Nominal Monoids

Mikołaj Bojańczyk*
Warsaw University
bojan@mimuw.edu.pl

March 18, 2013

Abstract

We develop an algebraic theory for languages of data words. We prove that, under certain conditions, a language of data words is definable in first-order logic if and only if its syntactic monoid is aperiodic.

1 Introduction

This paper is an attempt to combine two fields.

The first field is the algebraic theory of regular languages. In this theory, a regular language is represented by its syntactic monoid, which is a finite monoid. It turns out that many important properties of the language are reflected in the structure of its syntactic monoid. One particularly beautiful result is the Schützenberger-McNaughton-Papert theorem, which describes the expressive power of first-order logic.

> A regular language of finite words is definable in first-order logic if and only if its syntactic monoid is aperiodic.

For instance, the language "words where there exists a position with label a" is defined by the first-order logic formula (this example does not even use the order on positions $<$, which is also allowed in general)

$$\exists x\ a(x).$$

The syntactic monoid of this language is isomorphic to $\{0, 1\}$ with multiplication, where 0 corresponds to the words that satisfy the formula, and 1 to the words that do not. Clearly, this monoid does not contain any non-trivial group. There are many results similar to the theorem above, each one providing a connection between seemingly unrelated concepts of logic and algebra, see e.g. the book [13].

*Author supported by ERC Starting Grant "Sosna"

The second field is the study of languages over infinite alphabets, commonly called languages of data words. Regular languages are usually considered for finite alphabets. Many current motivations, including XML and verification, require the use of infinite alphabets. As an example of an infinite alphabet, consider an infinite set $\mathbb{D}$, whose elements are called data values. A typical language, which we use as our running example, is "some two consecutive positions have the same letter", i.e.

$$L_{dd} = \bigcup_{d \in \mathbb{D}} \mathbb{D}^* dd \mathbb{D}^* = \{d_1 \cdots d_n \in \mathbb{D}^* : d_i = d_{i+1} \text{ for some } i \in \{1, \ldots, n-1\}\}.$$

As another example, consider an alphabet $A = \{a, b\} \times \mathbb{D}$, and the language "some two letters agree on the second coordinate", that is

$$\bigcup_{d \in \mathbb{D}} A^*\big((a,d) + (b,d)\big)A^*\big((a,d) + (b,d)\big)A^*.$$

A number of automata models have been developed for such languages. The general theme is that there is a tradeoff between the following three properties: expressivity, good closure properties, and decidable (or even efficiently decidable) emptiness. The existing models strike different balances in this tradeoff, and it is not clear which balance, if any, should deserve the name of "regular language". Also, logics have been developed to express properties of data words. For more information on automata and logics for data words, see the survey [12].

The motivation of this paper is to combine the two fields, and prove a theorem that is analogous to the Schützenberger-McNaughton-Papert characterization theorem, but talks about languages of data words. If we want an analogue, we need to choose the notion of regular language for data words, the definition of first-order logic for data words, and then find the property corresponding to aperiodicity.

For the sake of simplicity, we assume that the alphabet is of the form $A = \Sigma \times \mathbb{D}$, where Σ is a finite set, whose elements are called *labels*, and where $\mathbb{D}$ is the infinite set of data values. Later, we will study a more general and abstract definition of an alphabet, which will cover other cases, e.g. the set of unordered pairs of data values.

For first-order logic over data words we use a notion that has been established in the literature. Formulas are evaluated in data words. The quantifiers quantify over positions of the data word, we allow two binary predicates: $x < y$ for order on positions, and $x \sim y$ for having the same data value. Also, for each label $a \in \Sigma$ we have a unary predicate $a(x)$ that selects positions with label a. An example of a formula of first-order logic is the formula

$$\exists x \exists y \qquad x < y \quad \wedge \quad x \sim y \quad \wedge \quad \neg(\exists z\ x < z \wedge z < y),$$

which defines the language L_{dd} in the running example.

As for the notion of regular language, we use the monoid approach. For languages of data words, we use the syntactic monoid, defined in the same way

as it is for words over finite alphabets. That is, elements of the syntactic monoid are equivalence classes of the two-sided Myhill-Nerode congruence. When the alphabet is infinite, the syntactic monoid is infinite for almost every language of data words. For instance, in the case of the running example language L_{dd}, two words $w, w' \in D^*$ are equivalent if and only if: either both belong to L_{dd}, or both have the same first and last letters. Since there are infinitely many letters, there are infinitely many equivalence classes.

Instead of studying finite monoids, we study something called *orbit-finite* monoids. Intuitively speaking, two elements of a syntactic monoid are considered to be in the same orbit if there is a renaming of data values that maps one element to the other[1]. For instance, in the running example L_{dd}, the elements of the syntactic monoid that correspond to the words $1 \cdot 7$ and $2 \cdot 3 \cdot 4 \cdot 8$ are not equal, but they are in the same orbit, because the renaming $i \mapsto i+1$ maps $1 \cdot 7$ to $2 \cdot 8$, which corresponds to the same element of the syntactic monoid as $2 \cdot 3 \cdot 4 \cdot 8$. It is not difficult to see that the syntactic monoid of L_{dd} has four orbits: one for the empty word, one for words outside L_{dd} where the first and last letters are not equal, and for words outside L_{dd} where the first and last letters are equal, and one for words inside L_{dd}. The monoid is illustrated in Figure 1.

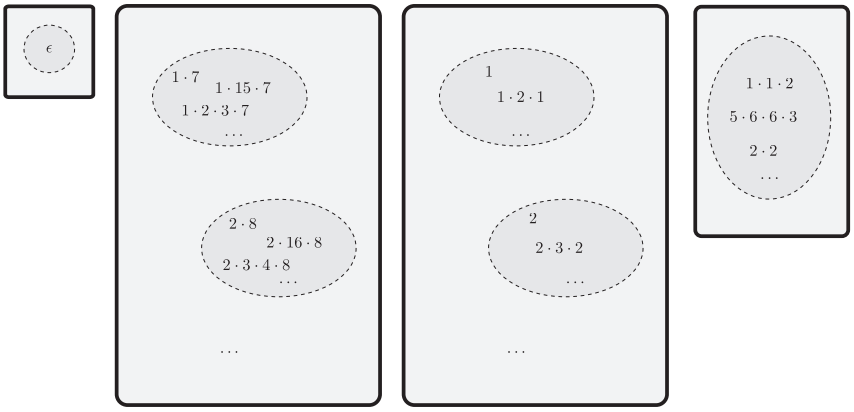

Figure 1: The syntactic monoid of L_{dd}. Rounded rectangles represent orbits, the middle two orbits are infinite. Dotted circles represent elements of the monoid, which are equivalence classes under two-sided Myhill-Nerode equivalence.

The contribution of this paper is a study of orbit-finite monoids. We develop the algebraic theory of orbit-finite monoids, and show that it resembles the theory of finite monoids. The main result of the paper is Theorem 9.1, which shows that the Schützenberger-McNaughton-Papert characterization also holds for languages of data words with orbit-finite syntactic monoids.

[1] This is related to Proposition 3 in [9].

Nominal sets. The key idea of using permutations of data values to act on the syntactic monoid of a language, or more generally, to act on any monoid, goes back to nominal sets. The theory of nominal sets originates from the work of Frankel in 1922, further developed by Mostowski in the 1930s. At that time, nominal sets were used to prove independence of the axiom of choice, and other axioms. In Computer Science, they have been rediscovered by Gabbay and Pitts in [7], as an elegant formalism for modeling name binding. Since then, nominal sets have become a lively topic in semantics. They were also independently rediscovered by the concurrency community, as a basis for syntax-free models of name-passing process calculi, see [10, 11].

The definition of monoid used in this paper is the same thing as a monoid in the category of nominal sets. That is why we call it a *nominal monoid*. The restriction on orbit-finiteness is what corresponds to finiteness in the category of nominal sets (more precisely, a nominal set is orbit-finite if and only if it is a finitely presentable object). In other words, the theory of syntactic monoids for languages of data words turns out to be the same theory as the theory of finite monoids in the category of nominal sets.

Other related work. The idea to present effective characterizations of logics on data words was proposed by Benedikt, Ley and Puppis. In [1], they show that definability in first-order logic is undecidable if the input is a nondeterministic register automaton. Also, they provide some decidable characterizations, including a characterization of first-order logic with local data comparisons within the class of languages recognized by deterministic register automata. This result is incomparable to the one in this paper, because we characterize a different logic (data comparisons are not necessarily local), and inside a weaker class of recognizers (nominal monoids have less expressive power than deterministic register automata).

There are two papers on algebra for languages over infinite alphabets. One approach is described by Bouyer, Petit and Thérien [5]. Their monoids are different from ours in the following ways: the definition of a monoid explicitly talks about registers, there is no syntactic monoid, monoids have undecidable emptiness (not mention questions such as aperiodicity or first-order definability). Our setting is closer to the approach of Francez and Kaminski from [6], but the latter talks about automata and not monoids, and does not study the connection with logics.

