Math. Struct. in Comp. Science (2013), *vol.* 23, *pp.* 1002–1031. © Cambridge University Press 2013
doi:10.1017/S0960129512000849 First published online 15 May 2013

Imaginary groups: lazy monoids and reversible computation

MURDOCH J. GABBAY[†] and PETER H. KROPHOLLER[‡]

[†]*School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, United Kingdom*
Website: `http://www.gabbay.org.uk`
[‡]*Mathematical Sciences, Southampton University, Highfield, Southampton SO17 1BJ, United Kingdom*
Email: `p.h.kropholler@southampton.ac.uk`

Received 30 September 2011; revised 21 April 2012

We use constructions in monoid and group theory to exhibit an adjunction between the category of partially ordered monoids and *lazy* monoid homomorphisms and the category of partially ordered groups and group homomorphisms such that the unit of the adjunction is injective. We also prove a similar result for sets acted on by monoids and groups.
We introduce the new notion of a *lazy homomorphism* for a function f between partially ordered monoids such that $f(m \circ m') \leqslant f(m) \circ f(m')$.
Every monoid can be endowed with the discrete partial ordering ($m \leqslant m'$ if and only if $m = m'$), so our constructions provide a way of embedding monoids into groups. A simple counterexample (the two-element monoid with a non-trivial idempotent) and some calculations show that one can never hope for such an embedding to be a monoid homomorphism, so the price paid for injecting a monoid into a group is that we must weaken the notion of a homomorphism to this new notion of a lazy homomorphism.
The computational significance of this is that a monoid is an abstract model of computation – or at least of 'operations' – and, similarly, a group models *reversible* computations/operations. With this reading, the adjunction with its injective unit gives a systematic high-level way of faithfully translating an irreversible system into a 'lazy' reversible one.
Informally, but perhaps informatively, we can describe this work as follows: we give an abstract analysis of how we can sensibly add 'undo' (in the sense of 'control-Z').

1. Introduction

1.1. *Monoids, groups, computation and inverses*

Suppose we have a computer. We run programs, and this changes the state of the computer.

[†] Murdoch Gabbay acknowledges the support of the Leverhulme trust and of grant RYC-2006-002131 at the Polytechnic University of Madrid.
[‡] Peter Kropholler acknowledges the support of Cornell University, New York and of the Institut Mittag-Leffler, Sweden.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

This can be thought of as a set X (representing *states*) and some functions $f : X \to X$ (representing *operations*) closed under function composition. The functions in $X \to X$ form a *monoid*, which is like a group but does not have inverses. Monoids provide an abstract model of state and computation.

Now suppose we want to undo the effects of an operation. However, since $f : X \to X$ need not be injective, given some $x \in X$, there may be several y such that $f(y) = x$, so monoids do not have inverses.

Yet 'undo' or 'rollback' operations are very common, and familiar from, for example, word processing or version control (subversion was used in the production of this paper). A notion of 'state, with rollback' called *transactional memory* is useful, especially in concurrency where updates from one process may need to be undone in the light of results later arrived at by a concurrent process (Herlihy and Moss 1993). In proof search, the difference between proof rules that are reversible, such as the left conjunction introduction ($\wedge\mathbf{L}$) below, which is also called *asynchronous*, and those that are not reversible, such as the right disjunction introduction ($\vee\mathbf{R}$) below, which is also called *synchronous*, has mathematical significance (Liang and Miller 2009).

$$\textit{Asynchronous}\quad \frac{\Gamma, A \wedge B \vdash C}{\Gamma, A, B \vdash C}\,(\wedge\mathbf{L}) \qquad \textit{Synchronous}\quad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\,(\vee\mathbf{R})$$

Note that ($\wedge\mathbf{L}$) just rearranges connectives (by turning a '$\wedge$' into a ',' if we read the rule bottom-up), but ($\vee\mathbf{R}$) eliminates B so is (from the point of view of proof search) an irreversible step.

Furthermore, the field of reversible computation, which has physical importance for heat dissipation and the design of biological and quantum computation, is devoted to the study of hand-tailored reversible computational mechanisms – see Section 7 for references and further discussion along with our conclusions.

Now let us return to our monoid $\mathcal{M}$. This is given to us and was *not* designed from the start to be reversible. How can we 'add an undo function' to $\mathcal{M}$ in some abstract sense? In view of the discussion above, we can distil this down mathematically to the following formal and rigorously answerable question:

Given a monoid $\mathcal{M}$, does there exist a group $\mathcal{M}^{\pm}$ and an **injective** *monoid map $\mathcal{M} \to \mathcal{M}^{\pm}$?*

We ask for the map to be injective so that distinct operations in the monoid correspond to distinct *reversible* operations in the group. That is, $\mathcal{M}^{\pm}$ needs to represent the original states and operations faithfully; otherwise, we could just map all of $\mathcal{M}$ to the trivial group with one element.

Given the injection above, we could have our cake and eat it: given any notion of computation, represent it as a monoid $\mathcal{M}$, calculate $\mathcal{M}^{\pm}$, compute in that group (with an undo function) and then, if we like the result, we can return it, and if we do not like it, we can always undo it.

1.2. *Monoids, lazy monoids and our solution*

Unfortunately, what we have asked for is impossible, and it is easy to prove this with a small counterexample.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Example 1.1. Consider the two-element monoid $\{0,1\}$ with

$$1 \circ x = x \quad \text{for} \quad x \in \{0,1\} \quad \text{and} \quad 0 \circ x = 0 \quad \text{for} \quad x \in \{0,1\}.$$

If we try to adjoin an element 0^{-1}, then $0 = 0 \circ 0$, so (applying 0^{-1} to both sides) $0 = 1$. Hence, we lose injectivity: if $\mathcal{M} = \{0,1\}$, then $\mathcal{M}^{\pm}$ is the trivial group in which computation is reversible because everything is equal to the identity operation, which does nothing at all.

In this paper we will indeed exhibit a construction that assigns to a monoid $\mathcal{M}$ a *partially ordered* group $\mathcal{M}^{\pm}$, and we will exhibit a *lazy* monoid homomorphism from $\mathcal{M}$ to $\mathcal{M}^{\pm}$. By moving to partially ordered structures and lazy homomorphisms, the contradiction noted in the previous paragraph is avoided.

Thus, it is possible to add 'undo' to a monoid after all – we just have to do it in the right way. How this should work is not immediately obvious.

Our solution supports an attractive computational interpretation: think of the monoid $\mathcal{M}$ as a set of operations and monoid composition as their composition, and both of these may be irreversible. Then the 'lazy' group $\mathcal{M}^{\pm}$ we build consists of operations that have been assembled but not yet carried out – they are 'suspended'. Each element in the group is a plan of action; we can compose plans, modify them, erase elements from them, and this is all reversible because nothing has been carried out. The partial order represents the operation of actually carrying out the plan; moving down along the partial order represents refining results as the plan is executed, which may lose information and is irreversible.

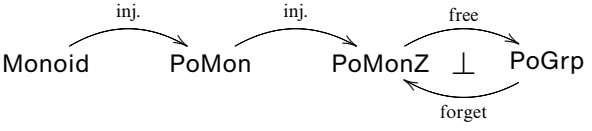
Our main result can be stated as follows:

> *There is an adjunction between the categories of partially ordered monoids and lazy homomorphisms and of partially ordered groups and group homomorphisms, and the unit of the adjunction is injective.*

See Theorems 3.22, 3.24, 3.25 and 3.30.

1.3. *Map of the paper*

Here is an eagle's eye view of how our construction works for monoids:



Monoid is the category of monoids and monoid homomorphisms and PoMon is the category of partially ordered monoids and monotone monoid homomorphisms. The injection from Monoid to PoMon is just obtained by giving a monoid the trivial partial order relating x to itself.

PoMonZ is the category of partially ordered monoids and lazy monotone monoid homomorphisms, where 'lazy' is defined in Definition 2.3. Every monotone monoid

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

homomorphism is a lazy monotone monoid homomorphism, so we obtain a trivial injection from PoMon to PoMonZ.

PoGrp is the category of partially ordered groups and monotone group homomorphisms. There is an easy forgetful functor from PoGrp to PoMonZ, and a *free* functor that is left adjoint to the forgetful functor mapping a monoid $\mathcal{M}$ to its free group with added 'imaginary' inverse $\mathcal{M}^{\pm}$. Constructing the underlying set of $\mathcal{M}^{\pm}$ is easy, as is defining its group structure and giving it a natural preorder (Definitions 3.3 and 3.6). What is not easy is proving that the preorder is a partial order, or showing that the unit of the adjunction is a monomorphism: see Theorems 3.22 and 3.24; see also Theorem 3.25.

In Section 4, we will discuss how this can be generalised in various ways, and in Section 5, we will extend the construction to sets with a lazy monoid or group action. The generalisations are essentially routine, and extending to sets with an action, though not entirely trivial, is no harder.

2. Basic definitions

2.1. *Partially ordered groups and monoids, and lazy homomorphisms*

Definition 2.1. Consider the following axioms:

(assoc)	$x \circ (x' \circ x'') = (x \circ x') \circ x''$
(id)	$\mathrm{id} \circ x = x \circ \mathrm{id} = x$
(inv)	$x \circ x^{-1} = x^{-1} \circ x = \mathrm{id}$

— A *group* $\mathcal{G}$ is a tuple $(|\mathcal{G}|, \circ, \mathrm{id}, ^{-1})$ of a *carrier set* $|\mathcal{G}|$, a *composition* $\circ$ mapping $|\mathcal{G}| \times |\mathcal{G}|$ to $|\mathcal{G}|$, an element $\mathrm{id} \in |\mathcal{G}|$ and an *inverse map* $^{-1}$ mapping $|\mathcal{G}|$ to $|\mathcal{G}|$, and satisfying **(assoc)**, **(id)** and **(inv)**.

— A *monoid* $\mathcal{M}$ is a tuple $(|\mathcal{M}|, \circ, \mathrm{id})$ as above, satisfying **(assoc)** and **(id)**.

Definition 2.2. A *(compatible) order* on a monoid $\mathcal{M}$ is a partial order $\leqslant \subseteq |\mathcal{M}| \times |\mathcal{M}|$ such that $x \leqslant y$ implies $x \circ z \leqslant y \circ z$ and $z \circ x \leqslant z \circ y$.

— A *partially ordered monoid* is a pair $(\mathcal{M}, \leqslant)$ of a monoid $\mathcal{M}$ and a compatible order on $\mathcal{M}$.

— A *partially ordered group* is a pair $(\mathcal{G}, \leqslant)$ of a group $\mathcal{G}$ and a compatible order on $\mathcal{G}$, considered as a monoid.

Definition 2.3. We call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ between monoids $\mathcal{M}$ and $\mathcal{M}'$ a *homomorphism* when it satisfies the two axioms

$$\mathrm{id}_{\mathcal{M}'} = f(\mathrm{id}_{\mathcal{M}})$$
$$f(x_1 \circ_{\mathcal{M}} x_2) = f(x_1) \circ_{\mathcal{M}'} f(x_2).$$

If $\mathcal{M}'$ is a group, the first equation can be deduced from the second, as can the identity $f(x^{-1}) = f(x)^{-1}$; but in general both axioms are required.

Suppose now that $\mathcal{M}$ and $\mathcal{M}'$ are partially ordered monoids. We call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ a *lazy homomorphism* when

$$\mathrm{id}_{\mathcal{M}'} \leqslant_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$$

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

and

$$f(x_1 \circ_{\mathcal{M}} x_2) \leqslant_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2).$$

We say $f \in |\mathcal{M}| \to |\mathcal{M}'|$ is *monotone* when $x \leqslant_{\mathcal{M}} y$ implies $f(x) \leqslant_{\mathcal{M}'} f(y)$.

We discuss this definition further in Section 2.2.

Definition 2.4.

— We write PoMonZ for the category of partially ordered monoids and monotone lazy homomorphisms between them *(Partially Ordered MONoids with laZy homomorphisms)*.
— We write PoGrp for the category of partially ordered groups and monotone lazy group homomorphisms between them *(Partially Ordered GRouPs and homomorphisms)*.

