Conservative Groupoids Recognize Only Regular Languages

Martin Beaudry [a], Danny Dubé [b], Maxime Dubé [b], Mario Latendresse [c], Pascal Tesson [b,*]

[a] *Département d'informatique, Université de Sherbrooke*
2500 Bld. de l'Université, Sherbrooke (Québec) J1K 2R1, Canada
[b] *Département d'informatique et de génie logiciel, Université Laval*
1065 av. de la Médecine, Québec (Québec) G1V 0A6, Canada
[c] *SRI International*
333 Ravenswood Ave, Menlo Park, CA 94025, USA

Abstract

The notion of recognition of a language by a finite semigroup can be generalized to recognition by finite groupoids, i.e. sets equipped with a binary operation '$\cdot$' which is not necessarily associative. It is well known that L can be recognized by a groupoid iff L is context-free. However it is also known that some subclasses of groupoids can only recognize regular languages.

A groupoid H is said to be *conservative* if $a \cdot b \in \{a, b\}$ for all $a, b \in H$. The first result of this paper is that conservative groupoids can only recognize regular languages. This class of groupoids is incomparable with the ones identified so far which share this property, so we are exhibiting a new way in which a groupoid can be too weak to recognize non-regular languages.

We also study the class $\mathcal{L}_{cons}$ of regular languages that can be recognized in this way and explain how it fits within the well-known Straubing-Thérien hierarchy. In particular we show that $\mathcal{L}_{cons}$ contains depth 1/2 of the hierarchy and is entirely contained in depth 3/2.

Keywords: groupoid, conservative algebra, algebraic automata theory

1. Introduction

A semigroup S is a set with a binary associative operation. It is a monoid if it also has an identity element. The algebraic point of view on automata, which is central to some of the most important results in the study of regular languages, relies on viewing a finite semigroup as a language recognizer. This makes it possible to classify a regular language according to the semigroups or monoids able to recognize it. There are various ways in which to formalize this idea but the following one will be useful in our context: a

*Corresponding author
Email addresses: `martin.beaudry@usherbrooke.ca` (Martin Beaudry), `danny.dube@ift.ulaval.ca` (Danny Dubé), `maxime.dube@ift.ulaval.ca` (Maxime Dubé), `latendresse@sri.com` (Mario Latendresse), `pascal.tesson@ift.ulaval.ca` (Pascal Tesson)

language $L \subseteq A^*$ is recognized by a finite monoid M if there is a homomorphism h from the free monoid A^* to the free monoid M^* and a set $F \subseteq M$ such that $w \in L$ iff $h(w)$ is a sequence of elements whose product lies in F. Since the operation of M is associative, this product is well defined. This framework underlies algebraic characterizations of many important classes of regular languages (see [13] for a survey).

These ideas have been extended to non-associative binary algebras, i.e. groupoids. If a groupoid is non-associative, a string of groupoid elements does not have a well-defined product so the above notion of language recognition must be tweaked. A language L is said to be recognized by a finite groupoid H if there is a homomorphism h from the free monoid A^* to the free monoid H^* and a set $F \subseteq H$ such that $w \in L$ iff the sequence of groupoid elements $h(w)$ can be bracketed[1] so that the resulting product lies in F. It has been shown that a language can be recognized by a finite groupoid iff it is context-free [9, 10, 18].

However, certain groupoids are too weak to recognize non-regular languages. The first non-trivial example was provided by Caussinus and Lemieux who showed that loops (groupoids with an identity element and left/right inverses) can only recognize regular languages [6]. Beaudry et al. later showed that the languages recognized by loops are precisely the regular open languages [3]. Along the same lines, Beaudry showed in [2] that if H is a groupoid whose multiplication monoid $\mathcal{M}(H)$ is in the variety **DA** then it can only recognize regular languages. ($\mathcal{M}(H)$ is the transformation monoid generated by the rows and columns of the multiplication table of H.) Finally, Beaudry et al. proved that this still holds if the multiplication monoid lies in the larger variety **DO** [4].

A groupoid H with operation '$\cdot$' is said to be *conservative* if $a \cdot b \in \{a, b\}$ for all $a, b \in H$. The simplest example of a non-associative and conservative groupoid is the one defined by the Rock-Paper-Scissors game. In this game, two players simultaneously make a sign with their fingers chosen among Rock, Paper, and Scissors. Rock beats Scissors, Scissors beats Paper, Paper beats Rock, and identical signs result in a tie. The associated groupoid has three elements r, p, s and the multiplication is given below (explicit list of products on the left, multiplication table on the right).

$$r \cdot r = r \cdot s = s \cdot r = r$$
$$p \cdot p = p \cdot r = r \cdot p = p$$
$$s \cdot s = s \cdot p = p \cdot s = s$$

	r	**p**	**s**
r	r	p	r
p	p	p	s
s	r	s	s

Note that H is indeed conservative and non-associative since

$$(r \cdot p) \cdot s = s \neq r = r \cdot (p \cdot s).$$

The main result of our paper is that conservative groupoids recognize only regular languages and in fact a very restricted class of regular languages. Our results are incomparable to those of Beaudry et al. since a straightforward calculation[2] shows that the

[1]If $h(w)$ is the empty word then it cannot be bracketed. To handle this exception, we consider that the result of the evaluation of ϵ is a special element $\eta \notin H$ which may or may not be a part of F. This technical issue is without incidence for the main results of this paper.

[2]This calculation is somewhat tangential to our results but we include it here for completeness. Let

multiplication monoid of the Rock-Paper-Scissors groupoid does not belong to the variety **DO** nor to the larger variety **DS**.

Conservative groupoids have already been studied in the literature (e.g. [7]) and anticommutative conservative groupoids were used as a model for undirected graphs in [19]. More generally, conservative algebras have also found applications in theoretical computer science. Most notably, they occur naturally in the study of the complexity of the list-homomorphism problem [1, 5].

1.1. Conservative Groupoids and Tournaments

It is convenient to think of conservative groupoids as defining a generalization of the Rock-Paper-Scissors game. For any conservative groupoid H, we define the game in which players 1 and 2 each choose an element of H (say a and b respectively) and player 1 wins iff $a \cdot b = a$. In fact, it is helpful to think of this game as a competition between elements of H.

Consider now a sequence $w \in H^*$ of elements of the groupoid. A bracketing of this sequence can be viewed as specifying a tournament structure involving the symbols of w, i.e. a specific way to determine a winner among the elements of w. For instance, if $w = abcd$, then $(a \cdot b) \cdot (c \cdot d)$ is the tournament that first pits a against b and c against d and then has the two winners of that first round competing. Similarly in the tournament $((a \cdot b) \cdot c) \cdot d$ we first have a facing b with the winner then facing c and the winner of that facing d. Note that this analogy makes sense because H is conservative and the "winner" of any such tournament (i.e. the value of the product given this bracketing) is indeed one of the participants (in the above example, one of a, b, c, or d). We intend to study languages of the form $\Lambda(a) = \{w \in H^* \mid w \text{ can be bracketed to give } a\}$ and we accordingly think of them as $\Lambda(a) = \{w \in H^* \mid \text{ an organizer can rig a tournament structure for } w \text{ to ensure that } a \text{ wins}\}$.

Let us define *contest trees*. We denote the set of all contest trees by $\mathcal{T}$. It is the smallest set that contains the single-node tree a for any $a \in H$, and such that the tree $t_1 \otimes t_2$ is also in $\mathcal{T}$, for any two trees t_1 and t_2 in $\mathcal{T}$. Let $T : H^+ \to 2^{\mathcal{T}}$ be the function that computes the set of possible contest trees over a given word.

$$\begin{aligned} T(a) &= \{a\} \\ T(w) &= \{t_1 \otimes t_2 \mid u, v \in H^+,\ uv = w,\ t_1 \in T(u),\ t_2 \in T(v)\} \qquad \text{if } |w| > 1. \end{aligned}$$

us first formally define the multiplication monoid of a groupoid H. To each element $a \in H$ one can associate the functions $t_a, q_a : H \to H$ defined by $t_a(x) = ax$ and $q_a(x) = xa$. The set T_H of functions from H to H naturally forms a monoid under function composition. The *multiplication monoid* of H is the submonoid of T_H generated by the set $\{t_a, q_a : a \in H\}$.

An element x of a monoid is idempotent if $x^2 = x$. It is well known that for any finite monoid M, there exists a positive integer ω such that x^ω is idempotent for all $x \in M$. A finite monoid belongs to the class **DO** if it satisfies the identity $(xy)^\omega(yx)^\omega(xy)^\omega = (xy)^\omega$ and belongs to the wider class **DS** if it satisfies the weaker identity $((xy)^\omega(yx)^\omega(xy)^\omega)^\omega = (xy)^\omega$.

Let H be the Rock-Paper-Scissors groupoid. Note that because H is commutative, we have $t_a = q_a$ for each a in H. Let us represent each element t of T_H as a triple $[t(r); t(p); t(s)]$. Thus $t_r = [r; p; r]$, $t_p = [p; p; s]$ and $t_s = [r; s; s]$. Now let $x = t_s$ and $y = t_r t_p = [p; p; r]$. We have $xy = [s; s; r]$ and $(xy)^2$ is the idempotent $[r; r; s]$. Moreover $yx = [p; r; r]$ and $(yx)^2$ is the idempotent $[r; p; p]$. Finally $(xy)^2(yx)^2(xy)^2 = [r; r; r]$ which is idempotent but different from $(xy)^2$. Therefore the multiplication monoid of H violates the defining identity of **DS**.

Note that, when performing the left-to-right traversal of a tree in $T(w)$, the leaves that we successively reach are the symbols that form w. Next, function $W : \mathcal{T} \to H$ computes the *winner* of a contest tree.

$$\begin{aligned} W(a) &= a \\ W(t_1 \otimes t_2) &= W(t_1) \cdot W(t_2) \end{aligned}$$

Note that the winner of a contest tree is unique. Next, we define the set of possible winners in a given contest w by overloading function W with an additional definition of type $H^+ \to 2^H$. We define $W(w)$ as $\{W(t) \mid t \in T(w)\}$. Finally as defined earlier, we denote by $\Lambda(a)$ the language of the words for which we can arrange a contest in which a is the winner, i.e. $\Lambda(a)$ as $\{w \in H^+ \mid a \in W(w)\}$. When drawing contest trees, we often label interior nodes with the winner of that subtree (see Figure 1).

