Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 225 (2009) 441–456
www.elsevier.com/locate/entcs

Reconstruction of Partial Orders and List Representation as Random Structures

Thierry Vallée[1] and Joseph Manning[1,2]

Centre for Efficiency-Oriented Languages (CEOL)
Department of Computer Science
University College Cork
Ireland
tv1@cs.ucc.ie / manning@cs.ucc.ie

Abstract

$\mathcal{MOQA}$ is a new programming language with the unique property that the average running time of its programs can be (semi-)automatically deduced in a modular way by a static analysis of the program code. This is based on the fact that to each $\mathcal{MOQA}$ action there corresponds an operation on partial orders, which associates with each partial order a sequence of partial orders.

All programs in $\mathcal{MOQA}$ use a special data structure and an associated suite of operations. This data structure consists of a pair $((X, \sqsubseteq), \ell)$, where $(X, \sqsubseteq)$ is a finite poset and $\ell : X \mapsto \mathcal{L}$ is a bijection from X to a totally ordered set of labels $\mathcal{L}$, satisfying the condition $x \sqsubseteq y \implies \ell(x) \leq \ell(y)$. Central to the analysis of $\mathcal{MOQA}$ programs is the set of all such pairs for a given $(X, \sqsubseteq)$ and a given $\mathcal{L}$; this set is called a *random structure*. The corresponding set of order-preserving bijections is called a *random structuring*.

This paper establishes a fundamental equivalence by showing that each poset is uniquely characterised by its associated random structuring, and derives algorithms to reconstruct a poset from its random structuring and to test if an arbitrary set of bijections forms a random structuring. It then develops some consequences of the previous results, and in particular a first characterisation of cardinalities of random structurings. These results open the way to the study of the representation of recursive sets of lists as random structures. This study is closely related to the implementation of list manipulation algorithms in $\mathcal{MOQA}$.

Keywords: posets, random structures, labelings, average running time

1 Introduction

This paper investigates certain basic aspects of $\mathcal{MOQA}$ ($\mathcal{MO}$odular $\mathcal{Q}$uantitative $\mathcal{A}$nalysis), a new programming language designed by Michel Schellekens at CEOL (Centre for Efficiency-Oriented Languages), and presented earlier in [15,16,14,17,19]. The most distinguishing feature of $\mathcal{MOQA}$ is the feasibility of determining the

[1] Research supported by Science Foundation Ireland under Grant 02/IN.1/181
[2] Research supported by Science Foundation Ireland under Grant 05/RFP/CMS0044

doi:10.1016/j.entcs.2008.12.091

average-case running time of its programs, merely by conducting an automated static analysis of the program code, and the ability to do so in a modular fashion.

Just as LISP is based on lists and operations that manipulate them, $\mathcal{MOQA}$ programs are likewise based on a special data structure and an associated suite of operations. This data structure consists of a pair $((X, \sqsubseteq), \ell)$, where X is a finite set, $\sqsubseteq$ is a partial order on X, and $\ell : X \mapsto \mathcal{L}$ is a bijection from X to a totally ordered set of labels $\mathcal{L}$, satisfying the condition $x \sqsubseteq y \implies \ell(x) \leq \ell(y)$. Such a pair is called a *labeled partial ordered (lpo)*, and the bijection ℓ a *labeling*.

A fundamental concept underlying the (semi-)automated average-case analysis of $\mathcal{MOQA}$ programs is the set of *all* lpo's for a given poset $(X, \sqsubseteq)$ and a given set of labels $\mathcal{L}$; this set is called a *random structure*. We introduce here the notion of a *random structuring* as the set of all *labelings* from a given poset $(X, \sqsubseteq)$ and set of labels $\mathcal{L}$. Such a set is denoted by $R_{\mathcal{L}}(X, \sqsubseteq)$.