Progress beyond the conference version. This article is a journal version of a conference paper [2]. The main difference is that the conference version was written without examining the connection to nominal sets; this shortcoming is fixed here. The paper is written entirely using the language of nominal sets; and theorems from the nominal literature are cited instead of reproved.

Using the abstract language of nominal sets requires more abstract and general definitions. An important example is our abstract notion of first-order logic and MSO logic for words in nominal sets. In the special case of data words with alphabets of the form $\Sigma \times \mathbb{D}$, the abstract logic coincides with the standard notion of logic for data words.

Unlike the conference version, we use the more general notion of nominal sets, from [4], which allows more structure on data values, such as order instead of equality only.

2 Nominal sets

In this section we describe nominal sets. The definition is based on [4], with slight modifications.

Group action. A (right) action of a group G on a set X is a function

$$(x, \pi) \in X \times G \quad \mapsto \quad x\pi \in X$$

subject to axioms

$$x1 = x \qquad x(\pi\sigma) = (x\pi)\sigma$$

for $x \in X$ and $\pi, \sigma \in G$, where 1 is the neutral element of G. A set equipped with such an action is called a *G-set*. The point of the axioms is to make unambiguous an expression like $x\pi\sigma$.

Nominal symmetry. The notion of nominal sets is parametrized by a set $\mathbb{D}$, which is called the *set of data values*, and a group G of bijections on $\mathbb{D}$. The group G need not contain all bijections of $\mathbb{D}$. The idea is that $\mathbb{D}$ might have some structure, and G contains the structure-preserving bijections (i.e. automorphisms). The pair $(\mathbb{D}, G)$ is called a *data symmetry*. In this paper, we will use the following symmetries as the examples:

- The set $\mathbb{D}$ is empty, and G has only the identity element. We call this the *classical symmetry*.
- The set $\mathbb{D}$ is a countable set, say the natural numbers. The group G consists of all bijections on $\mathbb{D}$. We call this the *equality symmetry*.
- The set $\mathbb{D} = \mathbb{Q}$ is the set of rational numbers, and G is the set of monotone bijections. We call this *total order symmetry*.

Nominal set. Consider a data symmetry $(\mathbb{D}, G)$. When C is a set of data values, we denote by G_C the subgroup

$$\{\pi \in G : \pi|_C \text{ is the identity on } C\}.$$

If X is a G-set, then a set C is called a *support* of $x \in X$ if $x = x\pi$ for all $\pi \in G_C$. In other words, the orbit of x under the action of G_C is a singleton.

Definition 2.1 *Consider a data symmetry $(\mathbb{D}, G)$, and a finite set $C \subseteq \mathbb{D}$ of data values. A nominal set with support C is a G_C-set where every element has some finite support.*

Let X and Y be two nominal sets, with support C and D, respectively. A function $f : X \to Y$ is called a *nominal function* if it commutes with the action of G_E for some finite set E. More precisely, there must be a finite set $E \supseteq C \cup D$ such that the following diagram commutes for every $\pi \in G_E$

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle \pi}\downarrow & & \downarrow{\scriptstyle \pi} \\ X & \xrightarrow[f]{} & Y \end{array}$$

Nominal sets and nominal functions form a category, which is parametrized by the data symmetry $(\mathbb{D}, G)$. We say a nominal set is *equivariant* if it has empty support. A nominal function is called equivariant if the set E in the definition is empty. In an equivariant function, the domain and codomain must be equivariant sets.

Observe that nominal sets in the classical symmetry are simply sets (equipped with the only possible action), and nominal functions are simply functions. Therefore the classical symmetry corresponds to classical set theory, without data values.

Nominal subsets. Let X be a nominal set with support C. A subset $Y \subseteq X$ is called a *nominal subset* if there is some finite set of data values $D \supseteq C$ such that

$$x \in Y \quad \text{iff} \quad x\pi \in Y \qquad \text{for every } y \in X \text{ and } \pi \in G_D.$$

In other words, Y is a nominal set with support D, when the set X is restricted to Y and the action is restricted to G_D.

Orbit finite sets. Let X be a nominal set with support C. We say that X is *orbit-finite with respect to* C if it has finitely many orbits under the action of G_C.

Assumptions on the data symmetry. In this paper, we only consider data symmetries that satisfy the following assumptions.

- **Least supports.** Every element of a nominal set has a finite support that is least with respect to inclusion. This assumption is very useful in proofs. Intuitively it says that an element of a nominal set can be canonically represented by identifying its least support.
- **Cartesian products.** Orbit-finite sets are preserved under Cartesian products. The usefulness of this assumption should be clear: it is very difficult to get any work done without pairs and other tuples.
- **Orbit refinement.** Let $C \subseteq D$ be supports of a nominal set X. If X is orbit-finite with respect to C, then it is also orbit-finite with respect to D. The reason for this assumption is that it establishes a robust notion of "finite" set, namely the notion of an orbit-finite set, see Lemma 2.2 below.

These assumptions are satisfied by the classical, equality and total order symmetries. This was shown in [4] for least supports and Cartesian products, and in [3] for orbit refinement. For other examples of data symmetries that satisfy these properties, see [4, 3].

An example of a data symmetry that violates all three assumptions is when the data values are integers $\mathbb{Z}$, and the group consists of translations $x \mapsto x + y$. The set $\mathbb{Z}$ of data values itself is a counter-example to all three assumptions. It does not have least supports, since $0 \in \mathbb{Z}$, or any other number, is supported by every singleton, but not by the empty set. The set $\mathbb{Z}$ is orbit-finite, since it has just one orbit, but $\mathbb{Z} \times \mathbb{Z}$ has infinitely many orbits, which are diagonals of the form $\{(x, y + x) : y \in \mathbb{Z}\}$. Finally, $\mathbb{Z}$ is orbit-finite with respect to the empty set of data values, but it shatters into singleton orbits with respect to any nonempty set of data values.

As mentioned above, a corollary of orbit refinement is we can simply say that a set is orbit-finite, without saying that it is finite with respect to some C, as stated by the following lemma.

Lemma 2.2 *Let C, D be supports of a nominal set X. Then X is orbit-finite with respect to C if and only if it is orbit-finite with respect to D.*

Proof
Suppose that X is orbit-finite with respect to C. By orbit refinement, it is orbit-finite with respect to $C \cup D \supseteq C$. Because the group $G_{C \cup D}$ is a subgroup of G_D, then orbit-equivalence with respect to $G_{C \cup D}$ refines orbit-equivalence with respect to G_D. It follows that X is also orbit-finite with respect to D. □

Background. The definition of nominal sets presented above is based on, but not identical to, the one from [4]. The definition used in this paper is slightly more relaxed than the one in [4]; in [4] the category of nominal sets used only equivariant sets and equivariant objects. The difference between the nominal sets from [4] and this paper, and the nominal sets of Gabbay and Pitts [7], is that here and in [4] there is the additional parameter of a data symmetry, while in [7] the data symmetry is always assumed to be the same, namely $\mathbb{D}$ is the natural numbers and G is the set of all permutations of natural numbers.

3 Nominal monoids

Fix a data symmetry $(\mathbb{D}, G)$. When talking about nominal sets and functions below, we mean nominal under this symmetry. In the category of nominal sets, there is a natural definition of a monoid:

Definition 3.1 *A* nominal monoid *is a monoid, where the carrier is a nominal set, and the concatenation operation is nominal. If M, N are nominal monoids, then a nominal monoid morphism $\alpha : M \to N$ is a function that is both a nominal function and a monoid morphism.*

Example: Consider the running example L_{dd}, which is in the equality symmetry. We now describe the syntactic monoid of this language, which is illustrated in Figure 1. The carrier of the monoid is the set

$$\{0, 1\} \quad \cup \quad \mathbb{D}^2.$$

The 0 and 1 are not atoms, but the traditional zero and identity elements of a monoid: anything multiplied by 0 is 0, and anything mulitplied by 1 is itself. This carrier has empty support. The action of G is natural: it does nothing to the elements 0 and 1, and it renames the coordinates of the pairs in $\mathbb{D}^2$. The carrier has four orbits: two singleton orbits $\{0\}$ and $\{1\}$, an orbit for the diagonal $\{(d, d) : d \in \mathbb{D}\}$, and an orbit for the co-diagonal $\{(d, e) : d \neq e \in \mathbb{D}\}$. It is easy to see that every element of the carrier has finite support: the elements 0 and 1 have empty supports, while an element (d, e) has support $\{d, e\}$. Finally, multiplication for elements that are not 0 or 1 is defined by

$$(d_1, d_2)(e_1, e_2) = \begin{cases} (d_1, e_2) & \text{if } d_2 \neq e_1 \\ 0 & \text{otherwise.} \end{cases}$$

This operation is clearly nominal, it has empty support. The language L_{dd} from the running example is recognized by the monoid morphism defined by

$$\alpha(d_1 \cdots d_n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } d_1 \cdots d_n \in L_{dd} \\ (d_1, d_n) & \text{otherwise.} \end{cases}$$

The accepting set is $\{1\}$. This particular accepting set is simply finite, but for the complement of L_{dd} we have an orbit-finite, but not finite, accepting set. □

The rest of this section is devoted to showing that the properties of monoids as language recognizers work just as well in the nominal setting, including the free monoid and the syntactic monoid

3.1 Free monoid.

Let A be any nominal set, with support C. Define A^* to be the set of words over A, with support C and the action defined coordinatewise:

$$(a_1 \cdots a_n)\pi = (a_1\pi) \cdots (a_n\pi) \qquad \text{for } \pi \in G_C.$$

It is easy to see that A^* is a nominal set, because the word $a_1 \cdots a_n$ is supported by the union of supports of its letters $a_1, \ldots, a_n$. (Observe that this would fail for infinite words.) The following lemma shows that A^* deserves to be called free.

