Toward Model Theory with Data Values

Mikołaj Bojańczyk and Thomas Place*

University of Warsaw

Abstract. We define a variant of first-order logic that deals with data words, data trees, data graphs etc. The definition of the logic is based on Fraenkel-Mostowski sets (FM sets, also known as nominal sets). The key idea is that we allow infinite disjunction (and conjunction), as long as the set of disjuncts (conjunct) is finite modulo renaming of data values. We study model theory for this logic; in particular we prove that the infinite disjunction can be eliminated from formulas.

1 Introduction

This paper uses Fraenkel-Mostowski sets (FM sets, also known as nominal sets) to study logics that describe properties of objects such as data words, data trees, or data graphs.

Suppose that $\mathbb{D}$ is an infinite set of *data values*, also called *atoms* or *ur-elements*, whose elements can only be compared for equality. A *data word* is a word (trees and graphs can also be considered, of course) whose positions are labelled by an alphabet that is not necessarily finite, but which can refer to data values in a finite way, such as in the following examples of alphabets.

$$\mathbb{D} \qquad \{0,1\} \times \mathbb{D} \qquad (\mathbb{D}^2 \cup \mathbb{D}) \qquad \{0,1\} \times \{\{c,d\} : c,d \in \mathbb{D}\}.$$

The statement "data values can only be compared for equality" is formalized by saying that properties of data words should be invariant under the action of the group of bijections of $\mathbb{D}$. The statement "refer to data values in a finite way" is formalized by saying that the alphabet contains finitely many elements, modulo bijections of data values. For instance, modulo bijections, the set $\mathbb{D}^2 \cup \mathbb{D}$ has three elements, which look like this: $(d,e), (d,d)$ and d.

Properties that are invariant under the action of the group of bijections include:

1. Data words over the alphabet $\mathbb{D}$ where all positions have different labels.
2. Data words over the alphabet $\mathbb{D}$ with at least six distinct letters.
3. Graphs with edges labelled by $\mathbb{D}^2$ where for each vertex, all outgoing edges have the same data value on the first coordinate.

There is a more relaxed notion of invariance: a property is called *finitely supported* if there is a finite set of data values $C \subseteq \mathbb{D}$, such that the property is invariant under the action of permutations that preserve C. The set C is called the *support*. For instance, if we choose some two elements $c, d \in \mathbb{D}$, then

* Both authors supported by ERC Starting Grant "Sosna". A full version of this paper can be found at `www.mimuw.edu.pl/~bojan`.

4. Data words over alphabet $\mathbb{D}$ which begin with c and end with d.

is a finitely supported property, namely supported by $C = \{c, d\}$.
To give examples of properties that are not finitely supported, one needs additional assumptions on $\mathbb{D}$. For instance, if we assume that $\mathbb{D}$ is the natural numbers, then "words in $\mathbb{D}^*$ which contain only even numbers" is not a finitely supported property of data words.
The notion of finitely supported sets is the cornerstone of "permutation models" of set theory, which were studied by logicians such as Fraenkel and Mostowski starting in the 1920's. Permutation models were rediscovered, under the name "nominal sets", by Gabbay and Pitts in [3], see also [4], as an elegant approach to deal with binding and fresh names in the syntax of programming languages and logical formulas. When dealing with syntax, one thinks of data values as being variable names. Finally, these sets were rediscovered by the automata community, as an approach to describing languages of data words [2].[1]

Logic on data words and data trees. The direct predecessor of this paper is [2], which uses FM sets (under the name nominal sets) to talk about automata on data words. In the present paper, we use FM sets to talk about logics on data words (and more general structures). We define:

- A notion of *FM relational structure*. This notion generalizes data words, data trees, data graphs, etc. One can apply a permutation of data values to an FM relational structure, and get another FM relational structure.
- A notion of *FM first-order logic*. The formulas are evaluated in FM relational structures. The formulas form an FM set. The previously stated examples of properties of data words and data graphs are definable in the logic, including example 4.