Note that in Definition 2.4 we take lazy homomorphisms in PoMonZ but 'real' homomorphisms in PoGrp. We could certainly define PoGrpZ, but it turns out that we will never use it: see Example 3.28 for an example illustrating why this is the case.

Remark 2.5. For convenience we will explicitly unpack what it means to be an arrow in PoMonZ. An arrow $f : (\mathcal{M}, \leqslant_{\mathcal{M}}) \to (\mathcal{M}', \leqslant_{\mathcal{M}'})$ is a function f from $|\mathcal{M}|$ to $|\mathcal{M}'|$ such that:

(1) $\mathrm{id}_{\mathcal{M}'} \leqslant_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$ (*laziness*).
(2) $f(x \circ_{\mathcal{M}} y) \leqslant_{\mathcal{M}'} f(x) \circ_{\mathcal{M}'} f(y)$ (*laziness*).
(3) $x \leqslant_{\mathcal{M}} y$ implies $f(x) \leqslant_{\mathcal{M}'} f(y)$ (*monotonicity*).

2.2. *Why lazy?*

There is a notion of a *lax functor* between 2-categories (see, for instance, Leinster (2004, Definition 1.5.8)). Specifically, for us here,[†] this suggests the following definition.

Definition 2.6. Suppose now that $\mathcal{M}$ and $\mathcal{M}'$ are partially ordered monoids. Call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ a *lax homomorphism* when

$$f(\mathrm{id}_{\mathcal{M}}) \leqslant_{\mathcal{M}'} \mathrm{id}_{\mathcal{M}'}$$

and

$$f(x_1 \circ_{\mathcal{M}} x_2) \leqslant_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2).$$

We call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ an *oplax* or *colax homomorphism* when

$$\mathrm{id}_{\mathcal{M}'} \leqslant_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$$

and

$$f(x_1) \circ_{\mathcal{M}'} f(x_2) \leqslant_{\mathcal{M}'} f(x_1 \circ_{\mathcal{M}} x_2).$$

[†] That is, specialising laxness to the case of a locally posetal 2-category on one object, also known as a partially ordered monoid.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

To see in elementary terms why lax homomorphisms are natural, note that if we consider the inequality

$$f(x_1 \circ_{\mathcal{M}} x_2) \leqslant_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2)$$

and imagine what its 0-ary version should be, we obtain $f(\mathrm{id}_{\mathcal{M}}) \leqslant \mathrm{id}_{\mathcal{M}'}$.

A lazy homomorphism sits in-between lax and oplax, combining the first axiom of oplax with the second axiom of lax. This is new.

We would like to give some intuition in elementary terms about why *lazy* homomorphisms are interesting.

(1) *An informal computational intuition:*

We can think of a partially ordered monoid or group as a set of operations that can be composed. We can then think of the partial order as describing the 'execution' of these operations: so $z \leqslant x \circ y$ means 'if we combine x with y and execute then we obtain z'. Then we can give the lazy inequalities the following informal reading:

— $\mathrm{id}_{\mathcal{M}'} \leqslant_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$ means *if we apply f to the identity and execute, we get the identity.*

— $f(x \circ_{\mathcal{M}} y) \leqslant_{\mathcal{M}'} f(x) \circ_{\mathcal{M}'} f(y)$ means *if we apply f to x and apply f to y and combine, then, if we execute, we can obtain the result of combining x and y in $\mathcal{M}$ first, and then applying f.*

We call this *lazy* by analogy with lazy evaluation in the λ-calculus.

(2) *A mathematical intuition:*

We want $\mathcal{M}$ to inject into the 'imaginary' group $\mathcal{M}^{\pm}$ that augments $\mathcal{M}$ with 'imaginary' inverses (see Section 3.1 for a formal definition). So let us consider a specific example and see what is required to make this work.

Let $\mathcal{M}$ be the one-element monoid $\{*\}$ with monoid composition $* \circ * = *$. Then $\mathcal{M}^{\pm}$ is isomorphic to the integers under addition: the unit is (); the number 1 corresponds to $+*$; the number 2 corresponds to $+*+*$; the number -1 corresponds to $-*$, and so on. The function taking $*$ to 1 is lazy because $0 \leqslant 1$ and $1 \leqslant 2$. This function is not lax, because $1 \not\leqslant 0$, and not oplax because $2 \not\leqslant 1$, but it is lazy.

So, with lazy homomorphisms, the integers $\mathbb{Z}$ considered as a group with their natural arithmetic ordering are exhibited as the free object over a single generator.

Interestingly, our proof of injectivity hinges on studying *index functions* $\iota(\text{-})_m$ from $\mathcal{M}^{\pm}$ to $\mathbb{Z}$: see Definition 3.12. So the meat of our proof (the technical action happens between Definition 3.12 and Lemma 3.16) hinges on a study of the structure of the functions and objects making up the following commuting square:

$$\begin{array}{ccc} \mathcal{M} & \xrightarrow{\eta} & \mathcal{M}^{\pm} \\ {\scriptstyle !\ \text{terminal map}}\downarrow & & \downarrow{\scriptstyle \iota(\text{-})} \\ \{*\} & \xrightarrow{*\mapsto 1} & \{*\}^{\pm} = \mathbb{Z} \end{array}$$

However, some care is needed here: though the total index function $\iota(\text{-})$ (a 'sum of exponents' function) is an arrow in PoMonZ, the proof uses measure functions $\iota(\text{-})_m$ that are not arrows in this category. So the abstract nonsense above is useful for organising the

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

material, but the proofs themselves cannot be conveniently phrased purely in categorical terms: we will use combinatorial arguments on concrete constructions.

For reference: $\mathcal{M}^{\pm}$ is defined in Definition 3.3; η is defined in Theorem 3.30; the proof that η is mono is Theorem 3.22; and commutativity is an easy calculation. With respect to later notation, we have abused notation in the diagram above and written $\mathcal{M}^{\pm}$ for $(\mathcal{M}^{\pm})^{\mathcal{I}}$ (see Definition 3.29).

3. The imaginary group

Our construction is the simplest possible: given a monoid $\mathcal{M}$, we take strings of elements-with-a-polarity + or – and quotient such that $+m{-}m$ becomes equal to the empty string. We immediately obtain an *imaginary* group $\mathcal{M}^{\pm}$. To be precise, it is the free group over $\mathcal{M}$ in the category of lazy monoids. This is Definition 3.3 (the hard part is not constructing it, but proving it is not trivial: see the conservative extension results below).

For the following step we need a little general theory about partial orders on groups. It is a fact that a partial order can be generated by a normal submonoid. This turns out to be useful: see Proposition 3.7.

We then define a preorder on $\mathcal{M}^{\pm}$ in Figure 1 in submonoid style. The order is sound by construction; the rule (**leq**) makes sure that if $n \leqslant m$ in $\mathcal{M}$, then $+n \leqslant +m$ in $\mathcal{M}^{\pm}$.

The hard work is *injectivity* and *conservative extension* results: *viz.* that the preorder is a partial order and is not 'too big' – in the spirit of Example 1.1, we do not want $+n \leqslant +m$ to hold because $+n = +m$ always. The main technical lemma is Proposition 3.19, and the main results are Theorems 3.22, 3.24 and 3.25.

Once we have this, it is a matter of fairly routine calculations to convert this into the adjunction promised in the Introduction, and culminating with Theorem 3.30.

3.1. *The free group*

This section describes an easy construction: given $\mathcal{M}$, we consider $|\mathcal{M}|$ and build the free group generated by this set.

Definition 3.1. Given a set X, we write $X^{\pm}$*string* for the set inductively defined by

$$s ::= () \mid s{+}x \mid s{-}x \qquad (x \in X).$$

We call () the *empty string*, and write string concatenation as $s{\cdot}t$. We may omit the initial (), so, for instance, we may write $+m$ as shorthand for $()+m$.

So if $X = \{1, 2, 3, \ldots\}$, then, intuitively, $X^{\pm}$*string* is the set of lists of non-zero integers.

Definition 3.2. We define $\sim$ to be the least equivalence relation on $|\mathcal{M}|^{\pm}$*string* such that:
(1) $()+m{-}m \sim () \sim ()-m{+}m$.
(2) If $s \sim s'$ and $t \sim t'$, then $s{\cdot}t \sim s'{\cdot}t'$.

Definition 3.3. Given a monoid $\mathcal{M}$, we define the *free group* $\mathcal{M}^{\pm}$ over $|\mathcal{M}|$ as follows:
— The carrier set is $|\mathcal{M}|^{\pm}$*string* quotiented by $\sim$.
— Group composition is given by concatenation of representatives.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

We may write m for both $m \in |\mathcal{M}|$ and for the $\sim$-equivalence class of $() + m$.

Lemma 3.4. $\mathcal{M}^{\pm}$ is indeed a group, with the natural composition, identity and inverse.

Example 3.5.

(1) Suppose $\mathcal{M}$ is the monoid of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ under addition (so 0 is the identity). Then $\mathcal{M}^{\pm}$ is *not* isomorphic to the integers. For instance, $+3{+}4 \in |\mathcal{M}^{\pm}|$ is not equal to $+7$. We will return to this in a mathematically richer context in Remark 3.18.

(2) Suppose $\mathcal{M}$ is the one-element monoid with underlying set $\{*\}$ and monoid composition $* \circ * = *$ (we also saw this in Section 2.2). Then $\mathcal{M}^{\pm}$ is isomorphic to the integers under addition: the unit is $()$; the number 1 corresponds to $+*$; the number 2 corresponds to $+*{+}*$; the number -1 corresponds to $-*$; and so on.
So $\mathcal{M}^{\pm}$ is *resource sensitive*. Note, however, that resources can be destroyed if we interpret $\geqslant$ as 'entailment', so $+*{+}* \geqslant +*$.[†]

3.2. *Positive cones*

This section develops some more standard theory. There is a bijection between normal submonoids of $\mathcal{G}$ and compatible partial orders. We will only need one direction of this for the following proofs – see Glass (1999, Section 1.1) for more details.

Definition 3.6.

— Suppose $\mathcal{G} = (|\mathcal{G}|, \circ, ^{-1})$ is a group. We say $X \subseteq |\mathcal{G}|$ is *normal* when $x \in X$ and $y \in |\mathcal{G}|$ implies $y \circ x \circ y^{-1} \in X$.

— Suppose $\mathcal{M}$ is a monoid. We call $X \subseteq |\mathcal{M}|$ a *submonoid* when $\mathrm{id} \in X$ and if $x \in X$ and $y \in X$ implies $x \circ y \in X$.

Suppose $P \subseteq |\mathcal{G}|$ is a normal submonoid of a group $\mathcal{G}$. Then we write $x \leqslant_P y$ when $y \circ x^{-1} \in P$.

Proposition 3.7. Suppose $\mathcal{G} = (|\mathcal{G}|, \circ, ^{-1})$ is a group and $P \subseteq |\mathcal{G}|$ is a normal submonoid of $\mathcal{G}$. Then $\leqslant_P$ from Definition 3.6 is a compatible preorder on $|\mathcal{G}|$.[‡]

Proof. We will consider each property in turn:

— By assumption, $\mathrm{id} \in P$, so it follows that $m \leqslant_P m$ and thus $\leqslant_P$ is reflexive.

— Suppose $x \leqslant_P y$ and $y \leqslant_P z$. This means that $y \circ x^{-1} \in P$ and $z \circ y^{-1} \in P$. Since P is a monoid, $z \circ x^{-1} \in P$ also. It follows that $\leqslant_P$ is transitive.

[†] In logical terminology, we are saying that we have contraction on the right but not on the left.
Contraction in logic is a rule that allows us to convert from $A \wedge A$ to A. If A represents an assertion and $\wedge$ represents logical conjunction, then this is immediate. If A represents a pound coin and $\wedge$ represents addition, then this may not be so.
'Contraction on the left' means we may go from A to $A \wedge A$ (like truth, or taxes), whereas 'contraction on the right' means we may go from $A \wedge A$ to A (like truth, or our bank account). More on this can be found in the field of *substructural logics* – see Restall (1999) for details.

