Electronic Colloquium on Computational Complexity, Report No. 35 (2011)

Typed Monoids – An Eilenberg-like Theorem for non regular Languages

Christoph Behle Andreas Krebs Stephanie Reifferscheid*

Wilhelm-Schickard-Institut, Universität Tübingen,
{behlec,krebs,reiffers}@informatik.uni-tuebingen.de

Abstract

Based on different concepts to obtain a finer notion of language recognition via finite monoids we develop an algebraic structure called typed monoid. This leads to an algebraic description of regular and non regular languages.

We obtain for each language a unique minimal recognizing typed monoid, the typed syntactic monoid. We prove an Eilenberg-like theorem for varieties of typed monoids as well as a similar correspondence for classes of languages with weaker closure properties.

1 Introduction

We present an algebraic viewpoint on the study of formal languages and introduce a new algebraic structure to describe formal languages. We extend the known approach to use language recognition via monoids and morphisms by equipping the monoids with additional information to obtain a finer notion of recognition.

In the concept of language recognition by monoids, a language $L \subseteq \Sigma^*$ is recognized by a monoid M if there exists a morphism $h : \Sigma^* \to M$ and a subset A of M such that $L = h^{-1}(A)$. The syntactic monoid, which is the smallest monoid recognizing L, has emerged as an important tool to study classes of regular languages. Besides problems applying this tool to non regular languages, even for regular languages this instrument lacks precision.

One problem is, that not only the syntactic monoid but also the syntactic morphism, i.e. the morphism recognizing L via the syntactic monoid, plays an important role: Consider the two languages L_{parity}, L_{even} over the alphabet $\Sigma = \{a, b\}$ where the first one consists of all words with an even number of a's while the latter consists of all words of even length. Both languages have C_2, the cyclic group of order two, as syntactic monoid. But there are well-known results [FSS81, Ajt89] that in the model

*the third author has been supported by Proyecto MTM2010-19938-C03-02, Ministerio de Ciencia y Tecnologia, Spain

ISSN 1433-8092

of circuit complexity classes the language L_{parity} is harder than L_{even}. This reflects in the recognition via morphism through the fact that in L_{parity} we need to distinguish between an "a" as input and a "b" as input, whereas in L_{even} we count only the number of inputs.

So one of our goals is to introduce an algebraic structure where we have control of the morphisms allowed, especially where the single letters are mapped to. The concept of limiting the images of morphisms in the setting of finite monoids has been studied in [ÉL03]. It was shown that this concept is useful and yields nice algebraic objects. A similar approach but with a different algebraic object was used in [PS05].

Limiting the morphisms is a great tool, but especially if we want to study non regular languages we need to add more information. Consider the language L_{Maj} over the alphabet $\Sigma = \{a, b\}$ of all words with more a's than b's. This is clearly a non regular language and hence not recognized by a finite monoid. The syntactic monoid of L_{Maj} is $\mathbb{Z}$, but $\mathbb{Z}$ is also the syntactic monoid of an undecidable language in unary encoding.

The standard approach to recognize L_{Maj} by $\mathbb{Z}$ is the following: $\eta : \Sigma^* \to \mathbb{Z}$ is defined by $\eta(a) = +1$ and $\eta(b) = -1$. Then $w \in L \Leftrightarrow \eta(w) > 1$. Hence L_{Maj} is recognized by $\mathbb{Z}, \eta$, and the accepting set $\mathbb{Z}^+$ (the positive numbers). While in general various, even undecidable, languages can be recognized by $\mathbb{Z}$ with suitable accepting sets, what happens if we restrict the accepting set to $\mathbb{Z}^+$? Let $h : \Sigma^* \to \mathbb{Z}$ be a morphism, then $h(w) = h(a) \cdot \#_a(w) + h(b) \cdot \#_b(w)$, since $\mathbb{Z}$ is commutative. Hence, any language accepted by such a morphism is defined by a linear inequality of the ratio of the letters occurring.

The idea to limit the allowed accepting subset has been studied in [Sak76] and applied to context free languages. We use a different approach here: Instead of one accepting subset, we will consider a set of subsets and then consider the Boolean algebra generated by these sets. Each element of that Boolean algebra can be an accepting subset. We loose precision there, because this forces us to be closed under complementation, but our aim is the application of our approach to circuit complexity and descriptive complexity. The use of a Boolean algebra helps a lot to obtain a neat definition for the block product which is an important tool to characterize such classes algebraically.

If we combine the two approaches, i.e. fix the set of acceptance subsets of the monoid and limit the allowed morphisms we obtain even better possibilities. If we take $\mathbb{Z}$ and allow only morphisms mapping letter to $\{-1, +1\}$, and have $\mathbb{Z}^+$ as accepting subset we can only recognize languages that partition the alphabet in two sets A and B and test if the letters of set A occur more often than the letters of set B. All these languages are "close" to L_{Maj} in the sense that they reduce by a length preserving morphism to it.

These two observations lead us to the definition of a typed monoid. A *typed monoid* is a triple $(S, \mathfrak{S}, \mathcal{E})$, where S is a finitely generated monoid, $\mathfrak{S}$ is a finite Boolean algebra over S, and $\mathcal{E} \subseteq S$ is a finite set. The elements of $\mathfrak{S}$ are called *types* and the elements of $\mathcal{E}$ are called *units*.

A language is recognized by $(S, \mathfrak{S}, \mathcal{E})$ if there is a morphism h from $\Sigma^* \to S$, $h(\Sigma) \subseteq \mathcal{E}$, and $L = h^{-1}(\mathfrak{s})$ for a type $\mathfrak{s} \in \mathfrak{S}$. More generally, we use the units to limit the allowed morphisms while only types may be acceptance sets.

The syntactic monoid plays an important role in the study of (regular) languages.

In the theory of finite monoids it can be shown that the syntactic monoid is the smallest monoid recognizing a language with respect to division. Furthermore, if a monoid M divides a monoid N, then all languages recognized by M are recognized by N and hence the partial order of division on monoids is meaningful in terms of language recognition. We will show the same properties for our typed monoids. On the other hand, we will see that we are even able to distinguish the typed syntactic monoids of languages like majority and prime numbers, although they have the same syntactic monoid, namely $\mathbb{Z}$, in the conventional case.

In the finite case the theorem of Eilenberg, stating a one-to-one correspondence between varieties of (finite) monoids and varieties of (regular) languages, is the origin of the algebraic study of formal languages. Recall that by results of Schützenberger and McNaughton and Papert [Sch65, MP71] the (regular) starfree languages are exactly the languages that can be recognized by (finite) aperiodic monoids, i.e. monoids that contain only trivial subgroups. This result lead to the decidability of the question whether a given regular language is starfree, and there are similar results for many varieties of regular languages.

In Section 6 we will define varieties of typed monoids and formulate a version of Eilenberg's theorem for typed monoids. Using this it is possible to obtain algebraic counterparts of varieties of (non-regular) formal languages [KLR07, Kre08].

Another important tool in the study of formal languages is there description via logic. So it was shown that the star free languages are exactly the languages describable by a first order logic fragment, namely FO[<]. The fact that FO is connected to the circuit class AC^0 which cannot recognize a group language [Ajt89, FSS81] lead to a study about the links between subclasses of regular languages and classes of circuits. These studies exhibited some interesting connections between some varieties of regular languages and certain circuit classes. For a survey we recommend [TT07].

While varieties play an important role, a lot of language classes defined by logic classes do not form a variety. Consider for instance the class FO[<, mod]: It can recognize all words of even length L_{even}, but it is known [Ajt89, FSS81] that this class cannot express L_{parity}. Hence it cannot be closed under arbitrary inverse morphisms, because there is a non length preserving morphism h such that $L_{parity} = h^{-1}(L_{even})$.

Eilenberg studied classes of transformation semigroups with weaker closure properties than varieties. We define weakly closed classes of typed monoids in Section 5 and show that for each weakly closed class of languages there exists a corresponding weakly closed class of typed monoids. This gives a weaker version of the variety theorem for classes of languages with closure properties like FO[<, mod].

We sum up the structure of the paper: In Section 3 we give the basic definitions of our algebraic objects, morphisms, and language recognition. In the following sections we transfer the Eilenberg program for (finite) monoids into the world of typed monoids. The typed syntactic monoid and its minimality are treated in Section 4. We define closure properties in Section 5 and show that weakly closed classes of typed monoids correspond to weakly closed classes of languages. In Section 6 we consider varieties and show that the correspondence of the previous section is one-to-one for varieties.