Logics for data words have been extensively studied in the special case of data words and data trees with alphabets of the form $A \times \mathbb{D}$, where A is a finite set. In this special case, the approach of [6] is to use: a binary predicate $x \sim y$ which says that two positions carry the same data value; as well as a unary predicate $a(x)$ for each $a \in A$. The satisfiability problem for the logic is undecidable for most variants, see [6]. In the special case of alphabets $A \times \mathbb{D}$, our abstract definition of FM first-order logic coincides with the existing definition.
Even for words, the choice of logic is not obvious for some alphabets. Consider the alphabet "sets of data values of size at most 3". A natural predicate would be $x \subseteq y$, saying that the set in the label of x is a subset of the set in the label

[1] There are two names for the sets that can be used: "FM sets" as in mathematical logic, or "nominal sets" as in the study of name binding. In this paper, we decided to use the name "FM sets". The main reason is that our application of Frankel-Mostowski set theory is not principally concerned with the use of names and their binding. An additional reason is that, like in Fraenkel-Mostwoski set theory, but unlike in the study of name binding, we are often interested in data values with additional structure, such as a linear order.

of y. Another kind of predicate, not definable in terms of $x \subseteq y$, could be

$$|x_1 \cup \cdots \cup x_n| = k \qquad \text{for } n, k \in \mathbb{N}.$$

Which predicates should be allowed in the logic? Our definition implies that they are all allowed. We do not address the question of a minimal choice of predicates, i.e. which predicates can be defined in terms of others.

Parse trees. On a definitional level, the principal idea in this paper is to allow parse trees of formulas where the branching degree is not finite, but finite modulo bijections of data values (we call this orbit-finite branching). In normal sets, the parse tree of an expression (a formula of first-order logic, a regular expression, an arithmetic expression, etc.) is a finite tree. In FM sets, one can have a more relaxed parse tree: for each node, the set of child subtrees is only required to be finite modulo bijections of data values[2]. For instance, if for each data value $d \in \mathbb{D}$ we have a formula φ_d, and the function $d \mapsto \varphi_d$ is finitely-supported, then it makes sense to consider the infinite disjunction $\bigvee_{d \in \mathbb{D}} \varphi_d$. On a technical level, the main contribution of this paper is Theorem 5.2 which says that the infinite disjunction can be eliminated from formulas.

Related work. A logic for nominal sets, called nominal logic, was studied by Pitts in [5]. Nominal logic and the logic from this paper have different goals: nominal logic is designed to axiomatise nominal sets, while the formulas in this paper are used to define languages of data words and similar objects. Also, the logics are defined differently: the formulas and models for nominal logic are defined in normal set theory; while the formulas and models in this paper are defined inside FM set theory[3]. Finally, the principal technical result of this paper is elimination of infinite disjunction, this result cannot be even stated in the language of [5].

Acknowledgement. We would like to thank Nathanaël Fijalkow, Bartek Klin, and the anonymous referees for their comments and suggestions.

2 Preliminaries

Data symmetry. The notion of FM sets is parametrized by a set of *data values* $\mathbb{D}$, and a group G of bijections on $\mathbb{D}$. The group G need not contain all bijections of $\mathbb{D}$. The idea is that $\mathbb{D}$ has some structure, and G contains the structure-preserving bijections. The pair $(\mathbb{D}, G)$ is called a *data symmetry*. In this paper, we use the following data symmetries:

[2] This appears already explicitly in [1], where terms of λ-calculus have orbit-finitely branching parse-trees. Implicitly, the idea goes back the work of Gabbay and Pitts, where the whole point of nominal sets was to model the use of binding.

[3] One could say that our logic is an internal logic for FM sets, while the logic of [5] is external.

– The set $\mathbb{D}$ is empty, and G has only the identity element. We call this the *classical symmetry*. FM sets in the classical symmetry are normal sets.
– 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*. FM sets in the equality symmetry are the same thing as nominal sets [3, 4].
– The set $\mathbb{D}$ is the vertices of the undirected countable homogeneous graph (also called the Rado graph), and the group G is the group of automorphisms of this graph. We call this the *graph symmetry*.