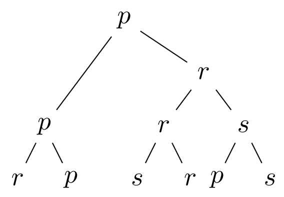

Figure 1: A contest tree on $rpsrps$ over the Rock-Paper-Scissors groupoid.

It is convenient to further abuse the above terminology and notation as follows. Let w be a word and let t be a contest tree in $T(w)$. We say that t is a *bracketing* of w and write $t(w)$ to denote the winner of the contest tree t. For instance if H is the Rock-Paper-Scissors groupoid then for $w = rpsps$ and for $t = r((ps)(ps))$, we obtain $t(w) = r$. This notation and terminology is particularly convenient because of the following observation.

Remark 1. *For any $t \in T(w)$ and any $x, y \in H^*$ we have $W(xt(w)y) \subseteq W(xwy)$. Indeed the right-hand side is the set of elements that can win under some bracketing of xwy whereas the left-hand side represents the possible winners in the special case where the segment w is bracketed according to t.*

1.2. The Straubing-Thérien Hierarchy

The *Straubing-Thérien hierarchy* consists of classes of regular languages and is one of the best-known examples of a so-called concatenation hierarchy (see e.g. [13]). A language $L \subseteq A^*$ is in depth 0 of the hierarchy if it is either A^* or $\emptyset$ and it is of depth $1/2$ if it is a union of languages of the form $A^*a_1A^*a_2A^* \dots a_kA^*$ with each $a_i \in A$. For $n \geq 1$ the rest of the hierarchy is defined inductively as follows: the language L is of depth n if it is a Boolean combination of languages of depth $n - 1/2$ and is of depth $n + 1/2$ if it is a union of languages of the form $L_0a_1L_1a_2 \dots a_kL_k$ with $a_i \in A$ and L_i of depth n. It is clear from the definition that the union of the classes in the Straubing-Thérien hierarchy is equal to the class of star-free languages, i.e. languages that can be represented by a regular expression using the union, concatenation and complement operators but without using the $*$ operator.

The Straubing-Thérien hierarchy has a nice logical interpretation [17, 11] since languages of depth $n + 1/2$ are exactly those which can be expressed by a Σ_{n+1} first-order sentence over words using the order predicate[3]. In the remainder of the paper we write Σ_1 (resp. Σ_2) instead of "languages of Straubing-Thérien depth 1/2" (resp. 3/2). We also denote as Π_1 (resp. Π_2) the class of languages whose complement lies in Σ_1 (resp. Σ_2). The following is a useful combinatorial characterization of Σ_2: a language is Σ_2 iff it is a finite union of languages of the form $A_0^* a_1 A_1^* a_2 \dots A_{k-1}^* a_k A_k^*$ where each $A_i \subseteq A$ [14].

The paper is organized as follows. In Section 2, we show that the languages that can be recognized by conservative groupoids are all regular and in fact lie in Σ_2. In Section 3, we discuss the class of languages recognized by conservative groupoids, its closure properties and its place in the Straubing-Thérien hierarchy.

A preliminary version of this paper appeared in the proceedings of the 6th International Conference on Language and Automata Theory and Applications (LATA 2012).

2. Main Theorem

The objective of this section is to establish our main theorem.

Theorem 2. *For any conservative groupoid H and $a \in H$, the language $\Lambda(a)$ is regular. Furthermore $\Lambda(a)$ lies in Σ_2, i.e. it can be written as a finite union of languages of the form $\sigma_0^* a_1 \sigma_1^* \dots \sigma_{k-1}^* a_k \sigma_k^*$ where the a_i lie in H and the σ_i are subsets of H.*

The demonstration proceeds in two steps. In the first step, we build a context-free grammar G that generates $\Lambda(a)$. In the second step, we analyze this grammar and show that the language it generates lies in Σ_2.

2.1. Initial Observations

We begin by establishing some further notation and auxiliary lemmas which are useful in the sequel. In particular our first objective is to provide tools which help identify the set of winners over a given string.

If H is a conservative groupoid and a, b are elements of H then we say that a is *left-favorable* to b if $ab = b$ (i.e. a loses when placed to the left of b) and that a is *right-favorable* to b if $ba = b$ (i.e. a loses when placed to the right of b). Note of course that for any $a \neq b$, it holds that a is right-favorable to b iff b is not left-favorable to a and vice-versa. We define the auxiliary functions $f_L : H \to 2^H$ and $f_R : H \to 2^H$ that, given a symbol b, return the symbols that are respectively left-favorable and right-favorable to b. Formally $f_L(b) = \{a \in H \mid a \cdot b = b\}$ and $f_R(b) = \{a \in H \mid b \cdot a = b\}$.

Let σ be a set of groupoid elements. We generalize our earlier definition of Λ by setting $\Lambda(\sigma) = \{w \in H^* \mid w$ can be bracketed to give some a in $\sigma\}$. We define $\Lambda^\epsilon(\sigma) = \Lambda(\sigma) \cup \{\epsilon\}$

[3]Details can be found in e.g. [17, 11]. A Σ_n sentence over words begins with n alternating blocks of quantifiers (starting with an existential block) that quantify over positions in the word. The quantifier-free part is built from predicates of the form $Q_a x$ (interpreted as "position x in the word holds an a") and comparisons between positions $x < y$. For instance, the language $A^*aA^*bA^*$ discussed in Section 3.2 is defined by the Σ_1 sentence $\exists x \exists y\ x < y \wedge Q_a x \wedge Q_b y$. On the other hand, the language A^*aaA^* of Proposition 20 can be defined by the Σ_2 sentence $\exists x \exists y \forall z\ x < y \wedge Q_a x \wedge Q_a y \wedge (x < z < y \to \neg Q_b z)$.

Lemma 3. *Let $a \in H$ and $u \in H^*$. Then, $a \in W(u)$ if and only if there is a factorization $u = vaw$ such that*

- *if $|v| \geq 1$, then there exists $p \in f_L(a) \cap W(v)$;*
- *if $|w| \geq 1$, then there exists $q \in f_R(a) \cap W(w)$.*

An alternative formulation of this statement is that for any $a \in H$, the language $\Lambda(a)$ is equal to the concatenation $\Lambda^\epsilon(f_L(a)) \cdot \{a\} \cdot \Lambda^\epsilon(f_R(a))$.

Proof.
($\Leftarrow$)
Suppose that $u = vaw$ and that $s \in T(v)$ and $t \in T(w)$ are such that $s(v) = p$ and $t(w) = q$ with $pa = a$ and $aq = a$. Consider the bracketing for u given by $(s(v)(at(w)))$. It evaluates to $(p(aq)) = (pa) = a$ and therefore $a \in W(u)$ as claimed.

($\Rightarrow$)
Proceed by induction on $|u|$. For the base case $|u| = 1$, note that if $a \in W(u)$, then in fact $u = a$ and we trivially obtain a factorization $u = \epsilon\, a\, \epsilon$.

For the induction step, suppose that $a \in W(u)$ and $|u| = k + 1$. Consider a contest tree t in $T(u)$ such that $t(u) = a$. Consider the left-child t_L and the right-child t_R of the root of t. Let x, y be the strings such that $u = xy$ and such that $t_L \in T(x)$ and $t_R \in T(y)$. Since $t_L(x)t_R(y) = a$ and since H is conservative, one of the following must hold:

1. $t_L(x) = a$ and $at_R(y) = a$ (i.e. $t_R(y) \in f_R(a)$);
2. $t_R(y) = a$ and $t_L(x)a = a$ (i.e. $t_L(x) \in f_L(a)$).

Assume that case 1 holds (case 2 is handled symmetrically). Since $a \in W(x)$ and since $|x| < |u|$ we know by induction that x can be factorized as $x = vaw$ such that there exist $p \in W(v) \cap f_L(a)$ (or v is empty) and $q \in W(w) \cap f_R(a)$ (or w is empty). (See Figure 2) If $w = \epsilon$ then $u = vay$ is a factorization with the properties required in the lemma's

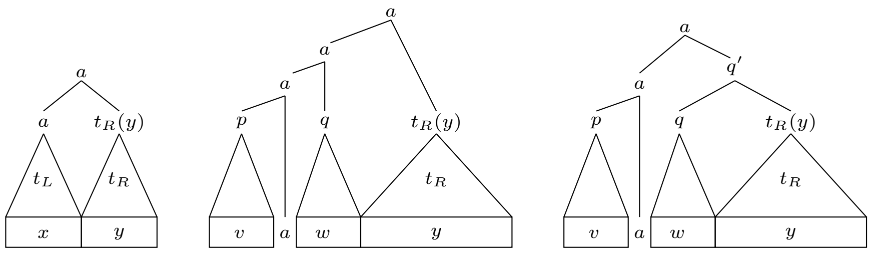

Figure 2: Steps in the proof of Lemma 3.

statement. Otherwise $u = vawy$ and we know that w and y can be bracketed to obtain q and $t_R(y)$ respectively. Since both of these elements are in $f_R(a)$, their product is also in $f_R(a)$ so there exists $q' \in W(wy) \cap f_R(a)$ and we are done. $\square$

Following the intuition behind this lemma, we say that an element $a \in H$ is *able to beat* a word $u \in H^*$ to its left (resp. to its right) if there exists $b \in W(u) \cap f_L(a)$

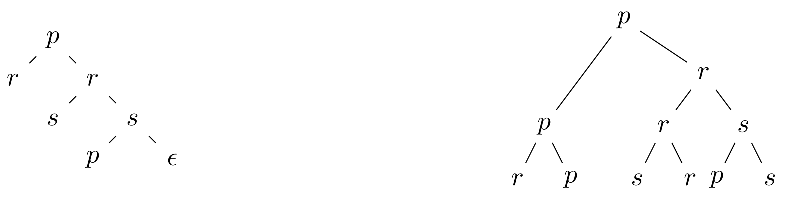

Figure 3: On the left, a favorable decomposition tree for p in the Rock-Paper-Scissors groupoid. The yield of this tree is $rpsrps$. By Lemma 3, p wins on this word. On the right, a contest tree over $rpsrps$ built from the decomposition at left.