2 Preliminaries

We define here most notions in a very basic way and presume the reader to be familiar with Eilenberg's variety theory, namely the concepts of language recognition via monoids and varieties. For a complete survey of monoids and language recognition we recommend [Pin86]. For readers who wish an in-depth study of monoids and varieties [Alm95] is a good source. We will bring up some logic classes from time to time, mostly used in examples and as motivation for some classes of languages. The reader can skip these parts but for those interested we refer to [Str94]. Most examples can be found there and we use the same notation. It also displays the connections of descriptive complexity, circuit complexity, and algebra.

Let A, B be nonempty sets; A mapping $f : A \mapsto B$ is called *surjective* if $h(A) = B$ and *injective* if for every $b \in B$ there is at most one $a \in A$ such that $h(a) = b$; f is *bijective* iff f is injective and surjective.

A *semigroup* S is a nonempty set equipped with a binary relation $\cdot$ which is associative. We call a semigroup M a *monoid* if it has a *neutral element* 1_M, i.e. an element such that $1_M \cdot x = x \cdot 1_M = x$ for all $x \in M$. A monoid G is called *group* if for each $g \in G$ there is a (unique) element $h \in G$ such that $g \cdot h = h \cdot g = 1_G$. As usual we write xy for $x \cdot y$ and 1 for 1_M if the context is understood.

A monoid M is *finite* if M is a finite set. A subset P of M *generates* M, denoted by $M = \langle P \rangle$, if each element in M can be written as a finite product of elements in P and the neutral element. M is *finitely generated* if there exists a finite subset P of M generating M. The *free monoid* with generator set A is usually denoted by A^*, i.e. we take the set of all finite sequences of elements of A together with the empty word as neutral element, the multiplication is defined as concatenation. Let M, N be monoids, a *(monoid-)morphism* h from M to N is a mapping $h : M \to N$ such that for all $m_1, m_2 \in M$: $h(m_1 m_2) = h(m_1)h(m_2)$ and $h(1) = 1$. A subset M' is a *submonoid* of M, if $xy \in M'$ for all $x, y \in M'$ and $1 \in M'$. A monoid N *divides* a monoid M ($N \preceq M$) if it is a morphic image of a submonoid of M, that is there exists a submonoid M' of M and a surjective morphism from M' to N.

A *congruence* on a monoid M is an equivalence relation $\sim$ on M, such that $x \sim y$ and $x' \sim y'$ imply $xx' \sim yy'$. Given a congruence $\sim$ on M, the set of equivalence classes together with the multiplication $[x][y] = [xy]$ form a monoid, the *quotient monoid* denoted by $M/\sim$. The mapping $x \mapsto [x]$ is a morphism, the so called *canonical epimorphism*. Conversely, a morphism $h : M \to N$ defines a congruence $\sim_h$ on M by setting $x \sim_h y \Leftrightarrow h(x) = h(y)$.

The set of integers $\mathbb{Z}$ with the usual addition is an infinite monoid generated by $\{-1, 1\}$; for each natural number k we denote the quotient monoid $\mathbb{Z}/k\mathbb{Z}$ of $\mathbb{Z}$ (corresponding to the congruence $x \sim y$ iff $x \equiv y \mod k$) by C_k and its elements by $0, \ldots, k-1$.

A *Boolean algebra* $\mathfrak{B}$ over a nonempty set S is a finite set of subsets of S containing $\emptyset, S$ and being closed under union, intersection and complement. Let $\mathfrak{B}, \mathfrak{C}$ be two Boolean algebras over sets S and T respectively. A *morphism* h from $\mathfrak{B}$ to $\mathfrak{C}$ is a mapping $h : \mathfrak{B} \to \mathfrak{C}$ such that $h(\emptyset) = \emptyset$, $h(S) = T$, $h(S \setminus B_1) = T \setminus h(B_1)$ and $h(B_1 \cup B_2) = h(B_1) \cup h(B_2)$ for all $B_1, B_2 \in \mathfrak{B}$.

If $\mathfrak{B}$ is a Boolean algebra over S and $\mathfrak{C} \subseteq \mathfrak{B}$ is again a Boolean algebra over S, then

$\mathfrak{C}$ is a Boolean subalgebra of $\mathfrak{B}$.

Languages. An *alphabet* Σ is a finite non-empty set and its elements are called *letters*. The elements of the *free* monoid Σ^* are called *words* and a *language* is a subset of Σ^*. Let w be a word then we can uniquely write it as a product of letters $w = w_1 \dots w_n$, where n is called the *length* of the word, denoted by $|w|$. For a letter a and a word w we define $\#_a(w) = |\{i \mid w_i = a\}|$, i.e. the number of occurrences of the letter a in w. A morphism h from Σ^* to Δ^* is called length preserving if $h(\Sigma) \subseteq \Delta$.

A monoid M *recognizes* a language $L \subseteq \Sigma^*$ iff there is a morphism $h : \Sigma^* \to M$ and a subset $A \subseteq M$ such that $L = h^{-1}(A)$.

Given a language $L \subseteq \Sigma^*$ the *syntactic congruence* is defined on Σ^* by $x \equiv_L y$ iff for all $w, v \in \Sigma^*$ holds $wxv \in L \Leftrightarrow wyv \in L$. The *syntactic monoid* $M(L)$ of L is defined as the quotient monoid $\Sigma^* / \equiv_L$. The canonical epimorphism for this congruence is called *syntactic morphism* and denoted by η_L. It can be shown that the syntactic monoid is a minimal monoid recognizing L with respect to division.

3 Typed monoids

In this section we introduce the notion of typed monoids. We start from the usual concept of language recognition by monoids. Instead of only considering the monoid M we want control over the possible morphisms, a concept already studied in [ÉL03, PS05]. We follow the approach of [ÉL03] and enhance the monoid with a subset, called units, and require morphisms to map units on units. Further, to reduce the power of the monoid we equip the structure with a finite Boolean algebra over the monoid, and require the accepting subsets to be elements of this Boolean algebra. This is in particular helpful when dealing with infinite monoids or non regular languages.

Definition 3.1 (Typed Monoid)**.** A *typed monoid* is a triple $(S, \mathfrak{S}, \mathcal{E})$, where S is a finitely generated monoid, $\mathfrak{S}$ is a finite Boolean algebra over S, and $\mathcal{E} \subseteq S$ is a finite set. The elements of $\mathfrak{S}$ are called *types* and the elements of $\mathcal{E}$ are called *units*.

For a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ we will write $(S, \{\mathfrak{s}_1, \dots, \mathfrak{s}_d\}, \mathcal{E})$, where the $\mathfrak{s}_i$ are types generating $\mathfrak{S}$. If the Boolean algebra $\mathfrak{S}$ is generated by a single set we will drop the braces: $(S, \{\mathfrak{s}\}, \mathcal{E}) = (S, \mathfrak{s}, \mathcal{E})$. We call a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ *free*, if the underlying monoid S is free.

Please note, that we do not require the units to generate S.

Example 3.2. We give some examples of typed monoids.

(a) Let $S = \mathbb{Z}$, $\mathfrak{S} = \{\emptyset, \mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}_0^-\}$ (where $\mathbb{Z}^+$ are the positive numbers and $\mathbb{Z}_0^-$ are the non positive numbers), and $\mathcal{E} = \{-1, 1\}$. Then $(S, \mathfrak{S}, \mathcal{E})$ is a typed monoid. As stated above we use the short hand notation $(\mathbb{Z}, \{\mathbb{Z}^+, \mathbb{Z}_0^-\}, \{-1, 1\})$ or even $(\mathbb{Z}, \mathbb{Z}^+, \{-1, 1\})$ to denote $(S, \mathfrak{S}, \mathcal{E})$.

(b) Let $S = \mathbb{Z}$, $\mathfrak{S} = \{\emptyset, 2\mathbb{Z}, 2\mathbb{Z} + 1, \mathbb{Z}\}$, and $\mathcal{E} = \{0, 1\}$. Then $(S, \mathfrak{S}, \mathcal{E})$ is a typed monoid. Again, we use the short hand notation $(\mathbb{Z}, \{2\mathbb{Z}, 2\mathbb{Z} + 1\}, \{0, 1\})$ or even $(\mathbb{Z}, 2\mathbb{Z}, \{0, 1\})$ to denote $(S, \mathfrak{S}, \mathcal{E})$.

(c) Let $S = C_4$, $\mathfrak{S} = \{\emptyset, \{0\}, \{1,2,3\}, \{0,1,2,3\}\}$, and $\mathcal{E} = \{1\}$. Then $(S, \mathfrak{S}, \mathcal{E})$ is a finite typed monoid. As we will see later this monoid is more restricted than C_4 in the untyped world.

(d) Each finite monoid S can be seen as a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ where the types are the subsets of S (that is, $\mathfrak{S}$ is the powerset of S) and the set of units is $\mathcal{E} = S$.