Lemma 3.2 *Let M be a nominal monoid and A a nominal set. Every nominal function $\alpha : A \to M$ can be uniquely extended to a nominal monoid morphism $[\alpha] : A^* \to M$.*

Proof
The proof is routine. Suppose that C is a set of constants that includes all the constants in A, α and M. If the extension $[\alpha]$ is to be a nominal monoid morphism, it needs to satisfy

$$[\alpha](a_1 \cdots a_n) = \alpha(a_1) \cdots \alpha(a_n) \qquad \text{for every } a_1, \ldots, a_n \in A.$$

Therefore, if $[\alpha]$ exists, then it is unique. Since every word $w \in A^*$ has a unique decomposition $w = a_1 \cdots a_n$, the above can be seen as the definition of $[\alpha]$. It remains to check that $[\alpha]$ is a nominal function, that is

$$([\alpha](a_1 \cdots a_n))\pi = [\alpha]((a_1 \cdots a_n)\pi) \qquad \text{for every } \pi \in G_C.$$

By definition of the function $[\alpha]$, the left side of the equality becomes:

$$(\alpha(a_1) \cdots \alpha(a_n))\pi.$$

Since the concatenation in M commutes with π, the above becomes

$$((\alpha(a_1))\pi) \cdots ((\alpha(a_n))\pi).$$

Because α is a nominal function, it commutes with π, and therefore the above becomes

$$(\alpha(a_1\pi)) \cdots (\alpha(a_n\pi)).$$

By definition of $[\alpha]$, the above becomes

$$[\alpha]((a_1\pi) \cdots (a_n\pi)).$$

Finally, by definition of the concatenation operation in A^*, the above becomes

$$[\alpha]((a_1 \cdots a_n)\pi),$$

which is what we wanted to prove.□

Recognition. A nominal monoid morphism $\alpha : A^* \to M$ is said to *recognize* a nominal language $L \subseteq A^*$ if there is a nominal subset $F \subseteq M$ such that

$$L = \alpha^{-1}(F).$$

In this paper, we are mainly interested in the case when M is orbit-finite.

3.2 Syntactic monoid.

A *nominal alphabet* is any nominal orbit-finite set.

Examples of alphabets. Here are some examples of nominal alphabets, in the equality symmetry.

- The set $\mathbb{D}$ of data values.
- The product $\Sigma \times \mathbb{D}$, where Σ is a finite set and the action is defined by

$$(a, d)\pi = (a, d\pi).$$

- Pairs of data values: $\mathbb{D}^2$.
- Unordered pairs of data values: $\{\{d, e\} : d \neq e \in \mathbb{D}\}$.
- Recall that in the definition of a nominal set, Definition 2.1, a nominal set comes with a support C. In the examples above, the support was empty. An example of a set with nonempty support $\{d\}$ is the set of data values with some chosen value d being excluded: $\mathbb{D} - \{d\}$.

In all examples except for the last one, the support of the alphabet is empty. In the last example, the support is $\{d\}$.

Nominal language. Consider a nominal alphabet A. A *nominal language over A* is a nominal subset of A^*. It is easy to see that any language recognized by a nominal monoid morphism is a nominal language. Using the syntactic morphism, we will also prove the converse in Lemma 3.3.

Two-sided Myhill-Nerode congruence. As usual for any language $L \subseteq A^*$, one can define the two-sided Myhill-Nerode equivalence relation $\equiv_L$ on A^* by

$$w \equiv_L w' \qquad \text{iff} \qquad uwv \in L \text{ iff } uw'v \in L \qquad \text{for all } u, v \in A^*.$$

The *syntactic monoid* of L is defined to be the set of equivalence classes of A^* under $\equiv_L$, with concatenation defined by

$$[w]_{\equiv_L}[v]_{\equiv_L} \qquad \stackrel{\text{def}}{=} \qquad [wv]_{\equiv_L},$$

where wv denotes concatenation of words in A^*. As is well known, the above is a proper definition (it does not depend on the choice of w and v), the resulting concatenation operation is associative, its neutral element is the equivalence class of the empty word, and the function $w \mapsto [w]_{\equiv_L}$ is a monoid morphism, which is denoted by α_L and called the *syntactic morphism.* The syntactic morphism recognizes L.

Lemma 3.3 *If $L \subseteq A^*$ is a nominal language, then its syntactic monoid is a nominal monoid, and the syntactic morphism is a nominal monoid morphism.*

Proof
The assumption in the lemma is that for some support D that contains the support of the alphabet, $L = L\pi$ holds for all $\pi \in G_D$.

We first define an action of G_D on the syntactic monoid, which consists of equivalence classes under $\equiv_L$. The action is defined by

$$[w]_{\equiv_L}\pi = [w\pi]_{\equiv_L} \qquad \text{for all } \pi \in G_D.$$

We first show that this is a correct definition, i.e. it does not depend on the choice of w in the equivalence class. Suppose then that $w \equiv_L w'$. We need to show that $w\pi \equiv_L w'\pi$. This follows immediately from the definition of $\equiv_L$ and the assumption that $L\pi = L$. It is easy to see that every element of the syntactic monoid has finite support, namely $[w]_{\equiv_L}$ is supported by whatever supports w. This shows that the syntactic monoid is a nominal set.

Because the syntactic morphism is $w \mapsto [w]_{\equiv_L}$, the very definition of the action in the syntactic monoid proves that the syntactic morphism is a nominal function. □

The following lemma shows that the syntactic morphism is the "best morphism" for recognizing a language.

Lemma 3.4 *Consider a nominal language $L \subseteq A^*$. For every surjective nominal monoid morphism $\alpha : A^* \to M$ that recognizes L, there is a unique β such that $\beta \circ \alpha$ is the syntactic morphism.*

Proof
If there is an extension, then it is unique, so the only question is whether the extension exists. Suppose that $\alpha : A^* \to M$ is a surjective morphism that recognizes L. Let $w, w' \in A^*$ be such that $\alpha(w) = \alpha(w')$. Then for all words $u, v \in A^*$, we have

$$\alpha(uwv) = \alpha(u)\alpha(w)\alpha(v) = \alpha(u)\alpha(w')\alpha(v) = \alpha(uw'v)$$

and therefore, because membership in L depends only on the image under α, it follows that $uwv \in L$ if and only if $uw'v \in L$. We have just shown that if $\alpha(w) = \alpha(w')$, then $w \equiv_L w'$. Therefore, it makes sense to define a function

$$\beta : M \to M_L \qquad \beta(\alpha(w)) = [w]_{\equiv_L},$$

which shows that α extends to the syntactic morphism. □

4 Logic

In this section, we give a definition of MSO and first-order logics for words in any data symmetry. The definition is abstract, but in the special case of the equality symmetry it specializes to the well known MSO and first-order logics, which access data values via a binary data equality predicate.

Let A be an orbit-finite alphabet. Define a relational vocabulary τ_A, which has one binary relation symbol $<$, and one n-ary relation symbol R for every n-ary nominal subset $R \subseteq A^n$. The vocabulary is infinite.

A word $w = a_1 \cdots a_k \in A^*$ can be treated as relational structure $\underline{w}$ over the vocabulary τ_A. The carrier of the structure is the set $\{1, \ldots, k\}$ of positions. There relation $<$ is interpreted as the linear order on positions. Finally, a relation $R \subseteq A^n$ is interpreted in $\underline{w}$ as the set

$$\{(i_1, \ldots, i_n) \in \{1, \ldots, k\}^n : (a_{i_1}, \ldots, a_{i_k}) \in R\}.$$

When talking about *nominal first-order logic for words over A*, we refer to first-order logic over the vocabulary τ_A. A sentence is said to be true in a word w if it is true in the structure $\underline{w}$. Likewise for MSO.