An lpo can be seen as the synthesis of two points of view. The first is that of programmers who, while writing list manipulation programs, deal with lists of values from a totally-ordered set (represented by the labelings) and are interested in the semantics (formalised or not) of their programs. The second is that of complexity analysts, who deal with random structures. Indeed, when the inputs of a program form a random structure, the special design of $\mathcal{MOQA}$ operations allows for control on the distribution of the program outputs. Moreover, it guarantees that the outputs themselves form a sequence of random structures. This strong property of $\mathcal{MOQA}$ operations allows an easy way to determine the average time of programs on random structures. In this context, it becomes interesting to explore the notion of random structuring and in particular to try and characterise to what extent it captures the notion of recursive sets of lists. A first step towards this characterisation is Theorem 5.5 below, where the possible cardinalities of finite random structurings are characterised.

In the sequel, we study the notion of random structuring for a class of poset called *height-stratifiable posets (hs-posets)*, a class which includes all finite posets, and some which are infinite. These are introduced in Section 2. Then, in Section 3, we present the *Reconstruction Problem*, that is, how to build a partial order starting from its labelings, and we show that it suffices to first establish an *Equivalence Theorem* which establishes that the notion of poset and random structuring are equivalent, that is, that two partial orders are equal iff their sets of labelings are equal. In Section 4, we prove this theorem, using the fact that every *well-founded locally finite* poset is uniquely determined by its set of labelings. This proof relies on the notion of *height-canonical labeling* and uses a technique of *locally height-canonical labeling*. A pair of reconstruction algorithms are also presented. Section 5 is then devoted to additional results and in particular to a characterisation of cardinalities of random structurings.

We conclude this introduction with some remarks concerning an equivalent representation of labelings as linear extensions of posets. A *linear extension* of $(X, \sqsubseteq)$ is a poset $(X, \sqsubseteq')$ such that $\sqsubseteq \subseteq \sqsubseteq'$ and $\sqsubseteq'$ is total. Since the seminal works of Szpilrajn [18] and Dushnik & Miller [8], linear extensions have received considerable

attention [1,2,3,4,6,7,12,13]. This attention focused in particular on the problems of generating all the linears extension of a poset, as well as the problem of finding its *dimension* (the minimal number of linear extensions needed to generate the poset). While these questions are certainly interesting, they are not our focus here. Moreover, they appeared in a context which seems quite different than that of $\mathcal{MOQA}$, where labelings are part of the lpo data structure. Thus, labelings seem the most natural and direct way to express and answer the particular questions raised by the development of the language. Finally, note that our Equivalence Theorem, which uses the labeling formulation, could itself be derived from an equivalent result of Szpilrajn in [18] using the linear extension theorem.

2 Preliminaries

2.1 Height-Stratifiable Posets

In this section, we recall some basic definitions concerning *partial orders*, and we then introduce the notion of *height-stratifiable* posets. We denote by $\mathbb{N}$ the set of natural numbers $\{0, 1, 2, 3, \dots\}$.

A *partially ordered set* (poset) is a pair $(X, \sqsubseteq)$ where X is a set and $\sqsubseteq$ is a *partial order (p.o.)* on X, that is, a reflexive, antisymmetric and transitive relation on $X \times X$. To each partial order $\sqsubseteq$, we associate a *strict partial order (s.p.o.)* $\sqsubset$ defined by $x \sqsubset y \Leftrightarrow x \sqsubseteq y \wedge x \neq y$. Conversely, every antisymmetric and transitive relation $\sqsubset$ on a set X induces a p.o. $\sqsubseteq$ (the reflexive closure of $\sqsubset$). In this article, to define a p.o., we often content ourselves with defining the corresponding s.p.o. We use "$\sqsubseteq$" to denote the p.o., and "$\sqsubset$" to denote the corresponding s.p.o.

We now introduce the usual definitions of the ceiling, floor, ascending tree, and descending tree of a (set of) node(s).