(e) There is a strong connection between a language $L \subseteq \Sigma^*$ and typed monoids. The structure $(\Sigma^*, \{\emptyset, L, \Sigma^* \setminus L, \Sigma^*\}, \Sigma) = (\Sigma^*, L, \Sigma)$ is a typed monoid. Note, that a language L and its complement $\bar{L}$ lead to the same typed monoid.

As mentioned before the units will limit the set of possible morphisms. On the free monoid Σ^* we usually pick Σ as the units; in this case every element has a unique representation as a product of the units and the "length" of an element is the number of units in its representation. Even on arbitrary monoids units give a kind of length property, this has been studied in [ÉL03].

We define the notion of a morphism for typed monoids.

Definition 3.3 (Typed Morphism)**.** A *(typed) morphism* $h : (S, \mathfrak{S}, \mathcal{E}) \to (S', \mathfrak{S}', \mathcal{E}')$ *of typed monoids* is specified by a triple $(h_S, h_\mathfrak{S}, h_\mathcal{E})$, where $h_S : S \to S'$ is a monoid morphism, $h_\mathfrak{S} : \mathfrak{S} \to \mathfrak{S}'$ is a morphism of Boolean algebras, $h_\mathcal{E} : \mathcal{E} \to \mathcal{E}'$ is a mapping of sets, and the triple fulfills the two compatibility requirements:

1. $\forall \mathfrak{s} \in \mathfrak{S}$ it holds $h_S(\mathfrak{s}) = h_\mathfrak{S}(\mathfrak{s}) \cap h_S(S)$,
2. $\forall u \in \mathcal{E}$ it holds $h_S(u) = h_\mathcal{E}(u)$.

Because of the compatibility conditions of this definition we can omit the indices of the morphisms.

Condition 2 forces h_S to map units to units and to be compatible with $h_\mathcal{E}$, and thus the compatibility conditions imply that h_S induces $h_\mathcal{E}$ and - in case h_S is surjective - also $h_\mathfrak{S}$. Note further that 1 implies that a nonempty type cannot be mapped to the empty type, and thus $h_\mathfrak{S}$ needs to be injective, i.e. $|\mathfrak{S}| \leq |\mathfrak{S}'|$.

The definitions of injective and surjective morphisms are straightforward.

Definition 3.4 (Injective, Surjective, Typed Submonoid, Division)**.** Let $(S, \mathfrak{S}, \mathcal{E}), (T, \mathfrak{T}, \mathcal{F})$ be two typed monoids.

- A typed morphism $h : (S, \mathfrak{S}, \mathcal{E}) \to (T, \mathfrak{T}, \mathcal{F})$ with $h = (h_S, h_\mathfrak{S}, h_\mathcal{E})$ is *injective* (resp. surjective, bijective) if $h_S, h_\mathfrak{S}$, and $h_\mathcal{E}$ are all injective (resp. surjective, bijective).
  If h is bijective we say that $(S, \mathfrak{S}, \mathcal{E})$ and $(T, \mathfrak{T}, \mathcal{F})$ are *isomorphic* (denoted $\cong$).
- A typed monoid $(T, \mathfrak{T}, \mathcal{F})$ is a *typed submonoid* of $(S, \mathfrak{S}, \mathcal{E})$ (we write $(T, \mathfrak{T}, \mathcal{F}) \leq (S, \mathfrak{S}, \mathcal{E})$) if T is a submonoid of S and there is an injective morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$.
- A typed monoid $(T, \mathfrak{T}, \mathcal{F})$ *divides* a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ (we write $(T, \mathfrak{T}, \mathcal{F}) \preceq (S, \mathfrak{S}, \mathcal{E})$) if $(T, \mathfrak{T}, \mathcal{F})$ is the morphic image of a typed submonoid of $(S, \mathfrak{S}, \mathcal{E})$.

As one should expect, concatenation of two (injective/surjective) morphisms is again a (injective/surjective) morphism. And given a typed morphism $h : (S, \mathfrak{S}, \mathcal{E}) \to (T, \mathfrak{T}, \mathcal{F})$, the image (preimage) of h is a submonoid in $(T, \mathfrak{T}, \mathcal{F})$ ($(S, \mathfrak{S}, \mathcal{E})$).

To clarify the notion of a typed morphism consider the following typed monoids. We let

$$(S, \mathfrak{S}, \mathcal{E}) = (C_4, \{\{0\}, \{1\}, \{2\}, \{3\}\}, \{0, 1\}),$$

$$(T, \mathfrak{T}, \mathcal{F}) = (C_4, \{\{0, 2\}, \{1, 3\}\}, \{0, 1\}), \text{ and}$$

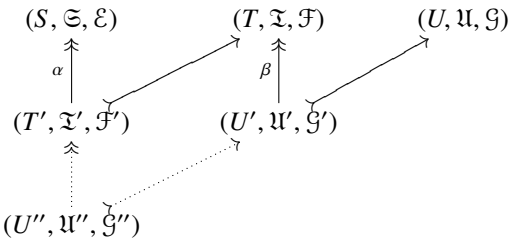
$$(U, \mathfrak{U}, \mathcal{G}) = (C_2, \{\{0\}, \{1\}\}, \{0, 1\}).$$

Since $|\mathfrak{S}| > |\mathfrak{T}|$, there is no typed morphism from $(S, \mathfrak{S}, \mathcal{E})$ to $(T, \mathfrak{T}, \mathcal{F})$. On the other hand the identity mapping yields an injective (but not surjective) typed morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$. Hence, $(T, \mathfrak{T}, \mathcal{F})$ is a typed submonoid of $(S, \mathfrak{S}, \mathcal{E})$.

There is no typed morphism from $(U, \mathfrak{U}, \mathcal{G})$ to $(S, \mathfrak{S}, \mathcal{E})$: Assume h is such a morphism. Then $h_{\mathcal{G}}(0) = h_U(0) = 0$ and $h_{\mathcal{G}}(1)$ can be 0 or 1. In the first case we hurt condition 1. Because then the following equation should hold: $0 \in h_{\mathfrak{U}}(\{0\}) \cap h_{\mathfrak{U}}(\{1\}) = h_{\mathfrak{U}}(\emptyset) = \emptyset$; contradiction. If we let $h_U(1) = 1$ then h_U is not a (monoid)morphism; in particular $(U, \mathfrak{U}, \mathcal{G})$ is no submonoid of $(S, \mathfrak{S}, \mathcal{E})$. Conversely, $(U, \mathfrak{U}, \mathcal{G})$ is a factor of $(S, \mathfrak{S}, \mathcal{E})$, since the mapping $h_T : (T, \mathfrak{T}, \mathcal{F}) \to (U, \mathfrak{U}, \mathcal{G})$, $i \mapsto i \pmod 2$ defines a typed morphism from $(T, \mathfrak{T}, \mathcal{F})$ onto $(U, \mathfrak{U}, \mathcal{G})$, and $(T, \mathfrak{T}, \mathcal{F})$ is a submonoid of $(S, \mathfrak{S}, \mathcal{E})$.

Lemma 3.5. *If* $(S, \mathfrak{S}, \mathcal{E}) \preceq (T, \mathfrak{T}, \mathcal{F})$ *and* $(T, \mathfrak{T}, \mathcal{F}) \preceq (U, \mathfrak{U}, \mathcal{G})$*, then* $(S, \mathfrak{S}, \mathcal{E}) \preceq (U, \mathfrak{U}, \mathcal{G})$.

Proof. Suppose that $(S, \mathfrak{S}, \mathcal{E}) \preceq (T, \mathfrak{T}, \mathcal{F})$ and $(T, \mathfrak{T}, \mathcal{F}) \preceq (U, \mathfrak{U}, \mathcal{G})$. By definition of divisor we have submonoids $(T', \mathfrak{T}', \mathcal{F}') \preceq (T, \mathfrak{T}, \mathcal{F})$ and $(U', \mathfrak{U}', \mathcal{G}') \preceq (U, \mathfrak{U}, \mathcal{G})$ and are in the following situation:



In the diagram above we define $(U'', \mathfrak{U}'', \mathcal{G}'') = \beta^{-1}((T', \mathfrak{T}', \mathcal{F}'))$, then $\alpha \circ \beta$ maps $(U'', \mathfrak{U}'', \mathcal{G}'')$ surjectively on $(S, \mathfrak{S}, \mathcal{E})$ and is a submonoid of $(U, \mathfrak{U}, \mathcal{G})$, hence $(S, \mathfrak{S}, \mathcal{E}) \preceq (U, \mathfrak{U}, \mathcal{G})$. □

A concept strongly connected to morphisms is the concept of congruences. Each morphism induces a congruence and vice versa. Congruences on typed monoids need to be compatible with the set of types.

