that is, $\phi(w)\neq\phi(v)$ does not imply anything for w,v in L, as the automaton can make as many (unnecessary) distinctions as it desires (this is related to the issue of minimality of automata). From this, we can for example conclude the following: if $L(\mathcal{A})=L$, then for $(M,C)\in\mathcal{B}_L$, there are $(M_i,R_i)\in\mathfrak{A}_{\mathcal{A}}$ for $i\in I$, such that $M=\bigcup_{i\in I}M_i$. However, this does **not** entail (as one might conjecture) that we have $M=(\bigcup_{i\in I}M_i)^{\blacktriangleright\blacktriangleleft}$, which by completeness of the lattice would entail that there is an automatic concept $(M,R)\in\mathfrak{A}_{\mathcal{A}}$.[3] In general, there is no homomorphic relation between the two structures, so there is no trivial way to extend completeness for SCL to completeness for automata-theoretic semantics via embeddings; instead, we have to recur to a peculiar automata-theoretic construction.

[3]Imagine the following situation: for every $w\in M$, there is $(r_w,r'_w)\in\phi(m)$, such that $\phi(x)\circ\{(r_w,r'_w)\}\circ\phi(y)\cap F\neq\emptyset$, but $(r_w,r'_w)\notin M^{\blacktriangleright}$. So every $w\in M$ has its own peculiar pair which makes sure $xwy\in L$. Obviously, for $\bigvee_{w\in M}(\{w\}^{\blacktriangleright\blacktriangleleft},\phi(w))=(N,R)$, we have $N\supseteq M$. Still there can be $v\in N$, $v\notin M$, because $\phi(v)\supseteq R$, but $\phi(v)$ does not contain *any* of the pairs which ensure that $xMy\subseteq L$, and in fact $xvy\notin L$.

4.3 The Universal Automaton

There are always infinitely many distinct automata recognizing a language (even modulo a labelled-graph based notion of automaton-isomorphism). We will now consider a particular automaton type which is uniquely specified for every language and which allows us to connect syntactic concepts to automatic concepts. This is the so-called **universal automaton** (see [?]). The observation that there is some connection between syntactic concepts and the universal automaton is due to A.Clark and has been elaborated in [?]. However, the direct correlation we establish here is new to my knowledge. The universal automaton is based on the notion of a factorization of a language. (X, Y) is a **factorization** of L, iff

1. $XY \subseteq L$, and
2. if $X \subseteq X', Y \subseteq Y'$ and $X'Y' \subseteq L$, then $X = X', Y = Y'$.

We denote the set of L-factorizations with $\mathit{fact}(L)$. So a factorization is a maximal decomposition of L into two factors. We denote the (unique) universal automaton for a language L by $U(L)$. The factorizations of L form the set of states of $U(L)$. We define I, the set of initial factorizations and F, the set of final factorizations as follows: $I = \{(X, Y) \in \mathit{fact}(L) : \epsilon \in X\}$, $F = \{(X, Y) \in \mathit{fact}(L) : \epsilon \in Y\}$. Then for $L \subseteq \Sigma^*$, one defines the **universal automaton** $U(L) := (\Sigma, \mathit{fact}(L), I, F, \delta)$, where for $a \in \Sigma$, $((X, Y), a, (X', Y')) \in \delta$ iff $Xa \subseteq X'$ iff $Y \supseteq aY'$. The latter bi-implication is easy to see: if $Xa \subseteq X'$, then $XaY' \subseteq L$, and so $aY' \subseteq Y$ (same for the other direction). The results of this subsection can be found in [?]; we present them as they are necessary for the proof of the isomorphism theorem, but we omit the proofs. Until now, we have given the "normal" presentation of universal automata. To proceed, we quickly need to bring the universal automaton into our "relational form" for automata: we put $U(L) = \langle \Sigma, \phi, I \times F \rangle$, where for all $a \in \Sigma$, we have $\phi(a) = \{((X, Y), (X', Y')) : (X, Y), (X', Y') \in \mathit{fact}(L) \text{ and } Xa \subseteq X'\}$. We define the maps $[-]^{\rightarrow}, [-]^{\leftarrow}$ by