Example 1. Consider the equality symmetry, and an alphabet $A = \Sigma \times \mathbb{D}$ where Σ is finite. The standard definition of first-order logic for data words has a predicate $<$ for order, a predicate $\sim$ for equality on the data value, and a unary predicate $a(x)$ for every $a \in \Sigma$. We also add a unary predicate $d(x)$ for every $d \in \mathbb{D}$, which selects positions that carry d as their data value. Call this *standard first-order logic for words over A*. We show that the predicates of the standard logic are available in the nominal logic. The order $<$ is built in. For the predicate $\sim$, it comes from the relation $R_\sim \subseteq A \times A$ defined by

$$R_\sim = \bigcup_{d \in \mathbb{D}} (\Sigma \times \{d\}) \times (\Sigma \times \{d\}),$$

which is nominal. The label predicate $a(x)$ comes from the relation $R_a \subseteq A$ defined by

$$R_a = \{a\} \times \mathbb{D},$$

which is also nominal. It follows that the nominal logic is at least as powerful as the standard logic. We will show in Theorem 4.1 that it actually has the same expressive power, in the case of alphabets of the form $\Sigma \times \mathbb{D}$. □

Example 2. Consider the equality symmetry, and an alphabet

$$A = \binom{\mathbb{D}}{2} \stackrel{\text{def}}{=} \{\{d, e\} : d \neq e \in \mathbb{D}\}.$$

This alphabet is a two-orbit nominal set, with the pointwise action. The standard definition of first-order logic for data words, discussed in the previous example, does not extend to alphabets such as A. A typical language we might study is the set of words where the first letter and the last letter have nonempty intersection:

$$L = \bigcup_{\substack{a,b \in A \\ a \cap b \neq \emptyset}} aA^*b.$$

This can be done in our nominal first-order logic for words over A, using the predicate $R \subseteq A \times A$,

$$R = \{(a, b) : a \cap b \neq \emptyset\},$$

which is a nominal subset of A^2.

Let us give the syntactic monoid of the language L. A nominal monoid morphism which recognises this language maps a nonempty word to its first last letter:

$$a_1 \cdots a_n \in A^* \qquad \mapsto \qquad (a_1, a_n) \in A^2.$$

For the empty word we need a special value, call it 1. The nominal monoid in the target of this morphism is therefore $A^2 \cup \{1\}$, with 1 as the identity and the monoid operation defined by

$$(a, b)(c, d) = (a, d).$$

It is not difficult to see that we have actually described the syntactic nominal monoid. Indeed, if two words w, w' disagree on their first or last letter, then one can choose words $x, y \in A^*$ such that exactly one of the words xwy and $xw'y$ is in the language. □

4.1 Nominal vs standard logic

We now prove the result that was announced in Example 1, namely that nominal and standard first-order logics coincide in the special case of the equality symmetry and alphabets of the form $\Sigma \times \mathbb{D}$.

Theorem 4.1 *Consider the equality symmetry, and an alphabet $A = \Sigma \times \mathbb{D}$, with Σ a finite set. Then the standard and nominal first-order logics for words over A have the same expressive power.*

We have already shown in Example 1 that the standard logic is included in the nominal logic. For the converse, we show that for every nominal relation

$$R \subseteq (\Sigma \times \mathbb{D})^n,$$

the predicate corresponding to R is definable in standard first-order logic. Toward this end, we use the following lemma.

Lemma 4.2 *R is a finite boolean combination of relations of the form:*

$$\begin{array}{ll} \{((a_1, d_1), \ldots, (a_n, d_n)) \in (\Sigma \times \mathbb{D})^n : a_i = a\} & \text{for } a \in \Sigma \text{ and } i \in \{1, \ldots, n\} \\ \{((a_1, d_1), \ldots, (a_n, d_n)) \in (\Sigma \times \mathbb{D})^n : d_i = d_j\} & \text{for } i, j \in \{1, \ldots, n\} \\ \{((a_1, d_1), \ldots, (a_n, d_n)) \in (\Sigma \times \mathbb{D})^n : d_i = d\} & \text{for } d \in \mathbb{D} \text{ and } i \in \{1, \ldots, n\} \end{array}$$

Furthermore, the relations of the last type are used only for data values d in the least support of R.

Proof
Let D be the least support of the alphabet. Let $C \subseteq \mathbb{D}$ be the least support of R, this set includes D. By definition of support of a set, R is a union of orbits under the action of the group G_C, i.e.

$$R = \bigcup_{\bar{a} \in R} \bar{a} G_C.$$

Because orbit-finite sets are closed under Cartesian products, then also A^n is orbit-finite. By the orbit-refinement property, A^n has finitely many orbits under

the action of any group with fixed support, in particular under the action of G_C. It follows that R can be described as a finite union

$$R = \bar{a}_1 G_C \cup \cdots \cup \bar{a}_k G_C \qquad \text{for tuples } \bar{a}_1, \ldots, \bar{a}_k \in (\Sigma \times \mathbb{D})^n.$$

To prove the lemma, it is enough to show that for every tuple $\bar{a} \in A^n$ and every finite set of data values C, the set $\bar{a}G_C$ is a finite boolean combination of languages as in the statement of the lemma. Let us inspect the coordinates of the tuple $\bar{a}$

$$\bar{a} = ((a_1, d_1), \ldots, (a_n, d_n)) \in (\Sigma \times \mathbb{D})^n$$

It is not difficult to show that a tuple

$$\bar{b} = ((b_1, e_1), \ldots, (b_n, e_n)) \in (\Sigma \times \mathbb{D})^n$$

belongs to $\bar{a}G_C$ if and only if

- For every $i \in \{1, \ldots, n\}$, the label b_i is a_i.
- For every $i \in \{1, \ldots, n\}$, if $d_i \in C$, then $e_i = d_i$.
- For every $i \in \{1, \ldots, n\}$, if $d_i \notin C$, then $e_i \notin C$.
- For every $i, j \in \{1, \ldots, n\}$ if $d_i = d_j$ then $e_i = e_j$.

All the conditions above are special cases of the relations in the statement of the lemma. □

Theorem 4.1 follows immediately from the lemma above, because the relations in standard first-order logic for data words can capture the predicates corresponding to the relations in the statement of Lemma 4.2. Observe also that, thanks to the "furthermore" clause in the lemma, if a formula of nominal first-order logic uses only equivariant predicates, then its corresponding formula in standard first-order logic does not need to use the predicates $d(x)$ for $d \in \mathbb{D}$.

The same proof as for Theorem 4.1 would work in other symmetries, such as the total order symmetry, except that $\sim$ would need to be replaced by the ordering on the data values.

5 Local finiteness

A monoid is called *locally finite* if every finitely generated submonoid of M is finite. For example, the monoid $\mathbb{N}$ with addition is not locally finite, because the submonoid generated by $\{1\}$ is the whole infinite set $\mathbb{N}$. Below is an example of an infinite but locally finite monoid.

Example 3. Let X be an infinite set. Consider the monoid where elements are subsets of X, and the monoid operation is set union. This monoid is infinite. However, any finite set of n generators will generate a submonoid of at most 2^n elements, hence the monoid is locally finite. □

In this section we show that every orbit-finite nominal monoid is locally finite. As we shall see, this implies that most of Green's theory can be used in orbit-finite monoids.

Theorem 5.1 *Every orbit-finite nominal monoid is locally finite.*

In the proof, we use the following lemma.

Lemma 5.2 *Let X be an orbit-finite nominal set, and C a finite set of data values. There are finitely many elements in X that are supported by C.*

Proof
We assume without loss of generality that X has one orbit. Let D be the least support of the set X itself.

If an element $x \in X$ is supported by C, then its least support (which exists by the least support assumption) is a subset $E \subseteq C$, which must satisfy $D \subseteq E$. There are finitely many subsets of C. Therefore, the lemma will follow once we show that for every $E \supseteq D$ there are finitely many elements $x \in X$ whose least support is E.

Consider the set $Y \subseteq X$ of elements with least support E. Because the set X has a single orbit, we know that for every $x, y \in X$ there is some $\pi_{xy} \in G_D$ such that $y = x\pi_{xy}$. In [4] it is shown that

$$E \text{ is the least support of } x \qquad \text{implies} \qquad \pi_{xy}(E) \text{ is the least support of } x\pi_{xy}$$

and therefore for every $x, y \in Y$, the function π_{xy}, when restricted to E, is a permutation of E. Toward a contradiction, suppose that Y is infinite. Because there are finitely many permutations of E, the pigeon-hole principle says that for every $x \in Y$ there must be different elements $y \neq z \in Y$ such that π_{xy} and π_{xz} induce the same permutation on E. It follows that

$$\pi_{xy}(\pi_{xz})^{-1} \in G_E.$$

Therefore, by assumption that E supports x, we get

$$x\pi_{xy}(\pi_{xz})^{-1} = x,$$

which contradicts our assumption on y, z being different:

$$y = x\pi_{xy} = x\pi_{xz} = z.$$

□

Proof (of Theorem 5.1)
Let M be an orbit-finite monoid. Suppose that $X \subseteq M$ is a finite set of generators, and let N be the submonoid of M generated by X. It is easy to see that if a set of data values C supports $m, n \in M$ and the monoid operation, then it also supports mn:

$$mn\pi = (m\pi)(n\pi) = mn \qquad \text{for } \pi \in G_C.$$

Let C be a finite set that supports every element from X, and which also supports the monoid. By the observation above, the set supports every element of N. The result follows by Lemma 5.2. □

Aperiodic monoid. A monoid M is called *aperiodic* if

$$\forall m \in M \quad \exists i \in \mathbb{N} \qquad m^i = m^{i+1}.$$

We say that a monoid M *contains a group* if there is some subset $G \subseteq M$, such that M restricted to G with the same operation is a group, possibly with the group identity not being the same as the monoid identity. The following lemma shows that for locally finite monoids, containing a non-trivial group is the same as violating aperiodicity.

