A study of substitution, using nominal techniques and Fraenkel-Mostowki sets

Murdoch J. Gabbay

http://www.gabbay.org.uk

Abstract

Fraenkel-Mostowski (FM) set theory delivers a model of names and alpha-equivalence. This model, now generally called the 'nominal' model, delivers inductive datatypes of syntax with alpha-equivalence — rather than inductive datatypes of syntax, quotiented by alpha-equivalence.

The treatment of names and alpha-equivalence extends to the entire sets universe. This has proven useful for developing 'nominal' theories of reasoning and programming on syntax with alpha-equivalence, because a sets universe includes elements representing functions, predicates, and behaviour.

Often, we want names and alpha-equivalence to model capture-avoiding substitution. In this paper we show that FM set theory models capture-avoiding subsitution for names in much the same way as it models alpha-equivalence; as an operation valid for the entire sets universe which coincides with the usual (inductively defined) operation on inductive datatypes.

In fact, more than one substitution action is possible (they all agree on sets representing syntax). We present two distinct substitution actions, making no judgement as to which one is 'right' — we suspect this question has the same status as asking whether classical or intuitionistic logic is 'right'. We describe the actions in detail, and describe the overall design issues involved in creating any substitution action on a sets universe.

Along the way, we think in new ways about the structure of elements of FM set theory. This leads us to some interesting mathematical concepts, including the notions of planes and crucial elements, which we also describe in detail.[1]

Keywords: Substitution, nominal techniques, Fraenkel-Mostowski set theory.

[1]Thanks to Arnon Avron. Supported by grant RYC-2006-002131 at the Polytechnic University of Madrid.

Contents

1 Introduction **2**
1.1 Design issues . . . 3

2 Fraenkel-Mostowski set theory **6**
2.1 Axioms, permutations, equivariance . . . 6
2.2 Support . . . 8
2.3 Orbits under the permutation action . . . 11
2.4 α-abstraction . . . 12

3 The substitution action **14**
3.1 Axioms, pointwise substitution action, syntactic substitution action . . . 14
3.2 A short quiz . . . 17
3.3 The planes of a set . . . 18
3.4 Construction of the first substitution action . . . 19
3.5 Substitutions on syntax, substitutions commuting . . . 26

4 New properties of FM: a-orbits and crucial elements **29**
4.1 a-orbits . . . 30
4.2 Crucial elements . . . 34
4.3 Operations on sets of crucial elements . . . 36

5 A second substitution action, using crucial elements **37**
5.1 Definition and properties . . . 38
5.2 Scope-extrusion and hereditarily finite crucial sets . . . 42

6 Summary, related work, and conclusions **45**
6.1 Summary of the two substitution actions . . . 46
6.2 Related work . . . 47
6.3 The rôle of Fraenkel-Mostowski sets . . . 48

1. Introduction

Fraenkel-Mostowski set theory [7, 40] was originally developed to prove the independence of the axiom of Choice from the other axioms of Zermelo-Fraenkel set theory. It was re-discovered and used — rather far from its original application — by the author and Pitts to provide a model of naming and abstraction in datatypes of abstract syntax [23].

Fraenkel-Mostowski set theory is a foundational theory and part of its usefulness is that the model of naming and abstraction which it provides [23, 15], extends smoothly from sets representing abstract syntax to all sets including those representing non-syntactic entities such as functions, domains, and predicates. It has been a success story of the continuing relevance of set-theoretic foundations to computer science.

This model of names has inspired programming languages with datatypes for names and constructors to bind those names [36, 8], logics [35, 21], it has been used to study

correctness properties of functional programming with storage [6], and in many other applications.

However, why should we wish naming and abstraction in the first place? In many cases, this is because we want a names with a capture-avoiding substitution action. For example, this is the behaviour displayed by variables in syntax.

In this paper we demonstrate by a non-trivial but elementary construction on sets that Fraenkel-Mostowski set theory also provides a model of capture-avoiding substitution on names. This model is compatible with the notion of abstraction from previous work; thus substitution is modelled in sets and we extend the Fraenkel-Mostowski story about names and abstraction from syntax to semantics, by showing how to model substitution on any set, not just sets representing syntax.

We shall show that in the case of sets representing abstract syntax, the substitution action coincides with the 'usual' substitution defined inductively on syntax. We show that on *all* sets — not just those representing syntax — the action satisfies the conditions laid out by an algebraic axiomatisation of substitution investigated in previous work [20, 22].

We will also show that more than one possible 'substitution action' exists: the behaviour on sets representing abstract syntax is constrained by the axioms; for more complex sets there is some freedom, and we explore the design space and specify two actions in full detail.

During the exploration of our two substitution actions we introduce some mathematical concepts which may be useful tools in their own right:

- Planes, in Subsection 3.3, particularly Theorems 3.12 and 3.13.
- a-orbits in Subsection 4.1. A fundamental technical result is Theorem 4.5, though Definition 4.11 and Remark 4.12 are more readable.
- Crucial elements in Subsection 4.2, notably Corollary 4.14, and the interaction of crucial elements with support, symmetric set difference, and permutation described in Corollary 4.15, Lemma 4.17, and Lemma 4.19.

1.1. Design issues

We now sketch the design issues involved in defining a substitution action on Fraenkel-Mostowski sets. Definitions and notation are given in full formal detail in this paper, and also also [23, 17].

Fraenkel-Mostowski sets contain atoms $a, b, c, \ldots$. These are empty ($x \in a$ is impossible) but they are not equal to the empty set. Following the nominal style introduced in [23] we use atoms to model variable symbols;

- In [23] we showed how to construct datatypes of syntax up to α-equivalence ('λ-terms', 'terms of first-order logic', 'π-calculus terms', and so on) such that substitution — and other functions — could be defined on the datatypes purely inductively.
- In this paper we give a substitution action on all of the sets universe, write it $z[a{\mapsto}x]$ and read it 'substitute a with x in z' (so z and x can be anything, not just elements representing syntax).

We want our new substitution to coincide with 'usual' substitution on elements representing syntax (i.e. representing labelled trees). This gives us useful information which we now sketch:

Atoms model variable symbols, therefore let

$$a[a{\mapsto}x] = x \quad \text{and} \quad b[a{\mapsto}x] = b$$

(x is any element, a and b are any two distinct atoms).

If Z is a finite set of elements then let substitution be pointwise:

$$Z[a{\mapsto}x] = \{z[a{\mapsto}x] \mid z \in Z\} \quad \text{if } Z \text{ is a finite set.}$$

This ensures that tuples $(z_1, z_2) = \{\{z_1\}, \{z_1, z_2\}\}$ satisfy

$$(z_1, z_2)[a{\mapsto}x] = (z_1[a{\mapsto}x], z_2[a{\mapsto}x]).$$

The same holds for n-tuples and also, modulo some technical details, for infinite tuples (Corollary 3.27).

In nominal style we model α-abstraction by taking an α-equivalence class of 'renamed variants' of a set. For example the nominal model of 'α-abstract the atom a in (a, b)', written $[a](a, b)$ (Definition 2.20) is

$$[a](a, b) = \{(a, (a, b)),\ (c, (c, b)),\ (d, (d, b)),\ (e, (e, b)),\ (f, (f, b)),\ \ldots\}.$$

This is the model of syntax up to α-equivalence presented in [23].

For the reader to understand what follows in this paper, and why the mathematics will be set up in the way that it will be, it is important to have some intuition of how this model of abstraction works.

We give another example: $[a](a, b, d)$ (where (z_1, z_2, z_3) is $(z_1, (z_2, z_3))$) is

$$[a](a, b, d) = \{(a, (a, b, d)),\ (c, (c, b, d)),\ (e, (e, b, d)),\ (f, (f, b, d)),\ \ldots\}.$$

Note that

$$(d, (d, b)) \in [a](a, b) \quad \text{and} \quad (d, (d, b, d)) \notin [a](a, b, d)$$

because d does not 'clash' with (a, b) but it does 'clash' with (a, b, d).

In nominal techniques, when we build equivalence classes of renamed variants like $[a](a, b)$ and $[a](a, b, d)$ above, we 'avoid name-clash'.

Now, we want substitution to distribute over abstraction in a capture-avoiding way. That is,

$$\text{we want} \quad ([a](a, b))[b{\mapsto}(b, d)] = [a](a, b, d).$$

We cannot obtain what we want by operating *pointwise* on the elements of $[a](a, b)$, i.e. by letting

$$Z[a{\mapsto}x] = \{z[a{\mapsto}x] \mid z \in Z\} \quad \text{always}$$

— not just for finite sets Z. For, if we operated pointwise, we would not know *not* to put into $([a](a, b))[b{\mapsto}(b, d)]$ the element $(d, (d, b))[b{\mapsto}(b, d)] = (d, (d, b, d))$. Substitution must detect α-equivalence classes and adjust for them by adding or removing elements with 'name-clash'.

One reason this is a difficult question is that not all sets look like the tame examples we selected above. Elements may be (amongst many other things) unions of abstractions

$$[a](a,b) \cup [a](a,b,d)$$

or equivalence classes which represent simultaneous abstraction by more than one atom

$$\begin{array}{l}\{\ \{a,b,c\},\ \{d,b,c\},\ \{e,b,c\},\ \{f,b,c\},\ \ldots \\ \ \ \{a,d,c\},\ \{b,d,c\},\ \{e,d,c\},\ \{f,d,c\},\ \ldots \\ \ \ \{a,e,c\},\ \{b,e,c\},\ \{d,e,c\},\ \{f,e,c\},\ \ldots\}\end{array}$$

— we can think of this as 'abstract a and b simultaneously in $\{a,b,c\}$' and write it in an informal notation which we do not make formal as

$$\{a,b\}\{a,b,c\}.$$

If we apply $[c{\mapsto}(b,d)]$ to this we want to 'avoid name-clash' with b and d, and 'α-convert b', and 'fill in renamed variants mentioning c', to obtain the element

$$\begin{array}{l}\{\ \{a,c,(b,d)\},\ \{d,c,(b,d)\},\ \{e,c,(b,d)\},\ \{f,c,(b,d)\},\ \ldots \\ \ \ \{a,e,(b,d)\},\ \{c,e,(b,d)\},\ \{f,e,(b,d)\},\ \{g,e,(b,d)\},\ \ldots \\ \ \ \{a,e,(b,d)\},\ \{c,e,(b,d)\},\ \{f,e,(b,d)\},\ \{g,e,(b,d)\},\ \ldots\}\end{array}$$

which we can write as

$$\{a,c\}\{a,c,(b,d)\}.$$

This is only a scratch on the surface of the complexity that exists in a model of sets but it is enough to illustrate our strategy for defining a substitution action. The problem with the pointwise definition of substitution is correct maintenance of 'points' in α-equivalence classes so:

> Given a set X, identify X as a union of some generalised notion of α-equivalence classes $X = \bigcup_i X_i$. Now carry out substitution on the individual classes, then put the results together again using sets union.

We can put it another way:

> Substitution *is* pointwise — but the points are not elements, they are (generalised) α-equivalence classes of elements.

Our generalisation of α-equivalence classes is *planes* $u\|_A$ (Subsection 3.3). The set above is $\{a,b,c\}\|_{\{c\}}$ which is the equivalence class of all renamed variants of $\{a,b,c\}$ under permutations of atoms fixing c (i.e. 'α-abstract a and b in $\{a,b,c\}$).

If we apply $[c{\mapsto}(b,d)]$ to this then we obtain $\{a,c,(b,d)\}\|_{\{b,d\}}$. Note that this is the same set as $\{a,e,(b,d)\}\|_{\{b,d\}}$ and $\{c,e,(b,d)\}\|_{\{b,d\}}$ — it is part of our model of α-equivalence on FM sets that we can 'α-rename abstracted atoms' freely.

Given these ideas, the only obstacle to defining substitution on FM sets is deciding how substitution interacts with planes. This is described formally in Subsection 3.4, with several examples.

We mentioned that we give not one but *two* substitution actions in this paper. The second substitution action (Section 5), write it $z[a{\mapsto}x]_2$, is based on the same ideas but with a twist. Instead of acting on all of z, the second substitution action isolates the parts of z which are responsible for it containing a (in nominal terminology; those elements responsible for $a \in \mathit{supp}(z)$ being true; see 'crucial elements' in Section 4). The second substitution action acts, in a suitable sense, only on planes involving these elements. Thus the second substitution action $z[a{\mapsto}x]_2$ is pointwise but only on points (generalised α-equivalence classes) that 'contain a' in a suitable sense given by crucial elements. This gives the second substitution action some different, and arguably better, properties. See Subsection 5.2 for examples and discussion.

2. Fraenkel-Mostowski set theory

2.1. Axioms, permutations, equivariance

The language of FM set theory is first-order logic with two binary predicates $=$ (set equality) and $\in$ (set membership) — just like the language of ZF set theory — *and* one constant symbol $\mathbb{A}$ called 'the set of atoms'.

Definition 2.1. The axioms of FM set theory are given in Figure 1.

We use standard definitional extensions of the language of sets:

$x = \{z \mid z \in x\}$	means	$\forall y.(\forall z.z \in x \Leftrightarrow z \in y) \Rightarrow x = y$
$y = \{z \in x \mid \phi(z)\}$	means	$\forall z.z \in y \Leftrightarrow (z \in x \wedge \phi(z))$
$z = \{F(y) \mid y \in x\}$	means	$\forall u.u \in z \Leftrightarrow \exists y.(F(y) = u \wedge y \in x)$
$z = \{x, y\}$	means	$\forall u.u \in z \Leftrightarrow (u = x \vee u = y)$
$z = \{y \mid \exists y'.(y \in y' \wedge y' \in x)\}$	means	$\forall y.y \in z \Leftrightarrow \exists y'.(y \in y' \wedge y' \in x)$
$z = \{y \mid y \subseteq x\}$	means	$\forall y.y \in z \Leftrightarrow \forall y'.(y' \in y \Rightarrow y' \in x)$
$\emptyset \in x$	means	$\exists z.z \in x \wedge z \notin \mathbb{A} \wedge \forall z'.z' \notin z$
$y \cup \{z\} \in x$	means	$\exists u.u \in x \wedge \forall u'.(u' \in u \Leftrightarrow u \in y \vee u = z)$

In addition $\mathcal{P}_{\mathit{fin}}$ is the finite powerset, and support is described below.

(We mention the intuition of the cumulative hierarchy model of these axioms in Remark 2.12.)

Remark 2.2. We use the following notational conventions:

- An **atom** is a set member of $\mathbb{A}$ (the set of atoms).
- *The permutative convention:* $a, b, c, \ldots$ range over *distinct* atoms unless stated otherwise.
- A, B, C, S, T range over sets of atoms. For example $A \subseteq \mathbb{A}$.
- X, Y, Z, U, V range over elements that are not atoms and may be empty. For example X might equal $\emptyset$ or $\{a, \emptyset\}$, but X cannot equal a.
- x, y, z, u, v range over arbitrary elements.

(**Sets**)	$\forall x.(\exists y.y \in x) \Rightarrow x \notin \mathbb{A}$	
(**Extensionality**)	$\forall x.x \notin \mathbb{A} \Rightarrow x = \{z \mid z \in x\}$	
(**Comprehension**)	$\forall x.\exists y.y \notin \mathbb{A} \wedge y = \{z \in x \mid \phi(z)\}$	(y not free in ϕ)
($\in$-**Induction**)	$\big(\forall x.(\forall y \in x.\phi(y)) \Rightarrow \phi(x)\big) \Rightarrow \forall x.\phi(x)$	
(**Replacement**)	$\forall x.\exists z.z \notin \mathbb{A} \wedge z = \{F(y) \mid y \in x\}$	
(**Pairset**)	$\forall x, y.\exists z.z = \{x, y\}$	
(**Union**)	$\forall x.\exists z.z \notin \mathbb{A} \wedge z = \{y \mid \exists y'.(y \in y' \wedge y' \in x)\}$	
(**Powerset**)	$\forall x.\exists z.z = \{y \mid y \subseteq x\}$	
(**Infinity**)	$\exists x.\emptyset \in x \wedge \forall y.y \in x \Rightarrow y \cup \{y\} \in x$	
(**AtmInf**)	$\mathbb{A} \notin \mathcal{P}_{\mathit{fin}}(\mathbb{A})$	
(**Fresh**)	$\forall x.\exists S \in \mathcal{P}_{\mathit{fin}}(\mathbb{A}).S$ supports x	

Figure 1: Axioms of FM set theory

Definition 2.3. As is standard, we will write $0 = \emptyset$ and $i + 1 = i \cup \{i\}$, and write $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

An atom $a \in \mathbb{A}$ is 'empty', or formally: $\forall x.x \notin a$. However atoms are not equal to the empty set $\emptyset$.

In ZF set theory two empty elements are equal by the axiom of extensionality. In FM set theory the axiom of extensionality (**Extensionality**) is weakened so that an empty element is the empty set $\emptyset$ — or is an atom.

(**AtmInf**) insists that there are infinitely many atoms.

Therefore, any element z is either an atom, or it is a set containing nothing but atoms and (possibly) more sets. For further details see elsewhere [15].

Definition 2.4. Write $(a\ b)$ for the **swapping** of a and b. This is the mapping from $\mathbb{A}$ to itself specified by

$$(a\ b)(a) = b \qquad (a\ b)(b) = a \qquad (a\ b)(c) = c.$$

Note that by our permutative naming convention a, b, and c are distinct.

Write $\circ$ for functional composition. Let π range over functions from $\mathbb{A}$ to $\mathbb{A}$ generated by functionally composing finitely many swappings, and call these **permutations**. A **permutation action** is defined by ϵ-induction [30]:

$$\pi X = \{\pi x \mid x \in X\}$$

The permutation action satisfies *equivariance*. This very useful property is also very easy to prove. Let $\phi(x_1, \ldots, x_n)$ range over predicates in the language of FM set theory which mention variables in $x_1, \ldots, x_n$. An n-rary function $F(x_1, \ldots, x_n)$ can be expressed by a predicate $\phi_F(x_1, \ldots, x_n, z)$ such that for each $x_1, \ldots, x_n$ there is a unique z making ϕ_F true. Then **equivariance** is the following two properties:

Theorem 2.5. *$\phi(x_1,\ldots,x_n) \Leftrightarrow \phi(\pi x_1,\ldots,\pi x_n)$, and*
$\pi(F(x_1,\ldots,x_n)) = F(\pi x_1,\ldots,\pi x_n)$
always hold.

Proof. The first part is by an easy induction on the syntax of ϕ. We consider just one case: $x \in y$ implies $\pi x \in \pi y$ follows directly from the fact that $\pi y = \{\pi y' \mid y' \in y\}$. The reverse implication uses π^{-1}.

The second part, for F, follows using the encoding of functions as predicates. □

2.2. Support

Definition 2.6. If $A \subseteq \mathbb{A}$ write

$$\mathsf{fix}(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}.$$

Say that $A \subseteq \mathbb{A}$ **supports** x when

$$\forall \pi \in \mathsf{fix}(A).\pi x = x.$$

- We write
$$\mathit{supp}(x) \quad \text{for} \quad \bigcap\{S \mid S \text{ is finite and supports } x\}$$
and call this the **support** of x. $\mathit{supp}(x)$ is well-defined because $(\mathbf{Fresh})$ insists that some finite S supporting x must exist.