$$M^{\rightarrow} = \{w : Mw \subseteq L\} \tag{5}$$

$$M^{\leftarrow} = \{w : wM \subseteq L\} \tag{6}$$

The compositions $[-]^{\rightarrow\leftarrow}, [-]^{\leftarrow\rightarrow}$ are closure operators, and $[-]^{\rightarrow}, [-]^{\leftarrow}$ establish a Galois connection between closed sets of strings (see [?] for the connection of $[-]^{\rightarrow}$ and $[-]^{\triangleright}$ etc.). A factorization is then exactly a pair of sets (M, N) such that $M^{\rightarrow} = N$, $N^{\leftarrow} = M$ (this entails $M = M^{\rightarrow\leftarrow}, N = N^{\leftarrow\rightarrow}$). Depending on L, there might be trivial factorizations $(\Sigma^*, \emptyset)$, $(\emptyset, \Sigma^*)$.

Lemma 12 *For $(X, Y), (X', Y') \in \mathit{fact}(L)$, $W \subseteq \Sigma^*$, the following are equivalent:*

1. $XW \subseteq X'$
2. $WY' \subseteq Y$

3. $XWY' \subseteq L$.

Lemma 13 *For every $L \subseteq \Sigma^*$, $w \in \Sigma^*$, for $U(L)$ we have $((X,Y),(X',Y')) \in \phi(w)$ iff $Xw \subseteq X'$ iff $wY' \subseteq Y$ iff $XwY' \subseteq L$.*

Lemma 14 $L(U(L)) = L$.

That is, the universal automaton of L recognizes L. It is a straightforward consequence of the Myhill-Nerode theorem that *fact*(L) is finite if and only if L is regular. This entails the following:

Lemma 15 *$U(L)$ is a finite automaton if and only if L is regular.*

4.4 An Isomorphism Theorem for ACL and SCL

We have said that there is no homomorphic (or in fact, any simple structural) relation between $SCL(L)$ and $ACL(\mathcal{A})$ for all $\mathcal{A}$ such that $L(\mathcal{A}) = L$. This is despite the fact that $\mathcal{A}$ must make the relevant distinctions between strings distinct modulo $\sim_L$. Things change if we look at the universal automaton instead of automata in general. For two algebras $\mathbf{B}, \mathbf{B}'$, we write $\mathbf{B} \cong \mathbf{B}'$ if there is an isomorphism from one to the other, that is a bijection which preserves all results of all operations. We can establish the following, surprisingly strong connection:

Theorem 16 *(Isomorphism theorem) $ACL(U(L)) \cong SCL(L)$*

That is, the automatic concept lattice for the universal automaton over L is isomorphic to the syntactic concept lattice of L. The following generalization of lemma 13 is quite simple, but will be very helpful in the proof of the isomorphism theorem. Let $[-]^{\blacktriangleright}, [-]^{\blacktriangleleft}$ below be defined with respect to $U(L)$.

Lemma 17 *For $(X,Y),(X',Y') \in$ fact(L), $((X,Y),(X',Y')) \in M^{\blacktriangleright}$ if and only if $XMY' \subseteq L$.*

Proof. *If*: Assume $XMY' \subseteq L$. Then for every $w \in M$, we have $XwY' \subseteq L$, hence $((X,Y),(X',Y')) \in \phi(w)$, hence $((X,Y),(X',Y')) \in M^{\blacktriangleright}$.

Only if: Assume $((X,Y),(X',Y')) \in M^{\blacktriangleright}$. Then for all $w \in M$, we have $((X,Y),(X',Y')) \in \phi(w)$. Hence for all $w \in M$, $XwY' \subseteq L$, so $XMY' \subseteq L$. $\square$

We can now show that universal automata are nuclear, so they provide a sound semantics for the full Lambek calculus.

Lemma 18 *Let $[-]^{\blacktriangleright}, [-]^{\blacktriangleleft}$ we defined with respect to $U(L)$ for some language L. Then $M^{\blacktriangleright} ; N^{\blacktriangleright} = (MN)^{\blacktriangleright}$. Hence for every language L, $U(L)$ is nuclear.*

Proof. $\subseteq$ Holds in general, by set-theoretic properties.

$\supseteq$ Assume $((X,Y'),(X',Y)) \in (MN)^{\blacktriangleright}$. Then $XMNY \subseteq L$.