Lemma 5.3 *For a locally finite monoid, the following conditions are equivalent*

- *M does not contain a nontrivial subgroup.*
- *M is aperiodic.*

Proof
Suppose first that M is aperiodic. We show that there can be no subgroup. For the sake of contradiction, suppose that G is a subgroup. Let $g \in G$ be an element different than the group identity. By aperiodicity, there must be some $i \in \mathbb{N}$ such that $g^i = g^{i+1}$, and by multiplying both sides by g^{-i}, we get that $1 = g$, where 1 is the identity of G.

In the converse implication, we use local finiteness. Suppose that M is not aperiodic, i.e. there is some a such that $a^i \neq a^{i+1}$ holds for all i. By local finiteness, the set $\{a^i\}_{i \in \mathbb{N}}$ is a finite. It follows that there must be some j such that $a^{i+j} = a^i$. In this case the set

$$\{a^i, a^{i+1}, \ldots, a^{i+j-1}\}$$

forms a cyclic group. □

Observe that the two conditions in Lemma 5.3 are not equivalent in all monoids. For instance, the group of integers with addition is aperiodic.

6 Local finiteness for MSO

As an added bonus, we prove a much stronger result than Theorem 5.1. This section is independent from the rest of the paper, and may be skipped by the reader who is only interested in the main result, Theorem 9.1

Theorem 6.1 *Let A be an orbit-finite alphabet, and let $L \subseteq A^*$ be a language definable in nominal MSO. Then the syntactic monoid of L is locally finite.*

Theorem 6.1 was stated in the conference paper but not proved. This theorem may come useful in future work. For instance, we might want to develop an algebraic theory of languages recognized by deterministic, or even nondeterministic, finite nominal automata, as defined in [4]. These automata can be simulated in MSO, and therefore Theorem 6.1 says that Green's theory can be used for them. The rest of this section is devoted to the proof of Theorem 6.1.

Fiber bounded morphism. A nominal morphism $f : A^* \to B^*$ is called *fiber bounded* if for every $w \in B^*$, the preimage $f^{-1}(w)$ is finite. Examples of nominal morphisms that are not fiber bounded include any morphism which maps some letter to the empty word, or the morphism

$$f : (\mathbb{D} \times \mathbb{D})^* \to \mathbb{D}^*$$

that maps all letters to their first coordinate.

Language characterization of MSO. When proving Theorem 6.1, we want to avoid formulas of logic, so that we do not have to talk about free variables and other nuisances. That is why we state below a characterization of MSO that is purely in terms of languages.

Lemma 6.2 *Languages definable in nominal MSO are included*[2] *in the smallest class of languages that contains languages recognized by orbit-finite nominal monoids, is closed under boolean combinations and images under fiber bounded morphisms.*

Proof (of Lemma 6.2)
The same proof as in the case of finite alphabets, see e.g. [14]. First, we eliminate first-order quantifiers from the language, by adding a subset predicate $X \subseteq Y$ for set variables, and replacing every k-ary predicate R by a set version

$$R' = \{(\{x_1\}, \ldots, \{x_k\}) : (x_1, \ldots, x_k) \in R\}.$$

The resulting logic has the same expressive power, and it does not use first-order variables. Next, we show how a formula with free variables can be treated as a language. For a word $w \in A^*$ and a set of X of positions in w, define $w \otimes X \in (A \times 2)^*$ by extending the label of each position with a bit, which says if the position belongs to X. The language of a formula $\varphi(X_1, \ldots, X_n)$ is defined as

$$L_\varphi = \{w \otimes X_1 \otimes \cdots \otimes X_n : w, X_1, \ldots, X_n \models \varphi\} \in (A \times 2^n)^*.$$

[2]The inclusion is strict. Consider the language described in Section III.A of an unpublished manuscript, `http://www.mimuw.edu.pl/ bojan/papers/atomturing.pdf`. This is an example of a language that is not definable in nominal MSO (because it is not even recognised by a deterministic Turing machine with atoms, and the machines are more powerful than nominal MSO), but which is an image, under a fiber bounded morphism, of a language definable in nominal MSO (because having an even number of conflicts, as defined in the manuscript, is definable in nominal MSO).

Under this interpretation, we see that the existential set quantifiers (boolean combinations, respectively) on the side of formulas correspond to morphic images (boolean combinations, respectively) on the side of languages. The morphisms are fiber bounded, because they simply remove one 2 component from the alphabet (i.e. the alphabet is halved).

It remains to show that the languages in the induction base, which correspond to predicates, are recognized by orbit-finite nominal monoids. Consider the language that corresponds to a formula

$$R'(X_{i_1}, \ldots, X_{i_n}),$$

which uses the set version R' of a predicate R. When R is the order predicate on positions, then the monoid is simply finite. When R is obtained from a nominal relation $R \subseteq A^k$, we use the following monoid M_R to recognize the language that corresponds to the relation R'. Elements of the monoid are partial functions

$$\eta : \{1, \ldots, k\} \to A,$$

plus an additional zero element 0. The zero element is used for words where the sets $X_{i_1}, \ldots, X_{i_n}$ are not singletons. The monoid operation is defined by

$$\eta_1 \eta_2 = \begin{cases} \eta_1 \cup \eta_2 & \text{if } \eta_1, \eta_2 \text{ are not 0 and have disjoint domains} \\ 0 & \text{otherwise} \end{cases}$$

The monoid identity is the completely undefined partial function. It is easy to see that the monoid is orbit-finite. We now give a monoid morphism

$$\alpha_R : (A \times 2^k)^* \to M_R$$

which recognizes the predicate R. A word

$$a_1 \cdots a_n \otimes X_1 \otimes \cdots \otimes X_n$$

is mapped by α_R to 0 if one of the sets $X_1, \ldots, X_n$ contains at least two positions. Otherwise, the word is mapped to the partial function

$$\eta(i) = \begin{cases} a_j & \text{if } X_i = \{j\} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

□

Example 4. We show in this example that it is important to consider images under fiber bounded morphisms, because other morphisms can lead to monoids that are not locally finite. Consider the language

$$L = \{d_1 \cdots d_k : d_1, \ldots, d_k \in \mathbb{D} \text{ are pairwise different and } k \text{ is prime}\} \subseteq \mathbb{D}^*.$$

The syntactic monoid of this language contains all infixes of words in L as distinct elements, and a special zero element for all other words. The monoid is locally finite, because it satisfies the following property:

$$m_1 \cdots m_k = 0 \qquad \text{whenever } m_i = m_j \text{ for some } i < j.$$

In particular, for a given set X of generators, the number of elements in the monoid generated by X is bounded by the number of words over X that use each letter at most once.

Consider now the morphism $f : \mathbb{D}^* \to a^*$ which replaces all letters by a. This morphism is not fiber bounded, because the inverse image of a is $\mathbb{D}$. The image of L under f is the set of words of prime length. The syntactic monoid of that language is not locally finite, because it is finitely generated by $\{a\}$ and infinite.

□

We now prove Theorem 6.1. By Lemma 6.2, it is enough to prove that the syntactic monoids of the class of languages in the lemma are locally finite. We first show two operations on monoids that correspond to boolean combinations and fiber bounded morphisms, respectively.

Monoid operations. The first operation is Cartesian product $M \times N$. It is easy to see that the Cartesian product of two nominal monoids is also monoid. The second operation is the finite powerset construction, which defines a monoid $\mathsf{P}_{\text{fin}}(M)$ based on a monoid M. Elements of $\mathsf{P}_{\text{fin}}(M)$ are finite subsets of M. The monoid operation is defined pointwise, by

$$XY = \{xy : x \in X, y \in Y\} \qquad \text{for } X, Y \subseteq M.$$

It is easy to see that $\mathsf{P}_{\text{fin}}(M)$ is nominal if and only if M is nominal. The following straightforward lemma, which is given without proof, shows that the monoid operations above capture the language operations from Lemma 6.2.