- We write $a\#x$ when $a \notin \mathit{supp}(x)$ and read this as a is **fresh for** x. We may write $a\#t_1, t_2$ for '$a\#t_1$ and $a\#t_2$', and so on.

The notion of support goes back to Fraenkel and Mostowski [29, Chapter 4]. The application in computer science came later [23, 15].

Remark 2.7. For example:

- $\mathit{supp}(\emptyset) = \emptyset$. $\quad \pi\emptyset = \emptyset$ for all $\pi \in \mathsf{fix}(\emptyset)$.
- $\mathit{supp}(\mathbb{A}) = \emptyset$.
$\pi\{a,b,c,\ldots\} = \{\pi(a),\pi(b),\pi(c),\ldots\} = \{a,b,c,d,\ldots\}$ for all $\pi \in \mathsf{fix}(\emptyset)$.
- $\mathit{supp}(a) = \{a\}$. $\quad \pi(a) = a$ for all $\pi \in \mathsf{fix}(\{a\})$.
- $\mathit{supp}(\{a\}) = \{a\}$. $\quad \pi(\{a\}) = \{a\}$ for all $\pi \in \mathsf{fix}(\{a\})$.
- $\mathit{supp}(\mathbb{A} \setminus \{a\}) = \{a\}$.
$\pi\{b,c,d,\ldots\} = \{\pi(b),\pi(c),\pi(d),\ldots\} = \{b,c,d,\ldots\}$ for all $\pi \in \mathsf{fix}(\{a\})$.
- $\mathit{supp}(\{a,b\}) = \{a,b\}$. $\quad \pi\{a,b\}{=}\{a,b\}$ for all $\pi{\in}\mathsf{fix}(\{a,b\})$.
- $\mathit{supp}(\mathbb{A} \setminus \{a,b\}) = \{a,b\}$.
$\pi\{c,d,e,\ldots\} = \{\pi(c),\pi(d),\pi(e),\ldots\} = \{c,d,e,\ldots\}$ for all $\pi \in \mathsf{fix}(\{a,b\})$.

- $supp(\{a,\{a\},\{c\},\{d\},\ldots\}) = \{a,b\}$.

$$\begin{aligned}\pi(\{a,\{a\},\{c\},\{d\},\ldots\}) &= \{\pi(a),\{\pi(a)\},\{\pi(c)\},\{\pi(d)\},\ldots\}\\ &= \{a,\{a\},\{c\},\{d\},\ldots\}\end{aligned}$$

  provided that $\pi \in \mathsf{fix}(\{a,b\})$.

Remark 2.8. Ideas from syntax match ideas from FM sets as follows:

- *variable symbols* matches *atoms* and
- *free variables* matches *support*.

It is possible to take the complement of a set, but not possible to take the complement of a syntax tree. It is therefore important to realise that sets are more general than syntax, and in particular that $a \notin X$ and $a\#X$ are *not* the same thing. $supp(X)$ measures how 'conspicuous' a is in X, either by its set membership or *lack* of set membership. For example:

$$\begin{array}{llll} a \in \mathbb{A} \text{ and } a\#\mathbb{A} & a \notin \emptyset \text{ and } a\#\emptyset & a \notin a \text{ and } a{\in}supp(a) \\ a \in \{a\} \text{ and } a{\in}supp(\{a\}) & a \notin \mathbb{A}\backslash\{a\} \text{ and } a{\in}supp(\mathbb{A}\backslash\{a\}) & \end{array}$$

Remark 2.9. Not every collection has finite support.

- $\{a,c,e,g,\ldots\}$ (the set of 'every other atom') is not finitely supported, and is excluded from the cumulative hierarchy model of Remark 2.12 below. There is no finite $S \subseteq \mathbb{A}$ such that if $\pi \in \mathsf{fix}(S)$ then $\pi\{a,c,e,g,\ldots\} = \{a,c,e,g,\ldots\}$.
- A set well-ordering atoms $\leq$ is not finitely supported, and is excluded from the cumulative hierarchy model of Remark 2.12:

$$\leq = \{\{a\},\ \{a,b\},\ \{a,b,c\},\ \{a,b,c,d\},\ \ldots\}$$

  There is no finite $S \subseteq \mathbb{A}$ such that if $\pi \in \mathsf{fix}(S)$ then $\pi \leq = \leq$.
- A set ϵ choosing from every unordered pair of atoms, an atom (the 'greater' one according to $\leq$ above), is not finitely supported and is excluded from the cumulative hierarchy model of Remark 2.12:

$$\begin{aligned}\epsilon = \{&\{\{a,b\},b\},\ \{\{a,c\},c\}\ \{\{a,d\},d\},\ \ldots\\ &\{\{b,c\},c\},\ \{\{b,d\},d\}\ \{\{b,e\},e\},\ \ldots\\ &\ldots\}\end{aligned}$$

  There is no finite $S \subseteq \mathbb{A}$ such that if $\pi \in \mathsf{fix}(S)$ then $\pi\epsilon = \epsilon$.

Theorem 2.10 was the reason Fraenkel-Mostowski set theory was originally created and studied:

Theorem 2.10. *The axiom of choice is inconsistent with the axioms of FM.*

Proof. Given the examples above it is easy to see that choice functions, and well-orderings on sets, are not in general finitely supported. For more details see [7] □

Remark 2.11. Theorem 2.10 may give the impression that the model of syntax-with-binding from Fraenkel-Mostowski techniques is inconsistent with the axiom of choice, but this is not so.

Set theory is untyped. It does no harm to admit the existence of elements that do not have finite support, so long as we do not try proving results involving freshness about them, i.e. all the non-trivial results which are the topic of this paper. We work in Fraenkel-Mostowski sets because it saves us writing 'if $x, y, z, \ldots$ have finite supporting sets' at the start of all our results.

Specifically, if we drop $(\mathbf{Fresh})$ from Figure 1, which asserts that every element must have finite support, then we obtain another standard set theory; Zermelo-Fraenkel set theory with atoms. This still satisfies equivariance (Theorem 2.5) because equivariance is a property of any first-order theory with atoms; models of Zermelo-Fraenkel set theory can contain elements representing well-orderings and choice functions (which may not have finite support); and all the results in this paper can easily be proven, subject to a clause 'provided $x, y, z, \ldots$ have a finite supporting set'.

Remark 2.12. (The cumulative hierarchy model) FM is a theory in first-order logic. As is often the case, we have a clear intuition in mind for a standard model; the *cumulative hierarchy* model is the collection $\mathcal{U}$ defined as follows:

$$\begin{aligned}\mathcal{U}_0 &= \mathbb{A}\\ \mathcal{U}_{i+1} &= \mathcal{U}_i \cup \{X \subseteq \mathcal{U}_i \mid X \text{ has a finite supporting set}\}\end{aligned}$$

Then $\mathcal{U} = \bigcup_i \mathcal{U}_i$. The reader can imagine all our constructions taking place in this model and no harm will come of it.

The cumulative hierarchy of Zermelo-Fraenkel set theory with atoms is obtained by taking *all* X instead of only X with finite support, at each stage. That is, the cumulative hierarchy of 'well-behaved' Fraenkel-Mostowski sets naturally embeds in a larger universe of Zermelo-Fraenkel sets with atoms, containing elements representing choice functions, well-orderings, and so on — but in which not all elements can be assigned substitution actions for atoms, and so on, as we shall now construct.

We now prove Theorem 2.13, Theorem 2.14, and Theorem 2.15. These are three basic results about support, which will be very useful in what follows:

Theorem 2.13. *If S and T support x and are finite, then so does $S \cap T$.*

As a corollary, $supp(x)$ is the unique smallest finite set supporting x.

Proof. The corollary follows by elementary calculations and $(\mathbf{Fresh})$.

Suppose κ fixes $S \cap T$ pointwise. We must show $\kappa x = x$.

Write K for $\{a \mid \kappa(a) \neq a\}$. Choose an injection ι of $T \setminus S$ into $\mathbb{A} \setminus (S \cup T \cup K)$ (we can say 'ι freshens $T \setminus S$'). Let $\pi(a) = \iota(a)$ and $\pi(\iota(a)) = a$ for $a \in T \setminus S$, and $\pi(a) = a$ otherwise. Note that $\pi = \pi^{-1}$. π fixes S pointwise so $\pi x = x$. Also $\pi \circ \kappa \circ \pi$ fixes T pointwise so $(\pi \circ \kappa \circ \pi)x = x$. We apply π to both sides and simplify and conclude that $\kappa x = x$ as required. □

Theorem 2.13 means that the action of π on x depends *only* on the values of π on the atoms in $supp(x)$.

Theorem 2.14. *S supports x if and only if πS supports πx. As a corollary, $\pi supp(x) = supp(\pi x)$.*

Proof. From Theorem 2.5. □

Theorem 2.15. $supp(F(x_1, \ldots, x_n)) \subseteq supp(x_1) \cup \cdots \cup supp(x_n)$.

Proof. Suppose $S = supp(x_1) \cup \ldots \cup supp(x_n)$. By Theorem 2.5 $\pi F(x_1, \ldots, x_n) = F(\pi x_1, \ldots, \pi x_n)$. If $\pi \in \mathsf{fix}(S)$ then $\pi \in \mathsf{fix}(x_1) \cap \ldots \cap \mathsf{fix}(x_n)$ and the result follows. □

2.3. Orbits under the permutation action

We introduce a new notion which we will make much use of in what follows:

Definition 2.16. Suppose $A \subseteq \mathbb{A}$. Write

$$u\|_A \quad \text{for} \quad \{\pi u \mid \pi \in \mathsf{fix}(A)\}.$$

(Recall that $\mathsf{fix}(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}$.) We call $u\|_A$ a set of **renamed variants** of u under permutations fixing atoms in A.

As discussed in Remark 2.8 the idea from syntax of 'free variables of' matches the idea in Fraenkel-Mostowski sets of 'support'. α-equivalence is how we remove free variables from terms. Taking equivalence classes is a standard way to abstract structure using sets. Thus, $u\|_A$ is a natural sets model of the intuition

> "simultaneously α-abstract in u for all atoms not in A."

This is made formal in Theorem 2.18 and Subsection 2.4.

For example:

$$\begin{aligned}
\{a\}\|_{\emptyset} &= \{a,\ b,\ c,\ d,\ e,\ f,\ \ldots\} \\
\{a\}\|_{\{a\}} &= \{a\} \\
\{b\}\|_{\{a\}} &= \{b,\ c,\ d,\ e,\ f,\ \ldots\} \\
\{a,b\}\|_{\{a,c\}} &= \{\{a,b\},\ \{a,d\},\ \{a,e\},\ \{a,f\},\ \ldots\}
\end{aligned}$$

Lemma 2.17. *If $\pi \in \mathsf{fix}(A)$ then $u\|_A = (\pi u)\|_A$.*

Proof. $u\|_A$ is the orbit of u under the action of $\mathsf{fix}(A)$. $\mathsf{fix}(A)$ is a group. The result follows. □

Note that $u\|_{\emptyset} = \{\pi u \mid \text{all } \pi\}$.

Theorem 2.18. *Suppose A is a finite set of atoms. Then:*

1. *If $supp(u) \subseteq A$ then $supp(u\|_A) = supp(u)$.*
2. *$supp(u\|_A) \subseteq A$ always.*

As a corollary,

$$supp(u) \setminus A \neq \emptyset \quad \textit{implies} \quad supp(u\|_A) = A.$$

Proof. 1. If $supp(u) \subseteq A$ then by the definition of support, if $\pi \in \mathsf{fix}(A)$ then $\pi u = u$ and $u\|_A = \{u\}$. It is easy to verify that $supp(\{u\}) = supp(u)$.

2. Suppose that $\pi \in \mathsf{fix}(A)$. We reason using Theorem 2.5 and Lemma 2.17:

$$\pi(u\|_A) = (\pi u)\|_{\pi A} = (\pi u)\|_A = u\|_A.$$

By Theorem 2.13 the result follows.

We illustrate why the corollary is 'obvious' with an example:

$$a\|_{\{b\}} = \{a,\ c,\ d,\ e,\ f,\ \ldots\} = \mathbb{A} \setminus \{b\}.$$

Here $supp(a) \setminus \{b\} = \{a\}$ and $supp(\mathbb{A} \setminus \{b\}) = \{b\}$.

Now for the proof. Suppose that $supp(u) \setminus A \neq \emptyset$. By part 2 of this result $supp(u\|_A) \subseteq A$. We prove the reverse inclusion. We note the following two points:

- Suppose $a \in A$. If $a \in supp(u) \cap A$ then by Theorem 2.14 $a \in supp(u')$ for all $u' \in u\|_A$. If $a \in A \setminus supp(u)$ then by Theorem 2.14 $a \notin supp(u')$ for all $u' \in u\|_A$.
- Suppose $a \notin A$. Since $supp(u) \setminus A \neq \emptyset$ by Theorem 2.14 there exist some $u' \in u\|_A$ such that $a \in supp(u')$, and some $u' \in u\|_A$ such that $a \notin supp(u')$.

Thus, we can recover A from $X = u\|_A$ using the function

$$F(X) = \{a \mid (\forall x \in X. a \in supp(x)) \vee (\forall x \in X. a \notin supp(x))\}.$$

By Theorem 2.15 it follows that $supp(A) \subseteq supp(u\|_A)$. But A is finite; by Lemma 2.22 $A = supp(A)$ and so we have $A \subseteq supp(u\|_A)$ as required.[2] □

2.4. α-abstraction

The model of α-equivalence in FM sets was introduced in [23, 15]. The presentation here is revised.

Definition 2.19 is the standard Kuratowski implementation of ordered pairs in sets [30].

Definition 2.19. Write (x, y) for $\{\{x\}, \{x, y\}\}$.

Definition 2.20. Define the **atoms-abstraction** by $[c]z = (c, z)\|_{supp(z)\setminus\{c\}}$.

[2] This argument is due to an anonymous referee.

We can read '$[c]z$' as the binding action of '$\lambda c.z$' or '$\forall c.z$', and the sets above correspond with α-equivalence classes of FM sets. There is no *a priori* notion of λ-abstraction or universal quantification in; this is just α-abstraction, on FM sets. So $[c]z$ is the α-equivalence class of (c, z) where c is abstracted, i.e. where we read (c, z) like '$\lambda c.z$' or '$\forall c.z$'. The avoids atoms in $supp(z) \setminus \{c\}$, thus, $supp(z)$ plays the rôle that 'the free variable symbols of' plays in syntax (but generalises it to all sets):

$$\begin{array}{rl}
[a]a & = \{(a,a),\ (b,b),\ (c,c),\ (d,d),\ (e,e),\ (f,f),\ \ldots\} \\
[a]\{a,b\} & = \{(a,\{a,b\}),\ (c,\{c,b\}),\ (d,\{d,b\}),\ (e,\{e,b\}),\ \ldots\} \\
[a](\mathbb{A}\backslash\{a\}) & = \{(a,\mathbb{A}\backslash\{a\}),\ (b,\mathbb{A}\backslash\{b\}),\ (c,\mathbb{A}\backslash\{c\}),\ \ldots\} \\
[a](\mathbb{A}\backslash\{a,b\}) & = \{(a,\mathbb{A}\backslash\{a,b\}),\ (c,\mathbb{A}\backslash\{c,b\}),\ (d,\mathbb{A}\backslash\{d,b\}),\ \ldots\}
\end{array}$$

Write U_{ab} for $\{a\} \cup \{\{a\},\{c\},\{d\},\{e\},\ldots\}$ for any a, b. Then:

$$\begin{array}{rl}
[a]U_{ab} & = \{(a,U_{ab}),(c,U_{cb}),(d,U_{db})(e,U_{eb}),\ldots\} \\
[c]U_{ab} & = \{(c,U_{ab}),(d,U_{ab}),(e,U_{ab}),\ldots\}
\end{array}$$

Definition 2.20 gives a result equal to the original definition of atoms-abstraction in FM set theory [23]:

Lemma 2.21. $[c]z = \{(n,(n\ c)z) \mid n \neq c \wedge n\#z\} \cup \{(c,z)\}$.
As a corollary, if $(x,y) \in [c]z$ *then* $supp(y) = (supp(z) \setminus \{c\}) \cup \{x\}$.

Proof. We note that $(c,z) \in [c]z$ and also that $(n\ c) \in \mathsf{fix}(supp(z) \setminus \{c\})$. By Theorem 2.5 we can calculate that

$$\{(n,(n\ c)z) \mid n \neq c \wedge n\#z\} \cup \{(c,z)\} \subseteq [c]z.$$

Conversely, by Theorem 2.13 if $\pi \in \mathsf{fix}(supp(z) \setminus \{c\})$ then $\pi x = (\pi c\ c)x$ and it follows by an easy calculation that

$$[c]z \subseteq \{(n,(n\ c)z) \mid n \neq c \wedge n\#z\} \cup \{(c,z)\}.$$

For the corollary, by Theorem 2.14 $supp((n\ c)z) = (n\ c)supp(z)$ and the result follows. □

Lemma 2.22.

- *If* $\bigcup\{supp(x) \mid x \in X\}$ *is finite then*

  $$supp(X) = \bigcup\{supp(x) \mid x \in X\}.$$

  In particular if X *is finite (so that* $\bigcup\{supp(x) \mid x \in X\}$ *is a finite union of finite sets and so is finite) then* $supp(X) = \bigcup\{supp(x) \mid x \in X\}$.

- $supp(\{x\}) = supp(x)$ *and if* $A \subseteq \mathbb{A}$ *is finite then* $supp(A) = A$.

Proof. $supp(X) \subseteq \bigcup\{supp(x) \mid x \in X\}$ always, by Theorem 2.15. We now prove the reverse inclusion. Suppose that $\bigcup\{supp(x) \mid x \in X\}$ is finite. Now suppose $a \in supp(x)$ for some $x \in X$. Choose some b such that $b\#X$ and $b\#x'$ for every $x' \in X$. By Theorem 2.14 $supp((b\ a)x) = (b\ a)supp(x)$. Since X has no element y such that $b \in supp(y)$, we know that $(b\ a)X \neq X$ and by Theorem 2.13 it must be that $a \in supp(X)$.

The second part follows easily. □

Lemma 2.23. $supp((x,y)) = supp(x) \cup supp(y)$.

Proof. From Definition 2.19 and Lemma 2.22. □

Corollary 2.24. $supp([c]z) = supp(z) \setminus \{c\}$.

Proof. By Lemma 2.23 $supp((c,z)) = supp(z) \cup \{c\}$. By definition $[c]z = (c,z)\|_{supp(z)\setminus\{c\}}$. The result follows by Theorem 2.18. □

So for example, we expect $(a\ d)[a]\{a,b\} = [a]\{a,b\}$ since $supp(\{a,b\}) = \{a,b\}$, and $\{a,b\}\setminus\{a\} = \{b\}$, and $(a\ d) \in \mathsf{fix}(\{b\})$. We can verify this by routine calculations:

$$\begin{array}{rl} [a]\{a,b\} & = \{(a,\{a,b\}), (c,\{c,b\}), (d,\{d,b\}), (e,\{e,b\}),\ \ldots\} \\ (a\ d)[a]\{a,b\} & = \{(d,\{d,b\}), (c,\{c,b\}), (a,\{a,b\}), (e,\{e,b\}),\ \ldots\} \end{array}$$

Lemma 2.25. $supp(X) \subseteq \bigcup\{supp(x) \mid x \in X\}$ *need not necessarily hold if* X *is not finite.*

Proof. It suffices to give a counterexample; we give two:
$supp(\mathbb{A}) = \emptyset$ but $\bigcup\{supp(a) \mid a \in \mathbb{A}\} = \mathbb{A}$.
$supp(\mathbb{A}\setminus\{c\}) = \{c\}$ but $\bigcup\{supp(a) \mid a \in \mathbb{A} \wedge a \neq c\} = \mathbb{A}\setminus\{c\}$. □

3. The substitution action

We recall our notational convention that a, b, c range over distinct atoms, A, B, C, S, T range over sets of atoms, lowercase x, y, z, u, v range over all elements, and uppercase X, Y, Z, U, V range over elements that are not atoms.