Definition 3.6 (Typed congruence)**.** Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid and $\sim$ be a congruence on S. Then $\sim$ is a *typed congruence* if $\forall \mathfrak{S} \in \mathfrak{S}, s_1, s_2 \in S : s_1 \sim s_2 \wedge s_1 \in \mathfrak{S} \Rightarrow s_2 \in \mathfrak{S}$.

In this case we say that $\sim$ is finer than $\mathfrak{S}$.

Example 3.7. The usual syntactic congruence $\equiv_L$ is a typed congruence on (Σ^*, L, Σ). To see this we have to show that $\equiv_L$ respects the types, but these are only $\emptyset, L, \bar{L}, \Sigma^*$ and L is a union of congruence classes of $\equiv_L$.

Let $\sim$ be a typed congruence on a typed monoid $(S, \mathfrak{S}, \mathcal{E})$. Let $\mathcal{E}/\sim = \{[x]_\sim \mid x \in \mathcal{E}\}$, $\mathfrak{s}/\sim = \{[x]_\sim \mid x \in \mathfrak{s}\}$ and $\mathfrak{S}/\sim = \{\mathfrak{s}/\sim \mid \mathfrak{s} \in \mathfrak{S}\}$. This is well defined, since $\sim$ is finer than $\mathfrak{S}$. We call $(S, \mathfrak{S}, \mathcal{E})/\sim = (S/\sim, \mathfrak{S}/\sim, \mathcal{E}/\sim)$ the typed quotient monoid of $(S, \mathfrak{S}, \mathcal{E})$ by $\sim$.

As in the classical case, a typed morphism defines a typed congruence via $s_1 \sim_h s_2 \Leftrightarrow h_S(s_1) = h_S(s_2)$. Likewise, a typed congruence $\sim$ on a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ defines a morphism from $(S, \mathfrak{S}, \mathcal{E})$ to $(S, \mathfrak{S}, \mathcal{E})/\sim$.

4 Typed Syntactic Monoid

We turn now to the recognition of languages by typed monoids. Instead of considering all morphisms into a monoid we limit the allowed morphisms to these mapping letters to units, and the only accepting subsets allowed are types. We can reduce the definition of language recognition to the notion of a typed morphism:

Definition 4.1. We say that $(S, \mathfrak{S}, \mathcal{E})$ *recognizes a language* $L \subseteq \Sigma^*$ if there is a typed morphism $h : (\Sigma^*, L, \Sigma) \to (S, \mathfrak{S}, \mathcal{E})$.

Since L and $\bar{L}$ are disjoint types they have to be mapped on two disjoint types. Hence, there is a type $\mathfrak{s} \in \mathfrak{S}$, such that $L = h_S^{-1}(\mathfrak{s})$. So the definition resembles the usual definition of language recognition via monoids. The standard notion of language recognition for finite monoids coincides with our notion in the following way: Given a finite monoid S then S and the typed monoid $(S, \mathcal{P}(S), S)$ recognize exactly the same languages.

Example 4.2. Let $L \subseteq \Sigma^*$ be a language, $M(L)$ its usual syntactic monoid, η_L the syntactic morphism, and $A = \eta_L(L)$ the accepting subset. Then L is recognized by the typed monoid $(M(L), A, \eta_L(\Sigma)) = (\eta_L(\Sigma^*), \eta_L(L), \eta_L(\Sigma))$.

The typed syntactic monoid will be defined using the syntactic congruence which is a typed congruence on the typed monoid (Σ^*, L, Σ).

Definition 4.3 (Typed Syntactic Monoid)**.** Let $L \subseteq \Sigma^*$ be a language, then $syn(L) = (\Sigma^*, L, \Sigma)/\equiv_L$ is the *typed syntactic monoid* of L.

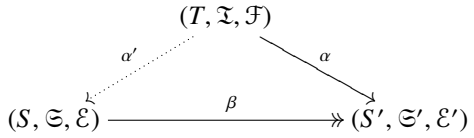
It is easy to see that L is recognized by its typed syntactic monoid via the typed morphism induced by the syntactic morphism η_L. We call that morphism the *typed syntactic morphism*.

In the finite case it is known that the syntactic monoid is the unique (up to isomorphism) minimal monoid recognizing the language (minimal with respect to division). Since in general, division is only a preorder (see example 4.4), in the classical case we only have that $M(L)$ is a minimal element recognizing the language L, i.e.: If N is a monoid recognizing L, then $M(L)$ divides N.

Example 4.4. Let S and T be the free monoid with two and three generators, respectively. Define the typed monoids $(S, \mathfrak{S}, \mathcal{E}) = (\langle a, b\rangle, \{\langle a, b\rangle\}, \{1\})$ and $(T, \mathfrak{T}, \mathcal{F}) = (\langle a, b, c\rangle, \{\langle a, b, c\rangle\}, \{1\})$. Then S and T divide each other but are not isomorphic: $\langle a, b, c\rangle$ is isomorphic to the submonoid $\langle a, ab, abb\rangle$ (this remains true with the given types and units) and $\langle a, b\rangle$ is isomorphic to the submonoid $\langle a, b\rangle$ of $\langle a, b, c\rangle$ (this also remains true for the given types and units).

We want to show that syn(L) divides $(S, \mathfrak{S}, \mathcal{E})$ if $(S, \mathfrak{S}, \mathcal{E})$ recognizes L.

Lemma 4.5. *If there is a surjective morphism $\beta : (S, \mathfrak{S}, \mathcal{E}) \to (S', \mathfrak{S}', \mathcal{E}')$, then every morphism from the free monoid $(T, \mathfrak{T}, \mathcal{F})$, where $\mathcal{F}$ is the standard generator set for T, to $(S', \mathfrak{S}', \mathcal{E}')$ factors though β. That is for every morphism α, there is a morphism α' such that the following diagram commutes, i.e. $\beta \circ \alpha' = \alpha$:*



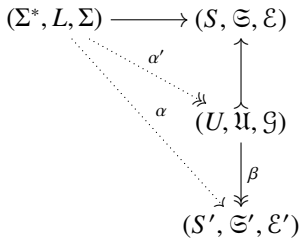
Proof. We need to define the typed morphism $\alpha' = (\alpha'_T, \alpha'_{\mathfrak{T}}, \alpha'_{\mathcal{F}})$. For each unit $s' \in \mathcal{E}'$ there is an element $s \in \mathcal{E}$ with $\beta(s) = s'$. For each s' pick such an element and call it $s_{s'}$. We define $\alpha' : (T, \mathfrak{T}, \mathcal{F}) \to (S, \mathfrak{S}, \mathcal{E})$ by setting $\alpha'_T(t) = s_{\alpha(t)}$ for all free generators t of T. Recall that this defines a morphism on T, and thus also the mapping $\alpha'_{\mathcal{F}}$. It remains to define the corresponding mapping for the types. Let $\mathfrak{T} \in \mathfrak{T}$ and set $\alpha'_{\mathfrak{T}}(\mathfrak{T}) = \max\{\mathfrak{S} \in \mathfrak{S} \ : \ \mathfrak{S} \cap \alpha'_T(T) = \alpha'_T(\mathfrak{T})\}$. Since every $\mathfrak{S} \in \beta^{-1}(\alpha(\mathfrak{T}))$ is in the above set, this is well defined. □

As in the finite case the syntactic monoid is minimal with respect to language recognition. We start by showing the following lemma which is a consequence of Lemma 4.5:

Lemma 4.6. *Let $(S, \mathfrak{S}, \mathcal{E})$, $(S', \mathfrak{S}', \mathcal{E}')$ be typed monoids, such that $(S', \mathfrak{S}', \mathcal{E}')$ divides $(S, \mathfrak{S}, \mathcal{E})$. Then every languages recognized by $(S', \mathfrak{S}', \mathcal{E}')$ is also recognized by $(S, \mathfrak{S}, \mathcal{E})$.*

In particular: If syn(L) *divides $(S, \mathfrak{S}, \mathcal{E})$, then $(S, \mathfrak{S}, \mathcal{E})$ recognizes L.*

Proof.