Lemma 6.3 *Let L, K be languages, recognized by monoids M, N respectively.*

- *Any boolean combination of L and K is recognized by the product $M \times N$.*
- *Any fiber bounded projection of L is recognized by $\mathsf{P}_{\text{fin}}(M)$.*

A corollary of Lemmas 6.2 and 6.3 is that every language definable in MSO is recognized by a monoid from the least class of monoids that contains orbit-finite monoids and is closed under Cartesian products and finite powersets. By Theorem 5.1, all orbit-finite nominal monoids are locally finite. In Lemmas 6.4 and 6.5 below, we show that locally finite monoids are closed under Cartesian products and finite powersets, respectively. It follows that every language definable in MSO is recognized by a locally finite monoid. Finally, by Lemma 6.6 below, and Lemma 3.4, it follows that the syntactic monoid of every language definable in MSO is locally finite.

Lemma 6.4 *If monoids M_1, M_2 are locally finite, then so is $M_1 \times M_2$.*

Proof
Let X be a finite subset of $M_1 \times M_2$. For $i \in \{1, 2\}$ let N_i be the submonoid of M_i generated by the projection of X to the i-th coordinate. By assumption on M_1, M_2 being locally finite, the submonoids N_1, N_2 are finite. The submonoid generated by X in $M_1 \times M_2$ is a subset of $N_1 \times N_2$, and therefore it is also finite. □

Lemma 6.5 *If monoid M is locally finite, then so is $\mathsf{P}_{\text{fin}}(M)$.*

Proof
Let $\mathcal{X}$ be a finite set of elements in $\mathsf{P}_{\text{fin}}(M)$. Let X be the union $\bigcup \mathcal{X}$, which is a finite set, because it is a finite union of finite sets. Let N be the submonoid of M that is generated by X. By local finiteness of M, the monoid N is finite. Let $\mathcal{N}$ be the submonoid of $\mathsf{P}_{\text{fin}}(M)$ generated by $\mathcal{X}$. By induction one shows that every element of $\mathcal{N}$ is a subset of N. By finiteness of N, there are finitely many elements in $\mathcal{N}$. □

Lemma 6.6 *Let $\alpha : M \to N$ be a surjective monoid morphism. If M is locally finite, then so is N.*

Proof
Let X be a finite subset of N. Using surjectivity, choose a finite subset Y of M such that $\alpha(Y) = X$. By local finiteness of M, the monoid generated by Y in M is finite; and therefore so is its image in N, which is the same as the monoid generated by X in N. □

7 Green's relations for nominal monoids

In this section we recall Green's relations. Suppose that M is a nominal monoid. Using the underlying monoid, one can define Green's relations on elements $m, n \in M$.

- $m \leq_{\mathcal{L}} n$ if $Mm \subseteq Mn$
- $m \leq_{\mathcal{R}} n$ if $mM \subseteq nM$
- $m \leq_{\mathcal{J}} n$ if $MmM \subseteq MnM$

We sometimes say that n is a *suffix* of m instead of writing $m \leq_{\mathcal{L}} n$. Likewise for *prefix* and *infix*, and the relations $\leq_{\mathcal{R}}$ and $\leq_{\mathcal{J}}$. Finally, we use the name *extension* for the converse of infix. Like in any monoid, these relations are transitive and reflexive, so one can talk about their induced equivalence relations, which are denoted by $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{J}$. Equivalence classes of these relations are called $\mathcal{L}$-classes, $\mathcal{R}$-classes and $\mathcal{J}$-classes. An $\mathcal{H}$-class is defined to be the intersection of an $\mathcal{L}$-class and an $\mathcal{R}$-class.

It is easy to see that Green's relations are nominal relations, with their support being the support of the monoid. The following lemma[3] shows that

[3] As pointed out by one of the anonymous referees, every locally finite monoid is group bound, which implies Lemma 7.1. See e.g. [8].

two key properties of Green's relations work in all locally finite monoids, which covers the case of orbit-finite nominal monoids thanks to Theorem 5.1.

Lemma 7.1 *The following properties hold in every locally finite monoid M.*

- *If $n \mathcal{J} m$ and $m \leq_{\mathcal{R}} n$, then $m \mathcal{R} n$. Likewise for $\mathcal{L}$.*
- *If M is aperiodic, then $m \mathcal{L} n$ and $m \mathcal{R} n$ imply $m = n$.*

Proof
It is well known that the properties above hold in finite monoids.

Consider the first property. By the assumptions, there exist elements $x, y, z \in M$ such that

$$m = nx \qquad n = ymz.$$

Consider the submonoid $N \subseteq M$ that is generated by $\{n, m, x, y, z\}$. This monoid is finite by the local finiteness assumption. In the submonoid N, the assumptions $n \mathcal{J} m$ and $m \leq_{\mathcal{R}} n$ are also satisfied, and therefore so is the conclusion $m \mathcal{R} n$. This means there is some $x' \in N$ with $n = mx'$. Because x' belongs also to M, it follows that $n \mathcal{R} m$ holds in M.

The same argument works for the second property. □

8 First-order definable functions

In this section, we define when a partial function

$$f : A^* \to B$$

is first-order definable, when A and B are nominal orbit-finite sets. We will use first-order definable functions as a technical tool in the proof of our characterization of first-order logic. Before defining the notion, we present some examples of functions that will turn out to be first-order definable.

Example 5. $B = 2$ and f says if some data value appears twice. More generally, the characteristic function of any first-order definable language is a first-order definable function, see Lemma 8.1. □

Example 6. $B = A$ and f returns the first letter. This is a partial function, because f is undefined on the empty word. Unlike in Example 5, the output set is infinite, and therefore f cannot be described just by using sentences of first-order logic. □

Example 7. B is the family of two element subsets of A, and f returns the unordered set containing the first and last letters. Again, f is a partial function. □

In the classical setting, where orbit-finite alphabets are simply finite, the notion of first-order definable function is just syntactic sugar on top of first-order definable languages. This is because a function $f : A^* \to B$ can be

described as finite set of sentences $\{\varphi_b\}_{b\in B}$, such that φ_b defines the language $f^{-1}(b)$. However, when orbit-finite sets are actually infinite, the ability to return letters and data values becomes important. In Section 8.1, we come back to the idea of representing a function as a set $\{\varphi_b\}_{b\in B}$.

Definition of first-order definable functions. The class of first-order definable functions is defined below.

- **Boolean properties.** Suppose that φ is a first-order sentence. Then the characteristic function $f : A^* \to 2$ is a first-order definable function.
- **Letter selection.** Suppose that $\varphi(x)$ is a first-order formula with one free variable. Then the function $f : A^* \to A$, which maps a word w to the label of the first position selected by $\varphi(x)$, is first-order definable. The function f is partial, because in some words no position is selected by $\varphi(x)$.
- **Product and case.** If functions $f : A^* \to B$ and $f : A^* \to C$ are first-order definable, then so are their product and sum

$$(f,g) \quad : \quad A^* \to B \times C \qquad w \mapsto (f(w), g(w))$$

$$f \text{ else } g \quad : \quad A^* \to B \cup C \qquad w \mapsto \begin{cases} f(w) & \text{if } f(w) \text{ defined} \\ g(w) & \text{otherwise} \end{cases}$$

  The product (f,g) is defined whenever both f and g are defined.
- **Information loss.** If the function $f : A^* \to B$ is first-order definable, and $g : B \to C$ is any nominal function, then the composition $f;g : A^* \to C$ is also first-order definable.

Observe that the codomain of every first-order definable function is orbit-finite. This is because orbit-finite sets are preserved under Cartesian products (an assumption that we made on the data symmetry), and under images of nominal functions (true in any data symmetry).

It is easy to see that all the examples from the beginning of this section are first-order definable. Consider the function from Example 7. We use the product of two letter selectors to define a function $h : A^* \to A^2$, which maps a word to the ordered pair of its first and last letters. Then, we use information loss to forget the order in the pair.

Lemma 8.1 *A language $L \subseteq A^*$ is first-order definable if and only if its characteristic function $f_L : A^* \to 2$ is first-order definable.*

Proof
The left-to-right implication is by definition of first-order definable functions. The right-to-left implication is by substituting formulas of first-order logic. □

8.1 Formulas as a nominal set

In this section we conjecture that first-order definable functions have a more elegant description than the one presented in the previous section.

Recall that the predicates in nominal first-order logic are obtained from nominal subsets $R \subseteq A^n$ (except for the order relation). Suppose that the alphabet A has support C, i.e. one can use G_C to act on elements of A. Extending this action coordinatwise to A^n, and then pointwise to subsets of A^n, we can apply any permutation $\pi \in G_C$ to a nominal relation $R \subseteq A^n$, and get a new nominal relation $R\pi \subseteq A^n$. We can then extend this action of G_C, by simple structural induction, to all formulas of nominal first-order logic, so that a permutation $\pi \in G_C$ maps one formula φ to another formula $\varphi\pi$. It is easy to see that the set of formulas of nominal first-order logic is also a nominal set, with support C. Call this set FO_A.

We conjecture that a function $f : A^* \to B$ is first-order definable if and only if there is a set of sentences $\{\varphi_b\}_{b \in B}$, such that

- For every b, the sentence φ_b defines the language $f^{-1}(b)$; and
- The correspondence $b \mapsto \varphi_b$ is a nominal function from B to FO_A.