3.1. Axioms, pointwise substitution action, syntactic substitution action

Definition 3.1. A **substitution action** on FM set theory is a function $z[a{\mapsto}x]$ expressed in the language of FM set theory (and so satisfying equivariance Theorem 2.5) taking any z, an atom a, and any x, satisfying:

$$\begin{array}{rl} (\alpha) & b\#z \Rightarrow z[a{\mapsto}x] = ((b\ a)z)[b{\mapsto}x] \\ (\#{\mapsto}) & a\#z \Rightarrow z[a{\mapsto}x] = z \\ (\mathbf{var}{\mapsto}) & a[a{\mapsto}x] = x \\ (\mathbf{id}{\mapsto}) & z[a{\mapsto}a] = z \\ (\mathbf{abs}{\mapsto}) & c\#x \Rightarrow ([c]z)[a{\mapsto}x] = [c](z[a{\mapsto}x]) \end{array}$$

Here by our convention x, y, z range over elements of the Fraenkel-Mostowski sets universe. a, b, and c range over distinct atoms. An equivariance rule from [22] is omitted here because it is guaranteed by Theorem 2.5.

Remark 3.2. FM set theory supports generalisations of the notions of 'name', 'freshness', and 'abstraction' which exist in syntax [23]. The author commented on the potential to extend this apparatus to a notion of substitution [15].

The axioms from Definition 3.1 came later in work with Mathijssen [22]. They are sound and complete for a model based on capture-avoiding substitution on syntax.

The axioms are only properties we require of the construction — our two substitution actions demonstrate that they specify, but not uniquely, substitution on sets. These axioms do rule out an 'obvious' definition, perhaps the first that we would consider:

Definition 3.3. The **pointwise** substitution action is defined by

$$a[a{\mapsto}x]_n = x \qquad b[a{\mapsto}x]_n = b \qquad Z[a{\mapsto}x]_n = \{z[a{\mapsto}x]_n \mid z \in Z\}.$$

Lemma 3.4. *Pointwise substitution does not satisfy* (α)*,* $(\#{\mapsto})$*, or* $(\mathbf{abs}{\mapsto})$*. Therefore, it is not a substitution action in the sense of Definition 3.1.*

Proof. It suffices to produce counterexamples. We do this for $(\#{\mapsto})$ and $(\mathbf{abs}{\mapsto})$. We expect that $\mathbb{A}[a{\mapsto}1]_n = \mathbb{A}$ since $a\#\mathbb{A}$. We also expect that $([c]a)[a{\mapsto}1]_n = [c](a[a{\mapsto}1])$ since $c\#1$. But:

$$\begin{aligned}
\mathbb{A}[a{\mapsto}1]_n =& (\mathbb{A} \setminus \{a\}) \cup \{1\} \\
([c]a)[a{\mapsto}1]_n =& \{(b,a),(c,a),(d,a),(e,a),\ldots\}[a{\mapsto}1]_n \\
=& \{(b,1),(c,1),(d,1),(e,1),\ldots\} \\
[c](a[a{\mapsto}1]_n) =& [c]1 \\
=& \{(a,1),(b,1),(c,1),(d,1),(e,1),\ldots\}.
\end{aligned}$$

□

Here is another attempt:

Definition 3.5. The **syntactic** substitution action is defined by:

$$\begin{gathered}
a[a{\mapsto}x]_t = x \qquad b[a{\mapsto}x]_t = b \qquad Z[a{\mapsto}x]_t = Z \quad (a\#Z) \\
Z[a{\mapsto}a]_t = Z \qquad Z[a{\mapsto}b]_t = (b\ a)Z \quad (b\#Z) \\
([c]z)[a{\mapsto}x]_t = [c](z[a{\mapsto}x]_t) \quad (c\#x) \\
Z[a{\mapsto}x]_t = \{z[a{\mapsto}x]_t \mid z \in Z\} \quad (Z \text{ finite}) \qquad Z[a{\mapsto}x]_t = \emptyset \quad (\text{otherwise}).
\end{gathered}$$

Theorem 3.6. $[a{\mapsto}x]_t$ *is well-defined and satisfies the axioms of substitution.*

The proof of this theorem is by easy but uninteresting calculations. Since we are in any case not happy with $[a{\mapsto}x]_t$ we do not include them.

$[a{\mapsto}x]_t$ reprises a similar definition on abstract syntax trees from the author's thesis [15, Definition 10.7.13], and a construction in the same spirit (but using diagrams) in presheaf categories [14, Definition of substitution by structural recursion, page 6]. $[a{\mapsto}x]_t$ is capture-avoiding substitution — on the subclass of (any model of) FM set theory that represents abstract syntax trees. On the rest of the FM sets universe the definition 'gives up' and defaults to $\emptyset$.

This could be suitable for providing a semantics to a generic substitution function, and might have applications, for example to generic programming [31, 9]. This is for future work.

The syntactic substitution action runs into another kind of trouble because of how it 'gives up' on sets that do not represent abstract syntax trees. For example

$$(\mathbb{A} \setminus \{a\})[a{\mapsto}1]_t = \emptyset.$$

If we expand definitions this becomes

$$\{b, c, d, e, f, \ldots\}[a{\mapsto}1]_t = \{\}.$$

Here we write $\mathbb{A} = \{a, b, c, d, e, f, \ldots\}$. Now

$$([c]a)[a{\mapsto}1]_t = [c]1.$$

If we expand definitions this becomes

$$\{(b,a),(c,a),(d,a),(e,a),(f,a),\ldots\}[a{\mapsto}1]_t = \{(a,1),(b,1),(c,1),(d,1),(e,1),(f,1),\ldots\}.$$

Thus $(\mathbb{A} \setminus \{a\})[a{\mapsto}1]_t$ and $([c]a)[a{\mapsto}1]_t$ return very different answers — yet both are equivalence classes of renamed variants in the sense of Definition 2.16. (In the examples above, the equivalence classes are renamed variants of c and (c, a) respectively.) For another example:

$$\{((b,b),a),((c,c),a),((d,d),a),((e,e),a),((f,f),a),\ldots\}[a{\mapsto}1]_t = \emptyset.$$

Compared to $[c]a$ the only difference is to place an ordered pair such as (b, b) in the first position, instead of just b. The effect on the result of a substitution is dramatic.

Clearly, this is an artefact of the definitions and has no deeper mathematical justification. The structures of the two sets

$$\begin{aligned}&\{((b,b),a),((c,c),a),((d,d),a),((e,e),a),((f,f),a),\ldots\}\\ [c]a =&\{(b,a),(c,a),(d,a),(e,a),(f,a),\ldots\}\end{aligned}$$

are sufficiently close that we could implement atoms-abstraction just as well using one as using the other. We expect them to behave in roughly analogous ways with respect to substitution, and they do not.

This issue is important:

- Equivalence classes of renamed variants are the mechanism by which FM sets model binding (Definition 2.20).
- They are also part of the mechanism by which $a\#x$ differs from 'a is in the transitive closure of x'. For example

  $$a \in \mathbb{A} = a\|_{\emptyset} \quad \text{yet} \quad a\#\mathbb{A}$$

  and

  $$a \notin (\mathbb{A} \setminus \{a\}) = b\|_{\{a\}} \quad \text{but} \quad a \in \mathit{supp}(\mathbb{A} \setminus \{a\}).$$

  (See Theorem 2.18 for the necessary results about $\mathit{supp}(u\|_A)$.)

Therefore, correct interaction of substitution with equivalence classes of renamed variables is directly relevant to motivation of this paper, which is to define a capture-avoiding substitution action on models of FM sets.

3.2. A short quiz

As we shall see, there is design freedom in building a substitution action. We motivate this with a short quiz. Some technical definitions will be useful:

Definition 3.7. If $A, S \subseteq \mathbb{A}$ are finite then define

$$A(a{\mapsto}S) = \begin{cases} (A \setminus \{a\}) \cup S & \text{if } a \in A \\ A & \text{if } a \notin A. \end{cases}$$

Lemma 3.8. *Any substitution action satisfies that*

$$supp(z[a{\mapsto}x]) \subseteq supp(z)(a{\mapsto}supp(x)).$$

Proof. By Theorem 2.15

$$supp(z[a{\mapsto}x]) \subseteq supp(z) \cup \{a\} \cup supp(x).$$

Choose some fresh b (so $b\#z, a, x$). By the axiom (α) $z[a{\mapsto}x] = ((b\ a)z)[b{\mapsto}x]$. By Theorem 2.15

$$supp(((b\ a)z)[b{\mapsto}x]) \subseteq supp((b\ a)z) \cup \{b\} \cup supp(x).$$

The result follows using Theorem 2.14. □

Remark 3.9. **[Quiz]** Recall from Definition 2.3 that we write $0 = \emptyset$ and $i{+}1 = i \cup \{i\}$, and write $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$. Note that $supp(i) = \emptyset$ for all $i \in \mathbb{N}$.

- **Q.** What is $((\mathbb{A} \setminus \{a\}) \cup \mathbb{N})[a{\mapsto}1]$?

  A. $\mathbb{A} \cup \mathbb{N}$.

  This makes the minimum change consistent with Lemma 3.8.[3]

- **Q.** What is $((\mathbb{A} \setminus \{a\}) \cup \mathbb{N})[a{\mapsto}1]$?

  A. $\mathbb{A} \cup (\mathbb{N} \setminus \{1\})$.

  This makes the minimum change consistent with Lemma 3.8 — and we replace the 'missing a' by a 'missing 1'. Since we are defining substitution on sets, we should think how substitution behaves on syntax denoting sets: substitution commutes with logical negation in the syntax of logic, and this is modelled by replacing 'missing elements' by other 'missing elements'. More on this in Subsection 5.2.

In this paper we will not attempt to argue whether one substitution action is better than another. We lay down two substitution actions representing two 'reasonable extremes' in the design space.

- The first one, presented immediately, belongs to the first class of substitutions in Remark 3.9 above.

[3] $supp(\mathbb{A} \setminus \{a\}) = \{a\}$ and $supp(\mathbb{A}) = \emptyset$.

- The second one, presented later beginning with Section 5, belongs to the second class of substitutions.

It is for future research to study the class of all possible substitution actions. In this paper we show some of what is possible and give methods by which these possibilities can be explored.

The reader might be tempted to expect some mathematical 'lattice of substitutions' similar perhaps to the 'lattice of λ-theories' which Salibra and others have studied [33]. This is just one of the interesting questions invited by the existence of a sets model of substitution. A moment of reflection shows that the answer is not obvious. Substitution is a function, not a relation — a λ-theory is a relation on λ-term syntax, so λ-theories are naturally ordered by inclusion.

3.3. The planes of a set

Definition 3.10. Suppose that $A \subseteq \mathbb{A}$ is finite. Call (u, A) a **plane** in Z when A is a minimal subset of $supp(Z)$ such that $u\|_A \subseteq Z$.

Write $\mathsf{plane}(Z)$ for the collection of planes in Z.

It is easy to see that if $(u, A) \in \mathsf{plane}(Z)$ then $u \in Z$.

Unpacking Definition 3.10, (u, A) is a plane in Z when $A \subseteq supp(Z)$ and $u\|_A \subseteq Z$ and for all (u', A'), if $A' \subseteq supp(Z)$ then

$$u\|_A \subseteq u'\|_{A'} \subseteq Z \quad \text{implies} \quad u'\|_{A'} = u\|_A.$$

For example:

1. $(a, \{a\}) \in \mathsf{plane}(\{a\})$ and $a\|_{\{a\}} = \{a\} \subseteq \{a\}$.
2. $(a, \{\}) \notin \mathsf{plane}(\{a\})$ because $a\|_{\{\}} = \mathbb{A} \not\subseteq \{a\}$.
3. $(a, \{a\}) \notin \mathsf{plane}(\mathbb{A})$ because $\{a\} \not\subseteq supp(\mathbb{A}) = \emptyset$.
4. $(a, \{a, b\}) \notin \mathsf{plane}(\{a\})$ because $a\|_{\{a,b\}} = \{a\} = a\|_{\{a\}}$ and $\{a\} \not\subseteq \{a, b\}$.
5. $(c, \{a\}) \in \mathsf{plane}(\mathbb{A}\backslash\{a\})$ and $c\|_{\{a\}} = \mathbb{A}\backslash\{a\} \subseteq \mathbb{A}\backslash\{a\}$.
6. $(a, \{\}) \in \mathsf{plane}(\mathbb{A})$ and $a\|_{\{\}} = \mathbb{A} \subseteq \mathbb{A}$.
7. $((c, a), \{a\}) \in \mathsf{plane}([c]a)$ and $(c, a)\|_{\{a\}} = \{(x, a) \mid x \neq a\} = [c]a \subseteq [c]a$.
8. $\mathsf{plane}(\{a\} \cup \{\{a\}, \{c\}, \{d\}, \ldots\}) = \{(a, \{a\})\} \cup \{(\{x\}, \{b\}) \mid x \in \mathbb{A},\ x \neq b\}$.
   $a\|_{\{a\}} = \{a\} \subseteq \{a\} \cup \{\{a\}, \{c\}, \{d\}, \ldots\}$.
   $\{x\}\|_{\{b\}} = \{\{a\}, \{c\}, \{d\}, \ldots\} \subseteq \{a\} \cup \{\{a\}, \{c\}, \{d\}, \ldots\}$.

Definition 3.11. If $S \subseteq \mathbb{A}$ is finite then define

$$\mathsf{plane}_S(Z) = \{(u, A) \in \mathsf{plane}(Z) \mid supp(u) \cap S \subseteq supp(u) \cap A\}.$$

We should think of $\mathsf{plane}_S(Z)$ as the planes (u, A) in Z such that the support of u 'avoids clashes' with atoms in S, where this is possible. For example

$$(a, \{\}) \in \mathsf{plane}_{\{\}}(\mathbb{A}) \quad \text{but} \quad (a, \{\}) \notin \mathsf{plane}_{\{a\}}(\mathbb{A}) \quad \text{and}$$
$$(c, \{a\}) \in \mathsf{plane}_{\{b\}}(\mathbb{A}\backslash\{a\}) \quad \text{but} \quad (b, \{a\}) \notin \mathsf{plane}_{\{b\}}(\mathbb{A}\backslash\{a\}).$$

It will later be useful to know that the planes of Z 'cover' Z in a suitable sense:

Theorem 3.12. *If $S \subseteq \mathbb{A}$ is finite then*

$$\bigcup\{u\|_A \mid (u,A) \in \mathsf{plane}_S(Z)\} = Z.$$

As a corollary taking $S = \emptyset$, $\bigcup\{u\|_A \mid (u,A) \in \mathsf{plane}(Z)\} = Z$.

Proof. We prove two set inclusions: The left-to-right inclusion is by construction. For the right-to-left inclusion, choose any $u \in Z$. Let $B = \{b_1, \ldots, b_k\}$ be equal to $supp(u) \setminus A$ and let $B' = \{b'_1, \ldots, b'_k\}$ be some set of entirely fresh atoms (so disjoint from $supp(u)$, A, S, and $supp(Z)$). Let $\pi = (b_1\ b'_1) \circ \ldots \circ (b_k\ b'_k)$.

By Theorem 2.14 we can calculate that

$$\begin{aligned} supp(\pi u) \cap S &= (supp(u) \cap A) \cap S \quad \text{and} \\ supp(\pi u) \cap A &= supp(u) \cap A. \end{aligned}$$

Therefore $supp(\pi u) \cap S \subseteq supp(\pi u) \cap A$. Also $(\pi u)\|_A = u\|_A$ by Lemma 2.17, so $(\pi u, A) \in \mathsf{plane}_S(Z)$. Finally we note that $u \in (\pi u)\|_A$. □

Planes do not just cover a set (Theorem 3.12), they cover its support, all of its support, and nothing but its support. Compare Lemma 2.25 with Theorem 3.13:

Theorem 3.13. *If $S \subseteq \mathbb{A}$ is finite then*

$$supp(Z) = \bigcup\{A \mid (u,A) \in \mathsf{plane}_S(Z)\}.$$

Proof. By definition if $(u,A) \in \mathsf{plane}_S(Z)$ then $A \subseteq supp(Z)$. It therefore suffices to prove that if for all $(u,A) \in \mathsf{plane}_S(Z)$ it is the case that $b \notin A$, then $b\#Z$.

Suppose $b \notin A$ for every $(u,A) \in \mathsf{plane}_S(Z)$. Choose any fresh $b'\#Z$, so that $b' \notin A$ for every $(u,A) \in \mathsf{plane}_S(Z)$. We reason as follows:

$$\begin{array}{lcll} (b'\ b)Z & = & (b'\ b)\bigcup\{(u\|_A) \mid (u,A) \in \mathsf{plane}_S(Z)\} & \text{Theorem 3.12} \\ & = & \bigcup\{(b'\ b)(u\|_A) \mid (u,A) \in \mathsf{plane}_S(Z)\} & \text{Definition 2.4} \\ & = & \bigcup\{u\|_A \mid (u,A) \in \mathsf{plane}(Z)\} & \text{Theorems 2.13 and 2.18} \\ & = & Z & \text{Theorem 3.12} \end{array}$$

Now $b \notin supp((b'\ b)Z)$ by Theorem 2.14 and the fact that $b'\#Z$. The result follows. □

3.4. Construction of the first substitution action

We can now define the substitution action. We use Theorem 3.12 to view an FM set Z as a union of planes; the 'capture-avoiding' aspect of substitution is easy to manage on a 'plane-by-plane basis'. Recall the definition of $A(a{\mapsto}S)$ from Definition 3.7.

Definition 3.14. Define the **substitution action** $z[a{\mapsto}x]$ and a 'helper' function $\delta(z,a,x)$ as follows:

- $a[a{\mapsto}x] = x$.
- $b[a{\mapsto}x] = b$.

- If $Z \notin \mathbb{A}$ then

$$Z[a{\mapsto}x] = \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u,A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)\}$$
$$\delta(u,a,x) = (supp(u)(a{\mapsto}supp(x))) \setminus supp(u[a{\mapsto}x]).$$

We consider some examples.