Definition 2.1 Let $(X, \sqsubseteq)$ be a poset, and $Y \subseteq X$. Then the following sets are, respectively, the *ceiling, floor, ascending tree,* and *descending tree of* Y *in* $(X, \sqsubseteq)$:

- $\lceil Y \rceil_{\sqsubseteq} = \{x \in X : \exists y \in Y, y \sqsubset x \wedge \forall z \in X\, (y \sqsubseteq z \sqsubseteq x \Rightarrow z = x \vee z = y)\}$
- $\lfloor Y \rfloor_{\sqsubseteq} = \{x \in X : \exists y \in Y, x \sqsubset y \wedge \forall z \in X\, (x \sqsubseteq z \sqsubseteq y \Rightarrow z = x \vee z = y)\}$
- $\Uparrow_{\sqsubseteq} Y = \{x \in X : \exists y \in Y, y \sqsubseteq x\}$
- $\Downarrow_{\sqsubseteq} Y = \{x \in X : \exists y \in Y, x \sqsubseteq y\}$

Notation: We will often use the above notations without mentioning the order. For instance, $\lceil Y \rceil_{\sqsubseteq}$ will simply be written as $\lceil Y \rceil$. Moreover, when $Y = \{x\}$, the sets defined above will simply be denoted by $\lceil x \rceil$, $\lfloor x \rfloor$, $\Uparrow x$, $\Downarrow x$. Observe that $(\Uparrow x \cap \Downarrow x) = \{x\}$, for every $x \in X$, and that $x \in \Uparrow z \Leftrightarrow z \in \Downarrow x$.

Definition 2.2 A *path from* x_0 *to* x_n in the poset $(X, \sqsubseteq)$ is a finite sequence $(x_0, \dots, x_n)$ of elements of X such that, for each $0 \leq i \leq n-1$, we have $x_i \sqsubset x_{i+1}$. The *length* of the path $x_0, \dots, x_n$ is n. In particular, (x_0) is the only path from x_0 to itself, and has length 0.

We now recall the usual notions of the maximal and minimal elements of a poset.

Definition 2.3 Let $(X, \sqsubseteq)$ be a poset. Define, respectively, the set of *maximal elements* and the set of *minimal elements* of $(X, \sqsubseteq)$ as:

- $M(X, \sqsubseteq) = \{x \in X : \forall y \in X,\ x \sqsubseteq y \Rightarrow y = x\}$
- $m(X, \sqsubseteq) = \{x \in X : \forall y \in X,\ y \sqsubseteq x \Rightarrow y = x\}$

Notation: For any set X, let $\Delta_X = \{(x, x) : x \in X\}$.

Remark 2.4 Note the following two special cases:

- $M(X, \sqsubseteq) = m(X, \sqsubseteq) = X \Leftrightarrow \sqsubseteq = \Delta_X$.
- $M(X, \sqsubseteq) = m(X, \sqsubseteq) = \emptyset \Leftrightarrow X = \emptyset$, for X finite.

We now introduce the posets which we will focus on in the sequel.

Definition 2.5 The poset $(X, \sqsubseteq)$ is:

- *locally finite* if, for every $x \in X$, both $\lceil x \rceil$ and $\lfloor x \rfloor$ are finite.
- *finitely well-founded* if $X = \Uparrow m(X, \sqsubseteq)$, with $m(X, \sqsubseteq)$ finite.
- *height-stratifiable (hs-poset)* if it is locally finite and finitely well-founded.

Clearly, every finite poset is height-stratifiable. The following facts are also clear:

Fact 2.6 *Let $(X, \sqsubseteq)$ be a hs-poset. Then for every $x \in X$:*

- $\Downarrow_{\sqsubseteq} x \cap m(X, \sqsubseteq)$ *is a non-empty finite set equal to $\{x\}$ if x is minimal.*
- *There are finitely many paths from any minimal element to x.*

Fact 2.6 justifies the following definition.