By definition there is a submonoid $(U, \mathfrak{U}, \mathcal{G})$ of $(S, \mathfrak{S}, \mathcal{E})$ and a surjective morphism $\beta : (U, \mathfrak{U}, \mathcal{G}) \to (S', \mathfrak{S}', \mathcal{E}')$. Since $(S', \mathfrak{S}', \mathcal{E}')$ recognizes the language $L \subseteq \Sigma^*$ there is a morphism $\alpha : (\Sigma^*, L, \Sigma) \to (S', \mathfrak{S}', \mathcal{E}')$. Hence by Lemma 4.5 there is a morphism

$\alpha' : (\Sigma^*, L, \Sigma) \to (U, \mathfrak{U}, \mathcal{G})$. Since $(U, \mathfrak{U}, \mathcal{G})$ is a submonoid of $(S, \mathfrak{S}, \mathcal{E})$ there is a morphism $i : (U, \mathfrak{U}, \mathcal{G}) \to (S, \mathfrak{S}, \mathcal{E})$, thus L is recognized by $(S, \mathfrak{S}, \mathcal{E})$ via the morphism $i \circ \alpha'$. □

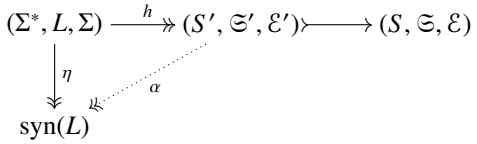
Although in the theory of typed monoids division is not a partial order (see example 4.4) we can show that the typed syntactic monoid of L is in fact the unique minimal typed monoid recognizing L.

Lemma 4.7. *Let $L \subseteq \Sigma^*$ be a language and $(S, \mathfrak{S}, \mathcal{E})$ a typed monoid.*

(a) $(S, \mathfrak{S}, \mathcal{E})$ *recognizes L if and only if* $\mathrm{syn}(L)$ *divides* $(S, \mathfrak{S}, \mathcal{E})$.

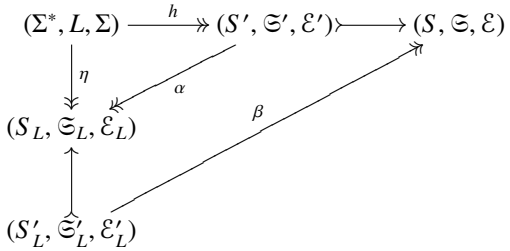
(b) $\mathrm{syn}(L)$ *is the unique minimal element (with respect to division) recognizing L, i.e. if $(S, \mathfrak{S}, \mathcal{E})$ recognizes L and $(S, \mathfrak{S}, \mathcal{E})$ divides* $\mathrm{syn}(L)$*, then $(S, \mathfrak{S}, \mathcal{E}) \cong \mathrm{syn}(L)$.*

Proof. (a) : We only need to show that $\mathrm{syn}(L)$ divides every typed monoid that recognizes $L \subseteq \Sigma^*$. So let $h : (\Sigma^*, L, \Sigma) \to (S, \mathfrak{S}, \mathcal{E})$ be a morphism. Further let $(S', \mathfrak{S}', \mathcal{E}')$ be the image of (Σ^*, L, Σ). We show that there is a surjective morphism α from $(S', \mathfrak{S}', \mathcal{E}')$ to $\mathrm{syn}(L)$ (and hence $\mathrm{syn}(L)$ divides $(S, \mathfrak{S}, \mathcal{E})$).



So we need to show that $\alpha(s) = \eta(h^{-1}(s))$ is well defined. By contradiction assume that there are w_1, w_2 with $h(w_1) = h(w_2)$ and $\eta(w_1) \neq \eta(w_2)$, then there are $u, v \in \Sigma^*$ with $uw_1v \in L$ and $uw_2v \notin L$, but $h(uw_1v) = h(uw_2v)$ and hence L is not recognized by S. It is obvious that α respects units and types and therefore defines a typed morphism.

(b) : Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid recognizing a language $L \subseteq \Sigma^*$ via a morphism h and dividing $\mathrm{syn}(L) := (S_L, \mathfrak{S}_L, \mathcal{E}_L)$. Then there is a submonoid $(S'_L, \mathfrak{S}'_L, \mathcal{E}'_L)$ of $(S_L, \mathfrak{S}_L, \mathcal{E}_L)$, a submonoid $(S', \mathfrak{S}', \mathcal{E}')$ of $(S, \mathfrak{S}, \mathcal{E})$, and morphisms α and β as shown below:



Since Σ generates Σ^* and h, η is surjective, we know $\mathcal{E}'$ generates S', and S_L is generated by $\mathcal{E}_L$. Moreover $|\mathcal{E}'| \leq |\mathcal{E}| \leq |\mathcal{E}'_L| \leq |\mathcal{E}_L| \leq |\mathcal{E}'|$, so $|\mathcal{E}'| = |\mathcal{E}| = |\mathcal{E}'_L| = |\mathcal{E}_L|$. In particular $(S_L, \mathfrak{S}_L, \mathcal{E}_L) \cong (S'_L, \mathfrak{S}'_L, \mathcal{E}'_L)$, and thus $\mathcal{E}$ generates S, which again implies $(S, \mathfrak{S}, \mathcal{E}) \cong (S', \mathfrak{S}', \mathcal{E}')$.

Now we have a surjective morphism from $(S, \mathfrak{S}, \mathcal{E})$ to $(S_L, \mathfrak{S}_L, \mathcal{E}_L)$ and converse, but this does not imply that α or β are isomorphisms. But it is clear that $\alpha \circ \beta$ is a permutation of $\mathcal{E}$, hence there is a power of $\alpha \circ \beta$ that is the identity on $\mathcal{E}$. But then this power is also an identity on $(S, \mathfrak{S}, \mathcal{E})$ and we have $(S_L, \mathfrak{S}_L, \mathcal{E}_L) \cong (S, \mathfrak{S}, \mathcal{E})$. □

5 Weakly Closed Classes

Motivated by Eilenberg's notion of a weakly closed class of transformation semigroups we consider weakly closed classes of typed monoids and languages. Weakly closed classes of transformation semigroups are closed under division and direct product. For typed monoids we add an additional operation which allows to identify typed monoids recognizing the same languages.

Definition 5.1 (Reduced Monoid/Trivial Extension)**.** Let $(S, \mathfrak{S}, \mathcal{E})$, $(T, \mathfrak{T}, \mathcal{F})$ be typed monoids such that there exists a surjective morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$. We call $(S, \mathfrak{S}, \mathcal{E})$ a *reduced monoid* of $(T, \mathfrak{T}, \mathcal{F})$ and conversely $(T, \mathfrak{T}, \mathcal{F})$ a *trivial extension* of $(S, \mathfrak{S}, \mathcal{E})$.

The following lemma formalizes the idea that from a language recognition perspective, reduced monoids and trivial extensions are equivalent.

Lemma 5.2. *If $(S, \mathfrak{S}, \mathcal{E})$ is a reduced monoid of $(T, \mathfrak{T}, \mathcal{F})$, then they recognize exactly the same languages.*

Proof. The fact that every language recognized by $(S, \mathfrak{S}, \mathcal{E})$ is also recognized by $(T, \mathfrak{T}, \mathcal{F})$ follows from Lemma 4.6, since $(S, \mathfrak{S}, \mathcal{E})$ is a divisor of $(T, \mathfrak{T}, \mathcal{F})$. On the other hand: Let L be a language recognized by $(T, \mathfrak{T}, \mathcal{F})$ via the typed morphism $h : (\Sigma^*, L, \Sigma) \to (T, \mathfrak{T}, \mathcal{F})$. Then L is recognized by $(S, \mathfrak{S}, \mathcal{E})$ via the typed morphism $\pi \circ h$, where π denotes the surjective typed morphism from $(T, \mathfrak{T}, \mathcal{F})$ to $(S, \mathfrak{S}, \mathcal{E})$. □

Similar to the case of the syntactic monoid, we can show that to each typed monoid there exists a unique minimal reduced monoid. This can be constructed - as the syntactic monoid - as a quotient monoid: Given a typed monoid $(S, \mathfrak{S}, \mathcal{E})$, define the relation $\equiv_{(S,\mathfrak{S},\mathcal{E})}$ by letting $x \equiv y$ $(x, y \in S)$ iff for all $z, z' \in S$ for all types $\mathfrak{s} \in \mathfrak{S}$ we have $zxz' \in \mathfrak{s} \Leftrightarrow zyz' \in \mathfrak{s}$. If we view a language as typed monoid with two nontrivial types, the definition above coincides with the definition of the syntactic congruence. It is easy to verify that the definition above forms a type preserving congruence. This allows us to define the minimal reduced monoid:

Definition 5.3 (Minimal Reduced Monoid)**.** Given a typed monoid $(S, \mathfrak{S}, \mathcal{E})$, *the minimal reduced monoid* is defined by $\widetilde{(S, \mathfrak{S}, \mathcal{E})} = (S, \mathfrak{S}, \mathcal{E})/\equiv_{(S,\mathfrak{S},\mathcal{E})}$.