FM set. Consider first the *cumulative hierarchy* of sets with data values, which is a hierarchy of sets indexed by ordinal numbers and defined as follows. The empty set is the unique set of rank 0. A set of rank α is any set whose elements are sets of rank smaller than α, or data values. A permutation π of data values can be applied to a set X in the hierarchy, by renaming the elements of X, and the elements of elements of X, and so on. The resulting set, which has the same rank, is denoted by $X \cdot \pi$.
A set C of data values is said to be a *support* of a set X in the cumulative hierarchy if $X \cdot \pi = X \cdot \sigma$ holds for every permutations π, σ in the group from the data symmetry which agree on elements of C. A set is called *finitely supported* if it has some finite support. We use the name *FM set* for a set in the cumulative hierarchy which is hereditarily finitely supported, which means that it is finitely supported, the sets in it are finitely supported, and so on[4].
The support of an FM set is not unique, e.g. supports are closed under adding data values. A set with empty support is called *equivariant*.
In many respects, FM sets behave like normal sets. For instance, if X, Y are FM sets, then $X \times Y$, $X \cup Y$, X^* and the finite powerset of X are all FM sets. Another example is the family of subsets of X that have finite supports. The appropriate notion of a function between FM sets X and Y is that of a *finitely supported function*, which is a function from X to Y whose graph is an FM set. Observe that FM sets in the classical symmetry are simply sets (equipped with the only possible action). Therefore the classical symmetry corresponds to classical set theory, without data values.

Orbit-finite FM sets. Suppose that X is an FM set. For a set of data values C, define the C-orbit of an element $x \in X$ to be the set $\{x \cdot \pi : \pi \in G_C\}$. If C supports X, then the C-orbits form a partition of X. The set X is called orbit-finite if it the partition into C-orbits has finitely parts, for some C which supports X. For some data symmetries, including the classical, equality and graph symmetries discussed in this paper, the notion of orbit-finiteness does not depend on the choice of support [1]. In other words, for these data symmetries, if two sets C and D support an FM set X, then X has finitely many C-orbits if and only if it has finitely many D-orbits.
In this paper, we are mostly interested in FM sets that are orbit-finite.

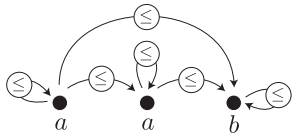
[4] The definition here is based on Definition 10.6 in [4], except that we use the name *FM set* for what [4] calls elements of $\mathcal{HFS}$.

3 Relational Structures

The discussion in this section – and the next Section 4 – makes sense in any data symmetry. Fix some data symmetry $(\mathbb{D}, G)$ for this section and the next.
One of the key ideas of finite model theory in computer science is that a combinatorial object, such as a word, tree, or graph, can be treated as a model for a logical formula. For instance, in the case of words over an alphabet $\{a, b\}$, a word with n positions can be interpreted as a relational structure where the domain is the set of positions $\{1, \ldots, n\}$, there are two unary predicates $a(x)$ and $b(x)$ for labels, and there is a binary predicate $x \leq y$ for the order on positions. Using this interpretation, one can define properties of words using first-order logic, e.g. the set of words that end with b is defined by the formula

$$\forall x \exists y \quad x \leq y \wedge b(y).$$

The goal of this paper is to define a similar notion of logic for combinatorial objects that contain data values. In particular, our definition should cover data words and data trees.
In standard sets, not FM sets, a relational structure can be seen as a hyper-edge colored directed hypergraph. For instance, the relational structure corresponding to the word aab is the following hypergraph.



We adapt this definition to FM sets as follows. For instance a binary predicate will not just say yes/no to each directed edge, but it can also color the edge, e.g. by a data value.
An *FM predicate* R consists of an orbit-finite FM set $\text{colors}(R)$ and a natural number $\text{arity}(R)$. An *FM signature* Σ is a finite set of FM predicates. An *FM relational structure* $\mathfrak{A}$ over Σ consists of:

- A set $\text{dom}(\mathfrak{A})$, called the *domain* of the structure, which is an FM set.
- For every predicate R in the signature, a finitely supported partial function

$$R^{\mathfrak{A}} : \text{dom}(\mathfrak{A})^{\text{arity}(R)} \dot{\to} \text{colors}(R),$$

  called the *interpretation of* R.

For a fixed FM signature Σ, the set of FM relational structures over Σ is itself a FM set, because an FM relational structure is nothing other than a domain (which has empty support, since it is equipped with the trivial action) and a finite tuple of finitely-supported partial functions[5]. Therefore, if $\mathfrak{A}$ is a FM relational

[5] Formally speaking, this is an FM class, because all the domains do not form a set.

structure and $\pi \in G$, then also $\mathfrak{A} \cdot \pi$ is a FM relational structure. Both $\mathfrak{A}$ and $\mathfrak{A} \cdot \pi$ have the same domains, only different interpretations. If R is a predicate of arity n, and $x_1, \ldots, x_n$ are in the domain of $\mathfrak{A}$, then

$$R^{\mathfrak{A}\cdot\pi}(x_1, \ldots, x_n) = (R^{\mathfrak{A}}(x_1, \ldots, x_n)) \cdot \pi.$$

In particular, either both sides of the equality above are defined, or neither are (recall that interpretations are partial functions.)
An FM relational structure is called finite if its domain is finite. In such a case, an interpretation is a finite tuple of colors. A tuple of objects taken from an FM set necessarily has finite support, and therefore in the case of finite relational structures, the requirement on finitely supported interpretations is redundant.