The advantage of this alternative characterization, if it is indeed true, is that it is shorter, more similar to the classical case, and it can easily be generalized to other logics, such as MSO or two-variable FO.

9 First-order logic

In this section we prove the main result of the paper, Theorem 9.1, which is an effective characterization of first-order logic in terms of aperiodic monoids. In the theorem, we talk about the $\mathcal{J}$-order, which is the relation $\leq_{\mathcal{J}}$ lifted to $\mathcal{J}$-classes.

Theorem 9.1 *Let L be a nominal language, and let M_L be its syntactic monoid. Assume that M_L is orbit-finite and has a well-founded $\mathcal{J}$-order. Then L is definable in first-order logic if and only if M_L is aperiodic.*

Lemma 9.2 shows that the easier implication holds even without the assumptions on orbit-finiteness and well-foundedness.

Lemma 9.2 *If L is definable in first-order logic, then M_L is aperiodic.*

Proof
This proof is the same as in the classical case, without data values.

Suppose that L is defined by a sentence φ of first-order logic. Let i be a number which is sufficiently large with respect to the quantifier rank of φ. Using an Ehrenfeucht-Fraïssé game, one shows that for every all $u, w, v \in A^*$,

$$uw^i v \models \varphi \qquad \text{iff} \qquad uw^{i+1} v \models \varphi. \tag{1}$$

The above says that for every word $w \in A^*$ and sufficiently large i, the words w^i and w^{i+1} are equivalent under the two-sided Myhill-Nerode equivalence relation. Since the syntactic monoid consists of these equivalence classes, it is aperiodic. □

In Section 9.1, we show the more difficult implication: if M_L is aperiodic then L is first-order definable. First, we give examples which illustrate the assumptions in the theorem. Both examples show syntactic monoids where the $\mathcal{J}$-order is not well-founded. It could be the case that if a syntactic monoid is aperiodic and has a well-founded $\mathcal{J}$-order, then it is orbit-finite. In particular, in Theorem 9.1, it could be that the assumption on orbit-finiteness is not necessary[4].

Example 8. Consider the total order symmetry, and the language of words with an even number of growing data values

$$L = \{d_1 \cdots d_{2n} : n \in \mathbb{N}, d_1 < d_2 < \cdots < d_{2n}\} \subseteq \mathbb{D}^*.$$

The syntactic monoid M_L of this language is

$$\{0,1\} \quad \cup \quad \mathbb{D} \quad \cup \quad \{(d,e,i) \in \mathbb{D}^2 \times 2 : d < e\}$$

where 0 stands for words outside L, 1 stands for the empty word, d stands for a single-letter word, and (d,e,i) stands for a word that begins with d, ends with e, and its length modulo 2 is i. This monoid is orbit-finite. However, the monoid fails the assumption on the $\mathcal{J}$-ordering being well-founded, as witnessed by the following infinite decreasing sequence

$$(-1,1,0) >_{\mathcal{J}} (-2,2,0) >_{\mathcal{J}} (-3,3,0) >_{\mathcal{J}} \cdots$$

That is why the implication in the theorem fails: the monoid is aperiodic, but the language L is not definable in first-order logic. □

Example 9. Let A be an infinite but orbit-finite alphabet, e.g. $\mathbb{D}$ in the equality symmetry. Consider the set of words where the number of distinct letters is even. The syntactic monoid of this language is the family of finite subsets of A, with union as the monoid operation. The monoid is aperiodic (and therefore a language that talks about parity can have an aperiodic syntactic monoid). However, the monoid is not orbit-finite, because subsets of different sizes are in different orbits. Also, the $\mathcal{J}$-order is not well-founded (because the $\mathcal{J}$-order is the superset relation, which is not well-founded). [5] □

Observe that in Example 8 we used the total order symmetry, and not the simpler equality symmetry. Indeed, it is impossible to provide an example that uses the equality symmetry, as shown by the following lemma.

[4] I would like to thank an anonymous reviewer for pointing this out.

[5] A previous version of this paper claimed that the monoid has a well-founded $\mathcal{J}$-order. This error was pointed out by the anonymous referee. I am not aware of a syntactic monoid that is aperiodic, not orbit-finite, but has a well-founded $\mathcal{J}$-order.

Lemma 9.3 *In the equality symmetry, the $\mathcal{J}$-order is well-founded in every orbit-finite monoid.*

Proof
Consider an orbit-finite monoid M. Suppose that there is an infinite decreasing chain

$$m_1 >_{\mathcal{J}} m_2 >_{\mathcal{J}} m_3 >_{\mathcal{J}} \cdots$$

Let C be a support of M. Because the monoid is orbit-finite, there must be some $i < j$ such that m_i is in the same orbit as j, with respect to G_C. We will show that m_i and m_j are in the same $\mathcal{J}$-class, contradicting the assumption on the decreasing chain.

Because m_j comes later in the chain than m_i, there must be elements $x, y \in M$ such that

$$m_j = x m_i y.$$

Because m_i and m_j are in the same orbit, there must be some $\pi \in G_C$ such that $m_i = m_j \pi$. Choose π so that, as a permutations on data values, it is the identity on all but a finite set of data values. This can be done in the equality symmetry, but not in other symmetries, such as the total order symmetry. By the assumption that the monoid operation is nominal, we see that

$$m_i = m_j \pi = (x m_i y)\pi = (x\pi)(m_i\pi)(y\pi).$$

Apply the above equation to itself k times, yielding

$$m_i = (x\pi)(x\pi^2)\cdots(x\pi^{k-1})(x\pi^k)(m_i\pi^k)(y\pi^k)(y\pi^{k-1})s(y\pi^2)(y\pi).$$

Because the permutation π is the identity on almost all data values, then π^k is the identity for some $k > 0$. It follows that

$$m_i = (x\pi)(x\pi^2)\cdots(x\pi^{k-1})(x m_i y)(y\pi^{k-1})\cdots(y\pi^2)(y\pi),$$

which means that $m_j = x m_i y$ is an infix of m_i, and therefore m_i is $\mathcal{J}$-equivalent to m_j.

Observe that we have proved a slightly stronger result: if a permutation from G_C can map one $\mathcal{J}$-class to another, then the $\mathcal{J}$-classes are either equal or incomparable in the $\mathcal{J}$-ordering. □

9.1 From aperiodic to first-order definable

In this section we finish the proof of Theorem 9.1, by showing the more difficult implication: if M_L is aperiodic then L is first-order definable. The proof is an induction, which needs a more detailed statement, as presented below.

Proposition 9.4 *Consider a nominal monoid morphism $\alpha : A^* \to M$ into a nominal monoid M that is orbit-finite and aperiodic. Let $X \subseteq M$ be a nominal subset that is upward closed under $\leq_{\mathcal{J}}$. Then the partial function α_X, which is α with domain restricted to X, is first-order definable.*

Let C be a set that supports both M and X. The proof is by induction on the number of orbits in X, under the action of the group G_C. The base of the induction is when there are zero orbits, in which case the function α_X is easy to define.

Because X is upward closed under $\leq_{\mathcal{J}}$, then it is a union of $\mathcal{J}$-classes. Choose a $\mathcal{J}$-class $J \subseteq X$ that is minimal under the $\mathcal{J}$-order, which is possible by the assumption that the $\mathcal{J}$-order is well founded. Let Y be the closure of J under the action of G_C, i.e.

$$Y = \bigcup_{m \in J} mG_C.$$

Because X is supported by C, it follows that $Y \subseteq X$. Consider the set $Z = X - Y$, which is also supported by C. The sets X, Y, Z are illustrated in Figure 2.

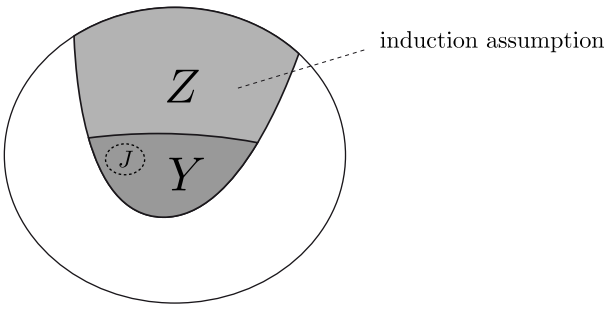

Figure 2: The sets Y and Z, which partition X. The $\mathcal{J}$-order is oriented vertically in the picture, with smaller elements at the bottom.