Lemma 5.4. *Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid and $(T, \mathfrak{T}, \mathcal{F})$ be a reduced monoid of $(S, \mathfrak{S}, \mathcal{E})$ then $\widetilde{(S, \mathfrak{S}, \mathcal{E})}$ is a reduced monoid of $(T, \mathfrak{T}, \mathcal{F})$.*

Proof. Let $h : (S, \mathfrak{S}, \mathcal{E}) \to (T, \mathfrak{T}, \mathcal{F})$ be a surjective morphism. Since $h_S(s_1) = h_S(s_2)$ implies that $s_1 \equiv_{(S,\mathfrak{S},\mathcal{E})} s_2$ the mapping $T \to \widetilde{(S, \mathfrak{S}, \mathcal{E})}$ which maps every t to the congruence class of an inverse image of t is well defined and gives the desired surjective morphism. □

A standard operation on monoids is the direct product which corresponds to the Boolean closure on the language side. We will get the same equivalence in the typed world. The definition of the direct product of two typed monoids is straightforward and sound in the category theory sense.

Definition 5.5 (Direct Product)**.** The *direct product* of two monoids $(S, \mathfrak{S}, \mathcal{E})$, $(S', \mathfrak{S}', \mathcal{E}')$, denoted by $(S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}')$, is defined as $(S \times S', \mathfrak{S} \times \mathfrak{S}', \mathcal{E} \times \mathcal{E}')$.

The direct product for typed monoids can express Boolean operations on the language side as in the case of conventional monoids:

Lemma 5.6. *Let $L_1, L_2 \subseteq \Sigma^*$ be languages recognized by typed monoids $(S, \mathfrak{S}, \mathcal{E})$ and $(S', \mathfrak{S}', \mathcal{E}')$ respectively. Then $L_1 \cap L_2$ and $L_1 \cup L_2$ are recognized by $(S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}')$.*

Proof. Let $h_1 := \pi_1(h) : \Sigma^* \to S$ and $h_2 = \pi_1(h') : \Sigma^* \to S'$ where h, h' denote the recognizing morphisms for L_1, L_2 respectively.

Define $\tilde{h} : \Sigma^* \to S \times S'$ by $\tilde{h}(\sigma) = (h_1(\sigma), h_2(\sigma))$. Then $L_1 \cap L_2 = \tilde{h}^{-1}(h_1(L_1) \times h_2(L_2))$ and $L_1 \cup L_2 = \tilde{h}^{-1}((h_1(L_1) \times S') \cup (S \times h_2(L_2)))$. By the definition of the direct product of typed monoids $h_1(L_1) \times h_2(L_2)$ and $(h_1(L_1) \times S') \cup (S \times h_2(L_2))$ are types of $(S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}')$, so $\tilde{h}$ defines typed morphisms $(\Sigma^*, L_1 \cap L_2, \Sigma) \to (S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}'))$ and $(\Sigma^*, L_1 \cup L_2, \Sigma) \to (S, \mathfrak{S}, \mathcal{E}) \times (S', \mathfrak{S}', \mathcal{E}'))$, and the assertion follows. □

Motivated by the definition of a weakly closed class of transformation semigroups ([Eil76, Chapter III]) we define:

Definition 5.7 (Weakly closed class of languages)**.** A *weakly closed class* of languages is a function $\mathcal{V}$ which associates to each alphabet A a nonempty set $A^*\mathcal{V}$ of languages over A such that

1. $A^*\mathcal{V}$ is closed under Boolean combinations, and
2. $\varphi^{-1}(L) \in B^*\mathcal{V}$ for every $L \in A^*\mathcal{V}$ and every length preserving morphisms $\varphi : B^* \to A^*$.

Many sets of languages defined in other contexts, e.g. descriptive complexity or circuit complexity, do not form varieties (see Section 6) but weakly closed classes.

Example 5.8. The languages described by the logic class FO[<, mod] form a weakly closed class. One can easily verify that $\mathcal{L}$(FO[<, mod]) is closed under length preserving morphisms and it is closed under Boolean operations. But since FO[<, mod] cannot recognize the language L_{parity}, it is not closed under non length preserving morphisms.

We will now define sets of typed monoids:

Definition 5.9 (Weakly Closed Class)**.** A *weakly closed class* of typed monoids is a nonempty set of typed monoids that is closed under trivial extensions, division and finite direct products.

Given a nonempty set $\mathbf{V}$ of typed monoids, we let $\mathfrak{L}(\mathbf{V})$ be a mapping which associates with every alphabet A the nonempty set of all languages over A that can be recognized by a typed monoid of $\mathbf{V}$.

Obviously we have

Lemma 5.10. *Let* $\mathbf{V}$, $\mathbf{W}$ *be sets of typed monoids.*

(a) If $\mathbf{V}$ *is closed under division, then* $A^*\mathfrak{L}(\mathbf{V})$ *is the set of all languages* $L \subseteq A^*$ *with* $syn(L) \in \mathbf{V}$.

(b) For two classes $\mathbf{V} \subseteq \mathbf{W}$ *we have* $A^*\mathfrak{L}(\mathbf{V}) \subseteq A^*\mathfrak{L}(\mathbf{W})$ *for every alphabet* A.

Our aim is a correspondence between weakly closed classes of languages and weakly closed classes of typed monoids, where the correspondence is given by the function $\mathfrak{L}$.

Proposition 5.11. *If* $\mathbf{V}$ *is a weakly closed class of typed monoids, then* $\mathfrak{L}(\mathbf{V})$ *is a weakly closed class of languages.*

Proof. We have to show that $\mathcal{V} = \mathfrak{L}(\mathbf{V})$ forms a weakly closed class of languages. We first show $\mathcal{V}$ to fulfill the closure under inverse length preserving morphisms: Let $L \subseteq \Sigma^*$ be a language in $\Sigma^*\mathcal{V}$ and $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid recognizing L via the typed morphism h. Assume that $L' \subseteq \Pi^*$ is a language such that $L' = \varphi^{-1}(L)$ where $\varphi : \Pi^* \to \Sigma^*$ is a length preserving morphism. Since φ is length preserving it can be seen as typed morphism from (Π^*, L', Π) to (Σ^*, L, Σ), thus $h \circ \varphi$ is a typed morphism from (Π^*, L', Π) to $(S, \mathfrak{S}, \mathcal{E})$, and therefore $L' \in \Pi^*\mathcal{V}$.

The other closure properties follow with Lemma 5.6. □

The next proposition ensures that every weakly closed class of languages can be characterized by a weakly closed class of typed monoids.

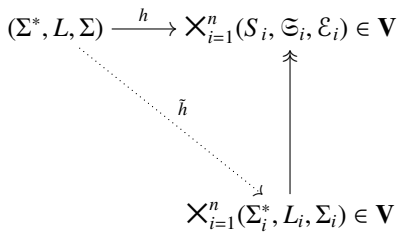
Proposition 5.12. *If* $\mathcal{V}$ *is a weakly closed class of languages, then there is a weakly closed class of typed monoids* $\mathbf{V}$ *with* $\mathfrak{L}(\mathbf{V}) = \mathcal{V}$.

Proof. Let $\mathbf{V}$ be the smallest weakly closed class that contains all syntactic monoids of $\mathcal{V}$. We have to show that $\mathfrak{L}(\mathbf{V}) \subseteq \mathcal{V}$, i.e. if $L \in \Sigma^*\mathfrak{L}(\mathbf{V})$ then $L \in \Sigma^*\mathcal{V}$. The other inclusion is obvious.

The outline of the proof is as follows: We start with a language $L \in \Sigma^*\mathfrak{L}(\mathbf{V})$ and want to show that it is also in $\Sigma^*\mathcal{V}$. We do this by constructing a language $L' \in \Pi^*$ as a Boolean combination of languages $L_i \in \Sigma_i^*\mathcal{V}$, where the L_i raise immediately from the typed monoid recognizing L, and constructing a length preserving morphism $\varphi : \Sigma^* \to \Pi^*$, such that $L = \varphi^{-1}(L')$.

L is recognized by a monoid in $\mathbf{V}$. We may assume that L is recognized via a typed morphism $h : (\Sigma^*, L, \Sigma) \to \bigtimes_{i=1}^{n}(S_i, \mathfrak{S}_i, \mathcal{E}_i)$, where $(S_i, \mathfrak{S}_i, \mathcal{E}_i)$ are syntactic monoids of some languages $L_i \in \Sigma_i^*\mathcal{V}$, in particular $(S_i, \mathfrak{S}_i, \mathcal{E}_i) \in \mathbf{V}$ (we can ignore the closure under division and trivial extension by Lemma 5.2). Further, the languages $L_i \subseteq \Sigma_i^*$ are recognized via surjective morphisms $\eta_i : (\Sigma_i^*, L_i, \Sigma_i) \to (S_i, \mathfrak{S}_i, \mathcal{E}_i)$.