Example 3.1. Data words can be modeled as FM relational structures. We use the name *FM alphabet* for any orbit-finite set A. To an FM alphabet A, we associate an FM signature Σ_A with two predicates:

- The alphabet predicate R_A, which has arity 1 and colors A.
- The order predicate $R_<$, which has arity 2 and only one color $\{<\}$.

For a data word $w = a_1 \cdots a_n \in A^*$, we define a corresponding FM relational structure $\mathfrak{A}_w$ over the signature Σ_A as follows. The domain $dom(\mathfrak{A}_w)$ is the set $\{1, \ldots, n\}$ of positions. The interpretation of the alphabet predicate maps each position to its label, and the interpretation of the order predicate maps a pair (i, j) to $<$ if and only if $i < j$.

Example 3.2. Edge-labelled directed data graphs can be modeled as FM relational structures. To an FM alphabet A, we associate an FM signature Σ_A with one predicate R_A with arity 2 and colors A, called the edge label predicate. A structure over this FM signature describes a directed graph, where edges are labelled by A. Because the interpretation is a partial function, every ordered pair of nodes is connected by zero or one edge.

4 Logic

In this section, we define a variant of first-order logic which is used to define properties of FM relational structures. Before giving the actual definition, we enumerate the postulates it should satisfy:

1. The set of formulas is itself an FM set, and the satisfaction relation is equivariant. That is

$$\mathfrak{A} \models \varphi \quad \text{iff} \quad \mathfrak{A} \cdot \pi \models \varphi \cdot \pi$$

   holds for every $\pi \in G$, FM relational structure $\mathfrak{A}$ and formula φ.
2. Orbit-finite disjunction is allowed. That is, if Γ is an orbit-finite FM set of formulas, then also $\bigvee \Gamma$ is a formula.

Below, in Section 4.1, we give a definition which satisfies the above postulates.

4.1 Definition of FM first-order logic

To choose the predicates for our logic, we use a semantic approach: every isomorphism-closed property of a tuple of elements is going to be a predicate. An isomorphism between two FM relational structures $\mathfrak{A}$ and $\mathfrak{B}$ is a finitely supported bijection

$$f : \mathrm{dom}(\mathfrak{A}) \to \mathrm{dom}(\mathfrak{A})$$

such that for every k-ary predicate R in the signature, we have

$$R^{\mathfrak{A}}(a_1, \ldots, a_k) = R^{\mathfrak{B}}(f(a_1), \ldots, f(a_k)) \qquad \text{for every } a_1, \ldots, a_k \in \mathrm{dom}(\mathfrak{A}).$$

Example 4.1. Consider the equality symmetry, and a signature with one unary predicate P, whose colors are $\mathbb{D}$. Suppose that $1, 2$ are data values. Let $\mathfrak{A}$ be a structure whose domain is $\{1\}$, and where the interpretation is $P^{\mathfrak{A}}(1) = 1$. Let $\mathfrak{B}$ be a structure whose domain is $\{2\}$, and where the interpretation is $P^{\mathfrak{B}}(2) = 1$. Then $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic.

An *atomic type of arity n* is (the isomorphism type of) a structure $\mathfrak{A}$, together with an n-tuple of elements, such that every element of the domain of $\mathfrak{A}$ appears in the tuple. The domain of the atomic type has at most n elements, but might be smaller if the tuple contains repetitions. We write $\mathrm{atoms}_n(\Sigma)$ for the set of atomic types of arity n. If $\mathfrak{A}$ is a structure, and $\bar{a}$ is a (possibly repeating) tuple of elements in $\mathrm{dom}(\mathfrak{A})$, then we define $\mathfrak{A}|\bar{a}$ to be the atomic type obtained from $\mathfrak{A}$ by only keeping the elements from $\bar{a}$.

Fact 1 *In the classical, equality and graph symmetries, the set* $\mathrm{atoms}_n(\Sigma)$ *is orbit-finite.*

A *basic type of arity n* is defined to be any finitely supported subset of $\mathrm{atoms}_n(\Sigma)$.