(resp. $b \in W(u) \cap f_R(a)$). Moreover, a *favorable decomposition tree* D for $a \in H$ is a binary tree labelled by $H \cup \{\epsilon\}$ such that for all nodes a of D, if b is a left (resp. right) child of a, then $b \in f_L(a) \cup \{\epsilon\}$ (resp. $b \in f_R(a) \cup \{\epsilon\}$). The yield $\lambda(D)$ of D is an inorder walk on D (see Figure 3). By Lemma 3, if D is a favorable decomposition tree for a, then $a \in W(\lambda(D))$. On the other hand, if $a \in W(u)$, then there is a favorable decomposition tree D for a such that $u = \lambda(D)$. It is important to distinguish contest trees and favorable decomposition trees and Figure 3 gives an example of the contrast.

Remark 4. *Any subtree r of a favorable decomposition tree D is a favorable decomposition tree for its root.*

Our proof of the main theorem relies on a generalization of Lemma 3.

Lemma 5. *For any $\sigma \subseteq H$ it holds that*

$$\Lambda(\sigma) = \bigcup_{b \in \sigma} \Lambda^\epsilon(f_L(b) \cup \sigma) \cdot \{b\} \cdot \Lambda^\epsilon(f_R(b) \cup \sigma).$$

Proof. Note that when σ is a singleton, the statement is exactly Lemma 3.

Let us first show the left to right containment.

$$\begin{aligned}
\Lambda(\sigma) &= \bigcup_{b \in \sigma} \Lambda(b) \\
&\subseteq \bigcup_{b \in \sigma} \Lambda^\epsilon(f_L(b)) \cdot \{b\} \cdot \Lambda^\epsilon(f_R(b)) && \text{(by Lemma 3)} \\
&\subseteq \bigcup_{b \in \sigma} \Lambda^\epsilon(f_L(b) \cup \sigma) \cdot \{b\} \cdot \Lambda^\epsilon(f_R(b) \cup \sigma)
\end{aligned}$$

For the right to left inclusion, we need to show that for any $b \in \sigma$ we have $\Lambda^\epsilon(f_L(b) \cup \sigma) \cdot \{b\} \cdot \Lambda^\epsilon(f_R(b) \cup \sigma) \subseteq \Lambda(\sigma)$. Suppose $w = xby$ with $x \in \Lambda^\epsilon(f_L(b) \cup \sigma)$ and $y \in \Lambda^\epsilon(f_R(b) \cup \sigma)$ and assume for now that x and y are non-empty. By definition of Λ there exists some $a \in W(x)$ with $a \in f_L(b) \cup \sigma$ and some $c \in W(y)$ with $c \in f_R(b) \cup \sigma$. Since $W(abc) \subseteq W(xby)$ it suffices to show that $W(abc) \cap \sigma \neq \emptyset$. If a and c both lie in σ then $W(abc) \subseteq \sigma$ and we are done. If $a \in f_L(b)$ and $c \in \sigma$ then $(ab)c = bc \in \{b, c\} \subseteq \sigma$. Symmetrically, if $c \in f_R(b)$ and $a \in \sigma$ then $a(bc) = ab \in \{a, b\} \subseteq \sigma$. Finally if $a \in f_L(b)$ and $c \in f_R(b)$ then $(ab)c = bc = b \in \sigma$.

The case where x or y is empty can be handled just like the case where a (resp. b) lies in $f_L(b)$ (resp. $f_R(b)$). □

2.2. A Context-free Grammar for $\Lambda(a)$

We construct, for a conservative groupoid H and any $a \in H$, a context-free grammar generating $\Lambda(a)$. This can be achieved in a number of ways but the following grammar suggested by Lemma 5 is particularly useful for our purpose. Let G_H be the grammar with the non-terminals $N = \{S_a\} \cup \{B_\sigma \mid \emptyset \neq \sigma \subseteq H\}$ (with S_a as the initial non-terminal) and the production rules

$$\begin{aligned} R &= \{S_a \to B_{\sigma'}\, a\, B_{\sigma''} \mid \sigma' = f_L(a),\, \sigma'' = f_R(a)\} \\ &\cup \{B_\sigma \to B_{\sigma'}\, b\, B_{\sigma''} \mid \emptyset \neq \sigma \subseteq H,\ b \in \sigma,\ \sigma' = \sigma \cup f_L(b),\ \sigma'' = \sigma \cup f_R(b)\} \\ &\cup \{B_\sigma \to \epsilon \mid \emptyset \neq \sigma \subseteq H\}. \end{aligned}$$

Lemma 6. *Let G_H be the grammar described above. For each non-terminal B_σ it holds that $L(B_\sigma) = \Lambda^\epsilon(\sigma)$ and the language generated by G_H is $L(S_a) = \Lambda(a)$.*

PROOF. This is almost immediate from Lemma 5. Formally, we show that $L(B_\sigma) \subseteq \Lambda^\epsilon(\sigma)$ for all σ by induction on $|u|$ for a $u \in L(B_\sigma)$. If $u = \epsilon$ then by definition $u \in \Lambda^\epsilon(\sigma)$. If $|u| = k+1$ and $u \in L(B_\sigma)$ then the first production used to derive u from B_σ is of the form $B_\sigma \to B_{\sigma'}\, b\, B_{\sigma''}$ with $\sigma' = f_L(b) \cup \sigma$ and $\sigma'' = f_R(b) \cup \sigma$. Therefore $u = xby$ with $x \in L(B_{\sigma'})$ and $y \in L(B_{\sigma''})$. By induction $x \in \Lambda^\epsilon(\sigma')$ and $y \in \Lambda^\epsilon(\sigma'')$ so $u \in \Lambda^\epsilon(\sigma)$ by Lemma 5.

To show $L(B_\sigma) \supseteq \Lambda^\epsilon(\sigma)$ we again use induction. For the base case, note that $\epsilon \in L(B_\sigma)$ since G_H contains the production $B_\sigma \to \epsilon$. If $|u| = k+1$ then by Lemma 5 we have $u = xby$ with $b \in \sigma$ and $x \in \Lambda^\epsilon(\sigma')$ and $y \in \Lambda^\epsilon(\sigma'')$. By induction we get $x \in L(B_{\sigma'})$ and $y \in L(B_{\sigma''})$ and thus $u \in L(B_\sigma)$ using a derivation that starts with the rule $B_\sigma \to B_{\sigma'}\, b\, B_{\sigma''}$.

It is now obvious that since the only rule for S_a is $S_a \to B_{f_L(a)}\, a\, B_{f_R(a)}$, Lemma 3 guarantees that $L(S_a) = \Lambda(a)$. □

2.3. From the grammar to a Σ_2 expression

Let H be a conservative groupoid with a some element of H and let $L = \Lambda(a) \subseteq H^*$. We are now ready to prove Theorem 2, and show that L is in fact in Σ_2, i.e. it is a finite union of sets of the form $\sigma_0^* a_1 \sigma_1^* \cdots a_n \sigma_n^*$, with each $a_i \in H$ and $\sigma_i \subseteq H$.

Build from H the context-free grammar G_H with the method of Section 2.2; its initial non-terminal is S_a. We say that a derivation δ is *nonerasing* if no production of the form $B \to \epsilon$ is used in it. An induction on the length of δ shows that a nonerasing derivation outputs a string $Y(\delta) = B_{\sigma_0} a_1 B_{\sigma_1} \cdots a_n B_{\sigma_n}$, where each B_{σ_i} is a non-terminal and each a_i a terminal. We slightly abuse notation and use δ to denote both the derivation and the corresponding derivation tree. Then, we also write $Y(\delta)$ to denote the output of a tree δ, i.e. the left-to-right sequence of leaves of δ. Erasing the non-terminals in $Y(\delta)$ we obtain a word $w(\delta) = a_1 \cdots a_n$. Replacing each non-terminal B_σ in $Y(\delta)$ with σ^*, we obtain a regular expression for the language $L(\delta) = \sigma_0^* a_1 \sigma_1^* \cdots a_n \sigma_n^* \subseteq L$.

Let Δ denote the set of all nonerasing derivations from S_a; we have

$$\bigcup_{\delta \in \Delta} L(\delta) \subseteq L = \{\ w(\delta) : \delta \in \Delta\ \} \subseteq \bigcup_{\delta \in \Delta} L(\delta) \tag{1}$$

so that L is a union of sets of the form $\sigma_0^* a_1 \sigma_1^* \cdots a_n \sigma_n^*$. The union in Equation 1 is infinite but we will prove that all but a finite numbers of the terms $L(\delta)$ it contains are redundant. More specifically, we say that a non-erasing derivation δ is *dominated* by a non-erasing derivation δ' if $L(\delta) \subseteq L(\delta')$. Note that if δ is a non-erasing derivation that contains γ as a subtree and if γ is dominated by some γ' with the same root then the tree δ' obtained by replacing γ in δ by γ' dominates δ.

We will define a finite set $\mathcal{F} \subset \Delta$ of non-erasing derivations such that every non-erasing derivation in Δ is dominated by one in $\mathcal{F}$. Consequently, we will have $L = \bigcup_{\delta \in \mathcal{F}} L(\delta)$, a finite union. We prove this in a sequence of steps, where each step uses a particular transformation on derivation trees. The successive transformations we present consist in collapsing *homogeneous subtrees*, eliminating *hiccup nodes*, straightening *paths* with multiple *angles*, and shortening paths that are *too long*. These transformations collaborate to reduce any redundant tree $\delta \in \Delta$—in particular, one whose depth is more than $3|H|$—into a tree $\delta' \in \mathcal{F}$ that dominates δ.

We say that γ is *homogeneous* for B_σ if B_σ is the only non-terminal involved in γ, i.e. every node in the derivation tree is either labeled by B_σ or by a letter in σ. In particular we have $w(\gamma) \in \sigma^*$ and therefore $L(\gamma) \subseteq \sigma^*$. We claim that every δ containing a homogeneous subtree γ for B_σ is dominated by the derivation obtained by replacing γ in δ by B_σ. By our earlier observation, it suffices to establish that the tree γ_σ consisting of the single node labeled B_σ dominates γ. This is obvious since $L(\gamma_\sigma) = \sigma^*$. Therefore any derivation δ is dominated by a δ' with $|\delta| \geq |\delta'|$ and such that δ' has no homogeneous subtree.