1. $\{a\}[a{\mapsto}x]$. There is one plane, $(a,\{a\})$.
$$\delta(a,a,x) = \{a\}(a{\mapsto}supp(x)) \setminus supp(x) = \emptyset.$$
$$\{a\}(a{\mapsto}supp(x)) \setminus \delta(a,a,x) = supp(x) \setminus \emptyset = supp(x)$$
$$\{a\}[a{\mapsto}x] = a[a{\mapsto}x]\|_{supp(x)} = x\|_{supp(x)} = x.$$

2. $(\mathbb{A}\setminus\{a\})[a{\mapsto}x]$. One plane is $(b,\{a\})$ where $b\#x$ (the others give the same result).
$$\delta(b,a,x) = \{b\}(a{\mapsto}supp(x)) \setminus \{b\} = \emptyset$$
$$\{a\}(a{\mapsto}supp(x)) \setminus \emptyset = supp(x)$$
$$(\mathbb{A}\setminus\{a\})[a{\mapsto}x] = b\|_{supp(x)} = \mathbb{A}\setminus supp(x)$$

3. $\mathbb{A}[a{\mapsto}x]$. One relevant plane is $(b,\emptyset)$ where $b\#x$ (the others give the same result).
$$\delta(b,a,x) = \emptyset \qquad \emptyset(a{\mapsto}supp(x)) \setminus \emptyset = \emptyset$$
$$\mathbb{A}[a{\mapsto}x] = b\|_{\emptyset} = \mathbb{A}$$

4. $([c]a)[a{\mapsto}x] = \{(b,a),(c,a),(d,a),\ldots\}[a{\mapsto}x]$.
One plane is $((c,a),\{a\})$ where $c\#x$.[4]
We omit calculations showing that $(c,a)[a{\mapsto}x] = (c,x)$; for a general result see Theorem 3.26 after these examples.
$$\begin{aligned}\delta((c,a),a,x) &= \{c,a\}(a{\mapsto}supp(x)) \setminus supp((c,x))\\ &= (supp(x)\cup\{c\}) \setminus (supp(x)\cup\{c\}) = \emptyset\end{aligned}$$
$$\{a\}(a{\mapsto}supp(x)) \setminus \emptyset = supp(x)$$
$$([c]a)[a{\mapsto}x] = (c,x)\|_{supp(x)} = [c]x$$
The other planes give the same result.

5. $U_b[a{\mapsto}\{b\}]$ where, for a given b, we write $U_b = \{a\}\cup\{\{a\},\{c\},\{d\},\ldots\}$.
Two planes are $a\|_{\{a\}}$ (a plane for $\{a\}$) and $\{a\}\|_{\{b\}}$ (a plane for $\{\{a\},\{c\},\{d\},\ldots\}$).
By calculations similar to the examples above, we calculate that
$$a[a{\mapsto}\{b\}] = \{b\} \quad \text{and}$$
$$\{\{a\},\{c\},\{d\},\ldots\}[a{\mapsto}\{b\}] = \{\{a\},\{c\},\{d\},\ldots\}$$
and that $U_b[a{\mapsto}\{b\}] = U$ where we write
$$U = \{\{a\},\{b\},\{c\},\{d\},\ldots\}.$$
The other planes give the same results.

[4] If $c \in supp(x)$ then $((c,a),\{a\}) \notin \mathsf{plane}_{supp(x)\cup\{a\}}([c]a)$.

6. $([c]U_b)[a{\mapsto}\{b\}] = \{(c,U_b),(d,U_b),\ldots\}[a{\mapsto}\{b\}]$.
   One plane is $((c,U_b),\{a,b\})$ (the other planes give the same result). Note that $\mathit{supp}(U) = \emptyset$ and $\mathit{supp}((c,U)) = \{c\}$, so that

$$\delta((c,U_b),a,\{b\}) = \{a,b,c\}(a{\mapsto}\{b\}) \setminus \{c\} = \{b\}$$
$$\{a,b\}(a{\mapsto}\{b\}) \setminus \{b\} = \emptyset$$
$$([c]U_b)[a{\mapsto}\{b\}] = (c,U)\|_\emptyset = [c]U.$$

   In all other examples δ is equal to $\emptyset$. Here, we see how δ is not equal to $\emptyset$. This corrects for the fact that $\mathit{supp}(U_b[a{\mapsto}\{b\}]) \neq \mathit{supp}(U_b)(a{\mapsto}\{b\})$.

Remark 3.15. Suppose that $A, S \subseteq \mathbb{A}$ are finite. Note that $A(a{\mapsto}S)$ and $A[a{\mapsto}S]$ do not coincide. For example, $\{a\}(a{\mapsto}\{a\}) = \{a\}$ whereas $\{a\}[a{\mapsto}\{a\}] = \{\{a\}\}$.

Now we show how this construction satisfies the axioms (α), $(\#{\mapsto})$, $(\mathbf{var}{\mapsto})$, $(\mathbf{id}{\mapsto})$, and $(\mathbf{abs}{\mapsto})$, and we investigate for what F it satisfies $(\mathbf{F}{\mapsto})$.

Theorem 3.16 $((\mathbf{id}{\mapsto}))$**.** $z[a{\mapsto}a] = z$.

Proof. We work by ϵ-induction.
The base cases of $z \in \mathbb{A}$ are easy:

- $a[a{\mapsto}a] = a$.
- $b[a{\mapsto}a] = b$.

Now suppose $Z \notin \mathbb{A}$ (we adhere to our notational convention and write capital 'Z'). Suppose the inductive hypothesis of all $u \in Z$. By definition

$$Z[a{\mapsto}a] = \bigcup\{(u[a{\mapsto}a])\|_{A(a{\mapsto}\{a\})\setminus\delta(u,a,a)} \mid (u,A) \in \mathsf{plane}_{\{a\}}(Z)\}$$

Suppose $(u,A) \in \mathsf{plane}_{\{a\}}(Z)$, so by definition

$$\mathit{supp}(u) \cap \{a\} \subseteq \mathit{supp}(u) \cap A.$$

By inductive hypothesis $u[a{\mapsto}a] = u$, so that $\mathit{supp}(u[a{\mapsto}a]) = \mathit{supp}(u)$. Then we can simplify:

$$A(a{\mapsto}\{a\}) \setminus (\mathit{supp}(u)(a{\mapsto}\{a\}) \setminus \mathit{supp}(u[a{\mapsto}a])) \\ = A \setminus (\mathit{supp}(u) \setminus \mathit{supp}(u)) = A.$$

So we have this:

$$Z[a{\mapsto}a] = \bigcup\{u\|_A \mid (u,A) \in \mathsf{plane}_{\{a\}}(Z)\}$$

The result follows by Theorem 3.12. □

For Theorem 3.21 we need Lemma 3.19, a technical lemma giving a form of 'capture-avoidance' result. The proof is detailed, but not very hard.

Definition 3.17. If $X, Y \notin \mathbb{A}$ then define the **symmetric difference** (the **XOR** or **exclusive or**) of X and Y by:

$$X \,\Delta\, Y = \{z \mid (z \in X \wedge z \notin Y) \vee (z \notin X \wedge z \in Y)\}$$

In words, $X \,\Delta\, Y$ means 'the set of elements in precisely one of X and Y'.

Lemma 3.18. *Fix any u, a finite set of atoms A, and any x. Suppose that π is a permutation, write*

$$C = \{x \in \mathbb{A} \mid \pi(x) \neq x\}.$$

Suppose that $C \cap (supp(x) \cup A \cup \{a\}) = \emptyset$ (so C is fresh for everything except for u). Then

$$u[a{\mapsto}x]\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} = (\pi u)[a{\mapsto}x]\|_{A(a\mapsto supp(x))\setminus\delta(\pi u,a,x)}.$$

Proof. By Theorem 2.13 $\pi x = x$. By Theorem 2.5 $\delta(\pi u, a, x) = \pi\delta(u, a, x)$ and so

$$\delta(u, a, x) \,\Delta\, \delta(\pi u, a, x) \subseteq C.$$

We assumed that $C \cap (A \cup supp(x)) = \emptyset$ and by elementary set calculations we deduce that

$$A(a{\mapsto}supp(x)) \setminus \delta(\pi u, a, x) = A(a{\mapsto}supp(x)) \setminus \delta(u, a, x).$$

Also $(\pi u)[a{\mapsto}x] = \pi(u[a{\mapsto}x])$ by Theorem 2.5. Therefore

$$\begin{aligned}(\pi u)[a{\mapsto}x]\|_{A(a\mapsto supp(x))\setminus\delta(\pi u,a,x)} &= (\pi(u[a{\mapsto}x]))\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} \\ &\overset{Theorem\ 2.18}{=} (u[a{\mapsto}x])\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)}\end{aligned}$$

as required. □

Lemma 3.19. *Fix $Z \notin \mathbb{A}$, $a \in \mathbb{A}$, and x. Suppose that $B = \{b_1, \ldots, b_n\}$ is a finite set of fresh atoms (so $b_i \# x, Z$ for $1 \leq i \leq n$). Then*

$$Z[a{\mapsto}x] = \bigcup\{(u[a{\mapsto}x])\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}\cup B}(Z)\}.$$

(Notice the B on the far right subscript.)

Proof. By definition

$$Z[a{\mapsto}x] = \bigcup\{(u[a{\mapsto}x])\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)\}.$$

Choose any

$$(u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z).$$

Unpacking definitions this means

$$(u, A) \in \mathsf{plane}(Z) \quad \text{and} \quad supp(u) \cap (supp(x) \cup \{a\}) \subseteq supp(u) \cap A.$$

Now choose some fresh $b'_1, \ldots, b'_n$ (so $b'_i \# u, A, B, x, Z$ for $1 \leq i \leq n$) and write $B' = \{b'_1, \ldots, b'_n\}$. Write

$$\pi = (b_1\ b'_1) \circ \ldots \circ (b_n\ b'_n).$$

We show that

1. $(\pi u, A) \in \mathsf{plane}(Z)$,
2. $supp(\pi u) \cap (supp(x) \cup \{a\} \cup B) \subseteq supp(\pi u) \cap A$ — so that

$$(\pi u, A) \in \mathsf{plane}_{supp(x) \cup \{a\} \cup B}(Z)$$

— and
3. $(\pi u)[a \mapsto x]\|_{A(a \mapsto supp(x)) \setminus \delta(\pi u, a, x)} = u[a \mapsto x]\|_{A(a \mapsto supp(x)) \setminus \delta(u, a, x)}$.

This will suffice to prove the result.

1. By Theorem 2.5 $(\pi u, \pi A) \in \mathsf{plane}(\pi Z)$. Now $\pi Z = Z$ by Theorem 2.13. Also by construction $A \subseteq supp(Z)$ and $B \cap supp(Z) = \emptyset = B' \cap supp(Z)$, so

$$B \cap A = \emptyset, \qquad B' \cap A = \emptyset, \qquad \text{and therefore} \quad \pi A = A.$$

   We conclude that $(\pi u, A) \in \mathsf{plane}(Z)$ as required.
2. $B \cap supp(x) = \emptyset = B' \cap supp(x)$ so $\pi x = x$ by Theorem 2.13. By Theorem 2.5

$$supp(\pi u) \cap (supp(\pi x) \cup \{a\}) \subseteq supp(\pi u) \cap \pi A$$

   and we simplify this to

$$supp(\pi u) \cap (supp(x) \cup \{a\}) \subseteq supp(\pi u) \cap A.$$

   But by Theorem 2.14 we know that $supp(\pi u) \cap B = \emptyset$ so by further elementary set calculations we conclude that

$$supp(\pi u) \cap (supp(x) \cup \{a\} \cup B) \subseteq supp(\pi u) \cap A$$

   as required.
3. We use Lemma 3.18. We need only check that

$$(B \cup B') \cap (supp(x) \cup A \cup \{a\}) = \emptyset$$

   and this is by construction. □

We use the following technical lemma in Theorem 3.21:

Lemma 3.20. *Fix some Z, a, and x. Suppose that*

for all b, if $b \# Z, x$ then $Z[a \mapsto x] \subseteq ((b\ a)Z)[b \mapsto x]$.

Then also

for all b, if $b \# Z$ then $Z[a \mapsto x] \subseteq ((b\ a)Z)[b \mapsto x]$.

Proof. Choose some fresh c (so $c \# Z, a, b, c$). By assumption

$$Z[a \mapsto x] = ((c\ a)Z)[c \mapsto x] \qquad ((b\ a)Z)[b \mapsto x] = ((c\ b)(b\ a)Z)[c \mapsto x].$$

The result follows by Theorem 2.13. □

Theorem 3.21 ((α))**.** *Suppose $Z \notin \mathbb{A}$. If $b\#Z$ then $Z[a{\mapsto}x] \subseteq ((b\ a)Z)[b{\mapsto}x]$.*
As a corollary, for any z if $b\#z$ then $z[a{\mapsto}x] = ((b\ a)z)[b{\mapsto}x]$.

Proof. We first prove the corollary. Suppose that $z \in \mathbb{A}$. Then there are three cases:

- $a[a{\mapsto}x] = x$ and $((b\ a)a)[b{\mapsto}x] = x$ and $x = x$.
- $b\#b$ is false, so there is nothing to prove for the case of $b[a{\mapsto}x]$.
- $c[a{\mapsto}x] = c$ and $((b\ a)c)[b{\mapsto}x] = c$ and $c = c$.

Now suppose that $Z \notin \mathbb{A}$ (we adhere to our convention and write capital 'Z'). Assume that $b\#Z$ implies $Z[a{\mapsto}x] \subseteq ((b\ a)Z)[b{\mapsto}x]$ for all b, Z, a, x.

Now suppose $b\#Z$. Then $Z[a{\mapsto}x] \subseteq ((b\ a)Z)[b{\mapsto}x]$. Also by Theorem 2.14 $a\#(b\ a)Z$ and it follows that $((b\ a)Z)[b{\mapsto}x] \subseteq ((b\ a)(b\ a)Z)[a{\mapsto}x]$. Now $(b\ a)(b\ a)Z = Z$, and the result follows.

For the first part, we work by ϵ-induction. Suppose $Z \notin \mathbb{A}$ and suppose $b\#Z$. By Lemma 3.20 we can also assume $b\#x$. Suppose the inductive hypothesis of every $u \in Z$. Using the definition of substitution and using Lemma 3.19 for the first equality (to add a $\{b\}$ to the subscript on plane; we cannot add $\{a\}$ to the subscript on plane in the second equality because we do not know $a\#x$)

$$Z[a{\mapsto}x] = \bigcup\{(u[a{\mapsto}x])\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} \mid (u,A) \in \mathsf{plane}_{supp(x)\cup\{a,b\}}(Z)\}$$
$$((b\ a)Z)[b{\mapsto}x] = \bigcup\{(u'[b{\mapsto}x])\|_{A'(b\mapsto supp(x))\setminus\delta(u',b,x)} \mid (u',A') \in \mathsf{plane}_{supp(x)\cup\{b\}}((b\ a)Z)\}.$$

Suppose $(u,A) \in \mathsf{plane}_{supp(x)\cup\{a,b\}}(Z)$. To prove our set inclusion we just need to exhibit some $(u',A') \in \mathsf{plane}_{supp(x)\cup\{b\}}((b\ a)Z)$ such that

$$u[a{\mapsto}x]\|_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} = u'[b{\mapsto}x]\|_{A'(b\mapsto supp(x))\setminus\delta(u',b,x)}.$$

We choose $u' = (b\ a)u$ and $A' = (b\ a)A$. Then $(u',A') \in \mathsf{plane}((b\ a)Z)$ by Theorem 2.5. Also by definition of $\mathsf{plane}_{supp(x)\cup\{a,b\}}(Z)$ we know that

$$supp(u) \cap (supp(x) \cup \{a,b\}) \subseteq supp(u) \cap A.$$

Now $b \notin A$ by construction and $b \in supp(x) \cup \{a,b\}$. Therefore $b\#u$. It is now not hard to use Theorem 2.14 and some elementary set calculations to calculate that

$$supp(u') \cap (supp(x) \cup \{b\}) \subseteq supp(u') \cap A'.$$

So $(u',A') \in \mathsf{plane}_{supp(x)\cup\{b\}}(Z)$. Since $b\#u$ by the inductive hypothesis $u'[b{\mapsto}x] = u[a{\mapsto}x]$.

It remains to show

$$A(a{\mapsto}supp(x)) \setminus \delta(u,a,x) = ((b\ a)A)(b{\mapsto}supp(x)) \setminus \delta((b\ a)u,b,x).$$

Recall that $b \notin A$. Then $A(a{\mapsto}supp(x)) = ((b\ a)A)(b{\mapsto}supp(x))$ is easily verified. Also

$$\delta((b\ a)u, b, x) = supp((b\ a)u)(b{\mapsto}supp(x)) \setminus supp(((b\ a)u)[b{\mapsto}x]).$$

Now $supp((b\ a)u)(b{\mapsto}supp(x)) = supp(u)(a{\mapsto}supp(x))$ is easily verified, and it follows by the inductive hypothesis that $((b\ a)u)[b{\mapsto}x]$, and we are done. □

Theorem 3.22 (($\#{\mapsto}$)). *If $a\#z$ then $z[a{\mapsto}x] = z$.*

Proof. We work by ϵ-induction. We start with the base cases:

- $a\#a$ is false so there is nothing to prove for $a[a{\mapsto}x]$.
- $b[a{\mapsto}x] = b$.

Now suppose $Z \notin \mathbb{A}$ (we adhere to our notational convention and write capital 'Z') and $a\#Z$ and suppose the inductive hypothesis of all $u \in Z$. By definition

$$Z[a{\mapsto}x] = \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)\}.$$

Consider any $(u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)$. By assumption

$$supp(u) \cap (supp(x) \cup \{a\}) \subseteq supp(u) \cap A.$$

By construction $a \notin A$, so $a\#u$ and by inductive hypothesis $u[a{\mapsto}x] = u$. Now $A(a{\mapsto}supp(x)) = A$, and

$$\delta(u, a, x) = supp(u)(a{\mapsto}supp(x)) \setminus supp(u[a{\mapsto}x]) = supp(u) \setminus supp(u) = \emptyset.$$

So

$$Z[a{\mapsto}x] = \bigcup\{u\|_A \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)\}.$$

The result follows by Theorem 3.12. □

Recall the notion of atoms-abstraction $[c]z$ from Definition 2.20.

Lemma 3.23. $\mathsf{plane}([c]z) = \{((c', (c'\ c)z), supp(z) \setminus \{c\}) \mid c'\#z\} \cup \{((c, z), supp(z) \setminus \{c\})\}$

Proof. By construction $(c, supp(z) \setminus \{c\}) \in \mathsf{plane}([c]z)$. Also

$$(c', (c'\ c)z)\|_{supp(z)\setminus\{c\}} = (c, z)\|_{supp(z)\setminus\{c\}} = [c]z$$

by Lemma 2.17. Therefore $((c', (c'\ c)z), supp(z) \setminus \{c\}) \in \mathsf{plane}([c]z)$ for each $c'\#z$.

