The Syntactic Concept Lattice: Another algebraic theory of context-free languages?

Alexander Clark
alexc@cs.rhul.ac.uk

October 11, 2012

Abstract

The syntactic concept lattice is a residuated lattice associated naturally with a given formal language; it arises naturally as a generalisation of the syntactic monoid in the analysis of the distributional structure of the language. In this paper we define the syntactic concept lattice and present its basic properties, and its relationship to the universal automaton and the syntactic congruence; we consider several different equivalent definitions, as Galois connections, as maximal factorisations and finally using universal algebra to define it as an object that has a certain universal (terminal) property in the category of complete idempotent semirings that recognize a given language, applying techniques from automata theory to the theory of context-free grammars.

This means that for any context free language, we can define a universal morphism that maps the nonterminals for any context free grammar for that language into a universal set of categories.

We can view this as changing the underlying objects manipulated by the grammar from being a collection of strings, to a collection of sets of strings; the syntactic congruence arising as a congruence of semigroups/monoids, while the syntactic concept lattice arises out of a closely related congruence of semirings.

1 Introduction

The theory of finite automata as representations of regular languages is now very well developed; there are three types of canonical representations. First the canonical minimal deterministic finite automation DFA, based on the Myhill-Nerode congruence; secondly, the universal automaton (UA), [11, 22] and finally, the canonical residual automaton [13] occupies a position between these two, being nondeterministic but quite close to the DFA. All of these are based in some sense on the relation between prefixes and suffixes: two strings p and s are related if their concatenation is in the language. Formally, we might write $p \sim_L s$ iff $ps \in L$. The Myhill-Nerode congruence says that two prefixes are identical if they are related to exactly same set of suffixes; this gives a partition of the set of prefixes. On the other hand, the UA is based on the Galois connection between prefixes and suffixes, though the standard presentations do not present it in this way. This gives, rather than a partition, an overlapping hierarchy, a lattice, whose automata theoretic interpretation is therefore naturally nondeterministic.

When we move to the theory of context free languages and grammars, we need to consider rather the relation between contexts and substrings, where a context is an ordered pair of strings or, equivalently, a string with a gap in it. We might write this as $(l, r) \sim_L u$ iff $lur \in L$. In this case we have two relevant structures: the syntactic congruence, which corresponds exactly to the Myhill-Nerode congruence, and the recently introduced syntactic concept lattice[5, 7, 8, 9] that corresponds to the universal automaton. This is very closely related to the syntactic semiring [23, 24]; indeed for regular languages they are isomorphic.[1] The syntactic congruence has been classically studied (e.g. [25]) and has recently given rise to a number of

[1]The syntactic semiring is a quotient of the set of finite sets of strings.

	$p \sim s$	$(l, r) \sim u$
Equivalence relation	Myhill-Nerode relation	Syntactic congruence
Lattice	Universal automaton	Syntactic concept lattice

Table 1: Relationship between congruences and lattices when considering regular grammars (on the left) and context free grammars on the right.

algorithms for the grammatical inference of context-free grammars [10, 6] that are analogous to the results for regular inference using deterministic finite automata (DFAs) obtained by Angluin [1, 2]. We note also the result of Bollig et al for learning residual automata [3]. However, whereas with regular languages, the restriction to deterministic automata does not reduce the class of languages that can be expressed, [2] with context-free grammars, the situation is quite different. The class of languages that can be expressed using nonterminals that correspond to congruence classes under the syntactic congruence is a subclass of context free languages, which while including all the regular languages, does not include such simple languages as the palindrome language over a two letter alphabet.

In this paper we define the syntactic concept lattice, and give its basic properties, focusing on its relationships to the syntactic congruence and the universal automaton. We also demonstrate that it has a desirable property analogous to that of the universal automaton: namely that it contains a morphic image of every context free grammar for the language. Alternatively, we show that it is the unique (up to isomorphism) smallest structure that interprets a CFG for the language; in other words it is the smallest and simplest object that provides a semantics for the grammar. We can view it as the syntactic monoid lifted from monoids (semigroups with unit) to semirings; as lifting from strings to sets of strings. Figure 1 shows the relationship between these structures.

In Section 2 we will define our notation, and define the basic algebraic structures we use: monoids, and complete idempotent semirings, as well as the basic notions of context-free grammars. Then in Section 3 we define the syntactic concept lattice, and explain its basic properties. We then, in Section 4 start to develop an algebraic theory for the semantics of context free grammars, algebraic structures that we use to interpret them. We then define the basic ideas of syntactic morphisms – i.e. morphisms of CFGs in Section 5, which leads us onto the universal morphism using the Syntactic Concept Lattice (Section 6.

We then look briefly at the theory of regular languages in Section 7, discussing the Universal Automaton, its relationship(s) to the Syntactic Concept Lattice, and the corresponding CFG that we call the Universal Context Free Grammar (which bears no relation to the term UG as used in generative linguistics). We then conclude.

2 Preliminaries

We use standard set theoretic notation; the empty set is $\emptyset$, and $\subseteq, \supseteq, \cup$ and $\cap$ have their normal meanings. We write $\mathcal{P}(X)$ for the power set of X, and $\mathcal{F}(X)$ for the collection of only finite sets of X.

We have a finite nonempty set Σ which we call an alphabet. Let Σ^* be the set of all finite length strings over Σ. We write λ for the empty string. We will use letters at the beginning of the alphabet, a, b, c to refer to elements of Σ and abusing notation slightly also to strings of length 1. We write $|u|$ for the length of a string $u \in \Sigma^*$; so $|\lambda| = 0$. We will write $|u|_a$ for the number of times the element $a \in \Sigma$ occurs in the string u. We will write concatenation of strings u, v as uv and sometimes $u \cdot v$ when we want to be explicit.

A language is any subset of Σ^*. For two such sets A, B we define the product $AB = \{uv \mid u \in A, v \in B\}$. A context is an ordered pair of strings $(l, r) \in \Sigma^* \times \Sigma^*$. We can combine a context (l, r) and a string u using the wrapping operation which we write $(l, r) \odot u = lur$. (λ, λ) is the empty context; clearly $(\lambda, \lambda) \odot u = u$ for any u. We will lift these operations to sets in the standard way. If C is a set of contexts, and S is a set of strings, then $C \odot S = \{lur \mid (l, r) \in C, u \in S\}$. If X, Y are sets of strings then $X \times Y$ is the set of contexts

[2] It is worth recalling here that the size of the minimal DFA for a given language may be exponentially larger than the size of the minimal nondeterministic automaton for the same language, so the two representations are not entirely equivalent.

$\{(x, y) \mid x \in X, y \in Y\}$. We also extend the wrap operation $\odot$ to contexts as $(x, y) \odot (p, q) = (xp, qy)$, so that $(x, y) \odot (p, q) \odot u$ is associative, and to sets of contexts in the natural way.

2.1 Monoids

We use a few standard algebraic structures; in order to make this self contained we define all the basic properties here.

A monoid is a set M together with a unit and an associative operation $\circ$. A monoid morphism from M to N is a function h such that $h(a \circ b) = h(a) \circ h(b)$ and $h(1_M) = 1_N$. The elements of the free monoid over Σ (which we write Σ^*) are finite length strings over Σ; the operation is simply concatenation.

2.2 Semirings

Rather than just a monoid (a semigroup with unit) we will consider a slightly more complicated structure which in addition to the concatenation operation $\circ$ has another operation which we will write as $\vee$.

Definition 1. *An idempotent semiring is a structure which we write as $\langle M, \vee, \circ, \bot, 1\rangle$ where*

- *$\langle M, \vee, \bot\rangle$ is a commutative idempotent monoid with $\bot$ as unit; i.e. $x \vee \bot = x = \bot \vee x$. Idempotent means that $x \vee x = x$.*
- *$\langle M, \circ, 1\rangle$ is a monoid*
- *$\bot \circ x = x \circ \bot = \bot$.*
- *$\circ$ distributes over $\vee$, so $a \circ (b \vee c) = (a \circ b) \vee (a \circ c)$ and $(b \vee c) \circ a = (b \circ a) \vee (c \circ a)$.*

The free idempotent semiring over Σ is the collection of finite sets of strings of Σ, $\mathcal{F}(\Sigma^*)$, where $1 = \{\lambda\}$, $\bot = \emptyset$, $\vee = \cup$ and $X \circ Y = XY$ is concatenation extended to sets of strings.

We need to add some infinitary operations here – in particular we want to be able to take the $\vee$ (the union, intuitively) over infinite sets. This structure is then a complete idempotent semiring [14, 19, 15].

Definition 2. *A complete idempotent semiring (CIS) is an idempotent semiring $\langle M, \vee, \circ, \bot, 1\rangle$ where additionally*

- *$\vee$ is defined over countable sets. Given a countable family of sets $\{x_1, x_2, \dots\}$, there is an element $\bigvee_{i\in\mathbb{N}} x_i$.*
- *$\circ$ distributes over countable $\vee$: if X is a countable collection of elements then $a \circ (\bigvee_{i\in\mathbb{N}} x_i) = \bigvee_{i\in\mathbb{N}} a \circ x_i$ and $(\bigvee_{i\in\mathbb{N}} x_i) \circ a = \bigvee_{i\in\mathbb{N}} (x_i \circ a)$*

The free structure generated by this over Σ is $\mathcal{P}(\Sigma^*)$.