[‡] Recall that a *preorder* is reflexive and transitive, but not necessarily antisymmetric. A *partial order* is an antisymmetric preorder.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

$$\frac{(n \leqslant_{\mathcal{M}} m)}{+m{-}n \in P^+}\ (\textbf{leq}) \qquad \frac{(m_1, m_2 \in |\mathcal{M}|)}{+m_1{+}m_2{-}(m_1 \circ_{\mathcal{M}} m_2) \in P^+}\ (\textbf{lazy})$$

$$\frac{p \in P^+ \quad (g \in \mathcal{M}^{\pm})}{g^{-1} \circ_{\mathcal{M}^{\pm}} p \circ_{\mathcal{M}^{\pm}} g \in P^+}\ (\textbf{norm}) \qquad \frac{p \in P^+ \quad p' \in P^+}{p \circ_{\mathcal{M}^{\pm}} p' \in P^+}\ (\textbf{mon})$$

Fig. 1. Rules determining the partial order on $\mathcal{M}^{\pm}$

— Suppose $x \leqslant_P y$. This means that $y \circ x^{-1} \in P$. To check that $\leqslant_P$ is compatible, we need to verify that for any $z \in |\mathcal{G}|$, we have $z \circ x \leqslant_P z \circ y$ and $x \circ z \leqslant_P y \circ z$.
Since P is a normal subgroup, $z \circ y \circ x^{-1} \circ z^{-1} \in P$ also, and this means that $z \circ x \leqslant_P z \circ y$.
Also, $y \circ x^{-1} = y \circ z \circ z^{-1} \circ x^{-1}$, so $x \circ z \leqslant_P y \circ z$. □

3.3. *The partial order*

The non-standard part begins here. We fix a partially ordered monoid $(\mathcal{M}, \leqslant_{\mathcal{M}})$. In Definition 3.3, we saw how to construct the free group $\mathcal{M}^{\pm}$ generated by $|\mathcal{M}|$.

We show how to extend $\leqslant_{\mathcal{M}}$ to a preorder $\leqslant_{\mathcal{M}^{\pm}}$ on $\mathcal{M}^{\pm}$. By Proposition 3.7, it suffices to exhibit a normal submonoid of $\mathcal{M}^{\pm}$. In fact, the preorder is also a partial order, but proving this takes more work: the main technical result is Proposition 3.19 and this quickly yields Theorem 3.22.

Definition 3.8. We define P^+ to be the least subset of $|\mathcal{M}^{\pm}|$ closed under rules (**leq**), (**lazy**), (**norm**), and (**mon**) from Figure 1. By rules (**norm**) and (**mon**), this is a normal submonoid of $\mathcal{M}^{\pm}$, so, by Proposition 3.7, P^+ determines a compatible preorder on $|\mathcal{M}^{\pm}|$. We write this $\leqslant_{\mathcal{M}^{\pm}}$.

Remark 3.9. A few words on the rules in Figure 1 are in order here:

— (**leq**) expresses the fact that $n \leqslant_{\mathcal{M}} m$ implies $n \leqslant_{\mathcal{M}^{\pm}} m$.
— (**lazy**) expresses the fact that $+(m_1 \circ_{\mathcal{M}} m_2) \leqslant_{\mathcal{M}^{\pm}} +m_1{+}m_2$.
— (**norm**) expresses the fact that if $p \leqslant_{\mathcal{M}^{\pm}} q$, then $p \circ_{\mathcal{M}^{\pm}} r \leqslant_{\mathcal{M}^{\pm}} q \circ_{\mathcal{M}^{\pm}} r$ (left-compatibility is automatic by construction – see Proposition 3.7).
— (**mon**) expresses the transitivity of $\leqslant_{\mathcal{M}^{\pm}}$.

The two base cases are (**leq**) and (**lazy**). Of these, rule (**leq**) introduces a string that has index 0 and rule (**lazy**) introduces a string that has index 1.

Lemma 3.10.

(1) If $n \leqslant_{\mathcal{M}} m$, then $+n \leqslant_{\mathcal{M}^{\pm}} +m$.
(2) If $n \leqslant_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$, then $+n \leqslant_{\mathcal{M}^{\pm}} +m_1{+}m_2$.

Proof. The first part follows directly from (**leq**) and Definition 3.6. The second part follows using (**lazy**). □

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

The converse implications also hold, but the proofs are harder: see Theorems 3.24 and 3.25.

Now (**lazy**) in Figure 1 corresponds to the condition

$$f(x_1 \circ_{\mathcal{M}'} x_2) \leqslant_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2)$$

from Definition 2.3, so what corresponds to the condition

$$\mathrm{id}_{\mathcal{M}'} \leqslant_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$$

from Definition 2.3? In fact, this is derivable.

Corollary 3.11. $() \leqslant_{\mathcal{M}^\pm} +\mathrm{id}_{\mathcal{M}}$. (Recall from Section 3.1 that () is the identity element of $\mathcal{M}^\pm$.)

Proof. By Lemma 3.10 (2), we have

$$+\mathrm{id}_{\mathcal{M}} \leqslant_{\mathcal{M}^\pm} +\mathrm{id}_{\mathcal{M}}+\mathrm{id}_{\mathcal{M}}.$$

We then apply $-\mathrm{id}_{\mathcal{M}}$ to both sides. □

3.4. *Technical lemmas*

This is where the maths begins to become non-trivial. In Definition 3.8 we defined a relation $\leqslant_{\mathcal{M}^\pm}$ on $|\mathcal{M}^\pm|$. It is easy to see that this is reflexive and transitive, but is it antisymmetric? In other words, is $\leqslant_{\mathcal{M}^\pm}$ a partial order as the caption to Figure 1 suggests, and not just a preorder? Also, is $\leqslant_{\mathcal{M}^\pm}$ a conservative extension of $\leqslant_{\mathcal{M}}$, or does it introduce spurious relations that should not be there?

This section develops the technical machinery we need to answer these questions. These lemmas will then be used to give concise answers to the questions in Section 3.5.

Definition 3.12. Suppose $m \in |\mathcal{M}|$. We define *index functions* $\iota(s)_m$ and $\iota(s)$ on $|\mathcal{M}|^\pm$*string* as follows:

$$\begin{aligned}
\iota(())_m &= 0 \\
\iota(s{+}m')_m &= \iota(s)_m \qquad (m' \neq m) \\
\iota(s{-}m')_m &= \iota(s)_m \qquad (m' \neq m) \\
\iota(s{-}m)_m &= \iota(s)_m - 1 \\
\iota(s{+}m)_m &= \iota(s)_m + 1
\end{aligned}$$

$$\iota(s) = \Sigma_{m \in |\mathcal{M}|} \iota(s)_m.$$

In words, we can say that $\iota(s)$ *sums exponents.*

Lemma 3.13. The index functions are well defined as functions on $p \in |\mathcal{M}^\pm|$. That is, if $p \in |\mathcal{M}^\pm|$, then for every $s, s' \in p$ and every m, we have $\iota(s)_m = \iota(s')_m$, and, therefore, $\iota(s) = \iota(s')$.

Proof. To prove the statement, we just consider each of the rules in Definition 3.2 and observe that they conserve the index. □

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Lemma 3.14. If $() \leqslant_{\mathcal{M}^\pm} p$ and $\iota(p) = 0$, then every proof that $p \in P^+$ does not mention (**lazy**).

Proof. By arithmetic, (**leq**) generates an element with index 0, (**lazy**) increases the index by 1 and (**norm**) and (**mon**) leave the index unchanged. The result then follows. □

Corollary 3.15. If $() \leqslant_{\mathcal{M}^\pm} p$ and $\iota(p)_m = 0$ for all m, then $p = ()$.

Proof. We first use Lemma 3.14 to show that the derivation of $() \leqslant_{\mathcal{M}^\pm} p$ cannot use rule (**lazy**). The rules (**norm**) and (**mon**) do not affect the index at all.

Let n be $\leqslant_{\mathcal{M}}$-minimal in p. This n must have been introduced by (**leq**) for some $n \leqslant_{\mathcal{M}} m$. Since n is $\leqslant_{\mathcal{M}}$-minimal in p, there is no other instance of (**leq**) for $n' \lneq_{\mathcal{M}} n$. By assumption, $\iota(p)_n = 0$, so it must be that $n = m$, and the result then follows. □

Lemma 3.16.

(1) If $p \in P^+$, then $\iota(p) \geqslant 0$.
(2) If $p \leqslant_{\mathcal{M}^\pm} q$, then $\iota(p) \leqslant \iota(q)$.

Proof.

(1) This part follows by a routine induction on the rules in Figure 1.
(2) Suppose $p \leqslant_{\mathcal{M}^\pm} q$. Then $() \leqslant_{\mathcal{M}^\pm} p^{-1} \circ_{\mathcal{M}^\pm} q$. By part (1) of this result, $\iota(p^{-1} \circ_{\mathcal{M}^\pm} q) \geqslant 0$. It is a fact that $\iota(p^{-1} \circ_{\mathcal{M}^\pm} q) = -\iota(p) + \iota(q)$. The result then follows. □

Corollary 3.17. If $+m_1 \ldots +m_k \leqslant_{\mathcal{M}^\pm} +n_1 \ldots +n_l$ and $+n_1 \ldots +n_l \leqslant_{\mathcal{M}^\pm} +m_1 \ldots +m_k$, then $k = l$.

Proof. We have

$$\iota(+m_1 \ldots +m_k) = k$$
$$\iota(+n_1 \ldots +n_l) = l.$$

We then use Lemma 3.16 (2). □

Remark 3.18. Recall Example 3.5 (1) and take $\mathcal{M}$ to be the monoid of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ under addition (so 0 is the identity). Then, by Corollary 3.17, we have $+3{+}4 \not\leqslant_{\mathcal{M}^\pm} +7$ since $\iota(+3{+}4) = 2$ and $\iota(+7) = 1$ and $2 \not\leqslant 1$.

In fact, we can get a much stronger result than Corollary 3.17, but the proof is harder – see Corollary 3.23.

Proposition 3.19 might seem a little formidable, but in fact it only expresses some common sense: suppose the index of p is 0 so that for every $+m$ in p there is a corresponding $-n$. Suppose there are more $+m$s in p than $-m$s so that there is a surfeit of ms in p and $\iota(p)_m > 0$. In order for everything to balance out, there must be some n such that $\iota(p)_n < 0$ (that is, such that there is a deficit of ns), and since p is a finite structure, we can find a $\leqslant_{\mathcal{M}}$-minimal such n in p. This is what Proposition 3.19 asserts, and this property turns out to be key.

Proposition 3.19. Suppose $p \in P^+$ and $\iota(p) = 0$. Suppose $\iota(p)_m > 0$. Then there exists some $\leqslant_{\mathcal{M}}$-minimal $n \lneq_{\mathcal{M}} m$ such that $\iota(p)_n < 0$.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Proof. The proof is by induction on the derivation that $p \in P^+$. We consider each rule in turn in a long, but essentially routine, analysis of cases (of which, (**mon**) is arguably the most interesting, and certainly the most complex):

— Rule (**leq**):
Then $n \leqslant_{\mathcal{M}} m$, but we still need to show that $\iota(p)_m > 0$ and $\iota(p)_n < 0$. By assumption, $\iota(+m-n)_m > 0$, so $m \neq n$. It then follows by arithmetic that $\iota(+m-n)_n < 0$.

— Rule (**lazy**):
If

$$\iota(+m_1+m_2-(m_1 \circ_{\mathcal{M}} m_2)) = 0,$$

then either $m_1 = m_2 = m = \mathrm{id}$ or $m_1 = m \neq \mathrm{id}$ and $m_2 = \mathrm{id}$ or $m_2 = m \neq \mathrm{id}$ and $m_1 = \mathrm{id}$. However, all these cases are impossible because we have assumed $\iota(p)_m > 0$ (for some m).