Example 4.2. Consider the FM alphabet $\binom{\mathbb{D}}{2}$, which is defined to be the family of two-element subsets of $\mathbb{D}$. Consider data words over this alphabet, as in Example 3.1. A basic type $\mathcal{B}$ of arity 2 could say that that the set in the label of the first distinguished position has non-empty intersection with the set in the label in the second distinguished position. This basic type is not only finitely supported, but also equivariant.

Example 4.3. This example also concerns data words over $\binom{\mathbb{D}}{2}$. In this example, as in subsequent examples, we assume that the data values are natural numbers. A non-equivariant basic type $\mathcal{B}_9$ of arity 2 says that the sets in the label of the first and second distinguished position both contain the data value $9 \in \mathbb{D}$.

We define *FM first-order logic for a relational signature Σ* as follows. As predicates, we use basic types in the following sense: a basic type $\mathcal{B}$ of arity n is a predicate of arity n, with the semantics

$$\mathfrak{A}, \bar{a} \models \mathcal{B}(x_1, \ldots, x_n) \qquad \text{iff} \qquad \mathfrak{A}|\bar{a} \in \mathcal{B}.$$

Furthermore, formulas can use boolean combinations $\{\vee, \wedge, \neg\}$ as well as quantifiers $\{\forall, \exists\}$. We will add one more connective, but to define this connective we need to discuss the action of G on formulas. When applying a permutation $\pi \in G$ to a formula φ, the structure of the formula, the connectives $\vee, \wedge, \neg, \forall, \exists$ as well as the variables are not changed. The only thing that changes is the basic types: a set of atomic types $\mathcal{B}$ is mapped to the set $\mathcal{B} \cdot \pi$.

Example 4.4. Consider the basic type $\mathcal{B}_9$ from Example 4.3, and the formula

$$\exists x \exists y \; x \neq y \wedge \mathcal{B}_9(x, y).$$

This formula, call it φ_9, says that the data value 9 appears in the label of at least two positions. Consider a permutation $\pi \in G$, which maps 9 to 8. Then the formula $\varphi_9 \cdot \pi$ says that the data value 8 appears in the label of at least two positions.

It is not difficult to see that every formula has finite support. The reason is that every formula uses a finite number of basic types, and each basic type has finite support by definition.
We now define the remaining connective, which is called *orbit-finite disjunction*. Consider an orbit-finite FM set of already defined formulas Γ. We allow a disjuction over this set $\bigvee \Gamma$, with the expected semantics. Orbit-finite disjunction is the last connective of the logic, and the definition of FM first-order logic is now complete.

Example 4.5. Consider the formula φ_9 in Example 4.4. This formula can be defined for any data value d, not just 9, and it is easy to see that the set $\{\varphi_d : d \in \mathbb{D}\}$ is an orbit-finite FM set of formulas. Therefore, we can use the orbit-finite disjunction

$$\bigvee \{\varphi_d : d \in \mathbb{D}\} \qquad \text{also written as} \qquad \bigvee_{d \in \mathbb{D}} \varphi_d.$$

The disjunction above says that some data value appears in the label of at least two positions. Observe that the above formula can be expressed, without orbit-finite disjunction, by using the predicate $\mathcal{B}$ from Example 4.2:

$$\exists x \exists y \; x \neq y \wedge \mathcal{B}(x, y).$$

Example 4.6. In the previous case, the set $\{\varphi_d : d \in \mathbb{D}\}$ was equivariant. One can also use non-equivariant sets, such as

$$\bigvee_{d \in \mathbb{D} - \{9\}} \varphi_d.$$

Non-equivariant sets are useful for nesting formulas, e.g.

$$\bigwedge_{e \in \mathbb{D}} \bigvee_{d \in \mathbb{D} - \{d\}} \varphi_d.$$

The formula above says that there are two data values that appear in the label of at least two positions.

5 Elimination of Orbit-Finite Disjunction

Recall that in Example 4.5, we were able to eliminate orbit-finite disjunction. The technique was to push the disjunction into the basic types. This technique can fail, e.g. in the graph symmetry, as shown by the following theorem.

Theorem 5.1. *Consider the graph symmetry. Let $L \subseteq \mathbb{D}^*$ be the set of words $d_1 \cdots d_n$ such that n is even, all letters are distinct, and for every $i, j \in \{1, \ldots, n\}$ there is no graph edge from d_i to d_j. This set is definable in FM first-order logic with orbit-finite disjunction, but not by a formula without orbit-finite disjunction.*

The main technical result of this paper is the following theorem, which says that orbit-finite disjunction can be eliminated in the equality symmetry.