Now take any $(c', z') \in [c]z$ (by our permutative convention, $c' \neq c$). By construction $(c', z') = (c', (c'\ c)z)$ for some $c'\#z$. The result follows. □

Theorem 3.24 (($\mathbf{abs}{\mapsto}$)). *If $c\#x$ then $([c]z)[a{\mapsto}x] = [c](z[a{\mapsto}x])$.*

Proof. If $a\#z$ then by Theorem 2.15 also $a\#[c]z$ and

$$([c]z)[a{\mapsto}x] = [c]z \qquad \text{and} \qquad [c](z[a{\mapsto}x]) = [c]z$$

follow by Theorem 3.22. So suppose $a \in supp(z)$. By Lemma 3.23

$$\mathsf{plane}([c]z) = \{((c', (c'\ c)z), supp(z) \setminus \{c\}) \mid c'\#z\} \cup \{((c, z), supp(z) \setminus \{c\})\}.$$

Using Lemma 3.18 we can calculate that if $c'\#x$ then

$$(c, z[a{\mapsto}x])\|_{A(a\mapsto supp(x))\setminus\delta((c,z),a,x)} = (c', ((c'\ c)z)[a{\mapsto}x])\|_{A(a\mapsto supp(x))\setminus\delta((c',(c'\ c)z),a,x)}.$$

We can also calculate using Theorems 2.13 and 2.5 that if $c'\#z[a{\mapsto}x]$ then

$$supp(z[a{\mapsto}x]) \setminus \{c\} = (c'\ c)(supp(z[a{\mapsto}x]) \setminus \{c\}) = supp((c'\ c)(z[a{\mapsto}x])) \setminus \{c'\}.$$

By Lemma 2.17 we also have

$$(c, z[a{\mapsto}x])\|_{supp(z[a\mapsto x])\setminus\{c\}} = (c', ((c'\ c)z)[a{\mapsto}x])\|_{supp((c'\ c)(z[a\mapsto x]))\setminus\{c'\}}.$$

By Theorems 2.5 and 2.13 $(c'\ c)(z[a{\mapsto}x]) = ((c'\ c)z)[a{\mapsto}x]$. The definitions now simplify to this:

$$\begin{aligned}
[c]z &= (c, z)\|_{supp(z)\setminus\{c\}} \\
([c]z)[a{\mapsto}x] &= (c, z[a{\mapsto}x])\|_{(supp(z)\setminus\{c\})(a\mapsto supp(x))\setminus\delta((c,z),a,x)} \\
[c](z[a{\mapsto}x]) &= (c, z[a{\mapsto}x])\|_{supp(z[a\mapsto x])\setminus\{c\}}
\end{aligned}$$

So it suffices to verify that

$$(supp(z) \setminus \{c\})(a{\mapsto}supp(x)) \setminus \delta((c, z), a, x) = supp(z[a{\mapsto}x]) \setminus \{c\}.$$

Now $\delta((c, z), a, x) = (supp(z) \cup \{c\})(a{\mapsto}supp(x)) \setminus (supp(z[a{\mapsto}x]) \cup \{c\})$ (we use Lemma 2.23 to calculate the support of a pairset) and the result follows by set calculations. □

Theorem 3.25. *Definition 3.14 is equivariant and satisfies* (α), $(\#{\mapsto})$, $(\mathbf{var}{\mapsto})$, $(\mathbf{id}{\mapsto})$, *and* $(\mathbf{abs}{\mapsto})$ *from Subsection 3.1.*

Proof. Equivariance is automatic by Theorem 2.5. $(\mathbf{var}{\mapsto})$ is direct from the definition. Each of (α), $(\#{\mapsto})$, $(\mathbf{id}{\mapsto})$, and $(\mathbf{abs}{\mapsto})$ is by one of the theorems proved above. □

3.5. Substitutions on syntax, substitutions commuting

We prove that on sets representing syntax substitution coincides with capture-avoiding substitution as we might inductively define it. This is Theorem 3.29 and Corollary 3.31.

Theorem 3.26. *If* $Z \notin \mathbb{A}$ *and* Z *is a* finite *set then* $Z[a{\mapsto}x] = \{z[a{\mapsto}x] \mid z \in Z\}$.

Proof. By definition,

$$Z[a{\mapsto}x] = \bigcup\{(u[a{\mapsto}x])\rVert_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u,A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)\}$$

Suppose $(u,A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(Z)$. Since $u\rVert_A \subseteq Z$ and Z is finite, $u\rVert_A$ is finite. It follows by Lemma 2.22 that $supp(u) \subseteq A$ and $u\rVert_A = \{u\}$.

Now recall that by Lemma 3.8 $supp(u[a{\mapsto}x]) \subseteq supp(u)(a{\mapsto}supp(x))$, and by definition

$$\delta(u,a,x) = (supp(u)(a{\mapsto}supp(x))) \setminus supp(u[a{\mapsto}x]).$$

It follows by set calculations that

$$supp(u[a{\mapsto}x]) \subseteq A(a{\mapsto}supp(x)) \setminus \delta(u,a,x)$$

and the result follows. □

Corollary 3.27. *The functions* $(x,y) = \{\{x\},\{x,y\}\}$, $\mathsf{inl}(x) = (x,\emptyset)$, *and* $\mathsf{inr}(x) = (x,\{\emptyset\})$, *all commute with the substitution action.*

Let $\mathbb{I}$ *be any set of elements with empty support, so if* $i \in \mathbb{I}$ *then* $supp(i) = \emptyset$ *(an example is* $\mathbb{N}$ *from Definition 2.3, a non-example is the set of atoms* $\mathbb{A}$*). Let* z_i *be an* $\mathbb{I}$*-indexed collection of elements and write* $(z_i)_{i\in\mathbb{I}} = \{(i,z_i) \mid i \in \mathbb{I}\}$. *Then*

$$(z_i)_{i\in\mathbb{I}}[a{\mapsto}x] = (z_i[a{\mapsto}x])_{i\in\mathbb{I}}.$$

Proof. The results for pairing, inl, and inr follow from Theorem 3.26.

The second part follows noting that the planes of $(z_i)_{i\in\mathbb{I}}$ are precisely $(z_i, supp(z_i))$ for $i \in \mathbb{I}$, since for all $i,j \in \mathbb{I}$ if $\pi i = j$ then $i = j$. □

Definition 3.28. Let Λ be the datatype of abstract syntax defined inductively by

$$\frac{a \in \mathbb{A}}{a \in \Lambda} \qquad \frac{x,y \in \Lambda}{(x,y) \in \Lambda} \qquad \frac{a \in \mathbb{A} \quad x \in \Lambda}{[a]x \in \Lambda}$$

It is not hard to show that Λ is isomorphic to λ-terms up to α-equivalence [23]; this is the FM standard construction of abstract-syntax-with-binding, slightly modified (see Remark 3.30 below). Atoms have zero elements, pairs have two, and abstractions have infinitely many, so no atom is a pair or an abstraction, and no pair is an abstraction.

Theorem 3.29. *If* $z,x \in \Lambda$ *then* $z[a{\mapsto}x] \in \Lambda$ *and* $z[a{\mapsto}x]$ *is equal to what we would normally call 'capture-avoiding substitution of* x *for* a *in* z*'.*

Proof. We work by induction on Λ.

- $a[a{\mapsto}x] = x$ and $b[a{\mapsto}x] = b$.
- $(z_1,z_2)[a{\mapsto}x] = (z_1[a{\mapsto}x], z_2[a{\mapsto}x])$ by Corollary 3.27.
- $([c]z)[a{\mapsto}x] = [c](z[a{\mapsto}x])$ providing $c\#x$ by Theorem 3.24. It is not hard to use Lemma 2.23 and Corollary 2.24 to prove that $c\#x$ corresponds precisely to 'c is not free in x' when $x \in \Lambda$. □

Remark 3.30. Theorem 3.29 is generic and works for any datatype of syntax. We must interpret atoms as themselves and not 'wrapped up' in the following sense: our construction is an isomorphic version of the datatype from [23] given by:

$$\frac{a \in \mathbb{A}}{\mathsf{inl}(\mathsf{inl}(a)) \in \Lambda} \qquad \frac{x, y \in \Lambda}{\mathsf{inl}(\mathsf{inr}((x, y))) \in \Lambda} \qquad \frac{a \in \mathbb{A} \quad x \in \Lambda}{\mathsf{inr}([a]x) \in \Lambda}$$

This is not suitable for Theorem 3.29 because atoms are wrapped up in $\mathsf{inl}(\mathsf{inl}(a))$ and $\mathsf{inl}(\mathsf{inl}(a))[a{\mapsto}x] = \mathsf{inl}(\mathsf{inl}(x)) \neq x$. No substitution action can anticipate our particular implementation of the tree-structure of datatypes; since atoms are a distinct class of elements in a model of FM set theory it is valid to insert them 'unwrapped' into the inductive type as we have done.

At first blush this appears to rule out datatypes with two or more different kinds of name (type variables, term variables, channel names, locations, and so on), since the datatypes concerned 'contain more than one copy of $\mathbb{A}$'. A good solution is to hypothesise many sets of atoms — for example a set of disjoint countably infinite sets of atoms, and to add to axiom (**Fresh**) a quantification over all the sets of atoms (that is, to insist on finite support with respect to each set). This makes no difference to the results in this paper aside from slightly complicating their presentation; our constructions require a countably infinite set of atoms; it makes no difference whether *other* sets of atoms are also present in the sets universe.

We can prove inductively that the usual commutativity property of substitution holds. Again, the proof is generic and would work as well for any FM-style datatype of abstract syntax with binding:

Corollary 3.31. *If* $a\#y$ *and* $x, y, z \in \Lambda$ *then* $z[a{\mapsto}x][b{\mapsto}y] = z[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]]$.

Proof. By induction on z. □

Theorem 3.29 and Corollary 3.31 seem empirically to be quite robust results. Analogous results hold also for other structures including coinductive datatypes such as the datatype of infinite lists $\mathbb{L}$ illustrated below, and also for 'name-generating' coinductive datatypes in the style of [16] such as $\mathbb{L}_{ng}$ illustrated below:[5]

$$\frac{x \in X \quad l \in \mathbb{L}}{x :: l \in \mathbb{L}} \qquad \frac{x \in X \quad l \in \mathbb{L}_{ng}}{x :: [a]l \in \mathbb{L}_{ng}}$$

(We can also specify this as $\mathbb{L} = X \times \mathbb{L}$ and $\mathbb{L}_{ng} = X \times [\mathbb{A}]\mathbb{L}_{ng}$.)

[5] $\mathbb{L}_{ng}$ is based on work in [16]. To see in what sense an element $\mathbb{L}_{ng}$ can represent an infinite name-generating — and infinitely name-generating — coinductive structure, consider the case where X is the set of finite lists of atoms. We will write $\langle\rangle$ for the empty list and $\langle x_1, \ldots, x_n\rangle$ for the n-element list with x_i in the ith position for $1 \leq i \leq n$, and we consider the element which we can then write as:

$$\langle\rangle :: [a]\langle a\rangle :: [b]\langle a, b\rangle :: [c]\langle a, b, c\rangle :: [d]\langle a, b, c, d\rangle :: \ldots$$

We can read this as an accumulator, with a state which is a list of atoms, and which at each step of its behaviour generates a fresh atom and pushes it onto the tail of its state.

Substitution commutes with the operations from which these sets are routinely built — atoms, tuples (of any size; see the second part of Corollary 3.27), and taking equivalence classes of renamed variants.

Remark 3.32. For less structured sets, substitutions need not commute:

$$\begin{aligned}\{a,\{a\},\{c\},\{d\},\ldots\}[a{\mapsto}\{b\}][b{\mapsto}\{c\}]&\\ =\{\{a\},\{b\},\{c\},\{d\},\ldots\}[b{\mapsto}\{c\}]&\\ =\{\{a\},\{b\},\{c\},\{d\},\ldots\}&\end{aligned}$$

$$\begin{aligned}\{a,\{a\},\{c\},\{d\},\ldots\}[b{\mapsto}\{c\}][a{\mapsto}\{\{c\}\}]&\\ =\{a,\{a\},\{b\},\{d\},\ldots\}[a{\mapsto}\{\{c\}\}]&\\ =\{\{a\},\{b\},\{\{c\}\},\{d\},\ldots\}&\end{aligned}$$

(The planes of interest here are $a\|_{\{a\}}$, $\{a\}\|_{\{b\}}$, and $\{a\}\|_{\{c\}}$.)

Intuitively, $\{a,\{a\},\{c\},\{d\},\ldots\}$ can be read as the predicate 'is the variable a, or is $\{x\}$ where x is a variable other than b'. Standard logics such as first-order logics cannot express predicates that reflect on their own syntax; intuitively, non-commutativity of substitution holds on sets that 'reflect on atoms' so that the results depend on the order in which those atoms are substituted. We can also approach this proof-theoretically [18].

An analogous phenomenon appears in work based on categories, treating names and behaviour in name-passing calculi that require a name-for-name substitution [13]. To model the syntax of the name-passing calculus and to model reasoning on it, a category in which inequality between names can be determined is required; to model behaviour, a category in which names can be substituted for other names is required. These two requirements seem incompatible since deciding inequality between names does not commute with name-for-name substitution. In [13] a nice solution is discussed which goes back to [28]: two categories, related by adjoint functors, are used for this purpose; $Set^{\mathcal{I}}$ for a category in which inequality between names can be determined, and $Set^{\mathcal{F}}$ for a category in which names can be substituted for other names; see [13, Equation 28].[6] Since set theory is untyped, the sets for which substitution commutes, and those for which it does not, all populate the same universe.

4. New properties of FM: a-orbits and crucial elements

To define our next substitution action we have to develop the theory of crucial elements. Given a set X we will define a set $C_a X$, the a-crucial elements of X, which

[6] $\mathcal{F}$ is the category of finite sets and maps between them. $\mathcal{I}$ is the category of finite sets and injections between them.

It may be worth noting that the name-for-name substitution action modelled in $Set^{\mathcal{F}}$ is not quite a special case of any substitution action based on the work in this paper, even allowing for the different presentations (categories versus sets). This is because in $Set^{\mathcal{F}}$ (and also in $Set^{\mathcal{I}}$), all presheaves are considered and not only presheaves preserving pullbacks of monos. The 'preserves pullbacks of monos' corresponds with Theorem 2.13 and ensures that 'every element has a unique least supporting set'. It is this property which makes possible the 'sets-based' presentations characteristic of nominal techniques. See [19] for a further discussion.

measures 'how far away X is from satisfying $a\#X$'.

For example

- $C_a\{a\}$ will be equal to $\{a\}$ and $\{a\}$ is 'one a away from satisfying $a\#\{a\}$' (because $a \in \{a\}$).
- $C_a(\mathbb{A} \setminus \{a\})$ will be equal to $\{a\}$ and $\mathbb{A} \setminus \{a\}$ is also 'one a away from satisfying $a\#\mathbb{A} \setminus \{a\}$' (because $a \notin \mathbb{A} \setminus \{a\}$).
- $C_a\mathbb{A}$ will be equal to $\emptyset$ and $\mathbb{A}$ already satisfies $a\#\mathbb{A}$ (even though $a \in \mathbb{A}$).

Remark 4.1. Recall the definition of symmetric difference from Definition 3.17:

$$X \mathbin{\Delta} Y = \{z \mid (z \in X \wedge z \notin Y) \vee (z \notin X \wedge z \in Y)\}$$

We shall show that $a\#(X \mathbin{\Delta} C_aX)$ always (Corollary 4.15). So if C_aX intuitively measures 'how far away X is from satisfying $a\#X$', then Δ is dimension along which to travel this 'distance'.

With reference to the examples above, $\{a\} \mathbin{\Delta} \{a\} = \emptyset$, and $(\mathbb{A} \setminus \{a\}) \mathbin{\Delta} \{a\} = \mathbb{A}$, and $\mathbb{A} \mathbin{\Delta} \emptyset = \mathbb{A}$, and in all cases a is fresh for the result.

An interesting analogy can be drawn between Δ and swapping $(a\ b)$:

Swapping is very much like atoms-renaming $[a{\mapsto}b]$. It is used in FM theory to do many of the things that are traditionally done using atoms-renaming [23]. An advantage of swapping is that it is *invertible*; FM techniques owe much of their convenience to this property.

Δ is a bit like like sets union $\bigcup$ and a bit like sets subtraction $\setminus$, and it can be used to do many of the same things. It is also an invertible operation in the sense that $X \mathbin{\Delta} Y \mathbin{\Delta} Y = X$. This invertibility will prove useful at important points in the proofs below, for example in Lemma 4.21.

We develop the theory of crucial elements in Subsection 4.2. First, we must discuss *a-orbits*:

4.1. a-orbits

If $a \in \mathbb{A}$ and any u, write

$$|u|_a \text{ for the set } u\|_{supp(u)\setminus\{a\}}.$$

Call $|u|_a$ the **a-orbit of** u.

For example:

$$|a|_a = \mathbb{A} \qquad |b|_a = \{b\}$$

$$|\{a,b\}|_a = \{\{a,b\},\{c,b\},\{d,b\},\ldots\} \qquad |\{a,b\}|_b = \{\{a,b\},\{a,c\},\{a,d\},\ldots\}.$$

To prove Theorem 4.5 below we need some easy technical lemmas:

Lemma 4.2. 1. *If $a\#u$ then $|u|_a = \{u\}$.*

2. *If $\neg a\#u$ then $|u|_a$ is an infinite set.*

Proof. 1. Suppose $a\#u$. Then $|u|_a = u\|_{supp(u)}$ and the result follows by the definition of $\|$ and by Theorem 2.13.
2. Suppose $\neg a\#u$. By Theorem 2.14 for $c, c'\#u$ it is the case that $supp((c\ a)u) \neq supp((c'\ a)u)$, so that $(c\ a)u \neq (c'\ a)u$. The result follows. □

Lemma 4.3. *If $\neg a\#u$ and $b\#u$ then $|u|_a = |(b\ a)u|_b$. As a corollary, $a\#(|u|_a)$.*

Proof. The first part is by Theorem 2.5 (a proof by calculations is also easy). For the second part, by Theorem 2.14 we can calculate that $a\#(b\ a)u$. Then the result follows by Theorem 2.15. □

Lemma 4.4. *If S is a finite set of atoms such that $\neg a\#S$ and $b\#S$ then $(b\ a)S = S[a{\mapsto}b]$.*

Proof. The permutation action is pointwise, so $(b\ a)S = \{(b\ a)x \mid x \in S\}$. The result follows by Lemma 2.22 and easy calculations. □

The main technical result on a-orbits is this:

Theorem 4.5. *Suppose $\neg(a\#u)$ and $\neg(a\#u')$. Then precisely one of the following three possibilities holds:*

1. *$|u|_a \cap |u'|_a$ is empty.*
2. *$|u|_a \cap |u'|_a$ is a singleton.*
   In this case all of the following hold:
   - *$supp(u) \setminus supp(u') = \{b\}$ for some $b\#u'$.*
   - *$supp(u') \setminus supp(u) = \{b'\}$ for some $b'\#u$.*
   - *$|u|_a \cap |u'|_a = \{(b\ a)u'\}$.*
   - *$|u|_a \cap |u'|_a = \{(b'\ a)u\}$.*
3. *$|u|_a = |u'|_a$ and $u = u'$.*

Proof. If $|u|_a \cap |u'|_a = \emptyset$ then we are done. So suppose that $z \in |u|_a \cap |u'|_a$.
By definition

$$|u|_a = \{(n\ a)u \mid n = a \vee n\#u\} \quad \text{and} \quad |u'|_a = \{(n'\ a)u' \mid n' = a \vee n'\#u'\}.$$

Here and for the rest of this proof, n and n' range over *all* atoms (and may be equal to a or each other).

Then $z = (n\ a)u$ and $z = (n'\ a)u'$ for some n and n', for which there are several possibilities:

- If $n = a$ and $n' = a$ then $u = u'$ and we fall into the third case.
- $n = a$ and $n' \neq a$ is impossible because $a \in supp(u)$ and by Theorem 2.14 $a \notin supp((n'\ a)u')$.
- Similarly if $n \neq a$ and $n' = a$.