— Rule (**norm**):
This follows from the fact that

$$\iota(p)_m = \iota(g^{-1} \circ_{\mathcal{M}^\pm} p \circ_{\mathcal{M}^\pm} g)_m.$$

— Rule (**mon**):
Suppose $\iota(p \circ_{\mathcal{M}^\pm} p')_m > 0$. There are two subcases:

– $\iota(p)_m > 0$:
By the induction hypothesis, there exists a $\leqslant_{\mathcal{M}}$-minimal $n \leqslant_{\mathcal{M}} m$ such that $\iota(p)_n < 0$. There are now two further subcases:

- $\iota(p')_n < -\iota(p)_n$:
In this case $\iota(p \circ_{\mathcal{M}^\pm} p')_n < 0$, and we are done.
- $\iota(p')_n \geqslant -\iota(p)_n$:
In this case, in particular, $\iota(p')_n > 0$. So there exists some $\leqslant_{\mathcal{M}}$-minimal $n' \leqslant_{\mathcal{M}} n$ such that $\iota(p')_{n'} < 0$. By assumption, n was minimal, so $\iota(p)_n = 0$. It follows that $\iota(p \circ_{\mathcal{M}^\pm} p')_{n'} < 0$. By transitivity, $n' \leqslant_{\mathcal{M}} m$, and we are done.

– $\iota(p)_m \leqslant 0 \ldots$:
This is symmetric to the case $\iota(p')_n < -\iota(p)_n$. □

Proposition 3.19 has a natural dual.

Proposition 3.20. Suppose $p \in P^+$ and $\iota(p) = 0$. Suppose $\iota(p)_m < 0$. Then there exists a $\leqslant_{\mathcal{M}}$-maximal $n \geqslant_{\mathcal{M}} m$ such that $\iota(p)_n > 0$.

Proof. We take $\mathcal{M}$ and p and consider $\mathcal{M}'$, which is the same as $\mathcal{M}$, but we invert $\leqslant_{\mathcal{M}}$ (we omit a formal definition), and p^{-1}. Thus, $\iota(p)_m < 0$ becomes $\iota(p^{-1})_m > 0$ and $n \geqslant_{\mathcal{M}} m$ becomes $n \leqslant_{\mathcal{M}'} m$, and '$\leqslant_{\mathcal{M}}$-maximal' becomes '$\leqslant_{\mathcal{M}'}$-minimal'. The result is then just Proposition 3.19. □

Corollary 3.21. Suppose $p \in P^+$ and $\iota(p) = 1$. Then there exist $n, m_1, m_2 \in |\mathcal{M}|$ such that $n \leqslant_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$ and $\iota(p)_n \leqslant -1$ and $\iota(p)_{m_1} \geqslant 1$ and $\iota(p)_{m_2} \geqslant 1$.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Proof. The proof is by induction on the derivation that $p \in P^+$. We consider each rule in turn:

— Rule (**leq**):
This case is not relevant since $\iota(+m-n) = 0$.

— Rule (**lazy**):
This case follows by construction since

$$m_1 \circ_{\mathcal{M}} m_2 \leqslant_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2.$$

— Rule (**norm**):
This case follows from the fact that

$$\iota(p)_m = \iota(g^{-1} \circ_{\mathcal{M}^\pm} p \circ_{\mathcal{M}^\pm} g)_m.$$

— Rule (**mon**):
Suppose $\iota(p \circ_{\mathcal{M}^\pm} p') = 1$. Then by Lemma 3.16(1), precisely one of $\iota(p) = 1$ and $\iota(p') = 0$ or $\iota(p') = 1$ and $\iota(p) = 0$ must hold.
Suppose $\iota(p) = 1$ and $\iota(p') = 0$; the other case is symmetric, so we will omit it. By the induction hypothesis, we obtain n, m_1 and m_2, as required, in p.
Now, if $\iota(p')_n \geqslant -\iota(p)_n$, we use Proposition 3.19 to obtain an $n' \leqslant_{\mathcal{M}} n$ such that $n' \leqslant_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$, and then use the transitivity of $\leqslant_{\mathcal{M}}$.
Similarly, if $\iota(p')_{m_1} \leqslant -\iota(p)_{m_1}$, we use Proposition 3.20 and the compatibility of the partial order with $\circ_{\mathcal{M}}$. And similarly for m_2. The result then follows. □

3.5. *The preorder is a partial order and the natural map is injective*

Theorem 3.22. If $p \leqslant_{\mathcal{M}^\pm} q$ and $q \leqslant_{\mathcal{M}^\pm} p$, then $p = q$.

Proof. Suppose $p \leqslant_{\mathcal{M}^\pm} q$ and $q \leqslant_{\mathcal{M}^\pm} p$. We write $P = q \circ_{\mathcal{M}^\pm} p^{-1}$ and $Q = p \circ_{\mathcal{M}^\pm} q^{-1}$. Thus $() \leqslant_{\mathcal{M}^\pm} P$ and $() \leqslant_{\mathcal{M}^\pm} Q$. Also, by Lemma 3.16(2), we have $\iota(p) = \iota(q)$, so $\iota(P) = 0$.

Now, if $\iota(P)_m = 0$ for all m, then, using Corollary 3.15, it follows that $p = q$, and we are done.

So suppose $\iota(P)_{m'} \neq 0$ for some m'. We will deduce a contradiction. If $\iota(P)_{m'} \neq 0$ for some m', then $\iota(P)_m > 0$ for some m. By Proposition 3.19, we have $\iota(P)_n < 0$ for some $n \leqslant_{\mathcal{M}} m$ and n is the $\leqslant_{\mathcal{M}}$-minimal such amongst the monoid elements mentioned in P. But then $\iota(Q)_n > 0$, and by Proposition 3.19 again, $\iota(Q)_{m''} < 0$ for some $m'' \leqslant_{\mathcal{M}} n$. This contradicts the minimality of n. □

We can immediately apply Theorem 3.22 to strengthen Corollary 3.17.

Corollary 3.23. If $+m_1 \ldots +m_k \leqslant_{\mathcal{M}^\pm} +n_1 \ldots +n_l$ and $+n_1 \ldots +n_l \leqslant_{\mathcal{M}^\pm} +m_1 \ldots +m_k$, then $k = l$ and $m_i = n_i$ for every $1 \leqslant i \leqslant k$.

Proof. By Theorem 3.22, we have

$$+m_1 \ldots +m_k = +n_1 \ldots +n_l.$$

The result then follows from Definition 3.3. □

Theorem 3.24 is a conservative extension result expressing the fact that $\mathcal{M}$ injects into $\mathcal{M}^\pm$ considered as a partially ordered monoid.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Theorem 3.24. $+n \leqslant_{\mathcal{M}^\pm} +m$ if and only if $n \leqslant_{\mathcal{M}} m$.

Proof. The right-to-left implication is Lemma 3.10 (1).

For the left-to-right implication, we reason as follows. If $m = n$, then $n \leqslant_{\mathcal{M}} m$ by reflexivity. So suppose $m \neq n$ and $+n \leqslant_{\mathcal{M}^\pm} +m$.

By Definition 3.6, this means that $+m{-}n \in P^+$. We note that $\iota(+m{-}n) = 0$ and $\iota(+m{-}n)_m > 0$. Using Proposition 3.19, it then follows that $n \leqslant_{\mathcal{M}} m$. □

$\mathcal{M}^\pm$ does not introduce any new 'lazy' relations.

Theorem 3.25. $+n \leqslant_{\mathcal{M}^\pm} +m_1{+}m_2$ if and only if $n \leqslant_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$.

Proof. The right-to-left implication is Lemma 3.10 (2). The left-to-right implication follows using Corollary 3.21. □

3.6. *The functor from* PoMonZ *to* PoGrp

Definition 3.26. We define a functor $\mathsf{imag} : \mathsf{PoMonZ} \to \mathsf{PoGrp}$ as follows:

— $\mathcal{M} \in \mathsf{PoMonZ}$ maps to $(\mathcal{M}^\pm, \leqslant_{\mathcal{M}^\pm})$ (Definitions 3.3 and 3.8).

— An arrow

$$f : \mathcal{M} \to \mathcal{M}' \in \mathsf{PoMonZ}$$

acts elementwise on representatives of $\sim$-equivalence classes. In full:

$$\begin{aligned} \mathsf{imag}(f)() &= () \\ \mathsf{imag}(f)(s{+}m) &= \mathsf{imag}(f)(s){+}f(m) \\ \mathsf{imag}(f)(s{-}m) &= \mathsf{imag}(f)(s){-}f(m). \end{aligned}$$

Lemma 3.27. imag from Definition 3.26 is well defined and is a functor from PoMonZ to PoGrp.

Proof. Since $\mathsf{imag}(f)$ acts elementwise,

$$\mathsf{imag}(f)(s{\cdot}s') = \mathsf{imag}(f)(s){\cdot}\mathsf{imag}(f)(s')$$

and $\mathsf{imag}(f)() = ()$ by construction. Thus, $\mathsf{imag}(f)$ is a group homomorphism.

We need to check that $\mathsf{imag}(f) : \mathsf{imag}(\mathcal{M}) \to \mathsf{imag}(\mathcal{M}')$ is well defined on $\sim$-equivalence classes. We will just sketch the two least trivial cases:

— $\mathsf{imag}() = \mathrm{id}_{\mathcal{M}'} = f(\mathrm{id}_{\mathcal{M}}) = \mathsf{imag}(\mathrm{id}_{\mathcal{M}})$. (This is slightly interesting because it depends on our assumption that $f(\mathrm{id}_{\mathcal{M}}) = \mathrm{id}_{\mathcal{M}'}$.)

— $\mathsf{imag}(s{+}m{-}m) = \mathsf{imag}(s){+}f(m){-}f(m) \sim \mathsf{imag}(s)$, and similarly for $s{-}m{+}m$.

Finally, we must check that $\mathsf{imag}(f)$ is monotone. To do this we just need to show that if $p \in P_{\mathcal{M}}$ then $\mathsf{imag}(p) \in P_{\mathcal{M}'}$. It suffices to check the following (note that the second case is not what one might immediately expect):

— *If* $n \leqslant_{\mathcal{M}} m$, *then* $f(n) \leqslant_{\mathcal{M}} f(m)$:
This follows because f is monotone by assumption.

— *If* $m = m_1 \circ_{\mathcal{M}} m_2$ *then* $f(m) \leqslant_{\mathcal{M}'} f(m_1) \circ_{\mathcal{M}'} f(m_2)$:
This follows because f is lazy. □

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Example 3.28. Consider the natural numbers $\{0, 1, 2, \dots\}$ with their usual ordering. This can be regarded as an object in PoMonZ, with 0 being the additive identity. We use $\mathbb{N}$ to denote this object. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(n) = n + 1 \text{ if } n \neq 0$$
$$f(0) = 0.$$

Then

$$\begin{aligned} f(n+m) &= n + m + 1 \\ &\leqslant n + 1 + m + 1 \\ &= f(n) + f(m), \end{aligned}$$

unless one or both of n and m are zero, in which case

$$f(n+m) = f(n) + f(m).$$

So f is an example of a monotone lazy homomorphism and is an arrow in PoMonZ from $\mathbb{N}$ to itself.

Then $\mathsf{imag}(\mathbb{N}) = \mathbb{N}^{\pm}$ is the free group on $\{0, 1, 2, \dots\}$ and $\mathsf{imag}(f)$ acts pointwise on strings so that, for example,

$$\mathsf{imag}(f)(+0+1+2) = +0+2+3.$$

This illustrates why we took homomorphisms in Definition 2.4, not lazy homomorphisms, between the partially ordered groups; $\mathsf{imag}(f)$ is always a homomorphism, by construction.

3.7. *The forgetful functor from* PoGrp *to* PoMonZ

Definition 3.29. We define a functor $\mathsf{forg} : \mathsf{PoGrp} \to \mathsf{PoMonZ}$ as follows:

— $\mathcal{G} \in \mathsf{PoGrp}$ maps to $\mathcal{G}$ considered as a monoid: we denote this by $\mathcal{G}^{\mathcal{M}}$.
— An arrow $g : \mathcal{G} \to \mathcal{G}' \in \mathsf{PoGrp}$ maps to itself considered as a (lazy) monoid homomorphism.