The elements of this are finite or infinite sets of finite length strings over Σ. If we had just an idempotent semiring, the free structure would be the collection of finite sets of strings – since the languages we use might be infinite, we need to allow infinite sets and therefore a complete semiring.

These CIS have a natural partial order.

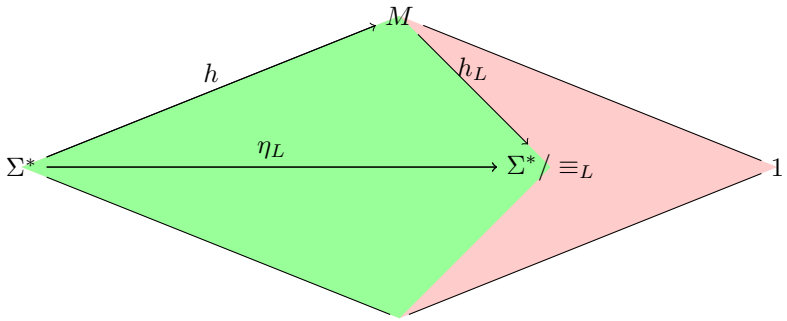
Proposition 1. *Given a CIS, we can define a relation $\leq$ by $X \leq Y$ iff $Y = X \vee Y$. This relation is a partial order.*

Proof. $X = X \vee X$ so $X \leq X$. If $X \leq Y$ and $Y \leq X$ then $X = X \vee Y = Y \vee X = Y$, so $\leq$ is antisymmetric. If $X \leq Y$ and $Y \leq Z$, then $Y = X \vee Y$ and $Z = Y \vee Z$ so $Z = (X \vee Y) \vee Z = X \vee (Y \vee Z) = X \vee Z$ so $X \leq Z$ so $\leq$ is transitive. □

Complete idempotent semirings are the appropriate structures for interpreting context-free grammars as systems of equations [18, 21].[3]

[3] I am grateful to Hans Leiss for making this point.

Figure 1: The syntactic congruence: the coarsest congruence that recognises L. Fine congruences are on the left and on the far right is the coarsest congruence of all with only one congruence class. Congruences that recognise L are shaded green. Congruences that do not are shaded red. If M is a monoid that recognises L through the morphism h then there is a homomorphism from M to $\Sigma^*/\equiv_L$, h_L such that $\eta_L = h \cdot h_L$.



2.3 Congruences

A congruence on an algebra is an equivalence relation on the elements that respects the operations of the algebra. A congruence of a monoid then is an equivalence relation $\cong$ such that if $u \cong u$ and $v \cong v'$ then $u \circ v = u' \circ v$. We write $[u]_\cong$ for the equivalence class of u.

The equivalence classes under a congruence form an algebra where $[1]_\cong$ is a unit, and the concatenation operation is $[u]]_\cong \circ [v]]_\cong = [uv]]_\cong$. This is well defined since it is a congruence.

A homomorphism is a function which respects the operations. The congruences and the surjective homomorphisms are closely related. Given a surjective homomorphism h from $A \to B$, the equivalence relation on A given by $u \cong v$ iff $h(u) = h(v)$ is a congruence. Conversely, given a congruence $\cong$, the natural map $h : u \to [u]_\cong$ is a surjective homomorphism.

Given a free algebra, like the free monoid Σ^*, the congruences form a lattice; and therefore also do the corresponding quotient monoids. The bottom element is Σ^*, the finest congruence, and the top is the coarsest congruence where everything is equivalent to everything else and there is only one element in the quotient. For any surjective homomorphism from Σ^* onto a monoid A, we can identify a congruence which is an element of this lattice. Similarly if we have two congruences, one of which is finer than the other, then there is a surjective homomorphism from the finer to the coarser.

The same applies, as we shall see, for congruences and homomorphisms of complete idempotent semirings.

2.4 The syntactic congruence

The syntactic concept lattice, which is the main topic of this paper, is best understood in relation to the syntactic congruence; therefore, we recall some of its elementary properties in a notation consistent with the rest of this paper.

Definition 3. *For a language L and $w \in \Sigma^*$; define the distribution of w to be*

$$C_L(w) = \{(l, r) \in \Sigma^* \times \Sigma^* \mid lwr \in L\}$$

.

Definition 4. *We say that two strings $u, v \in \Sigma^*$ are congruent w.r.t. a language L, written $u \equiv_L v$ iff $C_L(u) = C_L(v)$.*

Proposition 2. *$\equiv_L$ is an equivalence relation and a congruence of the monoid Σ^*; i.e. $u \equiv_L v$ implies that for any $z \in \Sigma^*$, $uz \equiv_L vz$ and $zu \equiv_L zv$.*

Definition 5. *We write $[u]_L$ for the equivalence class of u under $\equiv_L$.*

$$[u]_L = \{v \in \Sigma^* \mid u \equiv_L v\}$$

The fact that it is a congruence means that we have the following result, which is of crucial importance for constructing context free grammars.

Proposition 3. *For any language L and any strings, u, v*

$$[uv]_L \supseteq [u]_L[v]_L$$

This means that we can define context free grammars using congruence classes as nonterminals.
The following proposition is classical, attributed to Myhill [25].

Proposition 4. *The number of congruence classes of L is finite iff L is regular.*

Example 1. *Suppose $L = \Sigma^*$. Then there is one congruence class which consists of all strings.*

Example 2. *Consider the regular language $L = (ab)^*$. This has the following 6 congruence classes which are all regular sets. $(ab)^+$, $\{\lambda\}$, $a(ba)^*$, $b(ab)^*$, $(ba)^+$, and finally all other strings (i.e. not in the preceding 5 classes) which we can express using the regular expression: $\Sigma^*bb\Sigma^* \cup \Sigma^*aa\Sigma^*$.*

Example 3. *Consider the language $O_1 = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$. Define $c(w) = |w|_a - |w|_b$. Clearly $[u]_L = \{v | c(u) = c(v)\}$, and so this has an infinite number of congruence classes indexed by the integers where for $n \in \mathbb{Z}$, $C_n = \{w \mid c(w) = n\}$.*

Definition 6. *A congruence $\sim$ saturates a language L if $x \sim y$ implies that $x \in L$ iff $y \in L$.*

Informally, this just means that all of the classes are either inside or outside the language.

Proposition 5. *The syntactic congruence saturates L.*

Proof. If $u \in L$ and $u \equiv_L v$ then $v \in L$. □

Proposition 6. *The syntactic congruence is the coarsest congruence that saturates L.*

Proof. Suppose $\sim$ is a congruence that saturates L. and suppose $x \sim y$. Take some l, r. Then $lxr \sim lyr$ as it is a congruence. Since $\sim$ saturates L then $lxr \in L$ iff $lyr \in L$. Therefore $x \equiv_L y$ and so $\sim$ is finer than $\equiv_L$. □

Definition 7. *Given a monoid M and a morphism h from Σ^* to M, we say that the monoid recognizes the language L through h if there is a set $X \subseteq M$ such that $h^{-1}(X) = L$, i.e. such that*

$$\{w \in \Sigma^* \mid h(w) \in X\} = L$$

.

Note that if M recognises L through the morphism h, then $h(M)$ also recognizes L through the surjective morphism h. $h(M)$ is a submonoid of M is h is a homomorphism. In this case we are not interested in all of M but only in the submonoid $h(M)$. The rest of M consists of elements that are not related to the language at all. We will therefore restrict our consideration to the case where h is surjective.

Example 4. *The monoid Σ^* recognises L through the identity morphism id. We merely take L as the relevant subset; $id^{-1}(L) = L$.*

Definition 8. *The syntactic monoid, $\Sigma^*/\equiv_L$ is a monoid whose elements are the congruence classes of Σ^* under $\equiv_L$, concatenation being defined as $[uv] = [u] \circ [v]$, with left and right identity $[\lambda]$.*

Note that the operation in the quotient is well-defined, as usual, by the fact that it is a congruence.

Example 5. *Consider again the language O_1. This has a syntactic monoid which is isomorphic to the integers under addition.*

The syntactic monoid has a certain universal property [26].

Definition 9. *The natural map from $\Sigma^* \to \Sigma^*/\equiv_L$, written η_L is $u \to [u]_L$. This is a surjective monoid homomorphism.*

Proposition 7. *The syntactic monoid of L recognises L through the morphism η_L.*

Proof. This is perhaps too obvious to merit a detailed proof; but if $u \in L$ and $u \equiv_L v$ then v is in L; so clearly $L = \bigcup\{[u]_L \mid u \in L\}$. □

The syntactic monoid of L is the smallest (coarsest) monoid that recognizes L, and it is unique up to isomorphism.

We use the $\cdot$ symbol to represent function composition here. If $f : A \to B$ and $g : B \to C$ then $f \cdot g : A \to C$.

Proposition 8. *Suppose L is recognized by the monoid M via the surjective morphism h then there is a morphism h_L from M to the syntactic monoid such that $\eta_L = h \cdot h_L$.*

See the diagram in Figure 1 for an illustration of this.