- Suppose that $n \neq a$ and $n' \neq a$ and $n = n'$. Then $u = u'$ and we fall into the third case.
- Suppose that $n \neq a$ and $n' \neq a$ and $n \neq n'$. We have supposed that $(n\ a)u = (n'\ a)u'$, recall that $n\#u$ and $n'\#u'$. By Theorem 2.14

$$(n\ a)supp(u) = (n'\ a)supp(u').$$

Using Lemma 4.4 we can calculate that this can only happen if

$$supp(u) \setminus supp(u') = \{n'\} \quad \text{and} \quad supp(u') \setminus supp(u) = \{n\}.$$

We now verify by easy calculations that we fall into the second case. □

For example:

- $|a|_a \cap |(a,b)|_a = \emptyset$.
- $|(a,b)|_a \cap |(c,a)|_a = \{(c,b)\} = \{(b\ a)(c,a)\}$.
- $|(a,b)|_a \cap |(a,b)|_a = |(a,b)|_a$.

Corollary 4.6. *Suppose* $\neg(a\#u)$ *and* $\neg(a'\#u')$. *Then precisely one of the following holds:*

1. $|u|_a \cap |u'|_{a'}$ *is empty.*
2. $u = u'$ *and* $|u|_a \cap |u|_{a'} = \{u\}$.
3. $u \neq u'$ *and* $|u|_a \cap |u'|_{a'}$ *is a singleton.*
   In this case all of the following hold:
   - $supp(u) \setminus supp(u') = \{b, a\}$ *for some* $b\#u'$.
   - $supp(u') \setminus supp(u) = \{b, a'\}$ *for some* $b\#u$.
   - $|u|_a \cap |u'|_{a'} = \{(b\ a')u'\}$.
   - $|u|_a \cap |u'|_a = \{(b\ a)u\}$.
4. $|u|_a = |u'|_{a'}$, $a'\#u$, *and* $u = (a\ a')u'$.

Proof. Choose some fresh c (so $c\#a, a', u, u'$). By Theorem 2.14 $\neg(c\#(c\ a)u)$ and $\neg(c\#(c\ a')u')$. Also by Lemma 4.3 we know that

$$|u|_a = |(c\ a)u|_c \qquad \text{and} \qquad |u'|_{a'} = |(c\ a')u'|_c.$$

Using Theorem 4.5 we calculate that there are three possibilities:

1. $|u|_a \cap |u'|_{a'}$ is empty.
2. $|u|_a \cap |u'|_{a'}$ is a singleton $\{z\}$.
   In this case (here n and n' range over all atoms, and in particular may be equal to a, a', c, or each other):
   - $supp((c\ a)u) \setminus supp((c\ a')u') = \{n\}$ for some $n\#(c\ a')u'$.
   - $supp((c'\ a')u') \setminus supp((c\ a)u) = \{n'\}$ for some $n'\#(c\ a)u$.

- $|u|_a \cap |u'|_{a'} = \{(n\ c)(c\ a')u'\}$.
- $|u|_a \cap |u'|_{a'} = \{(n'\ c)(c\ a)u\}$.

Now we must consider some further possibilities:

- Suppose $n = a'$. Then we can use Lemma 4.4 to calculate it must be that $supp(u') = supp(u)$, and so that $n' = a$. Therefore $u = u'$ and $|u|_a \cap |u'|_{a'} = \{u\}$.
- The case $n' = a$ is similar.
- Suppose $n \neq a'$ and $n' \neq a$. Then:
  - Using Lemma 4.4 and our assumption that $a \in supp(u)$ we know that $supp(u) \setminus supp(u') = \{n, c\}$. (It easily follows that $u \neq u'$.)
  - Using Lemma 4.4 and our assumption that $a' \in supp(u')$ we know that $supp(u') \setminus supp(u) = \{n', c\}$.
  - By Theorem 2.13 and our assumption that $c\#u'$ we know that $|u|_a \cap |u'|_{a'} = \{(n\ a')u'\}$.
  - By Theorem 2.13 and our assumption that $c\#u$ we know that $|u|_a \cap |u'|_{a'} = \{(n'\ a)u\}$.

3. $|u|_a = |u'|_{a'}$ and $(c\ a)u = (c\ a')u'$. By Theorem 2.14 and our assumption that $c\#u, u'$ we deduce that $a'\#u$. By Theorem 2.13 and Theorem 2.5 we further deduce that $u = (c\ a)(c\ a')u' = (a\ a')u'$ as required. □

We need a new definition and a technical lemma before we prove Theorem 4.9:
We call $A \subseteq \mathbb{A}$ **cofinite** when $\mathbb{A} \setminus A$ is finite.

Lemma 4.7. *If $S \subseteq \mathbb{A}$ is cofinite then $supp(S) = \mathbb{A} \setminus S$.*

Proof. By Theorem 2.15 we know that $supp(\mathbb{A} \setminus S) \subseteq supp(S)$. By the same Theorem we know that $supp(S) \subseteq supp(\mathbb{A} \setminus S)$. The result follows by Lemma 2.22. □

The following result is a well-known property of FM set theory [23]:

Lemma 4.8. *$A \in \mathcal{P}(\mathbb{A})$ is either finite or cofinite, and not both.*

Proof. We know that $supp(A)$ exists and is finite by Theorem 2.13. So $\pi A = A$ for any $\pi \in \mathsf{fix}(supp(A))$. It follows by an easy calculation that A is either finite or cofinite. A cannot be both finite and cofinite, because $(\mathbf{AtmInf})$ insists that $\mathbb{A}$ be infinite. □

Theorem 4.9. *Suppose $\neg a\#u$ and $U \subseteq |u|_a$. $a\#U$ if and only if*

- *U is finite and $u \notin U$, or*
- *U is cofinite and $u \in U$.*

Proof. Let $S = \{n \mid (n\ a)u \in |u|_a\}$ (here n ranges over all atoms, and may equal a). Choose some fresh c (so $c\#S, u, U$). S is either finite or cofinite by Lemma 4.8. By Lemma 2.22 if S is finite then $supp(S) = S$. By Lemma 4.7 if S is cofinite then $supp(S) = \mathbb{A} \setminus S$.
Now we just consider the possibilities:

- Suppose S is finite and $a \notin S$. Then $u \notin U$.

  Also using Theorem 2.13 and the fact that $c\#u$ we have:

  $$\begin{aligned}(c\ a)U &= \{(c\ a)(n\ a)u \mid n \in S\}\\ &= \{(n\ c)(c\ a)u \mid n \in S\}\\ &= \{(n\ a)u \mid n \in S\}\\ &= U.\end{aligned}$$

  It follows that $a\#U$.

- Suppose S is finite and $a \in S$. Then $u \in U$.

  By similar reasoning we deduce that $(c\ a)U = (U \backslash \{u\}) \cup \{(c\ a)u\}$. Now we assumed that $\neg a\#u$ so by Theorem 2.14 we know that $u \neq (c\ a)u$. It follows that $\neg a\#U$.

- The cases for U cofinite are similar. □

4.2. Crucial elements

Δ is defined in Definition 3.17. The following properties are easy to prove —

- $X\ \Delta\ X = \emptyset$.
- $X\ \Delta\ \emptyset = X$.
- $X\ \Delta\ Y = Y\ \Delta\ X$.
- $(X\ \Delta\ Y)\ \Delta\ Z = X\ \Delta\ (Y\ \Delta\ Z)$.

— we use them without comment henceforth.

Lemma 4.10. $(b\ a)(X\ \Delta\ Y) = ((b\ a)X)\ \Delta\ ((b\ a)Y)$, *and if* $a\#X$ *and* $a\#Y$ *then* $a\#X\ \Delta\ Y$.

Proof. By Theorems 2.5 and 2.15. □

Definition 4.11. Call u a**-crucial for** X when

- $\neg a\#u$.
- $\neg(a\#(|u|_a \cap X))$.

Write $u\ \varepsilon_a\ X$ when u is a-crucial for X.

Remark 4.12. In nominal techniques it is known that an element can be as important by its absence from a set, as by its inclusion in that set. For example it is the 'missing a' in $\mathbb{A} \setminus \{a\}$ that is responsible for $supp(\mathbb{A} \setminus \{a\}) = \{a\}$.

Note that it is not necessarily the case that if $u\ \varepsilon_a\ X$ then $u \in X$. For example,

$$a\ \varepsilon_a\ \mathbb{A} \setminus \{a\}.$$

$u\ \varepsilon_a\ X$ is a measure of whether u is 'conspicuous' in X, at least as far as $a \in supp(X)$ is concerned — either by its absence or its presence. We make this measure by looking at whether u is in, or is not in, the intersection of its own a-orbit with X.

For example: the a-orbit of a is $\mathbb{A}$, and a is conspicuously absent from $\mathbb{A} \cap (\mathbb{A} \setminus \{a\}) = \mathbb{A} \setminus \{a\}$, so $a\ \varepsilon_a\ \mathbb{A} \setminus \{a\}$.

Write

$$C_a X \qquad \text{for} \qquad \{x \mid x\ \varepsilon_a\ X\}.$$

Theorem 4.13. *$a\#X$ if and only if for every u, if $\neg a\#u$ then $a\#(X \cap |u|_a)$.*

Proof. Suppose $a\#X$. Consider some u such that $\neg a\#u$ and write $U = X \cap |u|_a$. We use part 2 of Theorem 4.9:

- Suppose U is finite and $u \in U$. Choose some b such that $(b\ a)u \notin U$ and $b\#X$. We observe that $(b\ a)X \neq X$, which by Theorem 2.13 contradicts our assumption that $a\#X$ and $b\#X$. So $u \notin U$ and therefore $a\#U$.
- Suppose U is cofinite and $u \notin U$. We choose some b such that $(b\ a)u \in U$ and $b\#X$, and reason similarly.

Now suppose $a\#(X \cap |u|_a)$ for every u such that $\neg(a\#u)$. Choose any $b\#X$. We can calculate that $X = \bigcup\{X \cap |u|_a \mid \neg(a\#u)\}$ and so

$$\begin{aligned}(b\ a)X &= \bigcup\{(b\ a)(X \cap |u|_a) \mid \neg(a\#u)\} \\ &= \bigcup\{X \cap |u|_a \mid \neg(a\#u)\} \quad = X\end{aligned}$$

□

Recall from Remark 4.1 that we mentioned that $C_a X$ is a measure of 'how far X is away from satisfying $a\#X$'. Corollary 4.14 makes that formal:

Corollary 4.14. *$a\#X$ if and only if $C_a X = \emptyset$.*

Proof. Suppose $a\#X$. By Theorem 4.13 we know that $a\#(X \cap |u|_a)$ for every u such that $\neg(a\#u)$. It follows that $C_a X = \emptyset$.

Conversely suppose that $C_a X = \emptyset$. This means that if $\neg a\#u$ then $a\#(X \cap |u|_a)$. It follows by Theorem 4.13 that $a\#X$. □

As a matter of convenient notation, write

$$X - a \qquad \text{for} \qquad X\ \Delta\ C_a X.$$

Corollary 4.15. *$supp(X - a) \subseteq supp(X) \setminus \{a\}$.*

Proof. There are two parts to this result: we show that $a\#(X - a)$ always, and that if $b\#X$ then $b\#(X - a)$.

For the first part, by Theorem 4.13 it suffices to show that for every u such that $\neg(a\#u)$ it is the case that:

- If $(X - a) \cap |u|_a$ is finite then $u \notin (X - a) \cap |u|_a$.

- If $(X - a) \cap |u|_a$ is cofinite then $u \in (X - a) \cap |u|_a$.

Now $(X - a) \cap |u|_a$ is finite or cofinite if and only if $X \cap |u|_a$ is finite or cofinite respectively, since by construction $C_a X \cap |u|_a$ never has more than one element.

Also note that if $\neg(a\#u)$ then $u \in X - a$ if and only if $u \notin X$. This suffices to prove the first part.

For the second part, suppose that $b\#X$. Then $b\#C_a X$ by Theorem 2.15, and so $b\#(X - a)$ by the same theorem. □

Note that $supp(X - a)$ need not be equal to $supp(X) \setminus \{a\}$ in general. For example take $X = \{(a, b)\}$; then $C_a X = X$ and $X - a = \emptyset$.

4.3. Operations on sets of crucial elements

Lemma 4.16. $A \cap (X \,\Delta\, Y) = (A \cap X) \,\Delta\, (A \cap Y)$.

Proof. By easy calculations. □

Lemma 4.17. $C_a(X \,\Delta\, Y) = (C_a X) \,\Delta\, (C_a Y)$.

Proof. Pick any u such that $\neg a\#u$. So for any Z, $u \in C_a Z$ if and only if $\neg a\#(|u|_a \cap Z)$.

Write $A = |u|_a \cap X$ and $B = |u|_a \cap Y$. By Lemma 4.16 we can simplify:

- $u \in C_a(X \,\Delta\, Y)$ when $\neg a\#A \,\Delta\, B$.
- $u \in C_a X$ when $\neg a\#A$.
- $u \in C_a Y$ when $\neg a\#B$.

The result now follows reasoning by cases on whether A and B are finite, using part 2 of Theorem 4.9. □

Support is not preserved under taking unions in general. For example $supp(\mathbb{A}) = \emptyset$ and $supp(\{a\}) = supp(\mathbb{A} \setminus \{a\}) = \{a\}$, so it is not the case that $supp(X \cup Y)$ is related in any useful way to $supp(X) \cup supp(Y)$. Lemma 4.17 identifies Δ as a way of combining sets which is similar to sets union, but which interacts nicely with support.

Likewise, it is not in general the case that $a\#X$ if and only if $a\#x$ for all $x \in X$. Corollary 4.14 identifies $x \,\varepsilon_a\, X$, or equivalently $x \in C_a X$, as a notion of membership which is similar to sets membership, but which interacts nicely with support.

Lemma 4.18. *If $U \subseteq C_a Z$ then $C_a U = U$.*
In particular $C_a(C_a Z) = C_a Z$.

Proof. By the construction of C_a and by Theorem 4.5 we know that $U \cap |u|_a \subseteq \{u\}$. Since $u \in C_a Z$ we know that $\neg a\#u$.

Suppose $u \in U$. By Lemma 2.22 $\neg a\#\{u\}$ and so $u \in C_a U$.

Suppose $u \in C_a U$. Then $\neg a\#(U \cap |u|_a)$ and so $U \cap |u|_a = \emptyset$ is impossible and $U \cap |u|_a = \{u\}$. Therefore $u \in U$. □

The following lemma connects the permutation action with crucial elements:

Lemma 4.19. *If $b\#Z$ then $(b\ a)Z = (Z - a)\ \Delta\ (b\ a)C_aZ$.*

Proof. By elementary calculations $Z = (Z - a)\ \Delta\ C_aZ$ so by Lemma 4.10

$$(b\ a)Z = (b\ a)(Z - a)\ \Delta\ (b\ a)C_aZ.$$

By Corollary 4.15 we know $a\#(Z - a)$ and $b\#(Z - a)$. By Theorem 2.13 we know $(b\ a)(Z - a) = Z - a$. The result follows. □

It is useful to note a *non*-result.

Lemma 4.20.

- *It is not necessarily the case that $C_a(X \cap Y) = C_aX \cap C_aY$.*
- *Likewise it is not necessarily the case that $C_a(X \cup Y) = C_aX \cup C_aY$.*

Proof.

- $C_a(\mathbb{A} \cap \{a\}) = \{a\}$ but $C_a\mathbb{A} \cap C_a\{a\} = \emptyset \cap \{a\} = \emptyset$.
- $C_a((\mathbb{A} \setminus \{a\}) \cup \{a\}) = \emptyset$ but $C_a(\mathbb{A} \setminus \{a\}) = \{a\}$ and $C_a(\{a\}) = \{a\}$.

□

However, we do have the following useful 'pointwise' property in the case that sets are finite:

Lemma 4.21. *If X is a finite set then $C_aX = \bigcup\{x \in X \mid \neg a\#x\}$.*

Proof. Take u such that $\neg a\#u$. Then $u \in C_aX$ if and only if $\neg a\#(|u|_a \cap X)$. By Lemma 2.22 we know that $\neg a\#(|u|_a \cap X)$ if and only if $\neg a\#x$ for some $x \in X$ such that $\neg a\#x$. The result follows. □

Corollary 4.22. *If X is a finite set then C_aX is a finite set.*

Proof. By Lemma 4.21 $C_aX = \{x \in X \mid \neg a\#x\}$. So $C_aX \subseteq X$ and in particular C_aX is finite. □

The converse to Corollary 4.22 does not hold. For example $\mathbb{N}$ (the natural numbers implemented in set theory e.g. as $\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \ldots\}$) and $\mathbb{A}$ satisfy $C_a\mathbb{N} = C_a\mathbb{A} = \emptyset$, but they are not finite sets.

5. A second substitution action, using crucial elements

We now define a second substitution action. By shameless abuse of notation we shall use the same notation as we did for the first: $[a{\mapsto}x]$.

Where there is any danger of ambiguity we write the substitution action of Definition 3.14 as $[a{\mapsto}x]_1$ and the substitution action of Definition 5.1 below as $[a{\mapsto}x]_2$. All unannotated uses of the notation $[a{\mapsto}x]$ in this section, refer to Definition 5.1 below.

The second substitution action gives the *second* possible answer to the quiz in Remark 3.9. For example

$$\{a\}[a{\mapsto}1] = \{1\} \quad \text{and} \quad a\#(\mathbb{A} \cup \mathbb{N}),$$

and

$$((\mathbb{A} \setminus \{a\}) \cup \mathbb{N})[a{\mapsto}1]_2 = \mathbb{A} \cup (\mathbb{N} \setminus \{1\}).$$

Here we see that a 'missing a' is replaced by a 'missing 1'. This behaviour is symmetric between 'missing' and 'not missing';

$$(\mathbb{A} \setminus \{a\})[a{\mapsto}1]_2 = \mathbb{A} \cup \{1\}.$$

Here a 'missing a' is replaced by a 'missing missing 1'!

Intuitively, imagine sets as collections of binary switches describing what is and what is not an element of the set; the second substitution action flips these switches between 'on' and 'off'. More on this in Subsection 6.1.

5.1. Definition and properties

We recall our notational conventions:

- x, y, z, u, v range over arbitrary elements.
- X, Y, Z, U, V range over elements that are not atoms (i.e. 'sets').
- $a, b, c, a', \ldots$ range *permutatively* over atoms (so $a \neq a'$ and $a \neq b$).
- A, B, C, S, T range over sets of atoms.

Definition 5.1. Suppose x is any element. Define the **substitution action** $[a{\mapsto}x]$ by:

- $a[a{\mapsto}x] = x$.
- $b[a{\mapsto}x] = b$.
- If $Z \notin \mathbb{A}$ then

$$\begin{aligned} Z[a{\mapsto}x] &= (Z - a) \mathbin{\Delta} \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid \\ &\qquad\qquad (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(C_a Z)\} \\ \delta(u, a, x) &= (supp(u)(a{\mapsto}supp(x))) \setminus supp(u[a{\mapsto}x]). \end{aligned}$$

Lemma 5.2. *Suppose that $B = \{b_1, \ldots, b_n\}$ is a finite set of atoms and suppose that $b_i \# x, Z$ for $1 \leq i \leq n$. Then*

$$\begin{aligned} Z[a{\mapsto}x] &= (Z - a) \mathbin{\Delta} \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(u))\setminus\delta(u,a,x)} \mid \\ &\qquad\qquad (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}\cup B}(C_a Z)\}. \end{aligned}$$

Proof. By the same method as the proof of Lemma 3.19. □

Theorem 5.3 (($\mathbf{id}{\mapsto}$))**.** $z[a{\mapsto}a] = z$.