Theorem 3.30. $\mathsf{imag} : \mathsf{PoMonZ} \to \mathsf{PoGrp}$ is left adjoint to $\mathsf{forg} : \mathsf{PoGrp} \to \mathsf{PoMonZ}$. In symbols,

$$\mathsf{imag} \dashv \mathsf{forg} : \mathsf{PoGrp} \to \mathsf{PoMonZ}.$$

Proof. We will exhibit the unit and counit of the adjunction.

The unit $\eta_{\mathcal{M}} : \mathcal{M} \to (\mathcal{M}^{\pm})^{\mathcal{M}}$ is given by the function mapping $m \in |\mathcal{M}|$ to $+m \in |\mathcal{M}^{\pm}|$. By Lemma 3.10(1), this is monotone. By Lemma 3.10(2) and Corollary 3.11, it is a lazy homomorphism (Definition 2.3).

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

The counit $\epsilon_{\mathcal{G}} : (\mathcal{G}^{\pm})^{\mathcal{I}} \to \mathcal{G}$ is given by the function inductively defined on representatives as follows:

$$\begin{aligned}\epsilon_{\mathcal{G}}() &= \mathrm{id}_{\mathcal{G}}\\ \epsilon_{\mathcal{G}}(s{+}m) &= \epsilon_{\mathcal{G}}(s) \circ_{\mathcal{G}} m\\ \epsilon_{\mathcal{G}}(s{-}m) &= \epsilon_{\mathcal{G}}(s) \circ_{\mathcal{G}} m^{-1}.\end{aligned}$$

By construction, this is a group homomorphism. We prove inductively on the definition of $\leqslant_{(\mathcal{G}^{\mathcal{I}})^{\pm}}$ in Figure 1 that the map is monotone, that is, if $s \leqslant_{(\mathcal{G}^{\mathcal{I}})^{\pm}} s'$, then $\epsilon_{\mathcal{G}}(s) \leqslant_{\mathcal{G}} \epsilon_{\mathcal{G}}(s')$:

— *Rule* (**leq**):
Suppose $+n \leqslant_{(\mathcal{G}^{\mathcal{I}})^{\pm}} +m$ because $n \leqslant_{\mathcal{G}^{\mathcal{I}}} m$. Then $n \leqslant_{\mathcal{G}} m$ and we are done.

— *Rule* (**lazy**):
Suppose $+m \leqslant_{(\mathcal{G}^{\mathcal{I}})^{\pm}} +m_1{+}m_2$ because $m, m_1, m_2 \in |\mathcal{G}^{\pm}|$ and $m = m_1 \circ_{\mathcal{G}^{\mathcal{I}}} m_2$. Then, by construction,

$$\epsilon_{\mathcal{G}}(+m) = \epsilon_{\mathcal{G}}(+m_1{+}m_2),$$

and we are done.

— *Rules* (**norm**) *and* (**mon**):
These cases follow by the induction hypothesis and the compatibility and transitivity of $\leqslant_{\mathcal{G}}$.

It is not hard to see that the adjunction maps

— $g : \mathcal{M}^{\pm} \to \mathcal{G} \in \mathsf{PoGrp}$ to $\bar{g} : \mathcal{M} \to \mathcal{G}^{\mathcal{I}}$, which takes m to $g(+m)$, and
— $f : \mathcal{M} \to \mathcal{G}^{\mathcal{I}} \in \mathsf{PoMon}$ to $\bar{f} : \mathcal{M}^{\pm} \to \mathcal{G}$, which acts on generators by taking $+m$ to $f(m)$ and $-m$ to $f(m)^{-1}$.

Verifying naturality is routine. □

Remark 3.31. In Definition 3.12 we defined the *measure function* $\iota(\text{-})$ from $|\mathcal{M}^{\pm}|$ to integers $\mathbb{Z}$ considered as a group under addition for each partially ordered monoid $\mathcal{M}$. It is easy to check that $\iota(\text{-})$ is a monotone homomorphism of groups, so, by Theorem 3.30, it must correspond to a unique monotone lazy homomorphism from $\mathcal{M}$ to $\mathbb{Z}^{\mathcal{I}}$ (that is, to $\mathbb{Z}$ considered as a monoid under addition).

It is not hard to see that this monotone lazy homomorphism is the trivial function mapping every $m \in |\mathcal{M}|$ to $1 \in \mathbb{Z}$. The laziness conditions from Definition 2.3 then correspond to the arithmetic inequalities $0 \leqslant 1$ and $1 \leqslant 2$.

We could consider other measure functions coming from different functions out of $\mathcal{M}$; discovering whether any of them are useful is future research.

From the point of view of these general categorical observations, the injectivity and conservative extension results Theorems 3.22, 3.24 and 3.25 are proved by a detailed analysis of the fine structure of this measure function: we decompose it into $\iota(\text{-})_m$ for each m, which is not monotone and so is not an arrow in PoGrp, and then reason by a succession of counting arguments that culminate with Proposition 3.19.

4. Commutative and non-associative generalisation

In this brief section we make a few observations about generalising the proofs.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

4.1. *Commutative and other generalisations*

The key technical lemmas in Section 3.4 are by counting arguments, where the counting is performed by the index function of Definition 3.12. Because of this, it is not hard to verify that we can re-play all the proofs for the special cases where $\mathcal{M}$ and $\mathcal{M}^{\pm}$ are taken to be abelian.

That is, even if $x \circ_{\mathcal{M}} y = y \circ_{\mathcal{M}} x$ for all $x, y \in |\mathcal{M}|$, by our construction in Definition 3.2 it is not the case that

$$+x+y \sim +y+x.$$

What is true is that

$$+x+y \geqslant_{\mathcal{M}^{\pm}} +(x \circ_{\mathcal{M}} y) = +(y \circ_{\mathcal{M}} x).$$

However, none of the proofs depend on this; the proofs we have seen only count elements, and we can rearrange elements however we like. So if $\mathcal{M}$ is commutative, we could make $\mathcal{M}^{\pm}$ be commutative too, just by extending $\sim$ in Definition 3.2. Then we would obtain an adjunction between abelian monoids and abelian groups ((**mon**) in Figure 1 would become irrelevant).

Similarly, there is nothing in our proofs that depends on the associativity of $\mathcal{M}$, or even on the existence of a unit (we would drop the first laziness condition of Definition 2.3).

So it is not hard to generalise or specialise the results of this paper to, for example:

— abelian monoids (monoids in which $x \circ y = y \circ x$) and abelian groups – the theorems in Section 3.5 should survive unchanged except that Corollary 3.23 would have to take m_i and n_i up to some permutative rearrangement;
— semigroups (like monoids, but without a unit) and groups, so that we add both inverses *and* a unit;
— applicative structures (like monoids, but without a unit and without associativity) and applicative structures with inverses and a unit (but without associativity).

For instance, the elements of a Boolean algebra form a commutative monoid under $\wedge$, and we can turn this into a commutative group.

In this sense, the proofs of this paper are robust: so long as none of the congruences we choose to put in or take out of Definition 3.2 affect the index function of Definition 3.12, the proofs should survive.

4.2. *Generalisation to categories and groupoids*

It is not hard to add types to our results, that is, to generalise from monoids to categories. We will just sketch the relevant definitions here.

Definition 4.1. We say a category $\mathcal{C}$ is *locally posetal* when its hom-sets are enriched to be partial orders, and this is consistent with arrow composition.[†]

A *groupoid* is a category in which every arrow has an inverse.

[†] So if $f' \leqslant f$ are two arrows in $\mathcal{C}(X, Y)$ and $g' \leqslant g$ are two arrows in $\mathcal{C}(Y, Z)$, then $f \circ g \leqslant f' \circ g'$ in $\mathcal{C}(X, Z)$.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

A locally posetal category can be viewed as an instance of a *2-category* (Leinster 2004; Cheng and Lauda 2004) in which the hom-sets are posets.

Definition 4.2. A *lazy functor* on locally posetal categories $F : \mathcal{C} \to \mathcal{D}$ is a map from objects of $\mathcal{C}$ to objects of $\mathcal{D}$ and from hom-posets $\mathcal{C}(X, Y)$ to $\mathcal{D}(F(X), F(Y))$ such that:

— If $f' \leqslant f$ in $\mathcal{C}(X, Y)$, then $F(f') \leqslant F(f)$ in $\mathcal{D}(F(X), F(Y))$.
— $\mathrm{id}_{F(X)} \leqslant F(\mathrm{id}_X)$ for every $X \in Obj(\mathcal{C})$.
— $F(f \circ g) \leqslant F(f) \circ F(g)$ for every $f \in \mathcal{C}(X, Y)$ and $g \in \mathcal{C}(Y, Z)$.

We write PoCatZ and PoGrpoid for the categories of small locally posetal categories and lazy functors and small locally posetal groupoids and functors between them, respectively.

Definition 4.3. Given a locally posetal category $\mathcal{C}$, we define a locally posetal groupoid $\mathcal{C}^{\pm}$ as follows:

— Objects of $\mathcal{C}^{\pm}$ are objects of $\mathcal{C}$.
— An arrow is the quotiented well-typed[†] formal sum of arrows from $\mathcal{C}$ just as in Definition 3.3, where, if $f \in \mathcal{C}(X, Y)$, then $+f$ is given type $\mathcal{C}^{\pm}(X, Y)$ and $-f$ is given type $\mathcal{C}^{\pm}(Y, X)$.
— The partial order is defined on arrows just as in Figure 1 (and thus only on *well-typed* representatives).

With Definition 4.3, the adjunction of Section 3.7 extends to an adjunction between PoCatZ and PoGrpoid. The proof of injectivity is no different from the case of monoids; we just have to keep track of types. So, for instance, we have the following definition.

Definition 4.4. We define *index functions* $\imath(s)_m$ and $\imath(s)$ just as in Definition 3.12.

The well-typedness conditions are orthogonal to the counting arguments that follow, and it is easy to replay the proofs to obtain, for instance, the following theorem.

Theorem 4.5.

— $+f' \leqslant +f$ in $\mathcal{C}^{\pm}(X, Y)$ if and only if $f' \leqslant f$, for $f', f \in \mathcal{C}(X, Y)$.
— $+f \leqslant +g+h$ in $\mathcal{C}^{\pm}(X, Z)$ if and only if $f \leqslant g \circ h$ in $\mathcal{C}(X, Z)$, for $f \in \mathcal{C}(X, Z)$ and $g \in \mathcal{C}(X, Y)$ and $h \in \mathcal{C}(Y, Z)$.

We leave filling in the details of the relevant generalisations as an exercise. Since our proofs are by counting arguments, the generalisation to the typed case introduces extra richness, but no essential difficulties.

5. Sets with an action

We now extend the results of Section 3 to sets with a monoid action, showing how to extend a partially ordered set with an $\mathcal{M}$ monoid action to a partially ordered set with an $\mathcal{M}^{\pm}$ group action.

[†] We just do the obvious thing here: well-typed means, for instance, that we only allow ourselves to write $+f+g$ when $f \in \mathcal{C}(X, Y)$ and $g \in \mathcal{C}(Y, Z)$. It is easy to see that, for example, $+f-f$ is well typed and in $\mathcal{C}(X, X)$.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Examples of sets with a monoid action are common in computer science, and include:

— Syntax acted on by a monoid of substitutions.
— Integers $\mathbb{Z}$ acted on by the monoid of functions from $\mathbb{Z}$ to $\mathbb{Z}$.
— A domain model of the λ-calculus D acted on by $D^D \cong D$ considered as a monoid under functional composition.

So it is interesting to see how our results can be extended.

Section 5.1 contains the basic definitions, several key technical lemmas and one important result (Theorem 5.9). Section 5.2 defines the relevant adjunctions between the categories of partially ordered sets with a compatible $\mathcal{M}$-action and lazy functions between them, and of partially ordered sets with an $\mathcal{M}^{\pm}$-action and functions between them.

The proofs are easier than in Section 3, but they are not trivial, and do not follow by abstract nonsense. Certainly, if we can extend monoids to groups, we can extend sets acted on by monoids to sets acted on by groups in a standard way.

Just as in Section 3, what does *not* follow from abstract nonsense is injectivity results like Theorem 5.9 or Proposition 5.15, which are proved, just as in Section 3, by careful counting.