Lemma 9.5 *The set Z is upward closed under $\leq_{\mathcal{J}}$.*

Proof
Suppose that $n \in Z$ and $n \leq_{\mathcal{J}} m$. Because X is upward closed, it follows that $m \in X$. Suppose, for the sake of contradiction, that $m \notin Z$. Then $m \in Y$, and by definition of Y, there must be some $\pi \in G_C$ such that $m\pi \in J$. Because the order $\leq_{\mathcal{J}}$ has support C, we can multiply both sides of $n \leq_{\mathcal{J}} m$ by π, yielding

$$n\pi \leq_{\mathcal{J}} m\pi.$$

Because n belongs to Z, and Z is supported by C, it follows that $n\pi$ belongs to Z, and therefore $n\pi$ cannot belong to J, which is included in Y. This contradicts the assumption that $m\pi$ belongs to a minimal $\mathcal{J}$-class in X. $\square$

Thanks to Lemma 9.5, we can apply the induction assumption to Z, yielding a function α_Z that is first-order definable. The rest of this section is devoted to extending the function from Z to $X = Y \cup Z$

For a word $w \in A^*$, we use the name *type of* w for $\alpha(w)$. Define R_w to be the $\mathcal{R}$-class of the shortest prefix of w with type outside Z. Likewise, define L_w to be the $\mathcal{L}$-class of the shortest suffix of w whose type is outside Z. Both R_w and L_w might be undefined, if the appropriate prefixes or suffixes do not exist.

Lemma 9.6 *The partial function*

$$g : A^* \to M/\mathcal{R} \times M/\mathcal{L} \qquad g(w) = (R_w, L_w)$$

is first-order definable. Its domain is the set of words with type outside Z*.*

Proof
Suppose that a word w has type in Z. Because the $\mathcal{J}$-class of an infix decreases as the infix grows, and because Z is upward closed under the $\mathcal{J}$-ordering, it follows that all infixes of w have type in Z. Therefore, g is undefined on w.

Consider now a word w with type outside Z. Let u_1 be the longest prefix of w with type in Z. Let a_1 be the next letter after u_1 in w. By choice of a_1, we have that R_w is the $\mathcal{R}$-class of $u_1 a_1$. Likewise, let u_2 be the shortest suffix of w with type in Z, and let a_2 be the letter that precedes u_2 in w. Again, L_w is the $\mathcal{L}$-class of $a_2 u_2$.

Using the induction assumption, one shows that the partial function

$$h : A^* \to Z \times A \times A \times Z \qquad h(w) = (\alpha(u_1), a_1, a_2, \alpha(u_2))$$

is first-order definable (it is defined if and only if the type of w is outside Z). By composing the function with the monoid operation, and the nominal functions

$$m \mapsto [m]_{\mathcal{L}} \qquad m \mapsto [m]_{\mathcal{R}}$$

we conclude that the function g itself is first-order definable. □

Because R_w is an $\mathcal{R}$-class, and L_w is an $\mathcal{L}$-class, and because the monoid M is aperiodic, it follows from the second item of Lemma 7.1 that the intersection $R_w \cap L_w$ contains at most one element. This is the only place where we use the assumption that M is aperiodic. Consider the following partial function $f : A^* \to M$.

$$f(w) = \begin{cases} \alpha_Z(w) & \text{if } (R_w, L_w) \text{ is undefined and therefore } \alpha_Z(w) \text{ is defined} \\ m & \text{if } (R_w, L_w) \text{ is defined and } R_w \cap L_w = \{m\} \\ \text{undefined} & \text{otherwise, i.e. if } (R_w, L_w) \text{ is defined and } R_w \cap L_w = \emptyset \end{cases}$$

The partial function f is first-order definable, because first-order definable functions allow a case distinction.

Lemma 9.7 *If* w *has type in* X*, then that type is* $f(w)$*.*

Proof
Because X is partitioned into Y and Z, we consider two cases.

If w has type in Z, then the result follows from the definition of f.

Suppose that w has type in Y. In this case, R_w and L_w are defined. Let u be the shortest prefix of w with type outside Z, this is the word whose $\mathcal{R}$-class is R_w. Because u is a prefix of w, we have

$$\alpha(u) \geq_{\mathcal{R}} \alpha(w).$$

Because the type of u is not in Z, then it must be in Y. By the first item of Lemma 7.1, it follows that

$$\alpha(u) \ \mathcal{R} \ \alpha(w)$$

and therefore the $\mathcal{R}$-class of w is R_w. In the same way, we show that the $\mathcal{L}$-class of w is L_w. It follows that the intersection $R_w \cap L_w$ is nonempty, as it contains the type of w, and therefore $R_w \cap L_w = \{\alpha(w)\}$. $\square$

Lemma 9.8 *The set of words with type in Y is first-order definable.*

Proof
Because Y is $X - Z$, and we already have a formula for words with type in Z, it suffices to define words with type in X. We will define the complement of X, i.e. the words that have type outside X. Consider the following set K of words

$$K = \cup \begin{cases} \{ua : u \in A^*, a \in A, f(u) \text{ is defined and } f(u)\alpha(a) \notin X\} \\ \{au : u \in A^*, a \in A, f(u) \text{ is defined and } \alpha(a)f(u) \notin X\} \end{cases}.$$

Because it is defined in terms of the function f, the set K is a first-order definable language. We claim that a word w has type outside X if and only if it has an infix from K, the latter being a first-order definable property.

For the right-to-left implication, observe that all words in K have type outside X, and therefore also all words that contain an infix from K.

For the left-to-right implication, observe that the complement of X is downward closed in the $\mathcal{J}$-order, by assumption on X being upward closed. Therefore, a word has type outside X if and only if it has an infix outside X. If we choose the infix to have minimal length, then removing the first or last letter of the infix gives a word in with type in X, for which we can use Lemma 9.7. This infix is the word u in the definition of K. $\square$

We now conclude the proof of the induction step in Proposition 9.4. We need to define the function α_X. This is the disjoint union of the functions α_Y and α_Z. The latter function is defined by induction assumption. The former function is the restriction of f to the set of words with type in Y, by Lemma 9.7. By Lemma 9.8, this restriction can be done in first-order logic.

10 Further work

Characterize other logics. It is natural to extend the characterization of first-order logic to other logics. Candidates that come to mind include first-order logic with two variables, or various logics inspired by XPath, or piece-wise testable languages. Also, it would be interesting to see the expressive power of languages recognized by orbit-finite nominal monoids. This class of languages is incomparable in expressive power to first-order logic, e.g. the first-order definable language "some data value appears twice" is not recognized by an orbit-finite nominal monoid. It would be nice to see a logic, maybe a variant of monadic second-order logic, with the same expressive power as orbit-finite nominal monoids.

Use mechanisms more powerful than monoids. As language recognizers, orbit-finite nominal monoids are very weak. In most data symmetries, such as the equality and total order symmetries, orbit-finite nominal monoids are strictly less expressive than orbit-finite deterministic automata[6]. For example, the language "the first letter in the word appears also on some other position", is recognized by an orbit-finite deterministic automaton, but has a syntactic monoid with infinitely many orbits. Therefore, one can ask: is it decidable if an orbit-finite automaton recognizes a language that can be defined in first-order logic? We conjecture that this problem is decidable, and even that a necessary and sufficient condition is aperiodicity of the syntactic monoid (which need not be orbit-finite).

Acknowledgments. I would like to thank Clemens Ley for introducing me to the subject. I would like to thank Sławomir Lasota and Bartek Klin for many stimulating discussions. Finally, I would also like to thank Michael Kaminski and the anonymous referees for their many helpful comments.

References

[1] Michael Benedikt, Clemens Ley, and Gabriele Puppis. Automata vs. logics on data words. In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2010.

[2] Mikołaj Bojańczyk. Data monoids. In Thomas Schwentick and Christoph Dürr, editors, *STACS*, volume 9 of *LIPIcs*, pages 105–116. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[3] Mikołaj Bojańczyk, Laurent Braud, Bartek Klin, and Sławomir Lasota. Towards nominal computation. In John Field and Michael Hicks, editors, *POPL*, pages 401–412. ACM, 2012.

[6] This counterpart has the same expressive power as deterministic memory automata of Francez and Kaminski [9], as shown in [4].

[4] Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata with group actions. In *LICS*, pages 355–364. IEEE Computer Society, 2011.

[5] Patricia Bouyer, Antoine Petit, and Denis Thérien. An algebraic approach to data languages and timed languages. *Inf. Comput.*, 182(2):137–162, 2003.

[6] Nissim Francez and Michael Kaminski. An algebraic characterization of deterministic regular languages over infinite alphabets. *Theor. Comput. Sci.*, 306(1-3):155–175, 2003.

[7] Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.

[8] Peter M. Higgins. *Techniques of Semigroup Theory.* Oxford University Press, 1992.

[9] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[10] Ugo Montanari and Marco Pistore. Finite state verification for the asynchronous pi-calculus. In Rance Cleaveland, editor, *TACAS*, volume 1579 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 1999.

[11] Ugo Montanari and Marco Pistore. History-dependent automata: An introduction. In Marco Bernardo and Alessandro Bogliolo, editors, *SFM*, volume 3465 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 2005.

[12] Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In Zoltán Ésik, editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006.

[13] H. Straubing. *Finite Automata, Formal Languages, and Circuit Complexity.* Birkhäuser, Boston, 1994.

[14] Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III, pages 389–455. Springer, 1997.