Definition 2.7 Let $(X, \sqsubseteq)$ be a hs-poset. Then:

- For each $x \in X$, define the *(finite) height of x w.r.t. $\sqsubseteq$* as the maximal length of any path from a minimal element to x. In particular, the height of each minimal element of $(X, \sqsubseteq)$ is 0. We will write $height_{\sqsubseteq}(x)$, or simply $height(x)$, for the height of $x \in X$ relative to the order $\sqsubseteq$.
- For $X \neq \emptyset$, the *height of the poset* $(X, \sqsubseteq)$ is the maximal length of a path from a minimal element to an element of X; this value may be $+\infty$.
- The set $heights_{\sqsubseteq}(X)$, or simply $heights(X)$, is the set of all heights of elements of X relative to $\sqsubseteq$.

In the sequel, we suppose that $+\infty$ is greater than any positive integer, that is, we suppose that $\mathbb{N} \cup \{+\infty\}$ is totally ordered with greatest element $+\infty$.

The next fact follows easily from the definition of the height of an element:

Fact 2.8 *Let $(X, \sqsubseteq)$ be a non-empty hs-poset, and $x, y \in X$. Then:*

(a) *If $x \sqsubset y$, then $height(x) < height(y)$. In particular, if $x \in \lfloor y \rfloor$ ($\equiv y \in \lceil x \rceil$), then $height(x) < height(y)$.*

(b) *If $height(y) = i+1$, then, for every path $(x_0, \ldots, x_i, y)$ from a minimal element x_0 to y and every $0 \leq j \leq i$, we have $height(x_j) = j$.*

(c) *If* $height(y) = i + 1$, *then there exists an* $x \in X$ *such that* $height(x) = i$ *and* $y \in \lceil x \rceil$.

Lemma 2.9 *Let* $(X, \sqsubseteq)$ *be a non-empty hs-poset with height* h. *Then:*

(a) $H_i = \{x \in X : height(x) = i\}$ *is finite and non-empty, for each* $0 \leq i \leq h$.

(b) *The family* $(H_i)_{i \in heights(X)}$ *forms a partition of* X.

(c) h *is finite iff* X *is finite.*

Proof. To establish part (a), we use induction on i. For $i = 0$, since X is non-empty and $(X, \sqsubseteq)$ is finitely well-founded, $H_0 = m(X, \sqsubseteq)$ is finite and non-empty. For any $0 \leq i < h$, assume that H_i is finite and non-empty. Then there exists some $y \in X$ with $height(y) = k+1$ for some $k \geq i$. Let $(x_0, \ldots, x_k, y)$ be a path of length $k+1$ from a minimal element x_0 to y. By Fact 2.8(b), we have $height(x_{i+1}) = i+1$, and so $H_{i+1} \neq \emptyset$. Now, by Fact 2.8(c), for each $y \in H_{i+1}$, there is some $x \in H_i$ with $y \in \lceil x \rceil$, and so $H_{i+1} \subseteq \lceil H_i \rceil$. Since H_i is finite and $(X, \sqsubseteq)$ is locally finite, we now obtain that $\lceil H_i \rceil$ is finite and so too is H_{i+1}.
Part (b) follows easily from part (a), while part (c) follows from parts (a) and (b).□

As an immediate consequence of part (a) of this lemma, we have the following:

Corollary 2.10 *Every hs-poset is countable (either finite or denumerable).*

Remark 2.11 There is the dual notion of *depth-stratifiable* posets, as locally finite posets $(X, \sqsubseteq)$ such that $X = \Downarrow M(X, \sqsubseteq)$ and $M(X, \sqsubseteq)$ is finite. For such posets, the notion of *(finite) depth* can be defined, and for ds-posets, there exist dual results to those given for hs-posets. Nevertheless, to simplify our exposition, we limit ourselves to hs-posets, and we let the straightforward transpositions to ds-posets of the definitions and results given below to the interested reader.

2.2 Labelings and Random Structurings

In the following, a *set of labels* $\mathcal{L}$ is simply a linearly-ordered set (≡ totally-ordered), with its order denoted by $\leq$.

We recall that in [15,16], Schellekens defined the notion of *random structure* as the set of all lpo's on a given poset.