The technique we use in the rest of this proof and which we call "recursive top-down relabeling" is based on the following simple property of G_H.

Proposition 7. *For any two subsets $\sigma \subset \sigma'$ and any production $B_\sigma \to B_\rho a B_\tau$ in the grammar, there exists another production $B_{\sigma'} \to B_{\rho'} a B_{\tau'}$ with $\rho \subseteq \rho'$ and $\tau \subseteq \tau'$.*

This is immediate from the definition of G_H and in fact we can be more precise and establish that $\rho' = \rho \cup \sigma'$ and $\tau' = \tau \cup \sigma'$.

Let δ be a nontrivial derivation tree, let B_σ, B_{τ_1}, b and B_{τ_2} be the root and its sons, respectively, and let $Y(\delta) = B_{\varrho_0} a_1 B_{\varrho_1} \cdots a_k B_{\varrho_k}$. The first production used in the corresponding derivation is $B_\sigma \to B_{\tau_1} b B_{\tau_2}$. By the proposition, for every superset σ' of σ, there exists a production $B_{\sigma'} \to B_{\tau_1'} b B_{\tau_2'}$ with $\tau_1 \subseteq \tau_1'$ and $\tau_2 \subseteq \tau_2'$. We can relabel the nodes of δ, first replacing with $B_{\sigma'}$, $B_{\tau_1'}$, b and $B_{\tau_2'}$ the root and its sons, and then by doing similar replacements recursively in a top-down manner. The result is a derivation tree δ' with root labelled $B_{\sigma'}$, whose output is $Y(\delta') = B_{\varrho_0'} a_1 B_{\varrho_1'} \cdots a_k B_{\varrho_k'}$, where $\varrho_i \subseteq \varrho_i'$ for every $0 \leq i \leq k$, and therefore δ' dominates δ.

We first apply this technique to those derivations which involve a production of the form $B_\sigma \to B_\sigma b B_\sigma$, and therefore such that δ contains the pattern v:

$$\begin{tikzcd}
v : & & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & \\
& B_\sigma & b & B_\sigma
\end{tikzcd}$$

We say that a node, as above, with the same label as its left- and rightmost children is a *hiccup node*. We want to show that every tree δ containing a hiccup node is dominated by one of equal or lesser size that is hiccup-free.

Let γ be a subtree of δ rooted at a hiccup node x such that no ancestor of x is a hiccup node (i.e. we choose γ to be as close to the root of δ as possible). First note that if all non-terminal labels in γ are also labeled B_σ then, by the homogeneous case, γ can be replaced by B_σ.

Otherwise, γ contains a subtree u which breaks away from homogeneity:

$$\begin{tikzcd}
 & & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & & \\
u : & T_\sigma & b & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & \\
 & & T_\tau & c & T_\chi
\end{tikzcd}$$

where T_σ, T_τ, T_χ are subtrees with roots B_σ, B_τ, B_χ respectively and where at least one of $\tau \neq \sigma$ and $\chi \neq \sigma$ holds; the case where the leftmost B_σ is expanded is symmetric. The output of this subtree is $Y(u) = Y(T_\sigma)bY(T_\tau)cY(T_\chi)$.

We want to show that if the first break in homogeneity occurs at depth i in γ, then γ can be dominated by a γ' of identical size and with B_σ as its root but where the first break in homogeneity occurs at depth $i-1$. To do this, consider u and reverse in the derivation the order of productions $B_\sigma \to B_\sigma b B_\sigma$ and $B_\sigma \to B_\tau b B_\chi$, and apply our top-down relabeling technique to the left son of the root and its subtree; the result is a subtree u' which dominates u:

$$\begin{tikzcd}
 & & & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & \\
u' : & & B_\tau \arrow[dl] \arrow[d] \arrow[dr] & c & T_\chi \\
 & T_\varrho & b & T_\tau &
\end{tikzcd}$$

where T_ϱ is obtained from T_σ by replacing the root by B_τ and using top-down relabeling. Note that since either $\sigma \neq \tau$ or $\sigma \neq \chi$, the break in homogeneity has been moved up to the root of u' and by substituting u by u' in γ, we obtain, as claimed, a γ' in which the first break in homogeneity occurs at depth $i-1$. By iterating this construction $i-1$ times, we obtain a γ'' that dominates γ and has the same root B_σ but where the root is not a hiccup anymore. Note that this process might create new hiccups in the subtree γ'' but our construction can be iterated to eliminate these in turn. It is crucial to point out that our relabeling always replaces a B_σ by a $B_{\sigma'}$ where $\sigma \subseteq \sigma'$ so the depth of recursion in our hiccup elimination procedure is at most $|H|$.

We have thus far shown that any non-erasing derivation is dominated by one of equal or lesser size in which no two sons of a node carry the same label as their parent. There is still an infinite number of trees to consider since trees can contain a root-leaf path with an arbitrarily long sequence of nodes labelled with the same non-terminal.

Along such a sequence, we say that the path *moves to the left* (resp. right) from a given node if the left child (resp. right child) of that node is labeled B_σ. We show below that any subtree that begins with a B_σ path is dominated by one where the path is of length at most two and in fact consists of at most one move to the left followed by one move to the right.

We say that a *right B_σ-angle* occurs at a node if this node is a leftmost son and both the node, its father and its rightmost son carry the same label B_σ (see pattern p in the diagram below). We define a left B_σ-angle dually (see pattern p').

p

$$\begin{tikzcd}
& & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & \\
& B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & c & T_\chi \\
T_\rho & a & T_\sigma &
\end{tikzcd}$$

p'

$$\begin{tikzcd}
& B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & & \\
T_\rho & a & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & \\
& T_\sigma & c & T_\chi
\end{tikzcd}$$

In the above, T_ρ, T_σ and T_χ are trees with roots labeled B_ρ, B_σ and B_χ respectively. Seen as portions of a derivation tree, both patterns in this diagram have the same output so that any derivation tree γ which contains p is dominated by the tree γ' which contains p' instead of p (and vice versa). Observe that any other angle that may exist in γ is unaffected.

Therefore if γ is a hiccup-free subtree rooted at B_σ, we can repeat this substitution[4] process until the B_σ path starting at the root contains no more than one B_σ-angle and we assume without loss of generality that it is a right angle. In other words, the path consists of a certain number of moves to the left followed by a certain number of moves to the right. Suppose that the B_σ path begins by at least two moves to the left, i.e. we are in the following configuration:

$$\begin{tikzcd}
& & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & \\
& B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & c_2 & S_2 \\
T_\sigma & c_1 & S_1 &
\end{tikzcd}$$

where T_σ is a tree with root B_σ and S_1, S_2 are trees with roots B_{χ_1}, B_{χ_2} respectively and where χ_1, χ_2 are both strict supersets of σ.

[4]This substitution is reminiscent of the tree rotations that are performed on AVL trees and other balanced trees.

This subtree is dominated by the following one

$$\begin{tikzcd}
 & B_\sigma \arrow[dl] \arrow[d] \arrow[dr] & & \\
T_\sigma & c_1 & B_{\chi_1} \arrow[dl] \arrow[d] \arrow[dr] & \\
 & S_1 & c_2 & S_2'
\end{tikzcd}$$

where S_2' is the tree obtained by top-down relabeling S_2 using the root $B_{\chi_1 \cup \chi_2}$, following Proposition 7. Note that, since $\chi_1 \cup \sigma = \chi_1$, we can safely attach the subtree S_1 as the leftmost child of B_{χ_1}.

This shows that any subtree with a B_σ path starting at the root with $i \geq 2$ moves to the left can be dominated by a subtree where the B_σ path starts with $i-1$ moves to the left. Applying this argument $i-2$ times, we obtain a subtree where the B_σ path begins with one left move possibly followed by a sequence of right moves. A symmetric argument shows that we can assume that the latter sequence consists of at most one right move.

By applying the above transformations repeatedly, we can dominate any derivation tree by one of equal or smaller size which is hiccup-free and where every B_σ path is of length no more than 2. The set $\mathcal{F}$ of such trees is finite since they all have depth at most $3|H|$ so this translates into a finite Σ_2 expression for the language generated by the grammar. This concludes the proof of Theorem 2.

Example 8. *To illustrate the above proof, let us go back to the Rock-Paper-Scissors game and construct a regular expression for $\Lambda(p)$ the set of words in $\{r,p,s\}^*$ on which Paper can win. The grammar generating $\Lambda(p)$ is given by the rules:*

$$\begin{aligned}
S_p &\rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}} \\
B_{\{r,p\}} &\rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}} \mid B_{\{r,p,s\}}\, r\, B_{\{r,p,s\}} \mid \epsilon \\
B_{\{r,p,s\}} &\rightarrow B_{\{r,p,s\}}\, r\, B_{\{r,p,s\}} \mid B_{\{r,p,s\}}\, p\, B_{\{r,p,s\}} \mid B_{\{r,p,s\}}\, s\, B_{\{r,p,s\}} \mid \epsilon
\end{aligned}$$

Note that we can exclude the non-terminals $B_{\{r\}}, B_{\{p\}}, B_{\{s\}}, B_{\{r,s\}}, B_{\{p,s\}}$ which are in fact unreachable from S_p.