Firstly, we have $((X,Y'),((NY)^{\leftarrow},(NY)^{\leftarrow\rightarrow}) \in M^{\blacktriangleright}$: we have $(X,Y') \in$ *fact*(L) by assumption, $((NY)^{\leftarrow},(NY)^{\leftarrow\rightarrow}) \in$ *fact*(L) by definition of $[-]^{\leftarrow}$,

$[-]^{\rightarrow}$, and since $XMNY \subseteq L$, we also have $XM(NY)^{\leftarrow\rightarrow} \subseteq L$. So the claim follows from lemma 17.

Secondly, we have $(((NY)^{\leftarrow}, (NY)^{\leftarrow\rightarrow}), (X', Y)) \in N^{\blacktriangleright}$: $(X', Y) \in \mathit{fact}(L)$ by assumption, and we have $((NY)^{\leftarrow}NY \subseteq L$ by definition of $[-]^{\leftarrow}$, hence the claim follows again from lemma 17.

Consequently, by definition of **;**, we have $((X, Y'), (X', Y)) \in M^{\blacktriangleright}\mathbf{;}N^{\blacktriangleright}$. $\square$

In the sequel, $[-]^{\triangleright}, [-]^{\triangleleft}$ refer to SCL-closure w.r.t. to some fixed $L \subseteq \Sigma^*$, $[-]^{\blacktriangleright}, [-]^{\blacktriangleleft}$ to ACL-closure w.r.t. to $U(L)$ (referring to the same language!). Now comes the crucial lemma for the isomorphism theorem:

Lemma 19 *For all $M \subseteq \Sigma^*$, $M^{\triangleright\triangleleft} = M^{\blacktriangleright\blacktriangleleft}$.*

Proof. $M^{\triangleright\triangleleft} \subseteq M^{\blacktriangleright\blacktriangleleft}$. Assume $w \in M^{\triangleright\triangleleft}$. Then whenever $xMy \subseteq L$, then $xwy \in L$. If $((X, Y), (X', Y')) \in M^{\blacktriangleright}$, then $XMY' \subseteq L$ (by lemma 17). However, if $XMY' \subseteq L$, then $XwY' \subseteq L$, hence (by the equivalence in lemma 12) $Xw \subseteq X', wY' \subseteq Y$. Hence we have $((X, Y), (X', Y')) \in \phi(w)$ for all $((X, Y), (X', Y')) \in M^{\blacktriangleright}$. Hence we have $w \in M^{\blacktriangleright\blacktriangleleft}$.

$M^{\blacktriangleright\blacktriangleleft} \subseteq M^{\triangleright\triangleleft}$. Assume $w \in M^{\blacktriangleright\blacktriangleleft}$, and take an arbitrary $(x, y) \in M^{\triangleright}$. Put $X = (My)^{\leftarrow}$, $Y = (XM)^{\rightarrow}$. It is easy to see that 1. $x \in X, y \in Y$ (obvious), and 2. $((X, X^{\rightarrow}), (Y^{\leftarrow}, Y)) \in M^{\blacktriangleright}$ (by lemma 17). Since $w \in M^{\blacktriangleright\blacktriangleleft}$, we have $M^{\blacktriangleright} \subseteq \phi(w)$, and so $((X, X^{\rightarrow}), (Y^{\leftarrow}, Y)) \in \phi(w)$, which holds iff $XwY \subseteq L$, entailing $xwy \in L$. Hence $w \in M^{\triangleright\triangleleft}$ $\square$

This already entails that the operations and constants in the respective lattices yield the same result, because they are based on the same underlying set-operations, of which we simply take the (same) closure. We denote operations in $SCL(L)$ as usual; the operation in $ACL(U(L))$ corresponding to $\star$ in $SCL(L)$ will be denoted by $\star'$. We distinguish the constants of different structures by subscripts $\top_{SCL(L)}$ etc. As concepts are tuples, we write, for tuples $(X_1, X_2), (Y_1, Y_2)$, $(X_1, X_2) =_1 (Y_1, Y_2)$ iff $X_1 = Y_1$, that is, if their first components are identical.