Proof. Define the congruence $\equiv_h$ to be $u \equiv_h v$ iff $h(u) = h(v)$. This is a congruence that saturates L. Define h_L to be $h_L(h(x)) = \eta_L(x)$. Since $\equiv_h$ saturates L, this is welldefined. Suppose x and y are such that $h(x) = h(y)$; then $x \equiv_h y$ which implies $x \equiv_L y$ which implies $\eta_L(x) = \eta_L(y)$. □

Note that all of this follows quite directly from basic principles of universal algebra: This is an immediate consequence of Noether's third isomorphism theorem; namely that if we have two congruences of an algebra such that one is finer than the other then there is a homomorphism from the quotient of the finer to the quotient of the coarser. We do not rely on this body of mathematics in this paper but prove things directly each time. This leads to a little repetition and a lack of maximal generality, but makes the presentation more self-contained.

2.5 Context free grammars

A context free grammar over an alphabet Σ is a tuple $\langle \Sigma, V, S, P \rangle$, where V is a finite set of nonterminals, $S \in V$ is the unique start symbol, P is a finite set of productions written $N \to \alpha$ where $N \in V$ and $\alpha \in (V \cup \Sigma)^*$. We define the derivations in the standard way. We define a derivation relation by $\beta N \gamma \Rightarrow \beta\alpha\gamma$, when $N \to \alpha \in P$ and $\beta, \gamma \in (V \cup \Sigma)^*$. We use the notation $\alpha \stackrel{*}{\Rightarrow} \beta$ for a derivation using zero or more steps : the reflexive transitive closure of $\Rightarrow$.

Definition 10. *For a* CFG*, and a nonterminal N we define $\mathcal{L}(G, N) = \{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$ and $\mathcal{C}(G, N) = \{(l, r) \in \Sigma^* \times \Sigma^* \mid S \stackrel{*}{\Rightarrow} lNr\}$. We also extend $\mathcal{L}(G, \alpha)$ for $\alpha \in (V \cup \Sigma)^*$ in the usual way, with $\mathcal{L}(G, \lambda) = \{\lambda\}$ and $\mathcal{L}(G, \alpha\beta) = \mathcal{L}(G, \alpha)\mathcal{L}(G, \beta)$.*

Note that by this definition $(\lambda, \lambda) \in \mathcal{C}(G, S)$ since $S \stackrel{*}{\Rightarrow} S = \lambda S \lambda$, even if S does not occur on the right hand side of any production in G.

$L(G)$ is defined to be $\mathcal{L}(G, S)$, for the unique start symbol S. A context-free language is a language L_* such that there is a CFG G such that $L(G) = L_*$.

Proposition 9. *If we have a grammar G that defines a language L then for any nonterminal N in G,*

$$\mathcal{C}(G, N) \odot \mathcal{L}(G, N) \subseteq L$$

We will give the trivial proof now.

Proof. If $(l, r) \in \mathcal{C}(G, N)$, then there is a derivation $S \stackrel{*}{\Rightarrow} lNr$. If $u \in \mathcal{L}(G, N)$, then there is a derivation $N \stackrel{*}{\Rightarrow} u$. There is therefore a derivation $S \stackrel{*}{\Rightarrow} lNr \stackrel{*}{\Rightarrow} lur$, and so $(l, r) \odot u = lur \in L$. □

This proposition implies that a nonterminal in a CFG defines a decomposition or factorisation of L into a set of contexts and a set of substrings.

Proposition 10. *If there is a production $N \to \alpha$, then $\mathcal{L}(G, N) \supseteq \mathcal{L}(G, \alpha)$.*

Proposition 11. *More specifically, if the rule is of the form $N \to PQ$ then $\mathcal{L}(G, N) \supseteq \mathcal{L}(G, P)\mathcal{L}(G, Q)$. If the rule is of the form $N \to w$ then $\mathcal{L}(G, P) \supseteq \{w\}$.*

Therefore we have two relations between the grammar and the language it describes. One is that the nonterminal determines a factorisation into a set of contexts and a set of strings. The other is that a production determines an inclusion relation between the sets of strings that correspond to nonterminals. As we shall see later – and in exact analogy to the UA (see e.g. [22]) – we can reverse this process and define a grammar where the nonterminals are based on these decompositions – more precisely on the maximal decompositions, and the productions are the ones that satisfy the inclusion relations.

Later we will consider generalisations of CFGs where we allow the set of nonterminals and the set of productions to be infinite; in this case any language can be described using one of the objects.

In what follows we will occasionally restrict ourselves to grammars where the productions are either of the form $N \to PQ$ or $N \to a$ for $a \in \Sigma$, or $N \to \lambda$; a restriction somewhat along the lines of Chomsky normal form.

3 The Syntactic Concept Lattice

Given any fixed language L, we can define an object called the syntactic concept lattice [5, 7]. This is a canonical object in the sense that it is uniquely defined for each language. It has a number of important properties that we will discuss later for grammatical inference; at the moment we will focus on giving some clear definitions. They also define a class of sound and complete models for the Lambek calculus [27]. See also [9] for further discussion of the relation to Lambek calculus and Lambek grammars. There are several equivalent ways of defining this lattice; we will describe two.

The first is based on the Galois connection between sets of contexts and sets of strings [12]; this is exactly Formal Concept Analysis [17]. We define a *polar* map from sets of strings to sets of contexts. For a set of strings S, we define a set of contexts $S^{\triangleright}$ as follows.

$$S^{\triangleright} = \{(l, r) \in \Sigma^* \times \Sigma^* \mid \forall u \in S, lur \in L\} \tag{1}$$

This is the shared distribution of the set of strings S.

Dually, we define a polar map from a set of contexts C to the set of all strings that can occur in all of these contexts.

$$C^{\triangleleft} = \{u \in \Sigma^* \mid \forall (l, r) \in C, lur \in L\} \tag{2}$$

We now give some basic properties of these maps.

Proposition 12. *If $S \subseteq T$ are two sets of strings, then $S^{\triangleright} \supseteq T^{\triangleright}$. If S is empty, then $S^{\triangleright} = \Sigma^* \times \Sigma^*$.*

Proposition 13. *If X, Y are sets of strings then $(X \cup Y)^{\triangleright} = X^{\triangleright} \cap Y^{\triangleright}$. Similarly if X_i, for $i \in I$ is a family of sets, finite or infinite, then*

$$\left(\bigcup_{i \in I} X_i\right)^{\triangleright} = \bigcap_{i \in I} X_i^{\triangleright}$$

.

Dually we have the following two propositions.

Proposition 14. *If $C \subseteq D$ are two sets of contexts, then $C^{\triangleleft} \supseteq D^{\triangleleft}$.*

Proposition 15. *If C, D are sets of contexts then $(C \cup D)^{\triangleleft} = C^{\triangleleft} \cap D^{\triangleleft}$. Similarly if C_i, for $i \in I$ is a family of contexts, finite or infinite, then*

$$\left(\bigcup_{i \in I} C_i\right)^{\triangleleft} = \bigcap_{i \in I} C_i^{\triangleleft}$$

.

Proposition 16. *For any set of strings S, $S \subseteq S^{\triangleright\triangleleft}$. For any set of contexts C, $C \subseteq C^{\triangleleft\triangleright}$.*

Proof. Suppose $w \in S$. Then the set of contexts that w can occur in contains $S^{\triangleright}$. Therefore $w \in S^{\triangleright\triangleleft}$. Similarly if $(l, r) \in C$, the set of strings that can occur in (l, r) is a superset of $C^{\triangleleft}$ and so $(l, r) \in C^{\triangleleft\triangleright}$. □

Proposition 17. *For any set of strings S, $S^{\triangleright} = S^{\triangleright\triangleleft\triangleright}$.*
For any set of contexts C, $C^{\triangleleft} = C^{\triangleleft\triangleright\triangleleft}$.

Proof. $(S^{\triangleright})^{\triangleright\triangleleft} \supseteq S^{\triangleright}$ by the previous lemma(putting $C = S^{\triangleright}$). But $S^{\triangleright\triangleleft} \supseteq S$, so $(S^{\triangleright\triangleleft})^{\triangleright} \subseteq S^{\triangleright}$ by Proposition 12. This completes the first proof; the second is exactly parallel. □

Definition 11. *We say that a set of strings S is* closed *iff $S = S^{\triangleright\triangleleft}$. Similarly, we say that a set of contexts, C is* closed *iff $C = C^{\triangleleft\triangleright}$.*

Proposition 18. *For any set of strings S, $S^{\triangleright}$ is closed, and so is $S^{\triangleright\triangleleft}$. For any set of contexts C, $C^{\triangleleft}$ is closed, and so is $C^{\triangleleft\triangleright}$.*

Proposition 19. *For any language L, L is a closed set of strings. $(\lambda, \lambda) \in L^{\triangleright}$. Therefore if $w \in L^{\triangleright\triangleleft}$, w must occur in the context (λ, λ) which implies that $w \in L$.*

Definition 12. *A concept is an ordered pair $\langle S, C\rangle$ of a set of strings S and a set of contexts C such that $S^{\triangleright} = C$ and $C^{\triangleleft} = S$.*

Though it seems like these concepts might be hard to find, it is easy to prove the following proposition.

Proposition 20.

- *For any set of strings S, $\langle S^{\triangleright\triangleleft}, S^{\triangleright}\rangle$ is a concept.*
- *For any set of contexts C, $\langle C^{\triangleleft}, C^{\triangleleft\triangleright}\rangle$ is a concept.*

Indeed it is easy to see that if $\langle S, C\rangle$ is a concept, then S is a closed set of strings and C is a closed set of concepts. There is a bijection between the set of closed sets of strings and the set of closed sets of concepts; we can either consider the concepts as sets of strings, or sets of contexts, or both.

An alternative way of defining the concepts is as follows.