Theorem 5.2. *Consider the equality symmetry. Every formula of FM first-order logic is equivalent to a formula that does not use orbit-finite disjunction.*

The proof can be found in the full version of the paper. It uses a notion of functionality, which we believe to be of independent interest, and which is discussed in Section 6. When the colors used by the predicates are just the set $\mathbb{D}$ of data values, Theorem 5.2 is straightforward. The main difficulty is dealing with non-standard sets of colors. We illustrate this with the following examples which show Theorem 5.2 in action for increasingly complicated sets of colors.

Example 5.3. Consider data words over the alphabet $A = \{a, b\} \times \mathbb{D}$. If a position carries the letter $(\sigma, d) \in A \times \mathbb{D}$, then we say that that it has label σ and data value d. Consider the language: "some data value appears only on positions with label a". This language is expressed by the formula

$$\bigvee_d \exists x \; d(x) \wedge \forall x \; d(x) \Rightarrow a(x),$$

where $d(x)$ is the basic type which holds for positions where the data value is d. The orbit-finite disjunction in this formula can be eliminated by encoding the data value d by a position y:

$$\exists y \forall x \; x \sim y \Rightarrow a(x),$$

where $x \sim y$ says that x and y carry the same data value, also a basic type. The same trick, of encoding a data value by a position, works for every formula over this alphabet.

Example 5.4. For $k \in \mathbb{N}$, consider the alphabet

$$B_k = \{a, b\} \times P_{\leq k}(\mathbb{D}),$$

which is like in the previous example, except that the second coordinate is now not a single data value, but an (unordered) set of at most k data values. Let us

first study the case of $k = 2$. Consider the language "some data value appears (in the set) only on positions with label a".

$$\bigvee_d \exists x\ (d \in x) \wedge \forall x\ (d \in x) \Rightarrow a(x),$$

where $d \in x$ is a unary basic type, which selects positions that contain d. Let us use the name *witness* for a data value d which satisfies the formula $\forall x\ (d \in x) \Rightarrow a(x)$. The trick from Example 5.3 was to encode a witness by a position. This trick does not always work for the alphabet B. Consider for instance the word

$$(a, \{1\})(b, \{2, 3\})(a, \{1, 3\})(a, \{2, 4\})(a, \{5, 6\})(a, \{5, 6\})$$

The witnesses are the data values $1, 4, 5, 6$. The witness 1 can be defined in terms of the first position: it is the unique data value in the set $\{1\}$, which appears in the first position. The witness 4 can be defined in terms of two positions: it is the unique data value which appears in the set $\{2, 4\}$ on the fourth position but not in the set $\{2, 3\}$ on the second position. Finally, witnesses 5 and 6 can only be defined as a set of size two; they cannot be distinguished. One can see that these three types of witnesses are the only possible ones for $k = 2$. All of these three types can be captured by the following formula, which does not use orbit-finite disjunction:

$$\exists y_1 \exists y_2 \forall x (\emptyset \subsetneq y_1 \cap y_2 \subseteq x) \Rightarrow a(x),$$

where $\emptyset \subsetneq y_1 \cap y_2 \subseteq x$ is a basic type, which says that the intersection of the sets of data values in y_1 and y_2 is non-empty and included in the set in x.
For $k > 2$, one needs more complicated expressions to define some data values, such as: "the data value that appears in positions five, six and seven, but not eight and two". Also, one can have sets of up to k data values that cannot be distinguished from each other.

5.1 Standard data words

Consider the special case where the models are data words as in Example 3.1 and the alphabet is of the form $A_{\text{fin}} \times \mathbb{D}$, where A_{fin} is a finite set. As mentioned in the introduction, there is an established logic for words over this kind of alphabet, which has a predicate $x < y$ for the position order, a predicate $x \sim y$ for equal data values, and a label predicate $a(x)$ for every $a \in A_{\text{fin}}$ (note that this logic does not allow orbit-finite disjnuctions). A simple consequence of Theorem 5.2 is the following theorem:

Theorem 5.5. *Let A_{fin} be a finite set. Let L be an equivariant language over the alphabet $A_{\text{fin}} \times \mathbb{D}$. The following conditions are equivalent:*