5.1. *Construction of the sets with an action*

Definition 5.1. Suppose $(\mathcal{M}, \leqslant_{\mathcal{M}}) \in \mathsf{PoMonZ}$. A *poset with an $\mathcal{M}$-action* $\mathcal{X}$ is a tuple $(|\mathcal{X}|, \cdot_{\mathcal{X}}, \leqslant_{\mathcal{X}})$ of a *carrier set* $|\mathcal{X}|$, a partial order $\leqslant_{\mathcal{X}}$, and an action $x \cdot_{\mathcal{X}} m$ such that:

— If $x \leqslant_{\mathcal{X}} x'$ and $m \leqslant_{\mathcal{M}} m'$, then $x \cdot_{\mathcal{X}} m \leqslant x' \cdot_{\mathcal{X}} m'$ (*the order on $\mathcal{X}$ is compatible with that on $\mathcal{M}$*).
— $x \cdot_{\mathcal{X}} \mathrm{id}_{\mathcal{M}} = x$ and $(x \cdot_{\mathcal{X}} m) \cdot_{\mathcal{M}} m' = x \cdot_{\mathcal{X}} (m \circ_{\mathcal{M}} m')$ (*the action is a monoid action*).

We define a *poset with a $\mathcal{G}$ action* similarly for $(\mathcal{G}, \leqslant_{\mathcal{G}})$ a partially ordered group.

Suppose a poset with a compatible $\mathcal{M}$-action $\mathcal{X} = (|\mathcal{X}|, \cdot_{\mathcal{M}})$. The next definition shows how to extend this to a set with an $\mathcal{M}^{\pm}$-action.

Definition 5.2. We define $\leqslant$ to be the least transitive relation on $|\mathcal{X}| \times |\mathcal{M}^{\pm}|$ such that:

$$\frac{}{(x \cdot_{\mathcal{X}} m, g) \leqslant (x, +m \circ_{\mathcal{M}^{\pm}} g)} \textbf{(absorb)}$$

$$\frac{x \leqslant_{\mathcal{X}} x' \quad g \leqslant_{\mathcal{M}^{\pm}} g'}{(x, g) \leqslant_{\mathcal{X}^{\pm}} (x', g')} \textbf{(compat)}$$

$$\frac{(x, g) \leqslant (x', g') \quad (x', g') \leqslant (y, h)}{(x, g) \leqslant (y, h)} \textbf{(trans)}$$

Lemma 5.3. The ordering $\leqslant$ is *compatible*, in the sense that $(x, g) \leqslant (y, h)$ implies $(x, g \circ_{\mathcal{M}^{\pm}} g') \leqslant (y, h \circ_{\mathcal{M}^{\pm}} g')$.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Proof. The proof is by an easy induction on the rules in Definition 5.2. □

Remark 5.4. By assumption, $\leqslant_{\mathcal{X}}$ is reflexive, and by construction, so is $\leqslant_{\mathcal{X}^{\pm}}$, thus, by (**compat**), we know $\leqslant$ is too. We still need to prove that $\leqslant_{\mathcal{X}^{\pm}}$ is antisymmetric, that is, we need to show that $\mathcal{X}^{\pm}$ is a partially ordered set and not just a preordered set. This we do next, culminating with Theorem 5.9.

Lemma 5.5. If $(x, g) \leqslant (y, h)$, there exist $m_1, \ldots, m_k \in |\mathcal{M}|$ such that

$$+m_1 \ldots +m_k \leqslant_{\mathcal{M}^{\pm}} h \circ_{\mathcal{M}^{\pm}} g^{-1}.$$

Proof. The proof is by an easy induction on the rules in Definition 5.2. □

Lemma 5.6. If $(x, g) \leqslant (y, h)$, then $\iota(g) \leqslant \iota(h)$.

Proof. The proof is by induction on Definition 5.2:

— Rule (**absorb**):
It is a fact that $\iota(g) + 1 = \iota(m \circ_{\mathcal{M}^{\pm}} g)$.
— Rule (**compat**):
If $g \leqslant_{\mathcal{M}^{\pm}} h$, then by Lemma 3.16 (2), we have $\iota(g) \leqslant \iota(h)$.
— Rule (**trans**):
This is a property of numbers. □

Lemma 5.7. If $(x, g) \leqslant (y, h)$ and $(y, h) \leqslant (z, g)$, then $g = h$.

Proof. Suppose $(x, g) \leqslant (y, h)$ and $(y, h) \leqslant (z, g)$. By Lemma 5.6, we have $\iota(g) = \iota(h)$. Using this fact and Lemma 5.5, it follows that $g \leqslant_{\mathcal{M}^{\pm}} h$ and $h \leqslant_{\mathcal{M}^{\pm}} g$. Therefore, by Theorem 3.22, we have $g = h$. □

Lemma 5.8. If $(x, g) \leqslant (y, g)$, then $x \leqslant_{\mathcal{X}} y$.

Proof. The proof is by induction on the rules in Definition 5.2:

— Rule (**absorb**):
This rule cannot be used since $g = +m \circ_{\mathcal{M}^{\pm}} g$ is impossible because

$$\iota(g) \neq \iota(+m \circ_{\mathcal{M}^{\pm}} g) = 1 + \iota(g),$$

which contradicts Lemma 3.16 (2).
— Rule (**compat**):
This case is immediate.
— Rule (**trans**):
Suppose $(x, g) \leqslant (x', g')$ and $(x', g') \leqslant (y, g)$. By Lemma 5.7, we have $g' = g$. We then use the induction hypothesis and the transitivity of $\leqslant_{\mathcal{X}}$. □

Theorem 5.9. If $(x, g) \leqslant (y, h)$ and $(y, h) \leqslant (x, g)$, then $x = y$ and $g = h$.

Proof. The statement follows from Lemmas 5.7 and 5.8 and the fact that $\mathcal{X}$ and (by Theorem 3.22) $\mathcal{M}^{\pm}$ are partially ordered sets. □

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

5.2. *The natural adjunction*

Definition 5.10. Let $\mathcal{X}^{\pm}$ have underlying set $|\mathcal{X}| \times |\mathcal{M}^{\pm}|$ and action $\cdot_{\mathcal{X}^{\pm}}$ given by

$$(x, g)\cdot_{\mathcal{X}^{\pm}} h = (x, g \circ_{\mathcal{M}^{\pm}} h).$$

Definition 5.11. Suppose $\mathcal{M}$ is a partially ordered monoid. We write $\mathcal{M}$PoSetZ for the category such that:

— An object $\mathcal{X}$ is a poset with a compatible $\mathcal{M}$-action.
— An arrow $f : \mathcal{X} \to \mathcal{X}'$ is a function $f \in |\mathcal{X}| \to |\mathcal{X}'|$ such that:
  – $x \leqslant_{\mathcal{X}} x'$ implies $f(x) \leqslant_{\mathcal{X}'} f(x')$.
  – $f(x\cdot_{\mathcal{X}} m) \leqslant_{\mathcal{X}'} f(x)\cdot_{\mathcal{X}'} m$.

Suppose $\mathcal{G}$ is a partially ordered group. We write $\mathcal{G}$PoSet for the category such that:

— An object $\mathcal{Y}$ is a poset with a compatible $\mathcal{G}$-action.
— An arrow $f : \mathcal{Y} \to \mathcal{Y}'$ is a function $f \in |\mathcal{Y}| \to |\mathcal{Y}'|$ such that:
  – $y \leqslant_{\mathcal{Y}} y'$ implies $f(y) \leqslant_{\mathcal{Y}'} f(y')$.
  – $f(y\cdot_{\mathcal{Y}} g) = f(y)\cdot_{\mathcal{Y}'} g$.

Definition 5.12. Let $\mathsf{imag} : \mathcal{M}$PoSetZ $\to \mathcal{M}^{\pm}$PoSet have the following action:

— An object $\mathcal{X}$ maps to $\mathcal{X}^{\pm}$ (Definition 5.10).
— An arrow $f : \mathcal{X} \to \mathcal{X}'$ maps to $\mathsf{imag}(f) : \mathcal{X}^{\pm} \to (\mathcal{X}')^{\pm}$, which is defined by (x, g) maps to $(f(x), g)$.

Let $\mathsf{forg} : \mathcal{M}^{\pm}$PoSet $\to \mathcal{M}$PoSetZ have the following action:

— An object $\mathcal{Y}$ (which is a set with an $\mathcal{M}^{\pm}$ group action) maps to $\mathcal{Y}$ considered as a set with an $\mathcal{M}$ action, which we denote by $\mathcal{Y}_{\mathcal{M}}$.
— An arrow $f : \mathcal{Y} \to \mathcal{Y}' \in \mathcal{M}^{\pm}$PoSet maps to itself considered as a monoid homomorphism.

We will consider an example of Definition 5.12 in Section 6.

Lemma 5.13. imag from Definition 5.12 is a functor from $\mathcal{M}$PoSetZ to $\mathcal{M}^{\pm}$PoSet.

Proof. Suppose $f : \mathcal{X} \to \mathcal{Y} \in \mathcal{M}$PoSetZ. We have to check two things:

— $\mathsf{imag}(f)$ *is monotone*:
That is, we need to prove that if

$$(x, g) \leqslant (y, h)$$

(Definition 5.2) in $|\mathcal{X}| \times |\mathcal{M}^{\pm}|$, then

$$(f(x), g) \leqslant (f(y), h)$$

in $|\mathcal{X}'| \times |\mathcal{M}^{\pm}|$.
We do this by induction on the derivation of $(x, g) \leqslant (y, h)$: the case of (**absorb**) follows from the assumption that $f(x\cdot_{\mathcal{X}} m) \leqslant_{\mathcal{X}'} f(x)\cdot_{\mathcal{X}'} m$ and using (**compat**); the case of (**compat**) is a structural fact; transitivity also follows.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

— $\mathsf{imag}(f)$ *is a homomorphism*:
That is, we need to prove that

$$(f(x), g)\cdot_{\mathcal{X}^{\pm}} h = (f(x), g \circ_{\mathcal{M}^{\pm}} h).$$

But this follows directly from Definition 5.10. □

Proposition 5.14. $\mathsf{imag} \dashv \mathsf{forg} : \mathcal{M}^{\pm}\mathsf{PoSet} \to \mathcal{M}\mathsf{PoSetZ}$.

Proof. The unit $\eta_{\mathcal{X}}$ maps $x \in |\mathcal{X}|$ to $(x, ()) \in |(\mathcal{X}^{\pm})_{\mathcal{M}}|$. We need to check that

$$\eta_{\mathcal{X}}(x\cdot_{\mathcal{X}} m) \leqslant \eta_{\mathcal{X}}(x)\cdot_{(\mathcal{X}^{\pm})_{\mathcal{M}}} m.$$

We have

$$\begin{aligned}\eta_{\mathcal{X}}(x)\cdot_{(\mathcal{X}^{\pm})_{\mathcal{M}}} m &= (x, ())\cdot_{\mathcal{X}^{\pm}} +m \\ &= (x, +m) \\ &\geqslant (x\cdot_{\mathcal{X}} m) \\ &= \eta_{\mathcal{X}}(x\cdot_{\mathcal{X}} m).\end{aligned}$$

The counit $\epsilon_{\mathcal{Y}} : (\mathcal{Y}_{\mathcal{M}})^{\pm} \to \mathcal{Y}$ maps (x, g) to $x\cdot_{\mathcal{Y}} \epsilon_{\mathcal{M}^{\pm}}(g)$. We need to check that

$$\epsilon_{\mathcal{Y}}(x, g)\cdot_{\mathcal{Y}} h = \epsilon_{\mathcal{Y}}(x, g \circ_{\mathcal{M}^{\pm}} h).$$

We have

$$\begin{aligned}\epsilon_{\mathcal{Y}}(x, g)\cdot_{\mathcal{Y}} h &= (x\cdot_{\mathcal{Y}} g)\cdot_{\mathcal{Y}} h \\ &= x\cdot_{\mathcal{Y}} (g \circ_{\mathcal{M}^{\pm}} h) \\ &= \epsilon_{\mathcal{Y}}(x, g \circ_{\mathcal{M}^{\pm}} h).\end{aligned}$$

□

5.3. *More detailed analysis for the partial order*

In this section we will go deeper into the structure of objects in $\mathcal{M}^{\pm}\mathsf{PoSet}$ and ask the following question: if $x\cdot g \leqslant y\cdot h$ and $+m \leqslant g$, can $y\cdot h$ imitate this behaviour in some compatible sense?