Definition 13. *A concept is an ordered pair $\langle S, C\rangle$ of a set of strings S and a set of contexts C such that $C \odot S \subseteq L$ and S and C are both maximal. That is to say, if $\langle T, D\rangle$ is such that $D \odot T \subseteq L$ and $T \supseteq S, D \supseteq C$ then $T = S$ and $C = D$.*

It is easy to verify that these two definitions are equivalent; this second definition makes the links with the universal automaton more explicit.

Example 6. *Suppose $L = \Sigma^*$. Then there is exactly one concept: $\langle \Sigma^*, \Sigma^* \times \Sigma^*\rangle$.*

Example 7. *Suppose $L = \emptyset$. Then there are exactly two concepts: $\langle \Sigma^*, \emptyset\rangle$ and $\langle \emptyset, \Sigma^* \times \Sigma^*\rangle$.*

Proposition 21. *The collection of concepts form a complete bounded lattice denoted $\mathfrak{B}(L)$.*

- $\top = \langle \Sigma^*, (\Sigma^*)^{\triangleright}\rangle$
- $\bot = \langle \emptyset^{\triangleright\triangleleft}, \emptyset^{\triangleright}\rangle$

The partial order is defined by set inclusion between the sets of strings.

- $\langle S, C\rangle \leq \langle T, D\rangle$ *iff* $S \subseteq T$.
- $\langle S, C\rangle \vee \langle T, D\rangle = \langle (S \cup T)^{\triangleright\triangleleft}, C \cap D\rangle$
- $\langle S, C\rangle \wedge \langle T, D\rangle = \langle S \cap T, (C \cap D)^{\triangleleft\triangleright}\rangle$

We use the notation $\mathfrak{B}(L)$ to emphasize the link to concept analysis where the fraktur B stands for *Begriff*.

We now examine the relationship between the syntactic congruence and the syntactic concept lattice.

Proposition 22. $\{u\}^{\triangleright\triangleleft} = \{v\}^{\triangleright\triangleleft}$ *iff* $u \equiv_L v$.

Proposition 23. $[u] \subseteq \{u\}^{\triangleright\triangleleft}$

Proposition 24. *If $w \in S$ for some closed set of words S, then $[w] \subseteq S$. In other words closed sets of strings are unions of congruence classes: $S = \bigcup_{w\in S}[w]$.*

Proposition 25. *The lattice $\mathfrak{B}(L)$ has finitely many elements iff L is regular.*

Proof. Note that a language L has finitely many congruence classes iff it is regular. By Proposition 22 this means that if L is not regular then $\mathfrak{B}(L)$ is infinite. By Proposition 24, if L is regular, then $\mathfrak{B}(L)$ is finite. □

Whereas the syntactic monoid is always countable; and infinite iff the language is regular, the lattice may have uncountably many elements, even for a context-free language.

Proposition 26. *There is a context-free language L such that $\mathfrak{B}(L)$ has uncountably many elements. (Yoshinaka, p.c.)*

Proof. Let $O_1^c = \{w \in \{a, b\}^* \mid |w|_a \neq |w|_b\}$. Here $|w|_a$ means the number of occurrences in the string w of the symbol a. Define $c(w) = |w|_a - |w|_b$, and $c(l, r) = c(lr)$. $u \equiv_L v$ iff $c(u) = c(v)$. $\mathfrak{B}(L)$ is isomorphic to $2^{\mathbb{Z}}$. where if X is a set of integers, it corresponds to the concept: $\langle \{w \mid c(w) \in X\}, \{(l, r) \mid c(l, r) \notin X\}\rangle$. L corresponds to the set of non-zero integers. □

Note that in this example, the syntactic monoid of O_1^c is isomorphic to $\mathbb{Z}$ under addition.

Proposition 27. *The lattices may have infinite ascending and descending chains.*

Proof. $\mathfrak{B}(O_1^c)$ has an infinite ascending chain: $\{1\}, \{1, 2\}, \ldots$ and and infinite descending chain $\mathbb{Z}, \mathbb{Z}\setminus\{1\}, \ldots$ □

3.1 Concatenation

We now define the natural concatenation operation and its associated residuals.

Definition 14. *For two closed sets of strings S, T define $S \circ T = ST^{\triangleright\triangleleft}$.*

Definition 15. *Given a language L, we define the unit in the lattice to be $\lambda_L = \langle \{\lambda\}^{\triangleright\triangleleft}, \{\lambda\}^{\triangleright}\rangle$.*

Proposition 28. $\mathfrak{B}(L), \circ, \lambda_L$ *is a monoid.*

The closure operation interacts nicely with the concatenation operation. Just as distributional equivalence ($\equiv_L$) is a congruence of the monoid, implying that $[uv] \supseteq [u][v]$, so we have a related result for the lattice operations, lifted to sets of strings. Indeed as we shall see later, this defines a congruence on sets of strings – i.e. on $\mathcal{P}(\Sigma^*)$.

Proposition 29. *Suppose we have two sets of strings Y, Z. Then*

$$(YZ)^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft}$$

.

Proof. [16, 21] Suppose $y \in Y^{\triangleright\triangleleft}$ and $z \in Z^{\triangleright\triangleleft}$. So $\{y\}^{\triangleright} \supseteq Y^{\triangleright}$ and $\{z\}^{\triangleright} \supseteq Z^{\triangleright}$. Take $(l, r) \in (YZ)^{\triangleright}$. So $lYZr \subseteq L$. This means that $(l, Zr) \subseteq Y^{\triangleright}$. So $(l, Zr) \subseteq Y^{\triangleright} \subseteq \{y\}^{\triangleright}$. So $lyZr \subseteq L$. This means that $(ly, r) \subseteq Z^{\triangleright} \subseteq \{z\}^{\triangleright}$. so $lyzr \in L$. So $(l, r) \in (yz)^{\triangleright}$ and so $(l, r) \in (Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft})^{\triangleright}$. Therefore $(YZ)^{\triangleright} \subseteq (Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft})^{\triangleright}$. Therefore $(YZ)^{\triangleright\triangleleft} \supseteq (Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft})^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft}$. □

If $Y = \{u\}$ and $Z = \{v\}$ then this means that $\{uv\}^{\triangleright\triangleleft} \supseteq \{u\}^{\triangleright\triangleleft}\{v\}^{\triangleright\triangleleft}$; contrast this with the result in Proposition 3.

Proposition 30. *Suppose we have sets of strings X, Y, Z such that $X \supseteq YZ$. Then $X^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft}$.*

Proposition 31. *For any sets of strings X, Y, $(XY)^{\triangleright\triangleleft} = (X^{\triangleright\triangleleft}Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$.*

Proof. By Proposition , $XY^{\triangleright\triangleleft} \supseteq X^{\triangleright\triangleleft}Y^{\triangleright\triangleleft}$. So $XY^{\triangleright\triangleleft} = (XY^{\triangleright\triangleleft})^{\triangleright\triangleleft} \supseteq (X^{\triangleright\triangleleft}Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$. But $XY \subseteq X^{\triangleright\triangleleft}Y^{\triangleright\triangleleft}$, so $XY^{\triangleright\triangleleft} \subseteq (X^{\triangleright\triangleleft}Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$. □

We can also define the left and right residuals that are associated to this concatenation.

Definition 16. *Given two elements of $\mathfrak{B}(L)$, $X = \langle S_x, C_x \rangle$, and $Y = \langle S_Y, C_Y \rangle$ we define the left and right residuals X/Y and $X\backslash Y$. FIXME.*

The end result is a very

Proposition 32. *$\mathfrak{B}(L)$ forms a bounded residuated lattice and is therefore also a complete idempotent semiring.*

4 Universal property

Some rhetoric first – we want a system of syntactic categories to interpret our grammars [18]. We have at a minimum two operations – concatenation and union – if we restrict ourselves to CFGs. We want the simplest such system. We will show that for any language there is a unique such simplest system of categories.

We start from a finite set of primitive elements Σ. We can concatenate them finitely and take countable unions. Algebraically we also want the concatenation to distribute over union, which gives us a complete idempotent semiring (Hans Leiss) rather than a semigroup/monoid. So the simplest algebraic structure is something that has a monoid structure with an associative concatenation operation $\circ$ with a unit 1 and a union operation $\vee$ which must be at least associative, commutative, idempotent, and interact nicely with $\circ$ in some way; this is the complete idempotent semiring.

Definition 17. *Given a CIS, S, we say that an equivalence relation $\equiv$ on the elements of S is a CIS-congruence if for all $X, Y, Z \in S$:*

- *$X \equiv Y$ implies $X \circ Z \equiv Y \circ Z$ and $Z \circ X \equiv Z \circ Y$*
- *$X \equiv Y$ implies $X \vee Z \equiv Y \vee Z$*
- *For any countable collection of sets X_i, Y_i such that $X_i \equiv Y_i$, $\bigvee_I X_i \equiv \bigvee_i Y_i$.*

Definition 18. *Given two CIS, A and B a function $h : A \to B$ is a CIS-morphism iff*

- *$h(x \vee y) = h(x) \vee h(y)$*
- *For any countable collection x_i, $h(\bigvee_i x_i) = \bigvee_i h(x_i)$*

- $h(x \circ y) = h(x) \circ h(y)$.

