# Fraenkel-Mostowski Sets with Non-Homogeneous Atoms

Mikołaj Bojańczyk*, Sławomir Lasota**

University of Warsaw

Fraenkel-Mostowski sets (FM sets) are a variant of set theory, where sets can contain atoms. The existence of atoms is postulated as an axiom. The key role in the theory of FM sets is played by permutations of atoms. For instance, if $a, b, c, d$ are atoms, then the sets

$$\{a, \{a, b, c\}, \{a, c\}\} \qquad \{b, \{b, c, d\}, \{b, d\}\}$$

are equal up to permutation of atoms. In a more general setting, the atoms have some structure, and instead of permutations one talks about automorphisms of the atoms. Suppose for instance that the atoms are real numbers, equipped with the successor relation $x = y + 1$ and linear order $x < y$. Then the sets

$$\{-1, 0, 0.3\} \qquad \{5.2, 6.2, 6.12\}$$

are equal up to automorphism of the atoms, but the sets

$$\{0, 2\} \qquad \{5.3, 8.3\}$$

are not. (In the second example, the two sets can be mapped to each other by a partial automorphism, but not by one that extends to a automorphism of the real numbers.)

Here is the definition of FM sets. The definition is parametrized by a set of atoms. The atoms are given as a relational structure, which induces a notion of automorphism. (One can also consider atoms with function symbols, but we do not do this here.) Suppose that $X$ is a set which contains atoms (or contains sets which contain atoms, or contains sets which contain sets which contain atoms, and so on). If $\pi$ is an automorphism of atoms, then $\pi$ can be applied to $X$, by renaming all atoms that appear in $X$, and appear in elements of $X$, and so on. We say that a set $C$ of atoms is a *support* of the set $X$ if $X$ is invariant under every automorphism of atoms which is the identity on $C$. (For instance, the set of all atoms is supported by the empty set, because every automorphism maps the set to itself.) Equipped with these notions, we are ready to define the notion of an FM set: a set which is built out of atoms is called an *FM set* if it has some finite support, each of its elements has some finite support, and so on recursively.

FM sets were rediscovered for the computer science community, by Gabbay and Pitts [5]. In this application area, atoms have no structure, and therefore automorphisms are arbitrary permutations of atoms. It turns out that atoms are

* Author supported by ERC Starting Grant "Sosna"

** Author supported by FET-Open Project FoX, grant agreement 233599

a good way of describing variable names in programs or logical formulas, and the automorphisms of atoms are a good way of describing renaming of variables. FM sets are now widely studied in the semantics community, under the name of nominal sets (the name is so chosen because atoms describe variables names).

FM sets turn out to be a good framework for other applications in computer science. These other applications have roots in database theory, but touch other fields, such as verification or automata theory. The motivation in database theory is that atoms can be used as an abstraction for data values, which can appear in a relational database or in an XML document. Atoms can also be used to model sources of infinite data in other applications, such as software verification, where an atom can represent a pointer or the contents of an array cell.

FM sets are a good abstraction for infinite systems because they have a different, more relaxed, notion of finiteness. An FM set is considered finite if it has finitely many elements, up to automorphisms of atoms. (The formal definition is slightly technical, and we skip it here.) We call such a set *orbit-finite*. Consider for example FM sets where the atoms have no structure, and therefore automorphisms are arbitrary permutations. The set of atoms itself is orbit-finite, actually has only one orbit, because every atom can be mapped to every other atom by a permutation. Likewise, the set of pairs of atoms has two elements up to permutation, namely $(a, a)$ and $(a, b)$ for $a \neq b$. Another example is the set of $\lambda$-terms which represents the identity, with variables being atoms:

$$\{\lambda a.a : a \text{ is an atom}\},$$

this set has one orbit. Yet another example concerns automata with registers for storing atoms, as introduced by Francez and Kaminski in [6]: up to permutation, there are finitely many configurations of every such automaton.

The language of FM sets is so robust that one can meaningfully restate all of the results in a textbook on automata theory, replacing sets by FM sets and finite sets by orbit-finite sets, see [3] for examples. Some of the restated theorems are true, some are not. Results that fail in the FM setting include all results which depend on the subset construction, such as determinization of finite automata, or equivalence of two-way and one-way finite automata. Results that work in the FM setting include the Myhill-Nerode theorem, or the equivalence of pushdown automata with context free grammars (under certain assumptions on the structure of atoms, which will be described below). Under the same assumptions on the structure of atoms, the theory of computability still works in the FM setting. More specifically, one can design a programming language which manipulates orbit-finite FM sets, just like other programming languages manipulate lists or trees [2]. (This programming language is not a violation of the Church-Turing thesis – after some translation, the programming language can be executed on a normal computer.)

## 1 Non-homogeneous atoms

What are the assumptions on the atoms that are mentioned above, in results such as the equivalence of pushdown automata with context-free grammars? The

assumption is that the atoms are a *homogeneous structure*, which means that every partial automorphism of the atoms extends to a complete automorphism. For instance, the atoms with no structure are homogeneous, because every partial bijection extends to a complete bijection. Other examples of homogeneous structures include: the rational (or real) numbers with order, or the Rado graph (also called the random graph). When the atoms are a homogeneous structure, then orbit-finite sets are relatively well-behaved, in particular orbit-finite sets are closed under products and subsets.