We now construct L' and φ. The following diagram depicts the situation:



The typed monoids $(\Sigma_i^*, L_i, \Sigma_i)$ are trivial extensions of $(S_i, \mathfrak{S}_i, \mathcal{E}_i)$ and therefore exists a typed morphism $\tilde{h} : (\Sigma^*, L, \Sigma) \to \bigtimes_{i=1}^n (\Sigma_i^*, L_i, \Sigma_i)$. So $L = \tilde{h}^{-1}(\mathfrak{S})$ for some type $\mathfrak{S} = \bigtimes_{i=1}^n \mathfrak{S}_i$, where $\mathfrak{S}_i \in \{\emptyset, \Sigma_i^*, L_i, \Sigma_i^* \setminus L_i\}$. Note that $\tilde{h}_{\Sigma^*}(\Sigma^*) \subseteq (\bigtimes_{i=1}^n \Sigma_i)^*$ (since $\tilde{h}_\Sigma(\Sigma) \subseteq \bigtimes_{i=1}^n \Sigma_i$ and by the compatibility conditions of typed morphisms), thus $\tilde{h}_{\Sigma^*} : \Sigma^* \to (\bigtimes_{i=1}^n \Sigma_i)^*$ is a length preserving morphism. The assertion follows by setting $\Pi = (\bigtimes_{i=1}^n \Sigma_i)$, $L' = \mathfrak{S}$ and $\varphi = \tilde{h}_{\Sigma^*}$.

□

Thus, by Proposition 5.11 and Proposition 5.12 we get:

Theorem 5.13. *Let* $\mathbf{V}$ *be a weakly closed class of typed monoids, then* $\mathcal{L}(\mathbf{V})$ *is a weakly closed class of languages.*

Moreover, if $\mathcal{V}$ *is a weakly closed class of languages, then there is a weakly closed class* $\mathbf{V}$ *of typed monoids such that* $\mathcal{L}(\mathbf{V}) = \mathcal{V}$.

Note that this theorem does not guarantee a 1-1 correspondence: for a given weakly closed class of languages, there could be multiple weakly closed classes of typed monoids.

6 Varieties

In this section we prove our analogon to Eilenberg's theorem. Our notion of a language variety is the same that is found as $*$-variety in the literature [Eil76, Pin86].

The right quotient of a language $L \subseteq \Sigma^*$ by $w \in \Sigma^*$ is defined by $Lw^{-1} = \{x \in \Sigma^* \mid xw \in L\}$. The left quotient $w^{-1}L$ is defined analogously. A *variety* of languages is a weakly closed class $\mathcal{V}$ of languages such that for all alphabets A and B holds:

1. If $L \in A^*\mathcal{V}$, then $a^{-1}L, La^{-1} \in A^*\mathcal{V}$ for all $a \in A$ and $\varphi^{-1}(L) \in B^*\mathcal{V}$ where $\varphi : B^* \to A^*$ is a morphism.

To adapt this concept to the theory of typed monoids, we need the notion of shifts and unit relaxations. The following definition models the fact that varieties are closed under left and right quotients.

Definition 6.1 (Shifting)**.** Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid. Then $(S, \mathfrak{S}', \mathcal{E})$ is a *shift* of $(S, \mathfrak{S}, \mathcal{E})$, if there are $\lambda, \varrho \in S$ with $\mathfrak{S}' = \{\lambda^{-1}\mathfrak{S}\varrho^{-1} \mid \mathfrak{S} \in \mathfrak{S}\}$, where $\{\lambda^{-1}\mathfrak{S}\varrho^{-1}\} = \{s \in S \mid \lambda s \varrho \in \mathfrak{S}\}$.

The use of units let typed morphism correspond to length preserving morphisms for languages. In order to a notion for non-length-preserving morphism we allow our units to change:

Definition 6.2 (Unit Relaxation)**.** Let $(S, \mathfrak{S}, \mathcal{E})$ be a typed monoid. Then for any finite set $\mathcal{E}' \subseteq S$ we say $(S, \mathfrak{S}, \mathcal{E}')$ is a *unit relaxation* of $(S, \mathfrak{S}, \mathcal{E})$.

Adding these two closure properties to the requirement of a weakly closed class of typed monoids we obtain a variety of typed monoids.

Definition 6.3 (Variety of Typed Monoids)**.** A *variety* of typed monoids is a weakly closed class that is closed under shifting and unit relaxation.

Note that a variety of finite monoids in the sense of [Pin86] does not form a variety of typed monoids if we consider every finite monoid S as the typed monoid $(S, \mathcal{P}(S), S)$, since it is not closed under trivial extension leading to infinite typed monoids.

Proposition 6.4. *If* $\mathbf{V}$ *is a variety of typed monoids, then* $\mathfrak{L}(\mathbf{V})$ *is a variety of languages.*

Proof. Let $\mathcal{V} = \mathfrak{L}(\mathbf{V})$. By Proposition 5.11 it remains to show that $\mathcal{V}$ is closed under quotients and under inverse morphism. The closure under quotients is obviously given, since $\mathbf{V}$ is closed under shifting.

Assume that $\varphi : \Pi^* \to \Sigma^*$ is a morphism and $L \subseteq \Sigma^*$ is a language recognized by a typed monoid $(S, \mathfrak{S}, \mathcal{E}) \in \mathbf{V}$ (thus there is a typed morphism $h : (\Sigma^*, L, \Sigma) \to (S, \mathfrak{S}, \mathcal{E})$). We need to show that $L' = \varphi^{-1}(L)$ is also recognized by a monoid in $\mathbf{V}$. But this follows by unit relaxation, since we can consider φ as typed morphism from (Π^*, L', Π) to $(\Sigma^*, L, \varphi(\Pi))$ and thus $h \circ \varphi : (\Pi^*, L', \Pi) \to (S, \mathfrak{S}, h(\varphi(\Pi)))$ is a typed morphism to a monoid in $\mathbf{V}$. □

Proposition 6.5. *For two varieties* $\mathbf{V} \subseteq \mathbf{W}$ *we have* $\mathfrak{L}(\mathbf{V}) \subseteq \mathfrak{L}(\mathbf{W})$*, where equality occurs only if* $\mathbf{V} = \mathbf{W}$.

Proof. Let $\mathfrak{L}(\mathbf{V}) = \mathcal{V}$ and $\mathfrak{L}(\mathbf{W}) = \mathcal{W}$. By Lemma 5.10 we have $\mathcal{V} \subseteq \mathcal{W}$, so we need to show the equality statement for varieties.

Let $(S, \mathfrak{S}, \mathcal{E})$ be a monoid in $\mathbf{W}$ and assume $\mathcal{V} = \mathcal{W}$, we show $(S, \mathfrak{S}, \mathcal{E})$ is in $\mathbf{V}$. We denote the types in $\mathfrak{S}$ by $\mathfrak{s}_i$. Let G be a generating set of S containing $\mathcal{E}$, then $(S, \mathfrak{s}_i, G)$ is a typed monoid in $\mathbf{W}$ for all i. The set G generates S, thus there is a language $L_i \subseteq G^*$ recognized by $(S, \mathfrak{s}_i, G)$, moreover $\mathrm{syn}(L_i) \cong \widetilde{(S, \mathfrak{s}_i, G)} \in \mathbf{W}$. Thus L_i is in $G^*\mathcal{W} = G^*\mathcal{V}$ and consequently $\widetilde{(S, \mathfrak{s}_i, G)}$ and therefore $(S, \mathfrak{s}_i, G) \in \mathbf{V}$ for all i. Since $(S, \mathfrak{S}, \mathcal{E})$ divides $\bigtimes_i (S, \mathfrak{s}_i, G)$ we conclude $(S, \mathfrak{S}, \mathcal{E}) \in \mathbf{V}$ and hence $\mathbf{V} = \mathbf{W}$. □

Proposition 6.6. *For every variety of languages* $\mathcal{V}$ *there is a corresponding variety of typed monoids* $\mathbf{V}$*, such that* $\mathcal{V} = \mathfrak{L}(\mathbf{V})$.

Proof. The proof is similar to the proof of Proposition 5.12. We let $\mathbf{V}$ be the smallest variety that contains all syntactic monoids of $\mathcal{V}$. We need to show that $L \in \Sigma^*\mathfrak{L}(\mathbf{V})$ implies $L \in \Sigma^*\mathcal{V}$. We construct a language $L' \in \Pi^*\mathcal{V}$ and a morphism $\varphi : \Sigma^* \to \Pi^*$ such that $L = \varphi^{-1}(L')$.