Proposition 33. *If h is a surjective CIS-morphism from A to B, then $h(1_A) = 1_B$ and $h(\bot_A) = \bot_B$.*

Proof. $h(a) = h(1_A \circ a) = h(1_A) \circ h(a)$ and $h(a) = h(a \circ 1_A) = h(a) \circ h(1_A)$. So $1_B = h(1_A) \circ 1_B = h(1_A)$. $h(a) = h(a \vee \bot_A) = h(a) \vee h(\bot_A)$. So $h(\bot_A) = h(\bot_A) \vee \bot_B = \bot_B$. □

Proposition 34. *Suppose $x \leq y$; then $h(x) \leq h(y)$. Note that $x \leq y$ iff $x = x \vee y$. If this is true then $h(x) = h(x \vee y) = h(x) \vee h(y)$ implies $h(x) \leq h(y)$.*

We can define a notion of inverse of a map; in this case it is a residual rather than an exact inverse.

Definition 19. *If we have a CIS-morphism h, from CIS A to B, we define h^* to be the map from B to A $h^*(y) = \bigvee\{x \in A : h(x) \leq y\}$.*

Now h and h^* form a residuated pair.

Proposition 35. *For all $x \in A, y \in B$, $h(x) \leq y$ iff $x \leq h^*(y)$*

Proof. Suppose $h(x) \leq y$. $h^*(y) = \bigvee_{z \in A : h(z) \leq y} z$. But since $h(x) \leq y$ this means that one of these z will be x, so $h^*(y) \geq x$. Conversely suppose $x \leq h^*(y)$; Then $h(x) \leq h(h^*(y) = h(\bigvee_{z \in A : h(z) \leq y} z) = \bigvee_{z \in A : h(z) \leq y} h(z) \leq y$. Therefore $h(x) \leq y$. □

Proposition 36. *$h(h^*(y)) \leq y$ and $h^*(h(x)) \geq x$*

Proof. $h^*(y) \leq h^*(y)$ so $h(h^*(y)) \leq y$. $h(x) \leq h(x)$ so $x \leq h^*(h(y))$. □

Proposition 37. *Both h and h^* are monotonic.*

Note that if A is $\mathcal{P}(\Sigma^*)$ then the homomorphism is uniquely defined just by the values of the elements of Σ; once we know what $h(\{a\})$ is for each element $a \in \Sigma$ then this fixes the value for every string and thus every set of strings. This is rather like the way a linear map (a homomorphism of vector spaces) is fixed by the value of the function at the elements of a basis.

We are concerned exclusively with homomorphisms that arise from congruences of $\mathcal{P}(\Sigma^*)$and are therefore all surjective. In this case we can slightly strengthen the results.

Proposition 38. *If h is a surjective CIS-homomorphism then $h(h^*(y)) = y$.*

Proposition 39. *If $h : A \to B$ is a surjective CIS-homomorphism then for any $X, Y \in B$, $h^*(X \circ Y) = h^*(X) \circ h^*(Y)$.*

Proof. $X \circ Y = h(h^*(X \circ Y))$; and $h(h^*(X) \circ h^*(Y)) = h(h^*(X)) \circ h(h^*(Y) = X \circ Y$. □

Definition 20. *We say that a CIS B recognizes L if there is a surjective morphism h from $\mathcal{P}(\Sigma^*) \to B$ such that $h^*(h(L)) = L$.*

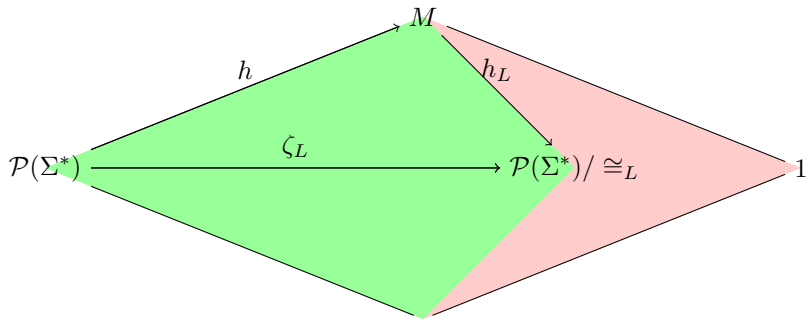
Example 8. *$\mathcal{P}(\Sigma^*)$ recognizes L through the identity morphism.*

Recall that this morphism is fixed just by the elements of $h(a)$ for $a \in \Sigma$. So if we have a morphism from $\mathcal{P}(\Sigma^*) \to B$ then $h(L)$ is the unique element that might represent L – $h^*(h(L)) \geq L$ by the previous proposition. The condition thus just states that this element does in fact correspond to L and not some other larger set of strings like Σ^*.

Having established these general properties of CIS-morphisms, and congruences we can now consider the specific examples we are interested in.

Definition 21. *Given a language L we define an equivalence relation on sets of strings i.e. on $\mathcal{P}(\Sigma^*)$, where the set of strings X is congruent to the set of strings Y, written $X \cong_L Y$ iff $X^\triangleright = Y^\triangleright$.*

Figure 2: The syntactic concept lattice: the coarsest congruence of $\mathcal{P}(\Sigma^*)$that recognises L. Fine congruences are on the left and on the far right is the coarsest congruence of all with only one congruence class. Congruences that recognise L are shaded green. Congruences that do not are shaded red. If M is a CIS that recognises L through the morphism h then there is a homomorphism from M to $\mathcal{P}(\Sigma^*)/\cong_L$, h_L such that $\zeta_L = h \cdot h_L$.



This congruence is on sets of strings – the syntactic congruence is a congruence of strings. They coincide on singleton sets of strings.

Proposition 40. *The relation $\cong_L$ is a CIS-congruence.*

Proof. For all sets of strings X, Y, W, Z

- If $X^\triangleright = Y^\triangleright$ and $W^\triangleright = Z^\triangleright$ then Now by Proposition 31, $XW^{\triangleright\triangleleft} = (X^{\triangleright\triangleleft}W^{\triangleright\triangleleft})^{\triangleright\triangleleft} = (Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft})^{\triangleright\triangleleft} = YZ^{\triangleright\triangleleft}$. So $(XW)^\triangleright = (YZ)^\triangleright$.
- If $X^\triangleright = Y^\triangleright$ and $W^\triangleright = Z^\triangleright$ then $(X \cup W)^\triangleright = X^\triangleright \cap W^\triangleright = Y^\triangleright \cap Z^\triangleright = (Y \cup Z)^\triangleright$.
- Suppose X_i, Y_i are a countable family of sets of strings such that $X_i \cong_L Y_i$ for all i. Then $(\bigvee_i X_i)^\triangleright = \bigcap_i X_i^\triangleright = \bigcap_i Y_i^\triangleright = (\bigvee_i Y_i)^\triangleright$.

□

Proposition 41. *For any language L, $\mathfrak{B}(L)$ is isomorphic to $\mathcal{P}(\Sigma^*)/\cong_L$ as a CIS.*

Proof. We define the natural function $\pi : \mathfrak{B}(L) \to \mathcal{P}(\Sigma^*)/\cong_L$, given by $\pi(X) = [X]_{\cong_L}$. We can verify that this is a bijection and that the operations of $\circ$ and $\vee$ are the same. □

So the SCL is the quotient of $\mathcal{P}(\Sigma^*)$ under the congruence $X \cong Y$ iff $X^{\triangleright\triangleleft} = Y^{\triangleright\triangleleft}$ iff $X^\triangleright = Y^\triangleright$. The relation to the syntactic congruence thus becomes even clearer: the syntactic monoid is the quotient of Σ^* under the congruence $u \equiv v$ iff $\{u\}^\triangleright = \{v\}^\triangleright$.

There is a potential confusion here between sets of sets of strings and sets of strings. The relation $\cong_L$ is a relation between sets of strings: $X \cong_L Y$ where X, Y are sets of strings. The equivalence class $[X]_{\cong_L}$ is thus a set of sets of strings; a collection whose elements are sets of strings that are congruent to X. Now it so happens that if Y, Z are two sets of strings in $[X]_{\cong_L}$ then so is $Y \cup Z$ (since it is a congruence wrt $\vee$).

Proposition 42. *Let $[X]_{\cong_L}$ be the set of all sets of strings congruent to X. Then $\bigcup\{Y \mid Y \cong_L X\} = X^{\triangleright\triangleleft} \cong_L X$. Or, $\bigcup\{Y \mid Y \cong_L X\} \in [X]_{\cong_L}$.*

Just to be clear: If $[X]_{\cong_L}$ is a set of sets of strings. The union of all the sets in $[X]_{\cong_L}$ is a set $\bigcup_{Y \in [X]} Y$ which is equal to $X^{\triangleright\triangleleft}$ which happens to be in $[X]$. So $\cong$ is a partition of sets of strings. A given set of strings will be in only one concept – its closure. A given string u may be in a variety of closed sets of strings, some large and some small – the largest will be Σ^* and the smallest will be $\{u\}^{\triangleright\triangleleft}$.

We can now start to state the universal property.

Proposition 43. *The natural map $\zeta_L : \mathcal{P}(\Sigma^*) \to \mathfrak{B}(L)$ given by $\zeta(X) = X^{\triangleright\triangleleft}$ is a CIS-homomorphism.*

Proposition 44. *The residual of this is $\zeta_L^* : \mathfrak{B}(L) \to \mathcal{P}(\Sigma^*)$which is just the identity on sets of strings; the natural embeding of the closes sets of strings in $\mathcal{P}(\Sigma^*)$.*

We will now show that this is the coarsest congruence that recognises L. If it is any coarser (i.e fewer larger classes) then it will not have L in it. Therefore this is the unique smallest structure that we can use to interpret this language.