This is Proposition 5.15 – those familiar with process calculi can think of this loosely as a *simulation* result in the sense of, for example, Milner (1999, Definition 3.3).

Proposition 5.15. Suppose $\mathcal{X}$ is a set with an $\mathcal{M}$-action and that $(x, g) \leqslant (y, h)$. Then:
(1) If $+m \leqslant_{\mathcal{M}^{\pm}} g$, there exists $n \in |\mathcal{M}|$ such that $+n \leqslant_{\mathcal{M}^{\pm}} h$ and $x\cdot_{\mathcal{M}} m \leqslant_{\mathcal{X}} y\cdot_{\mathcal{M}} n$.
(2) If $() \leqslant_{\mathcal{M}^{\pm}} g$, then either:
— $() \leqslant_{\mathcal{M}^{\pm}} h$ and $x \leqslant_{\mathcal{X}} y$; or
— there exists $n \in |\mathcal{M}|$ such that $+n \leqslant_{\mathcal{M}^{\pm}} h$ and $x \leqslant_{\mathcal{X}} y\cdot_{\mathcal{M}} n$.

Proof.
(1) The proof is by induction on the derivation of $\leqslant$:
— Rule (**absorb**):
By this rule,

$$(x\cdot_{\mathcal{X}} m, g) \leqslant (x, +m \circ_{\mathcal{M}^{\pm}} g).$$

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Suppose $+n \leqslant_{\mathcal{M}^\pm} g$. Then

$$+(m \circ_{\mathcal{M}} n) \leqslant_{\mathcal{M}^\pm} +m{+}n \leqslant_{\mathcal{M}^\pm} +m \circ_{\mathcal{M}^\pm} g,$$

and it is a fact that

$$(x \cdot_{\mathcal{M}} m) \cdot_{\mathcal{M}} n = x \cdot_{\mathcal{M}} (m \circ_{\mathcal{M}} n).$$

It then follows that

$$(x \cdot_{\mathcal{M}} m) \cdot_{\mathcal{X}} n \leqslant_{\mathcal{M}} x \cdot_{\mathcal{M}} (m \circ_{\mathcal{M}} n).$$

— Rule (**compat**):
Suppose $(x, g) \leqslant (x', g')$ because $x \leqslant_{\mathcal{X}} x'$ and $g \leqslant_{\mathcal{M}^\pm} g'$. Suppose also that $+m \leqslant_{\mathcal{M}^\pm} g$. Then, we also have $+m \leqslant_{\mathcal{M}^\pm} g'$ and $x \cdot_{\mathcal{M}} m \leqslant_{\mathcal{X}} x' \cdot_{\mathcal{M}} m$.

— Rule (**trans**):
Suppose $(x, g) \leqslant (y, h)$ because $(x, g) \leqslant (x', g')$ and $(x', g') \leqslant (y, h)$. Suppose also that $+m \leqslant_{\mathcal{M}^\pm} g$.
By the induction hypothesis, there exists $+m' \leqslant_{\mathcal{M}^\pm} g'$ such that $x \cdot_{\mathcal{X}} m \leqslant_{\mathcal{X}} x' \cdot_{\mathcal{X}} m'$. Also by the induction hypothesis, there exists $+n \leqslant_{\mathcal{M}^\pm} h$ such that $x' \cdot_{\mathcal{X}} m' \leqslant_{\mathcal{X}} y \cdot_{\mathcal{X}} n$. The result then follows by the transitivity of $\leqslant_{\mathcal{X}}$.

(2) This part is also proved by induction:

— Rule (**absorb**):
By this rule,

$$(x \cdot_{\mathcal{X}} m, g) \leqslant (x, +m \circ_{\mathcal{M}^\pm} g).$$

Suppose $() \leqslant_{\mathcal{M}^\pm} g$. Then it follows that

$$+m \leqslant_{\mathcal{M}^\pm} +m \circ_{\mathcal{M}^\pm} g$$

and it is a fact that

$$x \cdot_{\mathcal{M}} m = x \cdot_{\mathcal{M}} m.$$

If $m \leqslant_{\mathcal{M}^\pm} g$, we proceed as in the first part of the lemma.

— Rule (**compat**):
Suppose $(x, g) \leqslant (x', g')$ because $x \leqslant_{\mathcal{X}} x'$ and $g \leqslant_{\mathcal{M}^\pm} g'$. Suppose also that $() \leqslant_{\mathcal{M}^\pm} g$. Then we also have $() \leqslant_{\mathcal{M}^\pm} g'$ and $x \leqslant_{\mathcal{X}} x'$. If $m \leqslant_{\mathcal{M}^\pm} g$, we proceed as in the first part of the lemma.

— Rule (**trans**):
This case is similar. □

6. An example: substitutions

Consider some first-order syntax $\mathcal{L}$ (for example, $r ::= a \mid \mathsf{f}(r, \ldots, r)$ for some term-formers f and variables a), and consider substitution $r[a \mapsto s]$ on it (Baader and Nipkow 1998). We write Σ for the set of substitutions on $\mathcal{L}$ with functional composition.

Let σ range over elements of Σ. Substitutions form a submonoid of the function space on $\mathcal{L}$, that is, we can view Σ as a subset $\Sigma \subseteq \mathcal{L}^{\mathcal{L}}$ that is closed under functional

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

composition (so if $\sigma, \sigma' \in \Sigma$, then $\sigma \circ \sigma' \in \Sigma$ and it maps a to $(\sigma(a))\sigma'$) and contains the identity function id.

We consider Σ to be trivially partially ordered so that $\sigma \leqslant \sigma'$ if and only if $\sigma = \sigma'$. Similarly, we consider $\mathcal{L}$ to be trivially partially ordered.

Then we can form $\Sigma^{\pm}$, the imaginary group of substitutions, and $\mathcal{L}^{\pm}$, the set of terms lazily acted on by substitutions.

What do these look like?

6.1. *Substitutions just on variables*

Consider the special case where there are no term-formers so that $\mathcal{L}$ is just the set of variables $\{a, b, c, \dots\}$. For brevity, we write $\bullet$ for $\cdot_{\mathcal{L}^{\pm}}$. Then:

— $a \bullet [a \mapsto b] \geqslant b$.
— $b \bullet [a \mapsto b] \geqslant b$.
— $a \bullet [a \mapsto b]^{-1} \geqslant a$.
  We deduce this by taking $a \bullet [a \mapsto b] \geqslant b$ and applying $[a \mapsto b]^{-1}$ to both sides.
— $a \bullet [a \mapsto b]^{-1} \geqslant b$
  This is deduced similarly.

In view of the inequalities above, we can think of $a \bullet [a \mapsto b]^{-1}$ as being 'like the unordered pair $\{a, b\}$'. However, this is only an analogy since $a \bullet [a \mapsto b]^{-1}$ displays behaviour that is *not* like the behaviour of $\{a, b\}$. In particular, $a \bullet [a \mapsto b]^{-1} \neq b \bullet [b \mapsto a]^{-1}$, though (by symmetry) both are 'like $\{a, b\}$'.

6.2. *Substitutions on terms*

Now suppose there are term formers. Then $r \bullet [a \mapsto s] \geqslant r[s/a]$ (where $r[s/a]$ is the term obtained by really replacing a by s in r). Also, $r \bullet [a \mapsto s]^{-1} \geqslant r'$ for any r' such that $r'[s/a] = r$.

Now consider a rewrite rule $R = (s_1 \to s_2)$ in the sense of first-order rewriting (Baader and Nipkow 1998). Then, if r rewrites to r' with rule R, we have

$$(r \bullet [a \mapsto s_1]^{-1}) \bullet [a \mapsto s_2] \geqslant r'.$$

For instance, given 0-ary term-formers 1 and 2 (that is, constant symbols) it is easy to verify that

$$(1 \bullet [a \mapsto 1]^{-1}) \bullet [a \mapsto 2] \geqslant 2.$$

This encodes, in a certain sense, the rewrite rule $1 \to 2$.

By combining substitutions and their lazy inverses, it is not hard to encode arbitrary first-order rewriting in $\mathcal{L}^{\pm}$. As an easy corollary, $\leqslant_{\mathcal{L}^{\pm}}$ is undecidable – to see this, it suffices to write out a rewrite system for combinators S and K (Hindley and Seldin 2008, Section 2).

So equality in Σ is decidable (we just unfold the substitution), but equality in $\Sigma^{\pm}$ is not (given even two term formers, lazy substitutions can encode undecidable computations).

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

7. Conclusions

We set off wanting to inject every monoid homomorphically into a group. This is impossible, but we can do something similar that perhaps captures the spirit of this: we can inject every monoid into a partially ordered group.

There is an appealing computational interpretation for this, in which the groups are interpreted as 'executable plans' and the monoids as 'computation steps'.

In order to expand this observation to a categorical adjunction, we discover, somewhat surprisingly, an adjunction between the category of partially ordered monoids and lazy homomorphisms and the category of partially ordered groups and homomorphisms (we do not need the latter to be lazy).

We also obtain similar results for sets with an action. This is motivated by the spirit of Cayley's theorem, which allows every monoid and group to be represented as the action on a set.† These also display the lazy homomorphism/homomorphism structure.

The technical highlights of this paper are in the very specific design of the definitions, notably Definitions 3.8, 3.12 and 5.2, and in the proofs of Theorems 3.22, 3.24 and 5.9. The proofs involve some detailed combinatorial arguments, which rely on technical lemmas, the choice of which is non-evident: specifically, these are Lemma 3.14 to Proposition 3.19 and Lemma 5.5 to Lemma 5.8.‡

In conclusion, let us recall why this is interesting. Monoids are an abstract model of computation, which preserves only the idea that there are some operations (including a 'do nothing' identity operation), and operations can be composed. Groups are a similar model, but for reversible operations.

We have given a sense in which any monoid can systematically be injected into a universe of reversible operations. In that sense, the results in this paper give a new map for moving between irreversible and reversible computation.

Related work

To the best of our knowledge the connection we have observed between partially ordered monoids and lazy morphisms and partially ordered groups and morphisms is new. It sheds quite an interesting light on these structures and suggests a new way of (and indeed, a new motivation for) looking at them, a way that comes from the theory of logic and computation.

Conditions for being able to embed (homomorphically inject) a monoid in a group were considered by Mal'cev (Mal'cev 1937; Mal'cev 1939) and Lambek (Lambek 1951); these are infinite sets of Horn clauses that characterise the class of monoids that inject into some group. This was later generalised to categories and groupoids by Johnstone

† Some interesting historical comments on the attribution of Cayley's name to this family of results can be found in Nummela (1980) – Cayley certainly did not prove it for monoids, but 'Cayley's theorem' is the standard name for this kind of result.

‡ Framing the question and then finding the proof structure took us two years from start to finish.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

(Johnstone 2008). It should be clear that we are solving a distinct problem: we take all monoids, but weaken 'homomorphism/functor' to '*lazy* homomorphism/functor'.[†]

An inverse semigroup has a natural partial order given by $x \leqslant y$ when $x = ye$ for some idempotent e. A *prehomomorphism* of inverse semigroups is a function θ such that $\theta(xy) \leqslant \theta(x)\theta(y)$. Finally, an *inductive groupoid* is a groupoid whose hom-sets form a meet semilattice and an *ordered functor* on them is monotone on the ordering of the hom-sets. This extends by the *Ehresmann–Schein–Nambooripad* Theorem to an isomorphism between the categories of inverse semigroups and prehomomorphisms, and inductive groupoids. For full definitions and proofs, see Lawson (1998). This theorem expresses an equivalence between two ways of capturing the notion of 'partial symmetry ordered by the size of the domain': one notion expressed by semigroups (untyped) and the other by inductive groupoids (typed). There is some similarity here to our results, but they are not the same. Superficially, we use monoids (though we could drop the identity) and we map to partially ordered groups (though we could generalise to categories and locally posetal groupoids). A deeper difference is that in our adjunction the unit is not an identity map: we do not build an isomorphism. That is, our main result is not an equivalence, it is an embedding. It remains to be seen whether there is a variation of our result that is more aggressive in quotienting the partially ordered group when mapping back to monoids, which would allow us to recover something more like the above equivalence theorem as a special case.