Definition 2.12 Let X be a set, $\mathcal{L}$ a set of labels with the same cardinality as X, and $\sqsubseteq$ a p.o. on X. Then:

- An $\mathcal{L}$-*labeling* on $(X, \sqsubseteq)$ is an order-preserving bijection ℓ from X to $\mathcal{L}$, that is, a bijection such that $x \sqsubseteq y \Rightarrow \ell(x) \leq \ell(y)$, for all $x, y \in X$.
- The $\mathcal{L}$-*Random Structuring* on $(X, \sqsubseteq)$ is the set of all $\mathcal{L}$-labelings on $(X, \sqsubseteq)$, and is denoted by $R_{\mathcal{L}}(X, \sqsubseteq)$.

A *labeling* (resp. *random structuring*) is an $\mathcal{L}$-labeling (resp. $\mathcal{L}$-random structuring) for *some* set of labels $\mathcal{L}$ and poset $(X, \sqsubseteq)$. We often use the calligraphic letter $\mathcal{R}$ to denote a random structuring.

We recall that $(\emptyset, \emptyset)$ is itself a partial order.

Remark 2.13 The only set of labels $\mathcal{L}$ for which $R_{\mathcal{L}}(\emptyset, \emptyset)$ is defined is $\mathcal{L} = \emptyset$, and we have $R_{\emptyset}(\emptyset, \emptyset) = \{\emptyset\}$.

We now present two fundamental examples of random structurings.

Example 2.14 [Discrete Random Structuring]
The poset (X, Δ_X) is called the *discrete order*. Clearly, $R_{\mathcal{L}}(X, \Delta_X)$ consists of all bijections from X to $\mathcal{L}$; it is called the *discrete random structuring*.

Example 2.15 [Linear Random Structuring]
Each linear order $x_1 \sqsubset \ldots \sqsubset x_n$ induces a unique $\mathcal{L}$-labeling defined by $\ell(x_i) = a_i$, where $a_1, \ldots, a_n$ are the elements of $\mathcal{L}$, arranged in increasing order. Conversely, each bijection $\ell : X \mapsto \mathcal{L}$ induces a linear order defined by $x \sqsubset y \Leftrightarrow \ell(x) < \ell(y)$. If $\sqsubseteq$ is a linear order, then $R_{\mathcal{L}}(X, \sqsubseteq)$ is called a *linear random structuring*.

Definition 2.16 Let $(X, \sqsubseteq)$ be a poset. The bijection $\ell : X \mapsto \mathcal{L}$ is said to be *height-canonical* if $height_{\sqsubseteq}(x) < height_{\sqsubseteq}(y)$ implies $\ell(x) < \ell(y)$, for all $x, y \in X$.

By Fact 2.8(a), we have immediately:

Fact 2.17 *Let $(X, \sqsubseteq)$ be a poset and let $\ell : X \mapsto \mathcal{L}$ be a height-canonical bijection. Then ℓ is a labeling on $(X, \sqsubseteq)$.*

Example 2.18 [Height-Canonical Labeling]
Clearly, $\emptyset$ is a height-canonical labeling on the poset $(\emptyset, \emptyset)$. We now give a simple way of constructing a height-canonical labeling when $(X, \sqsubseteq)$ is a hs-poset with $X \neq \emptyset$. By Corollary 2.10, X must be finite or denumerable. We show how to proceed when X is denumerable; the case when X is finite can be handled similarly.

Let $\mathcal{L}$ be any set of labels with the same cardinality as X. For each $i \in \mathbb{N}$, let $H_i = \{x \in X : height(x) = i\}$, and $k_i = Card(H_i)$. By Lemma 2.9(a), each k_i is finite and non-zero. Moreover, the partition $(H_i)_{i \in \mathbb{N}}$ induces one on $\mathcal{L}$. Indeed, letting $a_1, a_2 \ldots$ be the labels of $\mathcal{L}$ arranged in increasing order, define a partition $(\mathcal{L}_i)_{i \in \mathbb{N}}$ of $\mathcal{L}$ by $\mathcal{L}_i = a_{n_i+1} \ldots a_{n_i+k_i}$, where $n_i = \sum_{j<i} k_j$, for each $i \in \mathbb{N}$. So $\mathcal{L}_1$ contains the k_1 smallest labels, $\mathcal{L}_2$ the k_2 smallest remaining labels, and so forth. In particular, if $a \in \mathcal{L}_i$ and $b \in \mathcal{L}_j$ with $i < j$, then we have $a < b$.