Every non-erasing derivation from S_p begins with $S_p \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}}$. In turn, derivations from $B_{\{r,p\}}$ start with either $B_{\{r,p\}} \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}}$ or $B_{\{r,p\}} \Rightarrow B_{\{r,p,s\}}\, r$ $B_{\{r,p,s\}}$. The first case immediately creates a hiccup node and can therefore be dominated and safely ignored. In the second case, one is left with two occurrences of $B_{\{r,p,s\}}$ but any further derivation from these non-terminals must also create hiccup nodes. We are therefore left with a Σ_2 expression containing only four useful terms corresponding to the following non-erasing derivations.

$$\begin{aligned}
&S_p \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}} \\
&S_p \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}} \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p,s\}}\, r\, B_{\{r,p,s\}} \\
&S_p \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}} \Rightarrow B_{\{r,p,s\}}\, r\, B_{\{r,p,s\}} p\, B_{\{r,p\}} \\
&S_p \Rightarrow B_{\{r,p\}}\, p\, B_{\{r,p\}} \Rightarrow B_{\{r,p,s\}}\, r\, B_{\{r,p,s\}}\, p\, B_{\{r,p,s\}}\, r\, B_{\{r,p,s\}}
\end{aligned}$$

Accordingly, $\Lambda(p)$ is represented by the regular expression

$$(r|p)^*p(r|p)^* \mid (r|p)^*p(r|p|s)^*r(r|p|s)^* \mid (r|p|s)^*r(r|p|s)^*p(r|p)^* \mid$$
$$(r|p|s)^*r(r|p|s)^*p(r|p|s)^*r(r|p|s)^*.$$

This expression basically says that Paper can win on a string x if and only if $x = upv$ where both u and v either consist only of Papers and Rocks or contain a Rock. The same language can be described more succinctly by the expression

$$[\,p^* \mid (r|p|s)^*r(r|p|s)^*]\,p\,[\,p^* \mid (r|p|s)^*r(r|p|s)^*].$$

3. Languages Recognized by Conservative Groupoids

We now know that languages recognized by conservative groupoids are regular. In this section we seek a more precise characterization.

3.1. Conservative Semigroups

One starting point is to consider conservative groupoids which are also associative, i.e. semigroups for which $x \cdot y \in \{x, y\}$. In particular, these satisfy $x^2 = x$ but we can give an exact characterization.

Lemma 9. *A semigroup S is conservative iff its set of elements can be partitioned into k classes $C_1, \ldots, C_k$ such that*

1. *$x \cdot y = y \cdot x = x$ whenever $x \in C_i$ and $y \in C_j$ for $i < j$;*
2. *$x \cdot y = x$ for all $x, y \in C_j$ (left-zero) or $x \cdot y = y$ for all $x, y \in C_j$ (right-zero) for any j.*

Proof. ($\Leftarrow$)

By definition, such a semigroup is conservative. Also, the operation defined above is associative. Indeed if x, y, z are three elements lying in the same class C_i then $(x \cdot y) \cdot z = x \cdot (y \cdot z) = x$ if C_i is left-zero and $(x \cdot y) \cdot z = x \cdot (y \cdot z) = z$ if C_i is right-zero. If x, y, z are not in the same class then associativity follows because the elements in the most absorbing class are the only ones that matter. Suppose for instance that x and z lie in the same class C_i while y lies in some C_j with $i > j$. Since $x \cdot y = x$ and $y \cdot z = z$ we clearly have $(x \cdot y) \cdot z = x \cdot (y \cdot z) = xz$.

($\Rightarrow$)

Conversely, suppose that S is a conservative semigroup. Let us recall the definition of Green's $\mathcal{J}$-preorder noted $\leq_{\mathcal{J}}$. Let $x, y \in S$. We write $x \leq_{\mathcal{J}} y$ if there exists $\alpha, \beta \in S$ such that $x = \alpha y \beta$. Finally, let us remind that $x \mathrel{\mathcal{J}} y$ is Green's $\mathcal{J}$-equivalence relation built with $\leq_{\mathcal{J}}$.

Let us denote the $\mathcal{J}$-classes of S by $C_1, \ldots, C_k$. Firstly, $x \leq_{\mathcal{J}} y$ or $y \leq_{\mathcal{J}} x$ for all $x, y \in S$ since S is conservative. Then, the $\mathcal{J}$-classes are totally ordered by $\leq_{\mathcal{J}}$ and we can assume that $C_1, \ldots, C_k$ are labelled such that $C_i \leq_{\mathcal{J}} C_j$ iff $i \leq j$.

(1). Let $x \in C_i$, $y \in C_j$ such that $i < j$. Thus, $xy \leq_{\mathcal{J}} x <_{\mathcal{J}} y$ and then, $xy \neq y$. Since S is conservative, $xy = x$. We can show $yx = x$ in the same way.

(2). Let $x, y \in C_i$. We suppose that $x \neq y$, otherwise the result is clearly true. Since $x, y \in C_i$, then $x \leq_{\mathcal{J}} y$ and $y \leq_{\mathcal{J}} x$; i.e. there exist $\alpha, \beta, \gamma, \rho \in S$ such that $x = \alpha y \beta$ and

$y = \gamma x \rho$. Since S is conservative, then we have one of $\alpha = x$ or $\beta = x$ and one of $\gamma = y$ or $\rho = y$. This is equivalent to having one of $xy = x$ or $yx = x$ and one of $yx = y$ or $xy = y$. Note that we cannot have $xy = x$ and $yx = x$ at the same time because neither of $xy = y$ or $yx = y$ would be true and that would cause a contradiction. If we have $xy = x$, then $yx = y$ and the left element wins in both cases. If we have $yx = x$, then $xy = y$ and the right element wins in both cases.

□

This characterization can be translated into a description of the languages recognizable by conservative semigroups. For an alphabet A, consider a partition $C_1, \dots, C_k$ each with an associated direction $d_1, \dots, d_k$ with $d_i \in \{\mathsf{L}, \mathsf{R}, \mathsf{C}\}$ given in the following way:

- if C_i has at least two elements and is a left-zero, $d_i = \mathsf{L}$;
- if C_i has at least two elements and is a right-zero, $d_i = \mathsf{R}$;
- otherwise C_i has one element and $d_i = \mathsf{C}$.

For $a \in C_j$ with $d_j = \mathsf{L}$ (resp. $d_j = \mathsf{R}$), define the language L_a (resp. R_a) of words with at least one a that contain no occurrence of letters in classes C_i with $i < j$ and where the first (resp. last) occurrence of a letter in C_j is an a. If $d_j = \mathsf{C}$, define the language C_a of words with at least one a that contain no occurrence of letters in classes C_i with $i < j$.

Corollary 10. *A language can be recognized by a conservative semigroup iff it is the disjoint union of some L_a, R_a and C_a.*

Note that the class of languages recognized by conservative semigroups does not have many closure properties. For instance, it is not closed under union or intersection: each of the languages A^*aA^* and A^*bA^* can be recognized but their union (or intersection) has a syntactic semigroup which is not conservative.

3.2. Basic Properties of the Non-Associative Case

The apparent absence of closure properties makes it difficult to provide a complete characterization of languages recognized by non-associative, conservative groupoids. Moreover the definition of recognition by a groupoid allows a homomorphism $h : A^* \to H^*$ that "translates" a word over the original alphabet into a string of groupoid elements and this can be surprisingly powerful. Consider for instance the alphabet $A = \{a, b\}$ and the language $K = A^*aA^*bA^*$. This language is not commutative (i.e. there exist x, y such that $xy \in K$ and $yx \notin K$) yet it can be recognized by the rock-paper-scissors groupoid $H = \{r, p, s\}$ which *is* commutative. Indeed, if one chooses the accepting set $F = \{p\}$ and if $h(a) = ps$ and $h(b) = r$ then it is possible to show that $w \in K$ iff $W(h(w)) \cap F \neq \emptyset$. Indeed if $w \notin K$ then $h(w) = r^n(ps)^m$ for some $n, m \in \mathbb{N}$ and by Example 8, it is impossible for p to win such a word. Conversely, suppose that $w \in K$. Consider the first p occurring in $h(w)$. On its left one finds r^n for some $n \geq 0$ and on its right there is at least one Rock because $w \in K$. Therefore by Example 8, Paper is able to win on $h(w)$.

In the rest of this section we prove a number of results which provide important insight into the place that languages recognizable by conservative groupoids occupy within the Straubing-Thérien hierarchy.

We begin by four simple lemmas.

Lemma 11. *The class of languages recognized by conservative groupoids is closed under inverse homomorphisms $h : B^* \to A^*$ from one free monoid to the other, i.e. if $L \subseteq A^*$ can be recognized by the conservative groupoid H then $h^{-1}(L)$ can also be recognized by H.*

Proof. This is a well-known straightforward consequence of the definition of recognition by a groupoid and does not depend on the fact that H is conservative. Indeed if L is recognized using the mapping $\phi : A^* \to H^*$ and the accepting subset F then by setting $\psi = \phi \circ h$ we obtain a homomorphism from B^* to H^* and we have

$$W(\psi(x)) \cap F \neq \emptyset \Leftrightarrow W(\phi(h(x))) \cap F \neq \emptyset \Leftrightarrow h(x) \in L \Leftrightarrow x \in h^{-1}(L).$$

□

Corollary 12. *Every language recognizable by a conservative groupoid lies in Σ_2.*

Proof. Suppose $L \subseteq A^*$ is recognizable by a conservative groupoid H using the homomorphism $h : A^* \to H^*$ and the accepting subset $F \subseteq H$. By definition, $L = h^{-1}(\Lambda(F))$ and $\Lambda(F)$ lies in Σ_2 by Theorem 2. It is known (see [13]) that Σ_2 is closed under inverse homomorphic images. □

Lemma 13. *If $L \subseteq A^*$ is recognizable by a conservative groupoid, then for any $B \subseteq A$ the language $L \cap B^*$ is also recognizable by a conservative groupoid.*

Proof. Suppose L is recognized by the conservative groupoid H using the homomorphism $h : A^* \to H^*$ and accepting subset $F \subseteq H$. Define H_0 by adding a new absorbing element 0 in H (i.e. $0x = x0 = 0$ for all $x \in H$). Note that the groupoid H_0 is still conservative. Now define $g : A^* \to H_0^*$ by setting $g(a) = h(a)$ if $a \in B$ and $g(a) = 0$ if $a \notin B$. For any $w \notin B^*$, we therefore have a 0 occurring in $g(w)$ and thus $W(g(w)) = \{0\}$. On the other hand if $w \in B^*$ then $g(w) = h(w)$ and therefore $L \cap B^*$ is precisely the set of words such that $W(g(w)) \cap F \neq \emptyset$. □

Lemma 14. *If $L \subseteq A^*$ is recognized by a conservative groupoid H then $L = L^+$ (where L^+ denotes LL^*).*