L is recognized by a monoid in $\mathbf{V}$. We may assume that L is recognized by $\bigtimes(S_i, \mathfrak{S}_i, \widetilde{\mathcal{E}_i})$ via some morphism $h : (\Sigma^*, L, \Sigma) \to \bigtimes(S_i, \mathfrak{S}_i, \widetilde{\mathcal{E}_i})$, where $(S_i, \mathfrak{S}_i, \mathcal{E}_i)$ are syntactic monoids of some languages $L_i \in \Sigma_i^*\mathcal{V}$ and $\widetilde{\mathcal{E}_i}$ are arbitrary finite subsets of S_i. Further, the languages $L_i \subseteq \Sigma_i^*$ are recognized via surjective morphisms $\eta_i : (\Sigma_i^*, L_i, \Sigma_i) \to (S_i, \mathfrak{S}_i, \mathcal{E}_i)$. Note that we may ignore the closure under shifting since $\mathcal{V}$ is closed under quotients.

The typed monoids $(\Sigma_i^*, L_i, \eta_i^{-1}(\widetilde{\mathcal{E}_i}))$ are trivial extensions of $(S_i, \mathfrak{S}_i, \widetilde{\mathcal{E}_i})$ and therefore exists a typed morphism $\tilde{h} : (\Sigma^*, L, \Sigma) \to \bigtimes_{i=1}^n (\Sigma_i^*, L_i, \eta_i^{-1}(\widetilde{\mathcal{E}_i}))$. So $L = \tilde{h}^{-1}(\mathfrak{S})$ for some type $\mathfrak{S} = \bigtimes_{i=1}^n \mathfrak{S}_i$, where $\mathfrak{S}_i \in \{\emptyset, \Sigma_i^*, L_i, \Sigma_i^* \setminus L_i\}$.

In contrast to the proof of Proposition 5.12 we can not immediately conclude that $\tilde{h}_{\Sigma^*}(\Sigma^*) \subseteq (\bigtimes \Sigma_i)^*$, since every entry in a tuple $\tilde{h}_{\Sigma^*}(w) \in (\bigtimes \Sigma_i^*)$ could have different length. Nevertheless, since we have the closure under unit relaxation we may assume that every language L_i has a neutral letter, i.e. a letter that gets mapped to the neutral elements by the morphism η_i. Thus we can identify $\tilde{h}_{\Sigma^*}(w) = (w_1, \dots w_n)$ with a word $(v_1, \dots, v_n) \in (\bigtimes \Sigma_i)^*$ where v_i is w_i padded with the neutral letter.

Now the assertion follows with φ is the morphism induced by $\tilde{h}_{\Sigma^*}$, $\Pi = (\bigtimes_{i=1}^n \Sigma_i)$ and $L' = \mathfrak{S}$. □

Summing up these results we obtain a one-to-one correspondence for varieties as in the finite case:

Theorem 6.7. *Varieties of typed monoids and varieties of languages are in a one-to-one correspondence:*

- *Let $\mathcal{V}$ be a variety of languages and $\mathbf{V}$ the smallest variety of typed monoids that recognizes all languages in $\mathcal{V}$, then $\mathcal{L}(\mathbf{V}) = \mathcal{V}$.*
- *Let $\mathbf{V}$ be a variety of typed monoids and $\mathbf{W}$ be the smallest variety that recognizes all languages of $\mathcal{L}(\mathbf{V})$, then $\mathbf{V} = \mathbf{W}$.*

As shown above, a variety of typed monoids contains always infinite typed monoids because of the closure under trivial extensions. But without this closure property there would be no one-to-one correspondence in the previous theorem. It is easy to construct a typed monoid $(S, \mathfrak{S}, \mathcal{E})$ such that any language recognized by $(S, \mathfrak{S}, \mathcal{E})$ is aperiodic but S contains a group.

7 Discussion

We have introduced typed monoids and shown that they can be used to describe languages, weakly closed classes, and varieties of languages. Typed monoids allow us to obtain a more precise description of language classes than with the usual approach with monoids or as mg-pairs or stamps as in [ÉL03] and [PS05].

In this paper we presented basic results about typed monoids. We have shown the existence and uniqueness of a typed syntactic monoid and the equivalent of Eilenberg's theorem for typed monoids as well as a weaker version for weakly closed classes. A number of questions are open when studying typed monoids. Green's relation are

a useful tool in the study of monoids. Many important varieties within the regular languages can be defined via some properties of the Green's relations of the monoids involved. This opens the question whether there is a useful notion of similar relations for typed monoids.

It usually makes a difference if one considers language recognition via monoids or semigroups. In the latter case languages are subsets of Σ^+ instead of Σ^* and the term +-varieties is used. The difference between these two approaches diminishes for typed monoids. The empty word must be mapped onto the neutral element of the typed monoid. But if the neutral element is not contained in the units, the morphism on $\Sigma^* \setminus \{\lambda\} = \Sigma^+$ behaves like a semigroup morphism. Conversely, if one considers a language $L \subseteq \Sigma^+$ recognized by a semigroup S with accepting set A, this language is recognized by the typed monoid $(S^1, A, \eta_L(\Sigma))$. Here, S^1 denotes S with an additional neutral element added. If for example a language L without a neutral letter is recognized by a semigroup S then $(S^1, A, \eta_L(\Sigma))$ recognizes L but not L with an additional neutral letter.

The motivation to study typed monoids stems from an interest in algebraic characterizations of logic and circuit classes. The program linking finite monoids [GL84, Imm87, BST90, BIS90, BCST92] to first order logic relied heavily on the block product [RT89] (or the bilateral semidirect product). It is possible to define the block product for typed monoids and, for example, obtain characterizations for first order logic with the majority quantifier [KLR07] as well as subclasses [BKM07]. To do so it was essential to limit the set of possible accepting subsets. The limitation of the acceptance set (types) was used in [Sak76] to obtain monoids for context free languages. By using a Boolean algebra we loose the ability to characterize language classes that are not closed under Boolean operations. We choose a Boolean algebra because circuit classes are usually closed under Boolean operations. Even more, the use of a Boolean algebra lead to a clean definition of a block product (or bilateral semidirect product) and eases the characterizations of logic classes with (arbitrary) Lindström quantifiers.

The approach of ordered monoids ([Pin95]) allows to examine language classes not closed under complement. A possible research area is to introduce "positive" and "negative" types, closed under union to characterize context free languages and one might be even able to define a block product for these kind of objects.

References

[Ajt89] Miklós Ajtai. First-order definability on finite structures. *Ann. Pure Appl. Logic*, 45(3):211–225, 1989.

[Alm95] Jorge Almeida. *Finite Semigroups and Universal Algebra*. World Scientific, Singapore, 1995.

[BCST92] David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. Regular languages in NC^1. *J. Comput. Syst. Sci.*, 44(3):478–499, 1992.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

[BKM07] Christoph Behle, Andreas Krebs, and Mark Mercer. Linear circuits, two-variable logic and weakly blocked monoids. In *MFCS*, pages 147–158, 2007.

[BST90] David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Inf. Comput.*, 89(2):109–132, 1990.

[Eil76] Samuel Eilenberg. *Automata, Languages and Machines, Vol. A+B*. Academic Press, 1976.

[ÉL03] Zoltán Ésik and Kim Guldstrand Larsen. Regular languages definable by lindström quantifiers. *ITA*, 37(3):179–241, 2003.

[FSS81] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *FOCS*, pages 260–270, 1981.

[GL84] Yuri Gurevich and Harry R. Lewis. A logic for constant-depth circuits. *Information and Control*, 61(1):65–74, 1984.

[Imm87] Neil Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987.

[KLR07] Andreas Krebs, Klaus-Jörn Lange, and Stephanie Reifferscheid. Characterizing TC^0 in terms of infinite groups. *Theory Comput. Syst.*, 40(4):303–325, 2007.

[Kre08] Andreas Krebs. *Typed Semigroups, Majority Logic, and Threshold Circuits.* PhD thesis, Universität Tübingen, 2008.

[MP71] Robert McNaughton and Seymour Papert. *Counter-free automata. With an appendix by William Henneman.* Research Monograph No.65. Cambridge, Massachusetts, and London, England: The M. I. T. Press. XIX, 163 p., 1971.

[Pin86] Jean-Eric Pin. *Varieties of formal languages*. Plenum, London, 1986.

[Pin95] Jean-Eric Pin. A variety theorem without complementation. 39:80–90, 1995. English version: Russian Mathem. (Iz. VUZ) 39 (1995),74-83.

[PS05] Jean-Eric Pin and Howard Straubing. Some results on C-varieties. *ITA*, 39(1):239–262, 2005.

[RT89] John L. Rhodes and Bret Tilson. The kernel of monoid morphisms. *J. Pure Applied Alg.*, 62:27–268, 1989.

[Sak76] Jacques Sakarovitch. An algebraic framework for the study of the syntactic monoids application to the group languages. In *MFCS*, pages 510–516, 1976.

[Sch65] Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[Str94] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, 1994.

[TT07] Pascal Tesson and Denis Thérien. Logic meets algebra: the case of regular languages. *Logical Methods in Computer Science*, 3(1), 2007.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de