Proof. We work by ϵ-induction.

The base cases of $z \in \mathbb{A}$ are easy:

- $a[a{\mapsto}a] = a$.
- $b[a{\mapsto}a] = c$.

Now suppose $Z \notin \mathbb{A}$ (we adhere to our notational convention and write capital 'Z'). Then using Lemma 5.2 we have:

$$Z[a{\mapsto}a] \;= (Z - a) \;\Delta\; \bigcup\{(u[a{\mapsto}a])\|_{A(a{\mapsto}a)\setminus\delta(u,a,a)} \mid (u, A) \in \mathsf{plane}_{\{a\}}(C_a Z)\}$$

By reasoning similar to that in Theorem 3.16 we can deduce that

$$\bigcup\{(u[a{\mapsto}a])\|_{A(a{\mapsto}a)\setminus\delta(u,a,a)} \mid (u, A) \in \mathsf{plane}_{\{a\}}(C_a Z)\} = C_a Z.$$

The result follows. □

Theorem 5.4 ((α)). *If $b\#z, a$ then $z[a{\mapsto}x] = ((b\ a)z)[b{\mapsto}x]$.*

Proof. We work by ϵ-induction.

The base cases of $z \in \mathbb{A}$ are easy:

- $a[a{\mapsto}x] = x = ((b\ a)a)[a{\mapsto}x]$ by definition.
- $b\#b$ is false so there is nothing to prove for the case $b[a{\mapsto}x]$.
- $c[a{\mapsto}x] = c = ((b\ a)c)[a{\mapsto}x]$ by definition.

Now suppose $Z \notin \mathbb{A}$ (we adhere to our notational convention and write capital 'Z'). We expand the definition of $((b\ a)Z)[b{\mapsto}x]$:

$$((b\ a)Z)[b{\mapsto}x] \;= ((b\ a)Z \;\Delta\; C_b(b\ a)Z) \;\Delta\; \bigcup\{(u'[b{\mapsto}x])\|_{A'(b{\mapsto}x)\setminus\delta(u',b,x)} \mid (u', A') \in \mathsf{plane}_{supp(x)\cup\{b\}}(C_b(b\ a)Z)\}$$

By Corollary 4.15 we know that

$$a\#((b\ a)Z \;\Delta\; C_b(b\ a)Z) \qquad \text{and} \qquad b\#((b\ a)Z \;\Delta\; C_b(b\ a)Z).$$

By Theorems 2.5 and 2.13 we deduce that

$$((b\ a)Z \;\Delta\; C_b(b\ a)Z) \;=\; Z - a.$$

So it would suffice if we show that

$$\bigcup\{(u'[b{\mapsto}x])\|_{A'(b{\mapsto}x)\setminus\delta(u',b,x)} \mid (u', A') \in \mathsf{plane}_{supp(x)\cup\{b\}}(C_b(b\ a)Z)\} = \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,b,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(C_a Z)\}$$

The proof of this is almost identical to the proof of Theorem 3.21. □

Theorem 5.5 ($(\#{\mapsto})$). *If $a\#z$ then $z[a{\mapsto}x] = z$.*

Proof. We work by ϵ-induction. We start with the base cases:

- $a\#a$ is false so there is nothing to prove for $a[a{\mapsto}x]$. We need not check that $a[a{\mapsto}x] = a$, because it is not the case that $a\#a$.
- $b[a{\mapsto}x] = b$ by definition.

Now suppose $Z \notin \mathbb{A}$ (we adhere to our notational convention and write capital 'Z'). The result now follows easily expanding definitions, noting that $C_a Z = \emptyset$ by Corollary 4.14. □

To prove Theorem 5.10 below we need a number of technical results:

Lemma 5.6. *If $\neg a\#z$ and $\neg c\#z$ then $C_a|z|_c = |z|_c$.*

Proof. Suppose that $u \in C_a|z|_c$. We want to show that $u \in |z|_c$. By construction $\neg a\#u$ and $\neg a\#(|u|_a \cap |z|_c)$. By Corollary 4.6 precisely one of the following holds:

1. $|u|_a \cap |z|_c = \emptyset$.
   It is a fact that $a\#\emptyset$ so we can discount this option.
2. $u = z$ and $|u|_a \cap |z|_c = \{u\}$.
   Then $u \in |z|_c$.
3. $u \neq z$ and $|u|_a \cap |z|_c$ is a singleton.
   Then $\mathit{supp}(z) \setminus \mathit{supp}(u) = \{b, a\}$ for some $b\#u$ and $|z|_c \cap |u|_a = \{(b\ a)u\} = \{(b\ c)z\}$. But this is not possible because $a\#(b\ a)u$ and also $\neg a\#(b\ c)z$ by Theorem 2.14.
4. $|u|_a = |z|_c$.
   This is impossible because by construction $|z|_c$ contains infinitely many v such that $\neg a\#v$, but $|u|_a$ contains only one; we use Theorem 2.14.

Now suppose that $u \in |z|_c$. So either $u = z$ or for some $b\#z$ it is the case that $u = (b\ c)z$. We want to show that $u \in C_a|z|_c$.

Suppose $u = z$. Since $\neg a\#z$ it suffices to also show that $\neg a\#|z|_a \cap |z|_c$. By Corollary 4.6 we know that $|z|_a \cap |z|_c = \{z\}$ and by Lemma 2.22 we know $\neg a\#\{z\}$ so we are done.

Suppose $u = (b\ c)z$ for some $b\#z$. By Theorem 2.14 we know $\neg a\#u$ so it suffices to show that $\neg a\#(|u|_a \cap |z|_c)$. We consider the various possibilities in Corollary 4.6 and see that it must be that $|u|_a \cap |z|_c = \{u\}$. The result follows again using Lemma 2.22. □

Lemma 5.7. *If $a\#z$ and $\neg c\#z$ then $C_a|z|_c = \emptyset$.*

Proof. By Theorem 2.15 and Corollary 4.14. □

Like $[a{\mapsto}t]_1$ from Definition 3.14, $[a{\mapsto}t]_2$ is pointwise on finite sets:

Lemma 5.8. *If Z is a finite set then*

$$Z[a{\mapsto}x] = \{z[a{\mapsto}x] \mid z \in Z\}.$$

Proof. Unpacking definitions,

$$Z[a{\mapsto}x] = (Z - a) \mathbin{\Delta} \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(C_a Z)\}.$$

By Lemma 4.21 $Z - a = \{u \mid u \in Z,\ a\#u\}$, and by Theorem 5.5

$$Z - a = \{u[a{\mapsto}x] \mid u \in Z,\ a\#u\}.$$

By reasoning similar to that in the proof of Theorem 3.26,

$$\bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(C_a Z)\} = \{u[a{\mapsto}x] \mid u \in Z,\ \neg a\#u\}.$$

The result follows. □

Corollary 5.9. $(z, z')[a{\mapsto}x] = (z[a{\mapsto}x], z'[a{\mapsto}x])$.

Proof. By two uses of Lemma 5.8. □

Theorem 5.10 ($(\mathbf{abs}{\mapsto})$)**.** *If* $c\#x$ *then* $([c]z)[a{\mapsto}x] = [c](z[a{\mapsto}x])$.

Proof. If $a\#z$ then by Theorem 2.15 also $a\#[c]z$ and

$$([c]z)[a{\mapsto}x] = [c]z \qquad \text{and} \qquad [c](z[a{\mapsto}x]) = [c]z$$

follow by Theorem 5.5. So suppose $a \in supp(z)$. We unpack definitions:

$$([c]z)[a{\mapsto}x] = ([c]z - a) \mathbin{\Delta} \bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(C_a([c]z))\}.$$

By Lemma 5.6 $C_a[c]z = [c]z$ and so $([c]z - a) = \emptyset$ and we simplify $([c]z)[a{\mapsto}x]$ to:

$$\bigcup\{(u[a{\mapsto}x])\|_{A(a{\mapsto}supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}([c]z)\}.$$

The result now follows by reasoning similar to that in the proof of Theorem 3.24. □

Theorem 5.11. *Definition 5.1 is equivariant and satisfies* (α), $(\#{\mapsto})$, $(\mathbf{var}{\mapsto})$, $(\mathbf{id}{\mapsto})$, *and* $(\mathbf{abs}{\mapsto})$ *from Subsection 3.1.*

Proof. Equivariance is automatic by Theorem 2.5. $(\mathbf{var}{\mapsto})$ is direct from the definition. Each of (α), $(\#{\mapsto})$, $(\mathbf{id}{\mapsto})$, and $(\mathbf{abs}{\mapsto})$ is by one of the theorems proved above. □

Theorem 5.12. *If* $z, x \in \Lambda$ *then* $z[a{\mapsto}x] \in \Lambda$ *and* $z[a{\mapsto}x]$ *is equal to what we would normally call 'capture-avoiding substitution of* x *for* a *in* z*'.*

Proof. By an easy induction similar to the proof of Theorem 3.29. □

Corollary 5.13. *If* $a\#y$ *and* $x, y, z \in \Lambda$ *then* $z[a{\mapsto}x][b{\mapsto}y] = z[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]]$.

Proof. By induction on z. □

Like Theorem 3.29 and Corollary 3.31, Theorem 5.12 and Corollary 5.13 are generic and work for any datatype of syntax implemented in FM set theory.

5.2. Scope-extrusion and hereditarily finite crucial sets

Scope-extrusion is an expression used to describe how binders can sometimes increase ('extrude') their scope over a connective — usually up to some notion of equivalence, usually subject to a freshness side-condition on free variables. Examples include behaviour displayed by π-calculus name-restriction and first-order logic universal quantification:

$$\begin{array}{llll} P \mid \nu a.Q & \text{is equivalent with} & \nu a.(P \mid Q) & \text{if } a \text{ not free in } P \\ \phi \Rightarrow \forall a.\psi & \text{is equivalent with} & \forall a.(\phi \Rightarrow \psi) & \text{if } a \text{ not free in } \phi \end{array}$$

$[a{\mapsto}x]_2$ satisfies a form of scope-extrusion with respect to Δ and subject to a nominal freshness side-conditions. With the results we have so far, the proof is easy:

Theorem 5.14. *Suppose $a \in \mathbb{A}$ and $Z \notin \mathbb{A}$ and $Z' \notin \mathbb{A}$.*
If $a\#Z$ then

$$(Z \mathbin{\Delta} Z')[a{\mapsto}x] = Z \mathbin{\Delta} (Z'[a{\mapsto}x]).$$

Proof. Recall that $C_a Z = \emptyset$ by Corollary 4.14. We unpack definitions using Lemma 4.17,

$$(Z \mathbin{\Delta} Z')[a{\mapsto}x] \;= Z \mathbin{\Delta} (Z' - a) \mathbin{\Delta} \bigcup\{(u[a{\mapsto}x])\rVert_{A(a\mapsto supp(x))\setminus\delta(u,a,x)} \mid (u, A) \in \mathsf{plane}_{supp(x)\cup\{a\}}(C_a Z')\},$$

and this is precisely equal to $Z \mathbin{\Delta} (Z'[a{\mapsto}x])$ as required. □

Remark 5.15. The condition that $Z \notin \mathbb{A}$ and $Z' \notin \mathbb{A}$ on Theorem 5.14 excludes the case that Δ is undefined (see Definition 3.17). There seems no plausible way to extend Δ to atoms so that Theorem 5.14 makes sense. The problem case is when $Z \notin \mathbb{A}$ and $Z' = a$. If we let $Z \mathbin{\Delta} a = Z$ (because a is empty; it has no set elements) then $(Z \mathbin{\Delta} a)[a{\mapsto}x] = Z$, which is not in general equal to $Z \mathbin{\Delta} x$. If we let $Z \mathbin{\Delta} a = a$ (just to see what happens) then $(Z \mathbin{\Delta} a)[a{\mapsto}x] = x$, which is again not in general equal to $Z \mathbin{\Delta} x$. Similarly if we let $Z \mathbin{\Delta} a = \emptyset$.

The question itself may be misguided: $Z \mathbin{\Delta} Z'$ is the symmetric difference of Z and Z' and so is all about the elements of Z and Z'. Atoms are not determined by their extension; they have no elements but are *not* equal to the empty set. Thus, Δ (an extensional creature) applied to atoms (intensional creatures) is a mild category ('typing') error. For this reason, we let Δ be undefined in Definition 3.17 and leave the side-conditions in Theorem 5.14.

Remark 5.16. Scope-extrusion is familiar from syntax. It is nice to see it reproduced for Δ. Scope-extrusion for Δ has a broader meaning. To understand why, we recall some of the research context of this work.

Recall our second answer in the quiz in Remark 3.9, which discussed substituting a 'missing a' for a 'missing 1'.

'Nominal techniques' were introduced by the author with Pitts in [23], emerging from Fraenkel-Mostowski sets. One of their distinctive features was the use of *permutations* — bijective renamings of names (modelled in FM sets by atoms) — to handle

α-equivalence. A benefit of permutation is that they move around 'missing atoms'. For example

$$(b\ a)(\mathbb{A} \setminus \{a\}) = \mathbb{A} \setminus \{b\}$$

— the 'missing a' has been replaced by a 'missing b'.[7]

From this we can take a lesson that a conspicuous *absence* of an element can be as important as its *presence*.

Corollary 5.17 makes formal a sense in which scope-extrusion (Theorem 5.14) gives the second substitution action the same kind of behaviour with respect to 'missing elements'.

Corollary 5.17. *Suppose that V is a set such that $a\#V$. Suppose also that $P \subseteq V$.*[8] *Then $(V \setminus P)[a{\mapsto}x] = V \ \Delta\ (P[a{\mapsto}x])$.*

Proof. We rewrite $V \setminus P$ as $V \ \Delta\ P$. The result follows by Theorems 5.14 and 5.5. □

The substitution action $[a{\mapsto}t]_2$, like $[a{\mapsto}t]_1$, is not necessarily pointwise on infinite sets:

Lemma 5.18. *It is not necessarily the case that if $z \in Z$ then $z[a{\mapsto}x] \in Z[a{\mapsto}x]$.*

Proof. It suffices to provide a counterexample. $a[a{\mapsto}\emptyset] = \emptyset$ and $\mathbb{A}[a{\mapsto}\emptyset] = \mathbb{A}$ by Theorem 5.5. We note that $\emptyset \notin \mathbb{A}[a{\mapsto}\emptyset]$. □

Definition 5.19. Call Z **finite crucial** when $C_a Z$ is finite for all atoms a. As a matter of convention call $a \in \mathbb{A}$ finite crucial as well.

We can use Theorem 5.14, along with the notion of finite crucial sets, to obtain an elementary characterisation of the substitution action on a broad class of sets:

Theorem 5.20. *If Z is finite crucial then $Z[a{\mapsto}x] = (Z - a) \ \Delta\ \{z[a{\mapsto}x] \mid z \in C_a Z\}$.*

Proof. By Theorem 5.14

$$Z[a{\mapsto}x] = (Z - a) \ \Delta\ (C_a Z)[a{\mapsto}x].$$

We use Lemma 5.8. □

Lemma 5.21.
1. *If Z is finite crucial then so is $C_a Z$.*
2. *If X and Y are finite crucial then so is $X \ \Delta\ Y$.*
3. *If Z is finite crucial then so is $Z - a$.*

Proof.
1. $C_a(C_a Z) = C_a Z$ by Lemma 4.18, and this is finite.
2. By Lemma 4.17 and some easy calculations on sets.
3. By the first two parts. □

Definition 5.22. Define the class of **hereditarily finite crucial** sets inductively by:

[7] This particular fact enters into the axioms for substitution via ($\mathbf{id}{\mapsto}$) in Definition 3.1.

[8] Note that $a\#P$ need not necessarily hold. For example $\{a\} \subseteq \mathbb{A}$.

- $a \in \mathbb{A}$ is hereditarily finite crucial.
- Z is hereditarily finite crucial if Z is finite crucial and every $z \in Z$ is hereditarily finite crucial.

The following technical properties will be useful soon:

Lemma 5.23. 1. *Any finite set is finite crucial.*
2. *A finite set is hereditarily finite crucial if all of its elements are.*
3. *If X and Y are hereditarily finite crucial then so is $X \mathbin{\Delta} Y$.*
4. *If Z is hereditarily finite crucial then so is $C_a Z$.*
5. *If Z is hereditarily finite crucial then so is $Z - a$.*

Proof. 1. By Corollary 4.22.
2. By the first part and from the inductive definition of hereditarily finite crucial.
3. Using part 2 of Lemma 5.21 and the fact that $X \mathbin{\Delta} Y \subseteq X \cup Y$.
4. Suppose Z is hereditarily finite crucial. Then by part 1 of Lemma 5.21 $C_a Z$ is finite crucial, so it suffices to show that all its elements are hereditarily finite crucial.
   Take any $z \in C_a Z$. By definition, either $z \in Z$ or that there is some $b\#z$ such that $(b\ a)z \in Z$. If $z \in Z$ then z is hereditarily finite crucial by our assumption that Z is hereditarily finite crucial. Otherwise $(b\ a)z \in Z$ and so $(b\ a)z \in Z$ is hereditarily finite crucial. By Theorem 2.5 it follows that z is hereditarily finite crucial.
5. The result follows by parts 3 and 4. □

Corollary 5.24. *If z and x are hereditarily finite crucial then $z[a{\mapsto}x]$ is hereditarily finite crucial.*

Proof. We reason by ϵ-induction. First we consider the base cases:

- $a[a{\mapsto}x] = x$. We assumed x is hereditarily finite and the result follows.
- $b[a{\mapsto}x] = b$. By definition b is hereditarily finite crucial and the result follows.

Now suppose $Z \notin \mathbb{A}$. We can rewrite Z as $(Z - a) \mathbin{\Delta} C_a Z$. By assumption $C_a Z$ is finite, by Corollary 4.15 also $a\#(Z - a)$. By Theorem 5.20 also

$$Z[a{\mapsto}x] = (Z - a) \mathbin{\Delta} \{z[a{\mapsto}x] \mid z \in C_a Z\}.$$

By parts 2 and 3 of Lemma 5.23 it now suffices to show that
$z[a{\mapsto}x]$ is hereditarily finite crucial for each $z \in C_a Z$. This follows by the inductive hypothesis (and Theorem 2.5; we do not know that $z \in Z$ but $\pi z \in Z$ for some π and that is enough). □

Theorem 5.25. *If z, x, and y are hereditarily finite crucial and if $a\#y$, then*

$$z[a{\mapsto}x][b{\mapsto}y] = z[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]].$$

Proof. We work by ϵ-induction. The base cases are easy:

- $a[a{\mapsto}x][b{\mapsto}y] = x[b{\mapsto}y] = a[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]]$.
- $b[a{\mapsto}x][b{\mapsto}y] = y = b[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]]$.
- $c[a{\mapsto}x][b{\mapsto}y] = c = c[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]]$.