Proof. Suppose L is recognized using $h : A^* \to H^*$ and accepting subset F. Consider a word of L^+ i.e. $x = x_1 \dots x_k$ with each x_i in L. We know that for each i there exists some $a_i \in F$ such that $h(x_i)$ can be bracketed to get a_i as the winner. Now $h(x) = h(x_1) \dots h(x_k)$ so $W(h(x)) \supseteq W(a_1 \dots a_k)$. Since H is conservative and since all a_i lie in F, the set $W(h(x))$ contains at least one of the a_i and $x \in L$. □

Let L be a language over A^*. The *syntactic pre-order* of L on A^* is defined by setting $x <_L y$ iff for all $s, t \in A^*$ it holds that $syt \in L \Rightarrow sxt \in L$. Note that $<_L$ is compatible with concatenation in the sense that $x <_L y \Rightarrow uxv <_L uyv$ for any $x, y, u, v \in A^*$. This pre-order and the corresponding equivalence relation ($x \equiv_L y$ if $sxt \in L \Leftrightarrow syt \in L$ for all $s, t \in A^*$) are central to algebraic automata theory. A theorem of Pin [12] states that every positive variety of languages, i.e. every class $\mathcal{L}$ of languages closed under union, intersection, inverse morphic images (if $K \subseteq A^*$ is in $\mathcal{L}$ and $h : B^* \to A^*$ is a homomorphism then $h^{-1}(K) \in \mathcal{L}$) and left and right quotients (if $K \in \mathcal{L}$ and $a \in A$ then $a^{-1}K \in \mathcal{L}$ and $Ka^{-1} \in \mathcal{L}$ where $a^{-1}K = \{x : ax \in K\}$ and $Ka^{-1} = \{x : xa \in K\}$) can be characterized by a (possibly infinite) set of defining identities of the syntactic pre-order. A formal treatment of identities can be found in [13] but the following two examples are somewhat typical and particularly relevant in our context. A language L lies in Σ_1 iff $x <_L \epsilon$ for all x (see e.g. [15, 13]). A language L lies in Σ_2 iff there exists some ω such that $x^\omega y x^\omega <_L x^\omega$ whenever the set of letters occurring in x is equal to the set of letters occurring in y [15].

The syntactic pre-order allows us to give a simple necessary condition for recognizability by a conservative groupoid.

Lemma 15. *If $L \subseteq A^*$ is recognized by a conservative groupoid then $x^2 <_L x$ for all $x \in A^*$.*

Proof. It suffices to show that for any conservative groupoid H and any $s, u, t \in H^*$ it holds that $W(sut) \subseteq W(su^2t)$.

Suppose $u = u_1 \dots u_k$. Pick any element in $W(u)$, i.e. fix some j such that u_j is a winner in u given the correct bracketing. By Lemma 3, there exist contest trees $\tau \in T(u_1 \dots u_{j-1})$ and $\tau' \in T(u_{j+1} \dots u_k)$ such that $\tau(u_1 \dots u_{j-1}) = \ell \in f_L(u_j)$ and $\tau'(u_{j+1} \dots u_k) = r \in f_R(u_j)$. Now consider the partial bracketing of $su^2t = su_1 \dots u_k u_1 \dots u_k t$ given by

$$su_1 \dots u_j \tau'(u_{j+1} \dots u_k)\tau(u_1 \dots u_{j-1})u_j \dots u_k t = su_1 \dots u_j r \ell u_j \dots u_k t.$$

In turn, the latter can be bracketed as

$$\begin{aligned} su_1 \dots u_{j-1}((u_j r)(\ell u_j))u_{j+1} \dots u_k t &= su_1 \dots u_{j-1}(u_j u_j)u_{j+1} \dots u_k t \\ &= su_1 \dots u_k t = sut. \end{aligned}$$

In particular $W(sut) \subseteq W(su^2t)$. ☐

Corollary 16. *The class of languages recognized by conservative groupoids*

1. *is not closed under complement*
2. *is not closed under union.*

Proof.

1. We showed earlier that $L = \Sigma^* a \Sigma^* b \Sigma^*$ is recognizable by the Rock-Paper-Scissors groupoid but Lemma 14 guarantees that its complement L^c is not recognizable by a conservative groupoid. Indeed note that L^c contains the words a, b and ϵ so $(L^c)^+ = \{a, b\}^* \neq L^c$.
2. The language $L_1 = a\{a, b\}^*$ can be recognized by the two element conservative groupoid $\{x, y\}$ where $xy = x$ and $yx = y$. (This groupoid is in fact associative.) Similarly, $L_2 = \{a, b\}^* b$ can be recognized by a two element conservative groupoid. Since $a \in L_1$ and $b \in L_2$ we have $a, b \in L_1 \cup L_2$ and therefore $ba \in (L_1 \cup L_2)^+$ even though $ba \notin L_1 \cup L_2$. By Lemma 14, this proves that $L_1 \cup L_2$ cannot be recognized by a conservative groupoid. ☐

Whether the class of languages recognizable by conservative groupoids is closed under intersection remains an open question.

Finally, the following lemma is useful for analyzing particular sequences of groupoid elements since it shows that repetitions of a given element can be eliminated without changing the set of potential winners.

Lemma 17. *Let H be a conservative groupoid and let $a \in H$. For any $s, t \in H^*$ we have $W(sat) = W(saat)$.*

Proof. By Lemma 15, we have $W(sat) \subseteq W(saat)$ for all $a \in H$.

Let $b \in W(saat)$. Let D be a favorable decomposition tree for b such that $\lambda(D) = saat$. Let a_1 and a_2 be the two instances of a. Without loss of generality, we can suppose that a_1 is a descendant of a_2 in D because a_1 and a_2 are neighbours. Moreover, there is a subtree r of D rooted at a_1 such that $saat = x\lambda(r)a_2t$. Let r' be the left subtree of a_1 in r. Since r is a favorable decomposition tree for a, there exists a contest tree $p \in T(\lambda(r')a_2)$ such that $p(\lambda(r')a_2) = a_2$.

For this reason, we just need to show that $b \in W(xa_2t)$. By replacing r by a leaf ϵ in D, we get a favorable decomposition tree D' for b such that $\lambda(D') = xa_2t = xat$. Thus, $b \in W(xat) \subseteq W(sat)$. □

Lemma 17 says that for any conservative groupoid H, any $h \in H$ and any $s, t \in H^*$ the language $\Lambda(h)$ has the following property

$$sat \in \Lambda(h) \Leftrightarrow saat \in \Lambda(h).$$

This property is known as *stutter invariance* and has been extensively studied, in particular in the context of automated verification (e.g. [8]).

3.3. Place in the Straubing-Thérien Hierarchy

Our main theorem shows that if L can be recognized by a conservative groupoid then L is in Σ_2 but examples in Corollary 16 show that the converse is not true. We begin this section by showing that each L in Σ_1 is recognizable by a conservative groupoid.

Lemma 18. *Let $A = \{a_{1,1}, \ldots, a_{1,k_1}, \ldots, a_{\ell,1}, \ldots, a_{\ell,k_\ell}\}$ be some alphabet. (Let us stress that the $a_{i,j}$ are all distinct.) Then*

$$A^*a_{1,1}A^* \ldots A^*a_{1,k_1}A^* \cup \ldots \cup A^*a_{l,1}A^* \ldots A^*a_{l,k_l}A^*$$

can be recognized by a conservative groupoid.

Proof. Let $H_i = \{\alpha_{i,1}, \beta_{i,1}, \ldots, \alpha_{i,k_i}, \beta_{i,k_i}\}$ and let $H = \bigcup_{1\le i\le l} H_i$. We choose the homomorphism $h : A \to H^*$ defined by $h(a_{i,j}) = \alpha_{i,j}\beta_{i,j}$ and the accepting subset $F = \{\beta_{1,k_1}, \ldots, \beta_{l,k_l}\}$. We define the conservative operation on H by first defining it within each H_i:

- $\alpha_{i,j}\alpha_{i,k} = \alpha_{i,k}$;
- $\alpha_{i,j}\beta_{i,k_i} = \begin{cases} \alpha_{i,j} & \text{for all } j \neq 1 \\ \beta_{i,k_i} & \text{if } j = 1 \end{cases}$

- $\alpha_{i,j}\beta_{i,k} = \begin{cases} \alpha_{i,j} & \text{for all } j \geq k \neq k_i \\ \beta_{i,k} & \text{for all } j < k \neq k_i \end{cases}$
- $\beta_{i,j}\alpha_{i,k} = \begin{cases} \alpha_{i,k} & \text{for all } j < k-1 \\ \beta_{i,j} & \text{for all } j \geq k-1 \end{cases}$
- $\beta_{i,j}\beta_{i,k} = \begin{cases} \beta_{i,j} & \text{for all } j \geq k \\ \beta_{i,k} & \text{for all } j < k \end{cases}$

  The above rules specify the multiplication within each H_i. In the rules below, we define the other products and therefore assume $i \neq r$.

- $\alpha_{i,j}\alpha_{r,k} = \alpha_{r,k}$;
- $\alpha_{i,j}\beta_{r,k} = \begin{cases} \beta_{r,k} & \text{if } k \neq k_r \\ \alpha_{i,j} & \text{if } k = k_r \end{cases}$
- $\beta_{i,j}\alpha_{r,k} = \begin{cases} \alpha_{r,k} & \text{if } j \neq k_i \\ \beta_{i,j} & \text{if } j = k_i \end{cases}$
- $\beta_{i,j}\beta_{r,k} = \begin{cases} \beta_{r,k} & \text{if } j \neq k_i \text{ and } k \neq k_r \\ \beta_{i,j} & \text{if } j = k_i \text{ or } k = k_r \end{cases}$

Let us point out a few important properties of H. First, for any i, the element β_{i,k_i} is weak when facing an opponent on its left since $f_L(\beta_{i,k_i}) = \{\alpha_{i,1}, \beta_{i,k_i}\}$. It is however strong when facing an element on its right since $f_R(\beta_{i,k_i}) = H$. Secondly, an element $\alpha_{i,j}$ loses on its right against any element outside of H_i with the sole exception of the β_{r,k_r}. The same is true for any element $\beta_{i,j}$ with $j \neq k_i$.

Claim (†). Let $u = h(v)$ and $|u| > 0$. There is an $\alpha_{i,j}$ in $W(u)$.