A number of computational systems that bear reversibility in mind have been studied, and have been motivated by three facts:

— Physical systems are time-reversible, so reversible computation is, in fact, the computation of the physical world as we know it. Most recently, this has been relevant in the design of DNA circuits, where the possibility of a computation running 'backwards' in some chemical soup in the test-tube must be taken into account.
— A corollary of *Landauer's principle* is that reversible computation dissipates less heat (that is, irreversibility is caused by increasing entropy, which corresponds to erasing information). Thus, from the point of view of designing circuits, it is relevant to keep track of where information is lost and heat is generated.
— Quantum computation has aroused much interest, and here the basic model of computation is again reversible (with irreversibility caused by measurement).

Some relevant publications in these areas are as follows:

— Toffoli (1980) introduced the *Toffoli gate*, which is a universal reversible logic gate (see also Storme *et al.* (1999), which studies universal reversible logic gates in general).[‡]
— Bennet's classic paper Bennett (1973) studied the notion of a reversible Turing machine.
— For reversible computation in biologically inspired computing, see Danos and Krivine (2004) and Phillips and Ulidowski (2007).

[†] One could study our construction in the special case where the Mal'cev or Lambek conditions are satisfied. If so, then something easier would be sufficient to begin with since our constructions are orthogonal to whether the monoid is commutative: a *commutative* monoid is embeddable in a group if and only if it satisfies *cancellation*, that is, $xy = xz \Rightarrow y = z$.

[‡] One can build any circuit out of NAND gates, so a NAND gate is a universal logic gate. Similarly, one can build any *reversible* circuit out of Toffoli gates.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

— For quantum computation, good places to start reading are Spiller (1998) and Kitaev *et al.* (2002, Section 7).
— The connection between reversibility, entropy and computation is discussed in Keyes and Landauer (1970) and Landauer (1961).

All of these papers are about concrete reversible computational structures, but our concern is more abstract. We consider how, *given* a possibly irreversible system represented abstractly as a monoid, we can generate a corresponding reversible system represented as a group, where the correspondence is natural in some way.

So, on the topic of reversible computation, we hope that the constructions in this paper are interesting theoretical tools, but it should be clear that we do not pretend to have provided evidence that they are directly computationally useful. In particular, we have not studied the representation of this group, aside from that given by its construction. We note that determining the partial order in the group may be strictly more complex than determining equality in the monoid, as Section 6.2 illustrates, where we observe that $\leqslant_{\mathcal{M}^{\pm}}$ may be undecidable even if $\leqslant_{\mathcal{M}}$ is decidable.

Of course, equality of most non-trivial computational systems is undecidable, so this is to be expected, and is not the end of the world. We can reasonably hope that the constructions of this paper might be useful as a framework with which to design invertible programming systems: in principle, the methodology would be to design a non-invertible programming system and then apply $(\text{-})^{\pm}$ (of course there is more to a programming language than that, but the underlying idea stands).

Future work

Briefly, we suggest the following areas for possible future work and applications:

— *Generalisation to weaker monoid-like structures*:
We noted in Section 4 that the proofs in this paper seem quite robust, and it should be possible to generalise the ideas in this paper to the cases of, for example, commutative monoids, semigroups and applicative structures. Similarly, a monoid is a special case of a category with one element. It is natural to take the results of this paper and extend them to categories. Thus, given an input category, we can generate a category (whose hom-sets are partially ordered) in which every arrow is invertible.
— *Algorithms enriched with inverses*:
Substitutions on syntax form a monoid. Therefore, it is possible to consider syntax enriched with invertible substitutions, as hinted at in Section 6.
A little more work is needed to enrich syntax with invertible substitutions, but it can be done: we need to build inductive definitions that include inverses of substitutions, but a corollary of Theorem 3.30 is that the 'lazy' functor preserves colimits, so it preserves least upper bounds of countably ascending chains, and thus, by general nonsense, Theorem 3.30 tells us that we can indeed interleave $(\text{-})^{\pm}$ with the construction of an inductive datatype.
We still need to study the properties of such a syntax, for example, to determine whether unification has most general unifiers. We believe that it does, because substitutions can

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

be inverted.[†] An algorithm would be theoretically interesting, though not necessarily practical since equality might not be decidable.

Of course, higher-order unification (Dowek 2001) is also not decidable, but that does *not* stop useful heuristics from being constructed that seem to work well in practice – see, for example, Huet (1975). So syntax with invertible substitutions is an interesting topic, and it might even be useful.

— *Computational models*:

Taking up the comments in Section 7 on related work, we still need to investigate the application of our construction to monoids or similar structures that are specifically designed to model computation: such as a monoid of functions of the untyped λ-calculus under functional composition; or a monoid of processes under parallel composition (Milner 1999; Danos and Krivine 2004); or the applicative structure of a model of the untyped λ-calculus.

— *Generalisation to richer structures*:

Suppose the monoid has extra structure – for example, the algebraic study of λ-models has met with some success (Manzonetto and Salibra 2010). This structure transfers to the lazy case.

We still need to study how algebraic properties of the monoid carry across. For example, how does the 'algebra' of $\mathcal{M}$ compare with a corresponding '(lazy) algebra' of $\mathcal{M}^{\pm}$? Also, if the partial order on the monoid is part of a domain (Abramsky and Jung 1994), what are the properties of the partial order on its free group?

— *Lattice properties of the free group*:

On a related note, we can ask: if $\mathcal{M}$ is a partial order with further structure, such as being a lattice, what further structure will $\mathcal{M}^{\pm}$ have? For instance, the elements of a Boolean algebra B form a commutative monoid under conjunction and a lattice under implication ($x \leqslant y$ when $x \wedge y = x$), so it is interesting to ask what kind of logical entity $B^{\pm}$ would be (possibly the commutative version, as discussed in Section 4). This seems related, at least in spirit, to the *anti-formulae* (formulae that when added to a database *delete* an item from it) considered in Gabbay *et al.* (2004). Is there some sense in which the 'logic' of B extends to a 'logic' of $B^{\pm}$?

Last but not least, the proofs in this paper are unambiguous, but a higher-level explanation of why they work is currently lacking. That is, the counting arguments in this paper work, but they do so 'by magic': the existence of some general conceptual framework of which the counting arguments are an instance is an open question.

[†] One of the original motivations for this work comes from the idea of *nominal unification* (Urban *et al.* 2004). This is based on permutations, which are invertible, and because of that, nominal unification has most general solutions. The question the first author asked that motivated this line of research was: *If substitutions also had inverses, would something like higher-order unification then have most general solutions?* Preliminary calculations suggest that the answer is *yes*.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Acknowledgments

We are very grateful to Phil Scott and to an anonymous referee, without whom this paper would not have attained its current form.

References

Abramsky, S. and Jung, A. (1994) Domain theory. In: Abramsky, S., Gabbay, D. M. and Maibaum, T. S. E. (eds.) *Handbook of Logic in Computer Science: Semantic Structures*, volume 3, Oxford University Press 1–168.

Baader, F. and Nipkow, T. (1998) *Term Rewriting and All That*, Cambridge University Press.

Bennett, C. H. (1973) Logical reversibility of computation. *IBM journal of Research and Development* **17** (6) 525–532.

Cheng, E. and Lauda, A. (2004) Higher-dimensional categories: an illustrated guide book. Notes prepared for IMA workshop.

Danos, V. and Krivine, J. (2004) Reversible communicating systems. In: Gardner, P. and Yoshida, N. (eds.) Proceedings of the 15th International Conference on Concurrency Theory (CONCUR 2004). *Springer-Verlag Lecture Notes in Computer Science* **3170** 292–307.

Dowek, G. (2001) Higher-order unification and matching. In: Robinson, A. and Voronkov, A. (eds.) *Handbook of automated reasoning*, volume 2, Elsevier 1009–1062.

Gabbay, D. M., Rodrigues, O. and Woods, J. (2004) Belief contraction, anti-formulae and resource overdraft: Part II, deletion in resource unbounded logics. In: Rahman, S., Symons, J., Gabbay, D. M. and van Bendegem, J. P. *Logic, Epistemology, and the Unity of Science*, volume 1, Springer-Verlag 291–326.

Glass, A. M. W. (1999) *Partially ordered groups*, Series in Algebra volume 7, World Scientific.

Herlihy, M. and Moss, J. E. B. (1993) Transactional memory: Architectural support for lock-free data structures. *ACM SIGARCH Computer Architecture News* **21** (2) 289–300.

Hindley, J. R. and Seldin, J. P. (2008) *Lambda-Calculus and Combinators, An Introduction*, 2nd edition, Cambridge University Press.

Huet, G. (1975) A unification algorithm for typed λ-calculus. *Theoretical Computer Science* **1** 27–57.

Johnstone, P. T. (2008) On embedding categories in groupoids. *Mathematical Proceedings of the Cambridge Philosophical Society* **145** 273–294.

Keyes, R. W. and Landauer, R. (1970) Minimal energy dissipation in logic. *IBM Journal of Research and Development* **14** 153–157.

Kitaev, A. Y., Shen, A. H. and Vyalyi, M. N. (2002) *Classical and quantum computation*, Graduate Studies in Mathematics volume 47, American Mathematical Society. (Translated from the Russian by Lester J. Senechal.)

Lambek, J. (1951) The immersibility of a semigroup into a group. *Canadian Journal of Mathematics* **3** 3443.

Landauer, R. (1961) Irreversibility and heat generation in the computational process. *IBM Journal of Research and Development* **5** 183–191.

Lawson, M. V. (1998) *Inverse semigroups: the theory of partial symmetries*, World Scientific.

Leinster, T. (2004) *Higher Operads, Higher Categories*, London Mathematical Society Lecture Note Series volume 298, Cambridge University Press.

Liang, C. and Miller, D. (2009) Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science* **410** (46) 4747–4768.

Mal'cev, A. I. (1937) On the immersion of an algebraic ring into a field. *Mathematische Annalen* **113** 686–691.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press

Mal'cev, A. I. (1939) On the embedding of associative systems in groups I. *Matematicheskii Sbornik* **6** (48) (2) 331–336. (In German: Über die Einbettung von assoziativen Systemen in Gruppen.)

Manzonetto, G. and Salibra, A. (2010) Applying universal algebra to lambda calculus. *Journal of Logic and computation* **20** (4) 877–915.

Milner, R. (1999) *Communicating and mobile systems: the pi-calculus*, Cambridge University Press.

Nummela, E. (1980) Cayley's theorem for topological groups. *American Mathematical Monthly* **87** (3) 202–203.

Phillips, I. and Ulidowski, I. (2007) Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming* **73** (1–2) 70–96.

Restall, G. (1999) *An Introduction to Substructural Logics*, Routledge.

Spiller, T. P. (1998) Basic elements of quantum information technology. In: Lo, H. K., Spiller, T. and Popescu, S. (eds.) *Introduction to Quantum Computation and Information*, World Scientific 1–28.

Storme, L., Vos, A. D. and Jacobs, G. (1999) Group theoretical aspects of reversible logic gates. *Journal of Universal Computer Science (J.UCS)* **5** (5) 307–321.

Toffoli, T. (1980) Reversible computing. In: de Bakker, J. and van Leeuwen, J. (eds.) Proceedings of the 7th Colloquium on Automata, Languages and Programming. *Springer-Verlag Lecture Notes in Computer Science* **85** 632–644.

Urban, C., Pitts, A. M. and Gabbay, M. J. (2004) Nominal Unification. *Theoretical Computer Science* **323** (1–3) 473–497.

https://doi.org/10.1017/S0960129512000849 Published online by Cambridge University Press