Now suppose $Z \notin \mathbb{A}$. We can rewrite Z using Lemma 4.17 to

$$\begin{aligned} Z &= (Z - a) \,\Delta\, C_a Z \\ &= ((Z - a) - b) \,\Delta\, C_b(Z \,\Delta\, C_a Z) \,\Delta\, C_a Z \\ &= ((Z - a) - b) \,\Delta\, C_b Z \,\Delta\, C_b C_a Z \,\Delta\, C_a Z. \end{aligned}$$

By Corollary 4.15 $b\#(Z-a)-b$ and $a\#(Z-a)-b$. Therefore by scope-extrusion Theorem 5.14 we can write:

$$Z[a{\mapsto}x][b{\mapsto}y] = ((Z - a) - b) \,\Delta\, \{z[a{\mapsto}x][b{\mapsto}y] \mid z \in C_b Z \,\Delta\, C_b C_a Z \,\Delta\, C_a Z\}.$$

It is a fact of the definitions that for each $z \in C_b Z \,\Delta\, C_b C_a Z \,\Delta\, C_a Z$ there is some permutation π such that $\pi z \in Z$. We use Theorem 2.5 and the inductive hypothesis:

$$Z[a{\mapsto}x][b{\mapsto}y] = ((Z-a)-b) \,\Delta\, \{z[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]] \mid z \in C_b Z \,\Delta\, C_b C_a Z \,\Delta\, C_a Z\}.$$

We now reverse our previous reasoning to obtain the result. □

Hereditarily finite crucial sets z generalise syntax trees. They need not be finite, but for each atom they do have only finitely many elements 'responsible' for that atom being in $supp(z)$.

6. Summary, related work, and conclusions

Our substitution actions were discovered in a set theory known for the better part of a century [7, 40]. The substitution arises spontaneously; there is nothing in the axioms of FM set theory to suggest that a substitution action should be possible. Even after the author and Pitts re-discovered Fraenkel-Mostowski set theory and applied it to syntax-with-binding [23], the substitution action remained hidden. This is good; it suggests that we have not built into our model the behaviour we want. 'Substitution' arises as a mathematical operation derived naturally from the basic foundational assumptions determined by the axioms of Fraenkel-Mostowski set theory. This is an interesting contribution for set theory, computer science, and their intersection.

We see three broad avenues for future work: to look at how substitution interacts with 'objects of behaviour', by this we mean sets representing for example traces of actions in process calculi; with graphs of functions represented as sets; and with equivalence classes of syntax quotiented for example by $\alpha\beta$-equivalence. We did consider this in a little more detail at the end of Subsection 6.1. Preliminary investigations, not written up in this paper, suggest that the techniques presented in this paper can be applied in all three cases above, modulo some technical adaptations to the specific domain. This paper provides the technical basis for investigating these, and we hope other applications, in future work.

6.1. Summary of the two substitution actions

In Subsection 1.1 we sketched the design issues involved in defining a substitution action on a sets universe. $[a{\mapsto}x]_1$ and $[a{\mapsto}x]_2$ both do the following to Z: they decompose Z into component planes, calculate substitution component-wise, and combine the results.

There may even be other notions of decomposition other than the planes used in this paper, but what is certain is that not every decomposition is suitable. For example $\mathbb{A} = \{a\} \cup (\mathbb{A} \setminus \{a\})$ but we do not want to base a substitution action on this decomposition since $a\#\mathbb{A}$ but $\neg a\#\{a\}$ and $\neg a\#(\mathbb{A} \setminus \{a\})$.

Care goes into the design of the expression $A(a{\mapsto}supp(x)) \setminus \delta(u, a, x)$ in Definition 3.14. This 'intelligently adjusts $\|$-orbits' for changes in support caused by substitution on representatives — part of the mechanism of getting generalised α-equivalence classes right — and it describes the interaction of substitution with planes. Other choices may also be possible here.

The design of $[a{\mapsto}x]_2$ introduces C_aZ, the a-crucial elements of Z. In a certain sense this only substitutes on those elements 'responsible' for $\neg a\#Z$ (if any). It is more complex but has somewhat better properties; notably it satisfies scope-extrusion Theorem 5.14, treats 'missing elements' and set-subtraction in an elegant way (Corollary 5.17), and interacts well with hereditarily finite crucial sets (Theorem 5.25).

On 'easy sets' the two substitution actions that we have defined coincide (Theorems 3.29 and 5.12). The important differences are in the treatment of 'missing elements'. For example (recall that $C_a(\mathbb{A} \setminus \{a\}) = \{a\}$):

$$\begin{array}{ll} (\mathbb{A} \setminus \{a\})[a{\mapsto}\emptyset]_1 = \mathbb{A} & (\mathbb{A} \setminus \{a\})[a{\mapsto}(a,a)]_1 = \mathbb{A} \\ (\mathbb{A} \setminus \{a\})[a{\mapsto}\emptyset]_2 = \mathbb{A} \cup \{\emptyset\} & (\mathbb{A} \setminus \{a\})[a{\mapsto}(a,a)]_2 = \mathbb{A} \cup \{(a,a)\} \end{array}$$

On (sets modelling) syntax the behaviour is as we expect:

$$\begin{array}{ll} (a,\emptyset)[a{\mapsto}\emptyset]_1 = (\emptyset,\emptyset) & (a,b)[a{\mapsto}b]_1 = (b,b) \\ (a,\emptyset)[a{\mapsto}\emptyset]_2 = (\emptyset,\emptyset) & (a,b)[a{\mapsto}b]_2 = (b,b) \end{array}$$

Behaviour may become counterintuitive, if we base our intuitions only on the behaviour for syntax and then extend these intuitions to the universe of sets:

$$\begin{array}{ll} ((\mathbb{A}\setminus\{a\})\cup\{\emptyset\})[a{\mapsto}\emptyset]_1 = \mathbb{A}\cup\{\emptyset\} & ((\mathbb{A}\setminus\{a\})\cup\{(a,a)\})[a{\mapsto}(a,a)]_1 = \mathbb{A}\cup\{(a,a)\} \\ ((\mathbb{A}\setminus\{a\})\cup\{\emptyset\})[a{\mapsto}\emptyset]_2 = \mathbb{A} & ((\mathbb{A}\setminus\{a\})\cup\{(a,a)\})[a{\mapsto}(a,a)]_2 = \mathbb{A} \end{array}$$

$$\begin{array}{ll} \{a,\emptyset\}[a{\mapsto}\emptyset]_1 = \{\emptyset\} & \{a,b\}[a{\mapsto}b]_1 = \{b\} \\ \{a,\emptyset\}[a{\mapsto}\emptyset]_2 = \emptyset & \{a,b\}[a{\mapsto}b]_2 = \emptyset \end{array}$$

We conclude by asking: what other technical demands can we make of a substitution action?

- What sets have most general unifiers (what is the theory of 'set unification')?
- What is the computational content of an element z considered as the function x maps to $z[a{\mapsto}x]$?
- Can we provide a denotational semantics for first-order rewriting [39]?

- Can we model the λ-calculus [4], the π-calculus [34] and its related name-passing calculi, or first-order predicate logic [27], using a substitution action to directly model substitution?

- It may be possible to model first-order logic by mapping sentences $\mathtt{P}, \mathtt{Q} \ldots$, open or closed, to subsets of an FM set U representing the universe of discourse; variables $\mathtt{x}, \mathtt{y} \ldots$ of a first-order language map to the set $\mathbb{A}$ of atoms; if a first-order sentence $\mathtt{P}$ is assigned the set $y \subseteq U$ as its semantic denotation, then the sentence $\forall \mathtt{x}\mathtt{P}$ has as its denotation the set $\bigcap_u \{y[a{\mapsto}u] \mid u \in U\}$.

  This is more than just a translation of the syntactic substitution-for-all-terms operation into another language, for $[a{\mapsto}u]$ represents an operation directly on the sets universe which can be constructed from the axioms of FM sets.

6.2. Related work

The axioms for substitution in this paper are taken from [22]. This paper continues that work by showing how the Fraenkel-Mostowski sets universe can be made a model of those axioms; in some sense which it may be possible to make formal in future work, it is a largest possible model.

Higher-order logic uses substitution to reflect function application in syntax [42]. Substitution does not exist in denotation, and neither do variables.

Calculi of explicit substitutions analyse the structure of carrying out a substitution, but to our knowledge they are purely operational systems without denotation [32].

Aczel's 'generalised set theory' [2] and 'universes with parameters' [3] model variable symbols (Aczel calls them *parameters*) as atoms in a ZFA-like set theory. However, Aczel imposes all the structure he needs as specific axioms on names. What Aczel does with his sets is also very different; it concerns coinductive structures and non-wellfounded set theory [1] as a semantics for behaviour. Aczel's substitution action is such that $Z[a{\mapsto}x] = \{z'[a{\mapsto}x] \mid z' \in Z\}$ always, like the pointwise substitution action $[a{\mapsto}x]_n$ from Definition 3.3.

CIAO Prolog [25, Extra-logical predicates] and a-logic [18] include a predicate $\mathsf{var}(x)$ which is true of x when x is a variable symbol. It is used to implement shallow embeddings of programming languages — for more details see [25]. The functional programming language CURRY [26] has similar constructs. The designers of these languages struggle to bring current denotational methods to bear on these constructs, because they put variables in the denotation, just as we have done. Making a connection with our work is current research.

Fine [12] has undertaken a philosophical axiomatic investigation of *dependency between objects*. For example, z and $2 * z$ are 'arbitrary numbers' but they are also connected. A formal connection might be established with our constructions if we take substitution to give a notion of 'correlation' between sets; for example if $z[a{\mapsto}x] = z'$, or $z[a{\mapsto}x] = z'[a{\mapsto}x]$, then z and z' are 'correlated'.

Nominal terms [41] come equipped with a substitution action, but this is for *context* variables and is not capture-avoiding. It should not be confused with the capture-avoiding substitution action for atoms of this paper.

Other axioms for substitution exist; for example in work by Salibra [37], by Beeson [5], by Crabbé [10], and by Feldmann [11]. The axioms of Crabbé and Feldmann are

not of capture-avoiding substitution; the axioms of Salibra and Beeson are. Salibra is interested in models but his models are abstract (he does universal algebra). Beeson is interested in theorem-proving. The axioms of Salibra and Beeson for substitution are embedded in larger theories of Turing-complete computation based on the λ-calculus, which gives their treatment a very different flavour from ours. Feldman's axioms are based on a concrete model using *environments* (functions mapping variables to denotational objects) which is very different from our model which is based on a set theory (a far larger construction) and in which variables inhabit the denotation. Fiore, Plotkin, and Turi consider substitution in [14]. The models they build use presheaf categories not dissimilar to the category of nominal sets from [23] (there, we called them 'FM-sets'). Modulo inessential differences, the 'definition of substitution by structural recursion' on pages 6 to 7 of that paper corresponds with the substitution by structural recursion in [24], while the treatment of 'substitution algebras' on page 7 corresponds with our axiomatisation in nominal algebra of capture-avoiding substitution [22].[9]

Schmidt-Schauß and Siekmann introduce unification algebra [38] and Williams introduces instantiation theory [43]. The idea is to free the theory of unification from (the shackles of) syntax. An axiomatic approach is taken to substitution. The work is 'bespoke' in the sense that the specification of the model includes a specification of the object of study; substitution, instantiation, and unification. In this paper we do not consider unification but we do extract substitution from an independent structure (the axioms of set theory).

6.3. The rôle of Fraenkel-Mostowski sets

We will conclude with some words on our use of Fraenkel-Mostowski sets.

The substitution actions we build in Definitions 3.14 and 5.1 rely on having a collection Z whose elements have a substitution action; in that case, we can define substitution actions on Z as well. So we can read our substitution actions informally as follows:

> Given a collection of elements with a substitution action, the collection of finitely supported subsets of that collection (the 'finitely supported powerset') also has a substitution action — and in at least two different ways!

So this is not quite about Fraenkel-Mostowski set theory; it is about sets with a substitution action, their finitely supported powersets, and ways to extend the substitution action from the base set to its set of finitely supported powersets.

The cumulative hierarchy model is built from the set of atoms, taking finitely supported powersets transfinitely (Remark 2.12). Thus, defining our substitution action on a cumulative hierarchy model lets us get on with studying a substitution action, without having first to express a theory of 'sets with a substitution action'.

[9] From the point of view of [14, 22] the existence of multiple substitution actions on a model of Fraenkel-Mostowski sets, as exhibited in this paper, corresponds with the observation that the underlying set of a substitution algebra as specified in [14], or of a substitution set as specified in [22], may support two or more distinct substitution actions.

This is convenient, but there is more to our design choice than convenience. It is an empirical observation that most interesting mathematical structures — probably, *all* the ones we normally consider — can be modelled by taking powersets and using sets comprehension to whittle down the resulting structures to something representing what we want. Powersets are a foundation for mathematics.

Our substitution actions act on every mathematical structure that we can model using powersets. This concrete model is not universal in a categorical sense; it *is* universal in the sense that it operates on mathematical universes and everything that can be built within them. That is something new and unexpected.

We illustrate this with a possible application. Our substitution actions make the atoms-concretion function from [23] into a total function by substituting for the abstracted atom; $[a]z$ concreted at x is $z[a{\mapsto}x]$.[10] This happens automatically, because the Gabbay-Pitts atoms-abstraction can be modelled using powersets.

References

[1] Peter Aczel. *Non-wellfounded Set Theory*. Number 14 in CSLI lecture notes. CSLI, 1988.

[2] Peter Aczel. Generalised set theory. *CSLI lecture notes*, 1(58):1–17, 1996.

[3] Peter Aczel and Rachel Lunnon. Universes and parameters. *CSLI lecture notes*, 2:3–24, 1991.

[4] H. P. Barendregt. *The Lambda Calculus: its Syntax and Semantics (revised ed.)*. North-Holland, 1984.

[5] Michael Beeson. Lambda logic. In *Second International Joint Conference on Automated Reasoning (IJCAR 2004)*, volume 3097 of *Lecture Notes in Computer Science*, pages 460–474. Springer, 2004.

[6] Nick Benton and Benjamin Leperchey. Relational reasoning in a nominal semantics for storage. In *Proc. of the 7th Int'l Conf. on Typed Lambda Calculi and Applications (TLCA)*, volume 3461 of *Lecture Notes in Computer Science*, pages 86–101, 2005.

[7] Norbert Brunner. 75 years of independence proofs by Fraenkel-Mostowski permutation models. *Mathematica Japonica*, 43:177–199, 1996.

[8] James Cheney and Christian Urban. Alpha-prolog: A logic programming language with names, binding and alpha-equivalence. In Bart Demoen and Vladimir Lifschitz, editors, *Proc. of the 20th Int'l Conf. on Logic Programming (ICLP 2004)*, number 3132 in Lecture Notes in Computer Science, pages 269–283. Springer, 2004.

[10] Thanks to Andrew Pitts for this observation.

[9] D. Clarke, R. Hinze, J. Jeuring, A. oh, and J. de Wit. The generic haskell user's guide. Technical Report UU-CS-2001-26, November 2001.

[10] M. Crabbé. On the notion of substitution. *Logic Journal of the IGPL*, 12 n.2:111–124, 2004.

[11] Norman Feldman. Axiomatization of polynomial substitution algebras. *Journal of Symbolic Logic*, 47(3):481–492, 1982.

[12] Kit Fine. *Reasoning with Arbitrary Objects*. Blackwell, 1985.

[13] Marcelo Fiore and Daniele Turi. Semantics of name and value passing. In *Proc. 16$^{\text{th}}$ LICS Conf.*, pages 93–104. IEEE, Computer Society Press, 2001.

[14] Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. Abstract syntax and variable binding. In *LICS '99: 14th Annual Symposium on Logic in Computer Science*, pages 193–202. IEEE, 1999.

[15] Murdoch J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, Cambridge, UK, 2000.

[16] Murdoch J. Gabbay. The pi-calculus in FM. In Fairouz Kamareddine, editor, *Thirty-five years of Automath*, volume 28 of *Kluwer applied logic series*, pages 247–269. Kluwer, November 2003.

[17] Murdoch J. Gabbay. A General Mathematics of Names. *Information and Computation*, 205(7):982–1011, July 2007.

[18] Murdoch J. Gabbay and Michael J. Gabbay. a-logic. In *We Will Show Them: Essays in Honour of Dov Gabbay*, volume 1. College Publications, 2005.

[19] Murdoch J. Gabbay and Martin Hofmann. Nominal renaming sets. In *LPAR'08*, 2008.

[20] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding Substitution as a Nominal Algebra. In *ICTAC 2006: Theoretical Aspects of Computing*, volume 4281 of *Lecture Notes in Computer Science*, pages 198–212, 2006.

[21] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. In *PPDP '06: Proc. of the 8th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming*, pages 189–200. ACM, 2006.

[22] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, January 2008.

[23] Murdoch J. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax with Variable Binding (journal version). *Formal Aspects of Computing*, 13(3–5):341–363, 2001.

[24] Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax Involving Binders. In *14th Annual Symposium on Logic in Computer Science*, pages 214–224. IEEE Computer Society Press, 1999.

[25] CLIP group. CIAO Prolog documentation project. `http://www.ciaohome.org`.

[26] Michael Hanus, Sergio Antoy, Herbert Kuchen, Francisco J. López-Fraguas, Wolfgang Lux, Juan José Moreno-Navarro, and Frank Steiner. Curry: An integrated functional logic language. 2003.

[27] Wilfrid Hodges. Elementary predicate logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 1, pages 1–131. Kluwer, 2001.

[28] Martin Hofmann. Semantical analysis of higher-order abstract syntax. In *14th Annual Symposium on Logic in Computer Science*, pages 204–213. IEEE, 1999.

[29] Thomas Jech. Set theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2002.

[30] P. T. Johnstone. *Notes on logic and set theory*. Cambridge University Press, 1987.

[31] Ralf Lämmel and Simon Peyton Jones. Scrap your boilerplate: a practical design pattern for generic programming. In *TLDI*, pages 26–37. ACM, 2003.

[32] Pierre Lescanne. From lambda-sigma to lambda-upsilon: a journey through calculi of explicit substitutions. In *POPL*, pages 60–69. ACM, 1994.

[33] S. Lusin and A. Salibra. The lattice of lambda theories. *Journal of Logic and Computation*, 14 n.3:373–394, 2004.

[34] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, 1992.

[35] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.

[36] A. M. Pitts and Murdoch J. Gabbay. A metalanguage for programming with bound names modulo renaming. In *MPC2000*, volume 1837 of *Lecture Notes in Computer Science*, pages 230–255. Springer, 2000.

[37] Antonino Salibra. On the algebraic models of lambda calculus. *Theoretical Computer Science*, 249(1):197–240, 2000.

[38] Manfred Schmidt-Schauß and Jörg Siekmann. Unification algebras: An axiomatic approach to unification, equation solving and constraint solving. Technical Report SEKI-report SR-88-09, FB Informatik, Universität Kaiserslautern, 1988.

[39] Terese. *Term Rewriting Systems*. Number 55 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

[40] J. Truss. Permutations and the axiom of choice. In H.D.Macpherson R.Kaye, editor, *Automorphisms of first order structures*, pages 131–152. OUP, 1994.

[41] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.

[42] Johan van Benthem. Higher-order logic. In *Handbook of Philosophical Logic, 2nd Edition*, volume 1, pages 189–244. Kluwer, 2001.

[43] James Williams. Instantiation theory. Technical Report M-90-04, The Mitre Corporation, 1990.