1. *L is definable by a formula of FM first-order logic, possibly including orbit-finite disjunction.*
2. *L is definable by a formula of the standard first-order logic for data words, which has predicates for the position order $x < y$, equal data value $x \sim y$, and the labels $\{a(x)\}_{a \in A_{\text{fin}}}$.*

6 Functionality and locality

In this section we define a key concept for the proof of Theorem 5.2. Our proof technique is to encode data values in elements of the domain of the relational structure (which corresponds to positions in the case of data words). As illustrated in Example 5.4:

1. Sometimes, more than one element is needed to define a data value;
2. Sometimes, a data value can only be defined in combination with some indistinguishable other data value;
3. Sometimes, both problems above hold simultaneously.

This section is devoted to a study of how one can define a data value, or more generally an element of some orbit finite set, in terms of a relational structure.

Functionality. In normal sets, without data values and group actions, the expression "f is a function of g" makes sense only when both f and g are functions with a common domain. For instance, one can say "the area of a circle is a function of its radius", which is formalized as two functions on the domain of circles, the area and radius functions. Another example: "a person's taste in football is a function of their sympathy for Real Madrid".
With data values, the notion of functionality makes sense for arbitrary objects. Suppose that x is an FM set or a data value, likewise y. Let C be a finite set of data values. We say that *y is a C-supported function of x* if there is a C-supported function

$$f : X \to Y \qquad \text{such that } x \in X \text{ and } y \in Y$$

which maps x to y. In the spacial case of $C = \emptyset$, we say that y is an equivariant function of x. (In the classical symmetry, which corresponds to normal sets, one can always take $X = \{x\}$, $Y = \{y\}$. In this case, every y is an equivariant function of every x, which is why the definition is not interesting.)

Example 6.1. The data value 2 is an equivariant function of the three-letter data word 123. In this case X is the set of data words of length three, Y is $\mathbb{D}$, and f maps a word to its second letter (there are other choices for X, Y and f). The data value 2 is an equivariant function of $\{2\} \in P(\mathbb{D})$. The data value 2 is not an equivariant function of the set $\{1, 2, 3\} \in P(\mathbb{D})$, or of the empty set $\emptyset \in P(\mathbb{D})$. The data value 2 is a $\{2\}$-supported function, but not a $\{3\}$-supported function, of the data value 1.

We can now state the main theorem of this section which concerns the first issue in the list at the beginning of this section: more than one element might be needed to define a data value.

Theorem 6.2 (Local Functionality Theorem). *Let X be an orbit-finite FM set, and Σ an FM relational signature. Let C be a finite set of data values that supports X and Σ. There is some $k \in \mathbb{N}$ such that for every $x \in X$ and every nominal relational structure $\mathfrak{A}$, the following conditions are equivalent*

– *x is a C-supported function of $\mathfrak{A}$;*
– *x is a C-supported function of $\mathfrak{A}|\bar{a}$, for some tuple $\bar{a} \in (\mathrm{dom}(\mathfrak{A}))^k$.*

The point of Theorem 6.2 is that the bound k depends on X, Σ and C, but not on $\mathfrak{A}$. When proving Theorem 5.2, we use this result in the following form: if a parameter $i \in I$ of an orbit-finite disjunction $\bigvee_{i \in I} \varphi_i$ is an equivariant function of a model, then it is an equivariant function of a small tuple $\bar{a}$, and the tuple can be captured using k existential quantifiers.

7 Conclusions

We have defined a notion of first-order logic for models that talk about data values. The main technical result is that orbit-finite disjunction can be eliminated in the equality symmetry. Possibilities of future work include:

– Using orbit-finite disjunction, one gets a natural notion of star-free languages of data words. Is this notion equivalent to FM first-order logic?
– Elimination of orbit-finite disjunction works in the equality symmetry, but not in the graph symmetry. In which symmetries does it work? We conjecture that it also works in the total order symmetry and the forest order symmetry, see [1]. We intend to investigate this issue further.
– Can one use the syntax of FM first-order logic to define new fragments of first-order logic on data words that have decidable satisfiability?

References

1. Mikolaj Bojanczyk, Laurent Braud, Bartek Klin, and Slawomir Lasota. Towards nominal computation. In *POPL*, pages 401–412, 2012.
2. Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata with group actions. In *LICS*, pages 355–364, 2011.
3. M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.
4. Murdoch James Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.
5. Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003.
6. Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, pages 41–57, 2006.