When the atoms are not homogeneous, then orbit-finite sets might no longer be closed under products and subsets. This means that almost all natural constructions might fail, e.g. languages recognized by automata with orbit-finite state spaces need no longer be closed under intersection (or even union in the case of deterministic automata). In this talk, I will discuss how one can try to work with FM sets when the atoms are not a homogeneous structure. I will illustrate this on two examples of non-homogeneous atoms, which are presented below.

### 1.1 Integers with order and successor

In this section, we assume that the atoms are integers, with the relations

$$y = x + 1 \qquad \text{and} \qquad x < y.$$

The automorphisms of this structure are the translations, i.e. functions of the form $x \mapsto x + y$, where $y \in \mathbb{Z}$. In other words, the automorphism group is isomorphic to the integers with addition. (Actually, as the reader can easily see, the notion of FM sets depends only on the automorphisms of the atoms, and the automorphisms stay the same when one keeps just one of the relations: the successor or linear order. Therefore, the notion of FM sets would be the same if we assumed, for example, that only the order relation was allowed.)

These atoms are not a homogeneous structure. For instance, the function

$$0 \mapsto 0 \qquad 2 \mapsto 3$$

is a partial automorphism, but it does not extend to a complete automorphism. This leads to all sorts of problems. [1] For instance, we cannot use the theorem that the product of orbit-finite sets is also orbit-finite. Indeed, this is actually false: the set of atoms $\mathbb{Z}$ has one orbit, but the set of pairs of atoms $\mathbb{Z} \times \mathbb{Z}$ has infinitely many orbits, one for every diagonal

$$\{(x, x + y) : x \in \mathbb{Z}\} \qquad \text{where } y \in \mathbb{Z}.$$

The notion of finite support trivializes. This is because everything is supported by the set $\{0\}$, because the identity is the only automorphism of integers

[1] This structure becomes homogeneous if one treats the successor as a function symbol. However, homogeneous structures with function symbols do not have all the good properties of homogeneous structures with only relation symbols.

that preserves 0. One troubling consequence is that orbit-finite sets are not closed under (finitely supported) subsets. For instance, the set of even atoms is not orbit-finite, but it is contained in the set of all atoms, which is orbit-finite.

Despite these pathologies, one can still try to get some work done.

Let us have a look at finite deterministic automata. Typical orbit-finite sets include the integers, or the integers modulo some finite natural number. One can show that these are essentially the only examples: every orbit-finite set is a finite disjoint union of such sets. Consider then an automaton, where the state space $Q$ and the input alphabet $A$ are orbit-finite sets. By the remarks above, we understand what the states and the alphabet look like. What about the transition function? Unfortunately, every possible function

$$\delta : Q \times A \to Q$$

is finitely supported, since being finitely supported is a trivially satisfied condition, because every set is supported by the set $\{0\}$. Therefore, the class of automata is too rich to be analyzed (in particular, there are uncountably many automata). To make the class more manageable, let us make two additional assumptions.

1. The transition function, the initial state, and the set of final states all have empty support. In other words, all of these objects are invariant under the action of arbitrary automorphisms of the integers.
2. The transition function is semilinear (under a natural notion of semilinearity).

The first assumption is perhaps natural, the second one is more ad hoc. Under these assumptions, one can try to do some constructions of automata theory. It turns out that emptiness is decidable for such automata, even nondeterministic automata. A more sophisticated result is that minimization of deterministic automata works, i.e. for every deterministic automaton satisfying the assumptions, the minimal automaton also satisfies the assumptions, and can be effectively computed. The minimization algorithm is non-trivial, it relies on decidability of quantifier-free Presburger arithmetic with divisibility predicates [7]. What is also interesting, a standard partition-refinement algorithm for minimization fails.

### 1.2 Real numbers with order and successor

In this section, we assume that the atoms are real numbers, with the relations

$$y = x + 1 \qquad x < y.$$

(The automorphism group of this structure works as follows. An automorphism is uniquely described by what it does to the half-open unit interval $[0; 1)$, this interval must be mapped homeomorphically to some other half-open unit interval.) These atoms are not homogeneous for the same reason as the integers. However, the notion of support does not trivialize, e.g. a subset of the interval $[0; 1)$ has

finite support if and only if it is a finite boolean combination of (half-open, open, or closed) intervals.

It turns out that in FM sets with these atoms, the notion of finite automaton is a generalization of timed automata [1], as introduced by Alur and Dill. Furthermore, classical results on timed automata, such as the emptiness check, can be recovered as a special case of more general algorithms working in FM sets. Finally, and what is perhaps most interesting, some new things can be done with timed automata which are made possible by the framework of FM sets. The first new thing is a robust notion of minimization for deterministic automata, connected with a Myhill-Nerode theorem. The second new thing is a machine-independent characterization of languages recognized by deterministic timed automata. These results are included in the paper [4].

## References


1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
2. Mikolaj Bojanczyk, Laurent Braud, Bartek Klin, and Slawomir Lasota. Towards nominal computation. In *POPL*, pages 401–412, 2012.
3. Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata with group actions. In *LICS*, pages 355–364, 2011.
4. Mikolaj Bojanczyk and Slawomir Lasota. A machine-independent characterization of timed languages. In *ICALP (2)*, pages 92–103, 2012.
5. Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.
6. Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.
7. L. Lipshitz. The Diophantine problem for addition and divisibility. *Trans. Amer. Math. Soc.*, 235:217–283, 1978.