Proposition 45. *$\mathfrak{B}(L)$ recognizes L, through the morphism $X \to X^{\triangleright\triangleleft}$.*

Proof. Immediate; since $L^{\triangleright\triangleleft} = L$, so $h(L) = L$ so $h^*(h(L)) = L$. □

We now prove a proposition that in a sense does all of the work; informally it says that any lattice that recognizes this must be 'finer' (larger, more elaborate) than the syntactic context lattice.

Proposition 46. *Suppose, A is some CIS that recognizes L through a morphism h. Suppose there are two sets of strings X, Y such that $h(X) = h(Y)$; then $\zeta_L(X) = \zeta_L(Y)$. (i.e. $X^{\triangleright} = Y^{\triangleright}$)*

Proof. Suppose $(l, r) \in X^{\triangleright}$. That means that $lXr \subseteq L$; so $h(lXr) \leq h(L)$ since h is monotonic. So $h(\{l\}) \circ h(X) \circ h(\{r\}) \leq h(L)$ using the fact that h is a homomorphism. So $h(\{l\}) \circ h(Y) \circ h(\{r\}) \leq h(L)$ since $h(X) = h(Y)$. So $h(\{l\}Y\{r\}) \leq h(L)$ since it is a homomorphism. So $h^*(h(\{l\}Y\{r\})) \leq h^*(h(L))$ since h^* is monotonic. Now $h^*(h(L)) = L$ since A recognises L So using the fact that $h^*(h(X)) \geq X$ we have $\{l\}Y\{r\} \leq h^*(h(\{l\}Y\{r\})) \leq h^*(h(L)) = L$ So $(l, r) \in Y^{\triangleright}$. Conversely, if $(l, r) \in Y^{\triangleright}$ then $(l, r) \in X^{\triangleright}$,and so $X^{\triangleright} = Y^{\triangleright}$. □

Proposition 47. *The SCL is the smallest CIS that recognizes L. If we have a lattice B that recognizes L through a morphism h, then there is a morphism h_L from B to $\mathfrak{B}(L)$ such that $\zeta_L = h \circ h_L$. Therefore the smallest CIS that recognises L is unique up to isomorphism.*

Proof. Define $h_L(h(X)) = \zeta_L(X)$. This is well defined by the previous lemma. We can verify that it is a morphism. Again this is a consequnce of the third isomorphism theorem. □

Proposition 48. *If a finite CIS recognizes L then L is regular. L is regular iff it is recognized by a finite CIS.*

5 CFG Morphisms

We can now make clear the relation between these morphisms between semirings, and representations such as CFGs. The semirings are merely a way of talking about CFGs and their morphisms in a more algebraic way; a method to talk about the 'semantics' of CFGS (in this context the syntax/semantics distinction refers to the distinction between the rules of the grammar, and the language it defines, and has nothing to do with the meaning of a sentence or utterance). We now make the link to CFGs; and in particular we will link the semantic notion of a morphism – a homomorphism between complete idempotent semirings – with the syntactic notion – a morphism between grammars.

We start by defining some standard ideas of mappings between CFGs.

Definition 22. *Assume we have two CFGS, $G_1 = \langle \Sigma, V_1, S_1, P_1 \rangle$ and $G_2 = \langle \Sigma, V_2, S_2, P_2 \rangle$, defined over the same terminal alphabet Σ. G_1 is a subgrammar of G_2 iff $V_1 \subseteq V_2$, $P_1 \subseteq P_2$ and $S_1 = S_2$.*

Proposition 49. *If G_1 is a subgrammar of G_2, then $L(G_1) \subseteq L(G_2)$.*

Definition 23. *Assume we have two CFGS, $G_1 = \langle \Sigma, V_1, S_1, P_1 \rangle$ and $G_2 = \langle \Sigma, V_2, S_2, P_2 \rangle$, defined over the same terminal alphabet Σ. Let ϕ be a function from V_1 to V_2; we extend it to a function from $V_1 \cup \Sigma$ to $V_2 \cup \Sigma$ by setting $\phi(a) = a$ for all $a \in \Sigma$, and thence to a function from $(V_1 \cup \Sigma)^*$ by saying that $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$ and $\phi(\lambda) = \lambda$. For a production $N \to \alpha$ we say that $\phi(N \to \alpha) = \phi(N) \to \phi(\alpha)$. A* morphism *$\phi$ from G_1 to G_2 is such a function from V_1 to V_2, such that $\phi(S_1) = \phi(S_2)$, and $\phi(P_1) \subseteq P_2$.*

Example 9. *Suppose G_1 has the three nonterminals S, A, B and productions $S \to AB, S \to ASB$ and $A \to a, B \to b$. $\mathcal{L}(G_1) = \{a^n b^n | n > 0\}$. Supoose G_2 has two nonterminals S, X and productions $S \to XX, S \to XSX$ and $X \to a, X \to b$. Then ϕ defined as $\phi(A) = X,' \phi(B) = X, \phi(S) = S$ is a morphism. $\phi(A \to a) = X \to a$, $\phi(B \to b) = X \to b$, and so on.*

Proposition 50. *If there is a morphism from G_1 to G_2 then $L(G_1) \subseteq L(G_2)$.*

We can state a more general result as follows.

Proposition 51. *If there is a morphism from G_1 to G_2 then for all nonterminals N in G_1:*

- $\mathcal{L}(G_1, N) \subseteq \mathcal{L}(G_2, \phi(N))$
- $\mathcal{C}(G_1, N) \subseteq \mathcal{C}(G_2, \phi(N))$

More precisely

Proposition 52. *If there is a morphism from G_1 to G_2 then for all nonterminals N in G_2:*

- $\bigcup_{M:\phi(M)=N} \mathcal{L}(G_1, M) \subseteq \mathcal{L}(G_2, N)$
- $\bigcup_{M:\phi(M)=N} \mathcal{C}(G_1, M) \subseteq \mathcal{C}(G_2, N)$

Proof. Take a derivation step $\beta N \gamma \Rightarrow_{G_1} \beta\alpha\gamma$ for some production $N \to \alpha \in P_1$. Then, since there is a morphism, ϕ, there is a production $\phi(N) \to \phi(\alpha)$ and thus $\phi(\beta)\phi(N)\phi(\gamma) \Rightarrow_{G_2} \phi(\beta)\phi(\alpha)\phi(\gamma)$.

Suppose $N \Rightarrow \alpha_1 \Rightarrow \dots \alpha_n \Rightarrow u$ is a derivation in G_1. Then $\phi(N) \Rightarrow \phi(\alpha_1) \Rightarrow \dots \phi(\alpha_n) \Rightarrow \phi(u) = u$ is a derivation of u from $\phi(N)$ wrt G_2.

Suppose $S \Rightarrow \alpha_1 \Rightarrow \dots \alpha_n \Rightarrow lNr$ is a derivation in G_1. Then $\phi(S) \Rightarrow \phi(\alpha_1) \Rightarrow \dots \phi(\alpha_n) \Rightarrow \phi(lNr) = l\phi(N)r$ is a derivation in G_2. □

Note here that there is not the usual duality or monotone/antitone pattern – rather both sets get bigger (or at least no smaller) when we go from G_1 to G_2.

Definition 24. *The image of G_1, $\phi(G_1) = \langle \Sigma, \phi(V_1), S_2, \phi(P_1) \rangle$ is a* CFG*, also which is called the morphic image of G_1 under the morphism ϕ. This wll be a subgrammar of G_2.*

Definition 25. *Given a* CFG *$G = \langle \Sigma, V, S, P \rangle$, and a function ψ from V into some arbitrary set X, we can define the morphic image of G to be the grammar $\psi(G) = \langle \Sigma, \psi(V), \psi(S), \psi(P) \rangle$. Clearly ψ is a morphism from G to $\psi(G)$.*

A brief discussion: if ψ is injective – i.e. it is not the case that two distinct nonterminals are mapped to the same symbol, then this is just a relabelling of G and it defines the same language. The more interesting case is where we have two or more nonterminals that are mapped to the same symbol: $\psi(M) = \psi(N)$. The new grammar then will be strictly smaller than the original grammar. In this case it may be that the new grammar will define a larger language. In the end we can map any grammar in CNF to the grammar with one nonterminals and productions $S \to SS, S \to a, S \to \lambda$ for $a \in \Sigma$ which defines the language Σ^*. This again is not very interesting – what we are most concerned with is the case when the resulting grammar defines the same language.

Proposition 53. *The identity function is a morphism. We can compose two morphisms to form a morphism. Given ϕ_1 a morphism from $G_1 \to G_2$ and ϕ_2 a morphism from $G_2 \to G_3$. $\phi_1 \circ \phi_2 : G_1 \to G_3$ is a morphism where $(\phi_1 \circ \phi_2)(N) = \phi_2(\phi_1(N))$.*

Definition 26. *A morphism is surjective/injective/bijective iff the underlying map of nonterminals is respectively surjective/injective/bijective.*

If there is a bijective morphism between two grammars, then we say they are isomorphic – they are identical up to a relabelling of nonterminals.

We now consider the case where the morphic image of the grammar defines the same language.