Corollary 20 1. *For $\star \in \{\wedge, \vee, \circ, /, \backslash\}$, $\star$ defined w.r.t. $SCL(L)$, $\star'$ defined w.r.t. $ACL(U(L))$, $(M, M^{\triangleright}) \star (N, N^{\triangleright}) =_1 (M, M^{\blacktriangleright}) \star' (N, N^{\blacktriangleright})$.*

2. $\top_{SCL(L)} =_1 \top_{ACL(U(L))}$

3. $\bot_{SCL(L)} =_1 \bot_{ACL(U(L))}$

4. $1_{SCL(L)} =_1 1_{ACL(U(L))}$

Now it is easy to construct an isomorphism $i : SCL(L) \to ACL(U(L))$: for every $(M, C) \in \mathcal{B}_L$, we put $i(M, C) = (M, M^{\blacktriangleright})$. This completes the proof of theorem 16. The isomorphism theorem thus establishes a surprisingly strong connection between the syntactic concept lattice and the universal automaton of a language.

We now consider the consequences of the isomorphism theorem for our investigations into the semantics of substructural logics. Automata-theoretic semantics is richer than simple language-theoretic semantics, because there is a many-one relationship of recognition between automata and languages. In order to ensure soundness, we already have to restrict interpretations to nuclear automata; then it follows from more general results. To obtain completeness, the isomorphism theorem can be applied in a straightforward fashion: just compose the SCL-interpretation of $\mathbf{FL}_\perp$ (or its fragments) with the isomorphism from $SCL(L)$ into $ACL(U(L))$, and we are done.

Theorem 21 *(Completeness of automata-theoretic semantics)*

1. $ACL \models \Gamma \vdash \alpha$ *iff* $\Vdash_{\boldsymbol{FL}_\perp} \Gamma \vdash \alpha$
2. $ACL_{\boldsymbol{FL}} \models \Gamma \vdash \alpha$ *iff* $\Vdash_{\boldsymbol{FL}} \Gamma \vdash \alpha$
3. $ACL_{\boldsymbol{L1}} \models \Gamma \vdash \alpha$ *iff* $\Vdash_{\boldsymbol{L1}} \Gamma \vdash \alpha$

It is obvious how to further strengthen these results: let $ACL(FIN)$ denote the class of automatic concept lattices over finite nuclear automata (i.e. nuclear automata with finite state set).[4] We can depart from completeness for $SCL(REG)$: for $\not\Vdash_{\mathbf{FL}_\perp} \Gamma \vdash \alpha$ we find a countermodel $SCL(L)$ where $L \in REG$. By the isomorphism theorem, we also have a countermodel $ACL(U(L))$ which is finite. Thus we have the following:

Theorem 22 *(Completeness for finite automata)*

1. $ACL(FIN) \models \Gamma \vdash \alpha$ *iff* $\Vdash_{\boldsymbol{FL}_\perp} \Gamma \vdash \alpha$
2. $ACL_{\boldsymbol{FL}}(FIN) \models \Gamma \vdash \alpha$ *iff* $\Vdash_{\boldsymbol{FL}} \Gamma \vdash \alpha$
3. $ACL_{\boldsymbol{L1}}(FIN) \models \Gamma \vdash \alpha$ *iff* $\Vdash_{\boldsymbol{L1}} \Gamma \vdash \alpha$

5 Conclusion

We have presented a new complete semantics for the full Lambek calculus and its various fragments, the so-called automata-theoretic semantics. It is based on an automata-theoretic construction we introduced, the automatic concept lattice. What is peculiar to this semantics is that it is both language-theoretic and relational, and thus brings together two prominent types of semantics for substructural logics. Our results are based on the construction of Galois connections, closure operators and nuclei: these allow us to give rather simple proofs for completeness. This illustrates (once more) the usefulness of Galois connections in the context of substructural logics and formal language theory. Another important result concerns finiteness of models, which corresponds to regularity of languages. We showed that our completeness results can be extended to this case.

As an outlook, we hope that we can use the results established in this paper to strengthen some of the canonical completeness results regarding L-models and relational models to regular languages and/or finite relations.

[4]This class is strictly smaller than the class of automata recognizing regular languages, as obviously there are infinite automata recognizing regular languages.

Acknowledgements

I would like to thank the anonymous reviewers – one in particular – for their extremely helpful comments.