Proof. By definition of h, the word u is a sequence of pairs $\alpha_{r_i,s_i}\beta_{r_i,s_i}$. We begin by considering the following partial bracketing of u:

$$(\alpha_{r_1,s_1}\beta_{r_1,s_1})(\alpha_{r_2,s_2}\beta_{r_2,s_2})\ldots(\alpha_{r_n,s_n}\beta_{r_n,s_n}).$$

The result within each pair of brackets is of α type and so any further bracketing will produce a winner of α type. □

Claim.

$$w \in A^*a_{1,1}A^*\ldots A^*a_{1,k_1}A^* \cup \ldots \cup A^*a_{l,1}A^*\ldots A^*a_{l,k_l}A^* \text{ iff } F \cap W(h(w)) \neq \emptyset.$$

Proof.
(⇒)
By hypothesis, $h(w) = u_1\alpha_{i,1}\beta_{i,1}u_2\ldots u_{k_i}\alpha_{i,k_i}\beta_{i,k_i}u_{k_i+1}$. By the claim †, there exists $\lambda_\ell \in W(u_\ell)$ for all $1 \leq \ell \leq k_i + 1$ such that λ_ℓ is of the form $\alpha_{i,j}$. Let t_ℓ be the contest tree over each u_ℓ such that $t_\ell(u_\ell) = \lambda_\ell$.

Consider the following partial bracketing of $u_1\alpha_{i,1}\beta_{i,1}u_2\dots u_{k_i}\alpha_{i,k_i}\beta_{i,k_i}u_{k_i+1}$

$$\begin{aligned} & (t_1(u_1)\alpha_{i,1})\beta_{i,1}(t_2(u_2)\alpha_{i,2})\dots(t_{k_i}(u_{k_i})\alpha_{i,k_i})(\beta_{i,k_i}t(u_{k_i+1})) \\ = \quad & (\lambda_1\alpha_{i,1})\beta_{i,1}(\lambda_2\alpha_{i,2})\dots(\lambda_{k_i}\alpha_{i,k_i})(\beta_{i,k_i}\lambda_{k_i+1})). \end{aligned}$$

Since $\alpha_{i,j}\alpha_{r,k}=\alpha_{r,k}$ for all i,j,r,k and $\beta_{i,k_i}\alpha_{r,k}=\beta_{i,k_i}$ for any i,r,k, the latter partial bracketing evaluates to

$$\alpha_{i,1}\beta_{i,1}\alpha_{i,2}\beta_{i,2}\dots\alpha_{i,k_i}\beta_{i,k_i}.$$

It thus suffices to show that β_{i,k_i} can win on this string and this is achieved through the following bracketing:

$$(\alpha_{i,1}(\beta_{i,1}(\dots(\beta_{i,k_i-2}(\alpha_{i,k_i-1}(\beta_{i,k_i-1}\alpha_{i,k_i})))\dots)\beta_{i,k_i}).$$

($\Leftarrow$)
By definition $h(w)$ is a sequence of pairs $\alpha_{i,k}\beta_{i,k}$ which we call *companion pairs*. Note first that for any favorable decomposition tree D, if $\lambda(D)=s\alpha\beta v$ where α,β are companion letters, then either α is the rightmost node of the left subtree of the subtree rooted at β in D or β is the leftmost node of the right subtree of the subtree rooted at α in D.

Suppose $\beta_{p,k_p}\in W(h(w))$ and let $h(w)=s\alpha_{p,k_p}\beta_{p,k_p}v$ where this occurrence of β_{p,k_p} is the eventual winner. By Lemma 3, there exists a favorable decomposition tree D for β_{p,k_p}. A *perfect* subtree D' of D is a favorable decomposition tree such that

- the root of D' is β_{q,k_q} for some $q\in\{1,\dots,l\}$;
- β_{q,k_q} is an ancestor of its companion element α_{q,k_q};
- any proper subtree D'' of D' is not a perfect subtree of D'.

There is at least one perfect subtree in D because if D does not have a perfect proper subtree, then D itself is a perfect subtree. Therefore let D' be a perfect subtree with root of label β_{q,k_q}. We observe the following facts about D':

- α_{q,k_q} is the rightmost node of the left subtree of β_{q,k_q} since α_{q,k_q} is not the ancestor of β_{q,k_q};

($\star$) D' does not contain another β_{z,k_z} which is the ancestor of its companion element α_{z,k_z} because this would imply that D' has a perfect proper subtree rooted at this β_{z,k_z};

- $\alpha_{q,1}$ is the left child of β_{q,k_q} because $f_L(\beta_{q,k_q})=\{\alpha_{q,1},\beta_{q,k_q}\}$ and by $\star$ the choice of β_{q,k_q} is excluded;
- let $1\le t<k_q$, then the right child of $\alpha_{q,t}$ can only be $\alpha_{q,t}$ or $\beta_{q,s}$ with $s\le t$ because the only other choices in $f_R(\alpha_{q,t})$ are of β_{z,k_z} type and this cannot happen by $\star$;

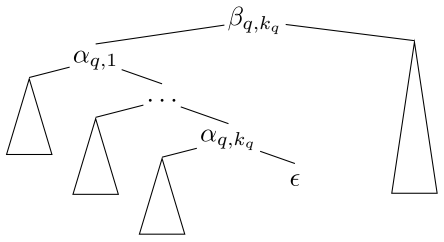

Figure 4: The structure of the perfect subtree D'

- let $1 \leq t < k_q$, then the right child of $\beta_{q,t}$ can only be $\alpha_{q,t+1}$, $\beta_{q,s}$ or $\alpha_{q,s}$ with $s \leq t$ because the only other choices in $f_R(\beta_{q,t})$ are of β_{z,k_z} type and this cannot happen by $\star$.

Now consider the left subtree D'' of β_{q,k_q} with root $\alpha_{q,1}$ and consider the rightmost path in that subtree (see Figure 4). This sequence of labels starts at $\alpha_{q,1}$, ends at α_{q,k_q} and by the above facts it must therefore include a subsequence $\alpha_{q,1}, \beta_{q,1}, \alpha_{q,2}, \dots, \alpha_{q,k_q}$.

Thus, $\lambda(D')$ includes the subsequence $\alpha_{q,1}, \beta_{q,1}, \dots, \alpha_{q,k_q}, \beta_{q,k_q}$, i.e.

$$\lambda(D') \in H^*\alpha_{q,1}H^*\beta_{q,1}H^* \dots H^*\alpha_{q,k_q}H^*\beta_{q,k_q}H^*,$$

so does $\lambda(D)$, and this implies that

$$w \in A^*a_{q,1}A^* \dots A^*a_{q,k_q}A^*$$

by the definition of h. □

Theorem 19. *Each language in Σ_1 is recognizable by a conservative groupoid.*

Proof. Suppose L is in Σ_1, i.e. that

$$L = B^*b_{1,1}B^* \dots B^*b_{1,k_1}B^* \cup \dots \cup B^*b_{\ell,1}B^* \dots B^*b_{\ell,k_\ell}B^*.$$

Lemma 18 establishes the theorem for the special case where all the $b_{i,j}$ are distinct. If they are not distinct, then let $A = \{a_{1,1}, \dots, a_{1,k_1}, \dots, a_{\ell,1}, \dots, a_{\ell,k_\ell}\}$ be a new alphabet in which all $a_{i,j}$ are distinct and define

$$K = A^*a_{1,1}A^* \dots A^*a_{1,k_1}A^* \cup \dots \cup A^*a_{\ell,1}A^* \dots A^*a_{\ell,k_\ell}A^*.$$

Since K is recognizable by a conservative groupoid, it is sufficient by Lemma 11 to give a homomorphism $h : B^* \to A^*$ such that $h^{-1}(K) = L$. Suppose $B = \{c_1, \dots, c_q\}$. We define

$$h(c_r) = a_{1,k_1}^{\delta_{r,1,k_1}} a_{1,k_1-1}^{\delta_{r,1,k_1-1}} \dots a_{1,1}^{\delta_{r,1,1}} a_{2,k_2}^{\delta_{r,2,k_2}} \dots a_{2,1}^{\delta_{r,2,1}} \dots a_{\ell,k_\ell}^{\delta_{r,\ell,k_\ell}} \dots a_{\ell,1}^{\delta_{r,\ell,1}}$$

where $\delta_{r,i,j} = 1$ if $b_{i,j} = c_r$ and $\delta_{r,i,j} = 0$ if $b_{i,j} \neq c_r$. Thus $a_{i,j}$ occurs in $h(c_r)$ iff $b_{i,j} = c_r$.

Let us first show that $h(L) \subseteq K$. Suppose $w \in L$ and without loss of generality that $w \in B^*b_{1,1}B^* \dots B^*b_{1,k_1}B^*$. Then $b_{1,1}b_{1,2} \dots b_{1,k_1}$ is a subsequence of w and $h(b_{1,1})h(b_{1,2}) \dots h(b_{1,k_1})$ is a subsequence of $h(w)$. By definition of h the letter $a_{i,j}$ occurs in $h(b_{i,j})$ so $a_{1,1}a_{1,2} \dots a_{1,k_1}$ is a subsequence of $h(w)$ and thus $h(w) \in K$.

Conversely, suppose $h(w)$ contains the subsequence $a_{1,1}a_{1,2}\dots a_{1,k_1}$. Each $a_{1,j}$ in this sequence comes from some $h(c_r)$ where $b_{1,j} = c_r$. A single $h(c_r)$ may contain more than one $a_{1,j}$ but note that these occur in decreasing order with respect to j. Therefore these $a_{1,j}$ must be the result of distinct occurrences of letters in w and therefore w contains the subsequence $b_{1,1}\dots b_{1,k_1}$ and $w \in L$. □

Our results thus far show that the class of languages recognizable by conservative groupoids contains Σ_1 and is contained in Σ_2. We complete this picture by establishing two partial results that clarify the place of this class within the Straubing-Thérien hierarchy.

The class of languages that lie both in Σ_2 and in Π_2 is well-studied and admits a long list of logical, algebraic and combinatorial characterizations [16]. It is therefore natural to ask if the class of languages recognizable by conservative groupoids is contained in Π_2. As the next example shows, this is in fact not the case.