Definition 27. *A morphism from G_1 to G_2 is called* exact *if $L(G_1) = L(G_2)$.*

Definition 28. *For a* CFG *G_1 if there is a grammar G_2 such that there is an exact morphism $\phi : G_1 \to G_2$, (i.e. such that $L(G_1) = L(G_2)$), and there are two nonterminals M, N such that $\phi(M) = \phi(N)$, then we say that M and N are mergible.*

Definition 29. *We say that a* CFG *is* minimal *if it does not have any mergible nonterminals.*

Note that this definition does not mean that this is the globally smallest grammar for the language $L(G)$. It might even have some redundant states that could be removed. Maybe we should use a different term? Hmm.

Clearly if a grammar has two nonterminals that are mergible then we can merge them and get another grammar which defines the same language but is smaller. Suppose we are interested in minimal CFGs; then we want to find CFGs that do not have mergible states; because if we do have mergible states then the grammar is not minimal.

6 The universal morphism

We can now join up the (syntactic) CFG-morphisms and the (semantic) CIS-homomorphisms through the following lemma; any CIS that recognizes a language induces a morphism of the grammar where we map two nonterminals together if they 'mean' the same thing. If the CIS recognises L then the morphism is exact.

Proposition 54. *Suppose B is a CIS and h a surjective CIS homomorphism from $\mathcal{P}(\Sigma^*) \to B$. Then let G be a grammar. Let ϕ be the* CFG*-morphism given by $\phi(N) = h(\mathcal{L}(G, N))$. Then for all nonterminals N in G, $\mathcal{L}(\phi(G), \phi(N)) \subseteq h^*(\phi(N)) = h^*(h(\mathcal{L}(G, N)))$.*

Proof. Suppose we have a rule $N \to PQ$ in G. Therefore $\mathcal{L}(G, N) \supseteq \mathcal{L}(G, P)\mathcal{L}(G, Q)$. So $h(\mathcal{L}(G, N)) \geq h(\mathcal{L}(G, P)) \circ h(\mathcal{L}(G, Q))$. So $h^*(h(\mathcal{L}(G, N))) \geq h^*(h(\mathcal{L}(G, P)) \circ h(\mathcal{L}(G, Q)))$ by monotonicity of h^*. So $h^*(h(\mathcal{L}(G, N))) \geq h^*(h(\mathcal{L}(G, P))) \circ h^*(h(\mathcal{L}(G, Q)))$ by Proposition 39. Similarly suppose we have a rule $N \to a$ in G. So $h(\mathcal{L}(G, N)) \geq h(\{a\})$, and $h^*(h((\mathcal{L}(G, N))) \geq h^*(h(\{a\}))$. Therefore suppose that $N' \stackrel{*}{\Rightarrow}_{\phi(G)} w$; we want to show that $w \in h^*(N')$. We then proceed by induction on the length of the derivation. Base case: the production is of length 1. Suppose $N' \to a$ is a production in $\phi(G)$. Then for any production $N \to a$ in G such that $N' = \phi(N)$, $h^*(h((\mathcal{L}(G, N))) \geq h^*(h(\{a\})) \geq \{a\}$. Suppose it is true for all derivations of length at most k, and let $N' \stackrel{*}{\Rightarrow}_{\phi(G)} w$ be a derivation of length $k + 1$ that starts with $N' \to P'Q' \to uv = w$, where $P' \stackrel{*}{\Rightarrow}_{\phi(G)} u$ and $Q' \stackrel{*}{\Rightarrow}_{\phi(G)} v$. these last two derivations must be of length at most k.

Then consider any production $N \to PQ$ in G such that $\phi(N) = N'$, $\phi(P) = P'$ and $\phi(Q) = Q'$ (of which there must be at least one). Now by the inductive hypothesis, $u \in h^*(P')$ and $v \in h^*(Q')$. So $uv \in h^*(P')h^*(Q') = h^*(h(\mathcal{L}(G, P))) \circ h^*(h(\mathcal{L}(G, Q)))$. Now since $h^*(h(\mathcal{L}(G, N))) \geq h^*(h(\mathcal{L}(G, P))) \circ h^*(h(\mathcal{L}(G, Q)))$ this means that $uv \in h^*(h(\mathcal{L}(G, N)))$. So $w \in h^*(\phi(N)$. By induction, it is therefore true for derivations of any length. □

Clearly this implies the obvious fact that if h is the identity (and therefore so is h^*) this gives an exact morphism; the morphism which merges nonterminals which generate the same sets of strngs.

As a simple corollary we have now the folllowing lemma, which explains our definition of a CIS recognising a language through a morphism.

Proposition 55. *Let G be a* CFG *that defines the language L and B a CIS that recognises L through the morphism h. Then the morphism induced by map $\phi : N \to h(\mathcal{L}(G, N))$ is an* exact CFG*-morphism from G to $\phi(G)$.*

Proof. Since it is a morphism we know that $\mathcal{L}(\phi(G)) \supseteq \mathcal{L}(G)$. Since B recognises L, $h^*(h(L)) = L$, so $\mathcal{L}(\phi(G), \phi(S)) \subseteq h^*(\phi(S)) = L$. □

Note that this is only an implication: we can have a homomorphism that does not recognise the language, but where the induced grammar morphism is still exact, i.e. defines the same language. This could occur for example when we merge two nonterminals M, N where $\mathcal{C}(G, M) \odot \mathcal{L}(G, N) \subseteq L$ and $\mathcal{C}(G, N) \odot \mathcal{L}(G, M) \subseteq L$.

We can now define the universal morphism.

Definition 30. *Suppose G is a grammar that defines a language L; the* universal morphismϕ_L *is the* CFG*-morphism that for any nonterminal N of G, maps $N \to \mathcal{L}(G, N)^{\triangleright\triangleleft}$.*

Note that the universal morphism is an exact morphism by the previous lemma.

We can now state the following rather informal proposition.

Proposition 56. *I a grammar is minimal then there is at most one nonterminal for each concept. More precisely if the grammar is minimal, then the universal morphism is injective.*

Proof. If there were two nonterminals for a concept, the image under the universal morphism would have one fewer nonterminal and would thus be smaller than the original grammar, contradicting the assumption of minimality. □

We may not be able to compute, for an arbitrary CFG whether two nonterminals are mergible, but that is to a certain extent irrelevant. This proposition tells us that we can restrict ourselves to nonterminals that are concepts. We have two consequences – one is that that for every context free language we have a grammar where the nonterminals are closed sets of strings; secondly the simplest grammarsfor every language will be of this type.

7 Regular languages

In the case of regular languages, we know the lattice will be finite and this simplifies the analysis in a number of respects. In this case, all of the concepts can be defined using a finite number of substrings. Accordingly, it is possible to define the same structure using only idempotent semirings rather than complete idempotent semirings. In this case the construction is called the *syntactic semiring* and was introduced in [23, 24]. We briefly summarise the syntactic semiring in a notation consistent with this paper. Define $\mathcal{F}(X)$ to be the set of all finite subsets of a set X. Then $\mathcal{F}(\Sigma^*)$ is the free idempotent semiring over Σ. Consider a language $L \subseteq \Sigma^*$; then $\mathcal{F}(X) \subseteq \mathcal{F}(\Sigma^*)$. We can define a congruence on $\mathcal{F}(\Sigma^*)$, given X, Y which are elements of $\mathcal{F}(\Sigma^*)$, $X \sim_L Y$ iff $X^{\triangleright} = Y^{\triangleright}$. That is to say merely the restriction of the congruence to the collection of finite sets of strings. The quotient $\mathcal{F}(\Sigma^*)/\sim_L$ is the syntactic semiring. It can be shown that this is isomorphic to the SCL in the case of regular languages. Note that since $\mathcal{F}(\Sigma^*)$ is countable so is the syntactic semiring. Consider the example O_1^c. In this case the syntactic semiring is countable, whereas the syntactic concept lattice is uncountable; and there is no element that corresponds to L.

We can also construct a canonical representation. This is the universal CFG for the language and will contain a morphic image of every (binarised) context free grammar for the language. We emphasize that in the case of a nonregular context-free language, we can do the same construction but the resulting object will be infinite and thus not strictly speaking a grammar. Nonetheless, it will contain as with the finite case an image of every grammar for the language.

Definition 31. *Given a regular language L, the universal context free grammar for the language $\mathfrak{G}(L)$ is the* CFG *$\langle V, \mathcal{C}(L), \Sigma, P\rangle$, where V is the set of concepts of L,and P is the set of productions $\{Q \to RS \mid Q \geq R \circ S\} \cup \{N \to a | a \in N, a \in \Sigma \cup \{\lambda\}\}$*

We can now demonstrate an exact correspondence between the derivations in the grammar, and the pair of sets of contexts and sets of strings that constitute a nonterminal in the grammar.