Proposition 20. *The language $\{a,b\}^*aa\{a,b\}^*$ is recognizable by a conservative groupoid. This language is known to lie outside Π_2 (see e.g. [16]).*

Proof. Let $w \in A^*$. Consider the groupoid H with the following multiplication table

	1	**2**	**3**	**4**
1	1	2	1	1
2	2	2	3	4
3	1	3	3	4
4	4	2	3	4

with $h(a) = 123$, $h(b) = 4$ and $F = \{2\}$. We claim that $w \in A^*aaA^*$ if and only if $2 \in W(h(w))$.

($\Rightarrow$)
If $w \in A^*aaA^*$, then $h(w) = x123123y$ with $x, y \in H^*$. Consider the partial bracketing $x12(31)(23)y = x1213y$. We claim that the 2 in this word can win. Indeed, to its left one finds $x1$ and we need to show that $x1 \in \Lambda(f_L(2))$. If we pick an arbitrary contest tree t in $T(x)$ then $(t(x)1) \neq 3$ since $31 = 1$. Therefore $(t(x)1) \in \{1,2,4\} = f_L(2)$. Similarly, to the right of 2 one finds $13y$. For any contest tree $s \in T(3y)$ we have $(1(s(3y))) \in \{1,2\}$ since 1 beats every element on its right except 2. Therefore $13y \in \Lambda(f_R(2))$ and by Lemma 3 we have $2 \in W(x123123y)$.

($\Leftarrow$)
For any $i, j \in \{1,2,3,4\}$, let $A_{i,j}$ denote the set of strings which begin with i, end with j and are substrings of $(1234)^n$ for some n. For instance $A_{3,2} = \{3412, 34123412, \dots, 34(1234)^i12, \dots\}$ and $A_{2,3} = \{23(4123)^i | i \geq 0\}$. In the following table, we compute upper bounds $V_{i,j}$ for the set of elements that can win on some word in $A_{i,j}$. To compute such bounds, it suffices to ensure that $V_{i,i}$ contains i and that for every i, j we have $V_{i,j} \supseteq \bigcup_k \{st | s \in V_{i,k}, t \in V_{k+1,j}\}$. We claim that Table 1 provides the minimal solution to these constraints although it is sufficient for our purposes to verify that it is a solution.

	1	**2**	**3**	**4**
1	{1,4}	{1,2,3,4}	{1,3,4}	{1,4}
2	{4}	{2,3,4}	{3,4}	{4}
3	{4}	{2,3,4}	{3,4}	{4}
4	{4}	{2,3,4}	{3,4}	{4}

Table 1: The set $V_{i,j}$ upper bounds the set of elements that can win on a substring of $(1234)^n$ that begins with i and ends with j.

Now suppose that $2 \in W(h(w))$ but assume for the sake of contradiction that $w \notin A^*aaA^*$. Because 2 wins on $h(w)$, then w has at least one a. If it has exactly one a, then $h(w) \in h(b^*ab^*) = 4^*1234^*$ and since we are interested in $W(h(w))$, we can use Lemma 17 and assume that the blocks of 4's are of length at most 1. If w has more than one a then

$$h(w) \in h(b^*ab^+ \dots b^+ab^*) = 4^*1234^+ \dots 4^+1234^*.$$

By Lemma 17, we can remove repeated occurrences of 4 and simply assume that $h(w)$ is in $4^*1234 \dots 12341234^*$ with the initial and final blocks of 4 of length at most 1. Therefore if $2 \in W(h(w))$ then 2 wins on a substring of $(1234)^n$ which begins with 1 or 4 and ends with 3 or 4 but this contradicts the upper bounds $V_{i,j}$ computed in the preceding table. $\square$

Lemma 21. *If $L \subseteq A^*$ is a language in Π_1 that can be recognized by a conservative groupoid then there exists $B \subseteq A$ such that either $L = B^*$ or $L = B^+$.*

Proof. If $L \in \Pi_1$ then $x >_L \epsilon$ for all x and this implies that $x^2 >_L x$ for all x. On the other hand, since L can be recognized by a conservative groupoid then $x^2 <_L x$ by Lemma 15. Therefore $x^2 \equiv_L x$ for all x. In particular $yz \equiv_L yzyz$ and since $x >_L \epsilon$ we get $yzyz >_L zy$. Thus $yz \equiv_L zy$. It is well known that if L satisfies $x^2 \equiv_L x$ and $yx \equiv_L xy$ then $x \equiv_L y$ whenever x and y contain the same set of letters. Now let $B = \{a \in A | a \in L\}$. Since $L = L^+$ by Lemma 14 we have $B^+ \subseteq L$ and if $\epsilon \in L$ we further have $B^* \subseteq L$. Suppose that there exists $x \in L - B^*$. This means that $x = ycz$ for some $c \notin B$. By definition of B we have $c \notin L$ but since $y >_L \epsilon$ and $z >_L \epsilon$ we get $ycz >_L c$. However this shows that $x \notin L$, a contradiction. Therefore $L \subseteq B^*$. $\square$

4. Conclusion and Future Work

We have shown that conservative groupoids can only recognize regular languages. Beaudry, Lemieux, and Thérien had previously exhibited a large class of groupoids with the same limitations [3, 2, 4] but our work is incomparable to theirs and our methods are, accordingly, quite different. It is natural to ask whether our approach can be generalized to find a wider class of "weak" groupoids and an obvious target are the 0-conservative groupoids, that is groupoids H with a 0 element such that $0 \cdot x = x \cdot 0 = 0$ for all $x \in H$ and $x \cdot y \in \{x, y, 0\}$ for all $x, y \in H$; i.e. all non-conservative products are 0.

Moreover, we have shown that the languages recognizable by conservative groupoids include all of Σ_1 and are contained in Σ_2. We also established some necessary conditions for recognizability by conservative groupoids but the picture is still incomplete and leads to some interesting open problems.

While we have shown that the class of languages recognizable by conservative groupoids is not closed under union or complement, we still do not know if it is closed under intersection. We conjecture that it is not but note that Lemmas 14 and 15 can be of no help in proving this since they are based on necessary conditions that are preserved by intersection.

Another interesting question concerns the optimality of our constructions. With $A = \{a\}$ the language $C_k = \{a^t : t \geq k\}$ is in Σ_1 since it is represented by the expression $A^*aA^*aA^* \ldots aA^*$ (with k as). The construction of Lemma 18 shows that C_k can be recognized by a conservative groupoid of size $2k$. Intuitively, one might expect that any conservative groupoid requires size at least k to recognize C_k since this language basically counts up to k. But surprisingly it is possible to count up to 6 with the following groupoid that only has five elements.

	1	**2**	**3**	**4**	**5**
1	1	2	1	1	5
2	2	2	3	4	2
3	1	3	3	3	5
4	4	2	3	4	5
5	5	2	3	5	5

We leave it as a (fun) exercise to check that if one sets $h(a) = 12345$ and $F = \{4\}$ then $h^{-1}(F) = \{a^t : t \geq 6\}$, i.e. that $W((12345)^t)$ contains 4 if and only if $t \geq 6$. We do not have any non-trivial lower bounds for the optimal size of a conservative groupoid recognizing C_k and our best upper bound is $2k$ (guaranteed by Lemma 18).

Acknowledgments

The comments of the anonymous reviewers helped to improve the paper. This work is funded in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] L. Barto, The dichotomy for conservative constraint satisfaction problems revisited, in: Proc. 26th Symp. on Logic in Comp. Sci. (LICS'11), pp. 301–310.

[2] M. Beaudry, Languages recognized by finite aperiodic groupoids, Theor. Comput. Sci. 209 (1998) 299–317.

[3] M. Beaudry, F. Lemieux, D. Thérien, Finite loops recognize exactly the regular open languages, in: Proc. 24th Int. Conf. Automata, Languages and Programming (ICALP'97), pp. 110–120.

[4] M. Beaudry, F. Lemieux, D. Thérien, Groupoids that recognize only regular languages, in: Proc. 32nd Int. Conf. Automata, Languages and Programming (ICALP'05), pp. 421–433.

[5] A. Bulatov, Tractable conservative constraint satisfaction problems., in: 18th IEEE Symp. on Logic in Comp. Sci. (LICS'03), pp. 321–331.

[6] H. Caussinus, F. Lemieux, The complexity of computing over quasigroups, in: Proc. Foundations of Software Technology and Theoretical Computer Science (FSTTCS'94), pp. 36–47.

[7] I. Chajda, Congruence semimodularity of conservative groupoids, Acta Universitatis Palackianae Olomucensis. Facultas Rerum Naturalium. Mathematica 35 (1996) 43–45.

[8] K. Etessami, Stutter-invariant languages, omega-automata, and temporal logic, in: Proc. 11th Int. Conf. on Computer Aided Verification (CAV'99), pp. 236–248.

[9] F. Gécseg, M. Steinby, Tree Automata, Akadémiai Kiadó, Budapest, 1984.

[10] J. Mezei, J.B. Wright, Algebraic automata and context-free sets, Information and Control 11 (1967) 3–29.

[11] D. Perrin, J.E. Pin, First-order logic and star-free sets., J. Comput. Syst. Sci. 32 (1986) 393–406.

[12] J.É. Pin, A variety theorem without complementation, Izvestiya VUZ Matematika 39 (1995) 80–90.

[13] J.E. Pin, Syntactic semigroups, in: Handbook of language theory, volume 1, Springer Verlag, 1997, pp. 679–746.

[14] J.É. Pin, H. Straubing, Monoids of upper triangular matrices, Colloquia Mathematica Societatis Janos Bolyai, Semigroups, Szeged 39 (1981) 259–272.

[15] J.E. Pin, P. Weil, Polynomial closure and unambiguous product, Theory Comput. Systems 30 (1997) 383–422.

[16] P. Tesson, D. Thérien, Diamonds are forever: the variety **DA**, in: Semigroups, Algorithms, Automata and Languages, WSP, 2002.

[17] W. Thomas, Classifying regular events in symbolic logic., J. Comput. Syst. Sci. 25 (1982) 360–376.

[18] L.G. Valiant, General context-free recognition in less than cubic time, J. Comput. Syst. Sci. 10 (1975) 308–314.

[19] B. Zelinka, Representation of undirected graphs by anticommutative conservative groupoids, Mathematica Bohemica 119 (1994) 231–237.