Proposition 57. *For any language L and any concept $X = (S, C)$, we have*

- $\mathcal{L}(\mathfrak{G}(L), (S, C)) = S$

- $\mathcal{C}(\mathfrak{G}(L), (S, C)) = C$

Proof. By induction on the length of a derivation. Suppose $S \stackrel{*}{\Rightarrow} lXr$. We know $\mathcal{C}(l) \stackrel{*}{\Rightarrow} l$ and $\mathcal{C}(r) \stackrel{*}{\Rightarrow} r$, by part 1, and $S \geq \mathcal{C}(l) \circ X \circ \mathcal{C}(r)$ which gives us the two rules. □

Proposition 58. *Suppose we have a regular language L and a* CFG *G such that $\mathcal{L}(G) = L$. Then there is an exact morphism from G to $\mathfrak{G}(L)$.*

Proof. The morphism $N \to (\mathcal{L}(G, N)^{\triangleright\triangleleft}, \mathcal{L}(G, N)^{\triangleright})$ is such a morphism. □

Proposition 59. *The universal grammar does not have any mergible states.*

Proposition 60. *Given a finite automaton generating a language L, we can effectively construct $\mathfrak{G}(L)$.*

We will now give a complete example for a regular language. Consider $L = (ab)^*$. This has the 7 closed sets of strings: $\Sigma^*, \emptyset, L, \{\lambda\}, a(ba)^*, b(ab)^*, (ba)^*$. We therefore have 7 nonterminals in $\mathfrak{G}(L)$ that we denote $\top, \bot, S, \lambda, A, B, R$. The lexical rules in the grammar are: $\top \to a, \top \to b, \top \to \lambda, L \to \lambda, R \to \lambda, A \to a, B \to b$. The start symbol is S. There are 7^3 possible binary rules of which the following are in the grammar:

- Any production of the form $\top \to XY$
- Any production of the form $X \to \bot Y$ or $X \to Y \bot$.
- Any production of the form $X \to \lambda X$ or $X \to X\lambda$
- $L \to AB|LL$
- $A \to AR, LA$
- $B \to BL, RB$
- $R \to BA|RR$

$\mathfrak{G}(L)$ is a large but finite CFG. For any (binarised) grammar G that defines the same language, ϕ_L is an exact morphism from G to $\mathfrak{G}(L)$. That is to say this grammar contains an image of every grammar for the language.

7.1 The Universal Automaton

We now consider the relation of the syntactic concept lattice to the universal automaton (UA) [11, 22]. We recall the definition of the UA using the terminology of [22]; interestingly the Galois lattice structure of the UA does not form part if the standard presentations of the UA.

Definition 32. *Given a language L, a factorisation of L is a maximal pair of sets of strings P, Q such that $PQ \subseteq L$.*

We can of course consider this as a Galois connection between the prefixes and suffixes.

Definition 33. *For a set of strings X, define $X^{\blacktriangleleft} = \{y | Xy \subseteq L\}$ and $X^{\blacktriangleright} = \{y | yX \subseteq L\}$.*

These are the left and right residuals. A left factor is a set X (of prefixes) such that it is left-closed: $X = X^{\blacktriangleleft\blacktriangleright}$; conversely a right factor is one where $X = X^{\blacktriangleright\blacktriangleleft}$. A factorisation (P, Q) is thus a pair where $P^{\blacktriangleleft} = Q$ and $Q^{\blacktriangleright} = P$; clearly P is left-closed and Q is right-closed.

Definition 34. *For a language L, let $UA(L)$ be the set of all factorisations of L.*

Definition 35. *The Universal Automaton of a regular language L is defined to be a finite automaton where the states are factorisations, and where the set of initial states is $\{(P, Q) | \lambda \in P\}$, the set of final states is $\{(P, Q) \mid \lambda \in Q\}$, and the set of transitions is defined as $\{(P, Q) \to^a (P', Q') \mid PaQ' \subseteq L\}$.*

We now consider a number of close relationships between the UA and the SCL for a given language.

Proposition 61. *Let (P, Q) be a factorisation of a language. Then both P and Q are closed sets of strings, in the lattice $\mathfrak{B}(L)$. i.e. $P = P^{\triangleright\triangleleft}$ and $Q = Q^{\triangleright\triangleleft}$.*

Proof. If P is left-closed, then it is closed since $P = \{(\lambda, P^{\blacktriangleleft})\}^{\triangleleft}$. Similarly if Q is right closed, then $Q = \{(Q^{\blacktriangleright}, \lambda)\}^{\triangleleft}$. □

However it is not necessarily the case that (P, λ) or (λ, Q) are closed sets of contexts.

Finally we note that the set of contexts of the empty string are equivalent to the factorisations.

Proposition 62.

$$\{\lambda\}^{\triangleright} = \bigcup_{(P,Q)\in UA(L)} P \times Q$$

Proof. Suppose (P, Q) is a factorisation; Therefore $P \times Q \odot \lambda = PQ \subseteq L$, so $P \times Q \subseteq \{\lambda\}^{\triangleright}$. Conversely if (l, r) in $\{\lambda\}^{\triangleright}$, then $lr \in L$ so (l, r) must be an element of at least one maximal factorisation. □

8 Conclusion

The SCL appears quite naturally under several equivalent definitions; as the Galois connection between strings and contexts; as the set of maximal decompositions of the language into strings and contexts, and as the unique minimal complete idempotent semiring that recognises a language. It can be seen as the lifting of the syntactic congruence to sets of strings rather than strings. It forms the basis for various algorithms for grammar induction, and can give a complete and sound semantics for the Lambek calculus. It also has deep relations with the syntactic congruence and the universal automaton. These considerations suggest that it is worthy of sustained analysis and study, as it potentially can give rise to new algorithms for various problems associated with context-free grammars.

Moreover, it appears therefore that learning algorithms based on the SCL are similar in certain respects to algorithms that learn using Bayesian or MDL based techniques: the simplest grammar for a language – or in Bayesian terms the grammars that are the modes of the posterior distribution – will have nonterminals that correspond to these closed sets of strings.

The phrase "an algebraic theory of context-free languages" and indeed the now rather dated term "algebraic language" for context free language, has been used in a number of different contexts [4], as part of a program to base language theory not on phrase structure grammars or on machine models (finite state automatons or push-down automata) but rather on some algebras (e.g. [20]).

The work we present here can be considered as part of the same research program; rather than dealing directly with the productions of a context-free grammar, we operate with an algebraic structure that interprets it; a complete idempotent semiring. Homomorphisms of this structure can then be used to model transformations, i.e. morphisms, of the original grammar. We then naturally arrive at a unique object for each grammar, the terminal element of a category, and thus a universal morphism for CFGs. This gives us a set of syntactic categories that can be used as nonterminals in a grammar; under the assumption that we want the grammar to be minimal in a fairly obvious sense, we can without loss of generality assume that the grammar uses only these nonterminals. Each production then has an easily interpretable form as an inequality (or equality) between these syntactic categories. This massively simplifies the process of constructing a grammar for a language, and allows for efficient learning algorithms under a number of different models.

Acknowledgments

I am very grateful to Ryo Yoshinaka and Hans Leiss.

References

[1] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.

[2] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[3] B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In *Proceedings of IJCAI 21*, 2009.

[4] N. Chomsky and M.P. Schützenberger. The algebraic theory of context-free languages. *Studies in Logic and the Foundations of Mathematics*, 35:118–161, 1963.

[5] Alexander Clark. A learnable representation for syntax using residuated lattices. In *Proceedings of the 14th Conference on Formal Grammar*, Bordeaux, France, 2009.

[6] Alexander Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In José Sempere and Pedro Garcia, editors, *Grammatical Inference: Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, pages 24–37. Springer, September 2010.

[7] Alexander Clark. Efficient, correct, unsupervised learning of context-sensitive languages. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 28–37, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[8] Alexander Clark. Learning context free grammars with the syntactic concept lattice. In José Sempere and Pedro Garcia, editors, *Grammatical Inference: Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, pages 38–51. Springer, 2010.

[9] Alexander Clark. Logical grammars, logical theories. In Denis Bechet and Alexandre Dikovsky, editors, *Logical Aspects of Computational Linguistics*, pages 1–20. Springer, 2012.

[10] Alexander Clark and Rémi Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, August 2007.

[11] J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, London, 1971.

[12] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[13] F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using RFSAs. *Theoretical Computer Science*, 313(2):267–294, 2004.

[14] S. Eilenberg. *Automata, languages, and machines*. Academic press, 1974.

[15] Z. Ésik and H. Leiß. Algebraically complete semirings and greibach normal form. *Annals of Pure and Applied Logic*, 133(1):173–203, 2005.

[16] N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated lattices: an algebraic glimpse at substructural logics*. Elsevier, 2007.

[17] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1997.

[18] S. Ginsburg and H.G. Rice. Two families of languages related to ALGOL. *Journal of the ACM (JACM)*, 9(3):350–371, 1962.

[19] J.S. Golan. *Semirings and their Applications*. Springer, 1999.

[20] G. Hotz. A representation theorem of infinite dimensional algebras and applications to language theory. *Journal of Computer and System Sciences*, 33(3):423–455, 1986.

[21] Hans Leiss. Learning cfgs with the finite context property: A note on A.Clark's algorithm. Manuscript, 2012.

[22] S. Lombardy and J. Sakarovitch. The universal automaton. In E Grädel, Flum J., and T Wilke, editors, *Logic and Automata: History and Perspectives*, pages 457–494. Amsterdam Univ Pr, 2008.

[23] L. Polák. Syntactic semiring of a language. In *Mathematical Foundations of Computer Science 2001*, pages 611–620. Springer, 2001.

[24] L. Polák. Syntactic semiring and universal automaton. In *Developments in language theory*, pages 162–162. Springer, 2003.

[25] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.

[26] J. Sakarovitch. *Elements of automata theory*. Cambridge Univ Pr, 2009.

[27] Christian Wurm. Completeness of full lambek calculus for syntactic concept lattices. In *Proceedings of the 17th conference on Formal Grammar 2012 (FG)*, 2012.