Now for every family of bijections $\Psi = (\psi_i)_{i \in \mathbb{N}}$ respectively from H_i to $\mathcal{L}_i$, define $\ell_\Psi = \cup_{i \in \mathbb{N}} \psi_i$. It is easy to verify that ℓ_Ψ is indeed height-canonical on $(X, \sqsubseteq)$.

Remark 2.19 Given a family $\Psi = (\psi_i)_{i \in \mathbb{N}}$ defined as in the previous example, the union of every family $\Psi' = (\psi'_i)_{i \in \mathbb{N}}$ where, for all $i \in \mathbb{N}$, the bijection ψ'_i is a permutation of ψ_i, is clearly a labeling on $(X, \sqsubseteq)$. Moreover, for any $\ell \in R_{\mathcal{L}}(X, \sqsubseteq)$, by taking $\psi_i(x) = \ell(x)$ for every $i \in \mathbb{N}$ and $x \in X$, it is clear that every height-canonical labeling on a denumerable hs-poset can be defined as the union of such a family of bijections. A similar remark holds for finite hs-posets.

By Remark 2.13 and Example 2.18, we get the following easy fact:

Fact 2.20 *For any hs-poset* $(X, \sqsubseteq)$ *and label set* $\mathcal{L}$*, with* $Card(X) = Card(\mathcal{L})$*,* $R_{\mathcal{L}}(X, \sqsubseteq) \neq \emptyset$*, and it contains at least one height-canonical labeling.*

3 Reconstruction and Applicability of $\mathcal{MOQA}$

3.1 The Reconstruction Problem

In this section, we are interested only in finite posets and finite random structurings, so in particular, every poset is a hs-poset.

The task of computing $R_{\mathcal{L}}(X, \sqsubseteq)$, starting from $\sqsubseteq$ and $\mathcal{L}$, is equivalent to the well-known problem of generating all *linear extensions* of $\sqsubseteq$, and there exist various algorithms to perform the latter computation. The *Reconstruction Problem* consists of the inverse task, that is, how to retrieve $\sqsubseteq$, starting from X, $\mathcal{L}$, and $R_{\mathcal{L}}(X, \sqsubseteq)$. This justifies the following somewhat informal terminology.

Definition 3.1 An algorithm $\mathcal{A}$ is called a *Reconstruction Algorithm* if, for every poset $(X, \sqsubseteq)$ and set of labels $\mathcal{L}$ with size $Card(X)$:

$$\mathcal{A}(R_{\mathcal{L}}(X, \sqsubseteq))) = \sqsubseteq \ .$$

But an immediate question arises:

- What if *different* posets could have the *same* random structurings?
- In other words, what if $\sqsubseteq_1 \neq \sqsubseteq$ but $R_{\mathcal{L}}(X, \sqsubseteq_1) = R_{\mathcal{L}}(X, \sqsubseteq_2)$?
- If this could happen, then the entire Reconstruction Problem becomes ill-defined and the very existence of a Reconstruction Algorithm becomes problematic!

To overcome this problem, we need to establish the following:

Theorem 3.2 (Equivalence Theorem) *For any* X, $\mathcal{L}$, $\sqsubseteq_1$ *and* $\sqsubseteq_2$:

$$\sqsubseteq_1 = \sqsubseteq_2 \quad \Longleftrightarrow \quad R_{\mathcal{L}}(X, \sqsubseteq_1) = R_{\mathcal{L}}(X, \sqsubseteq_2)$$

The main consequences of the Equivalence Theorem are:

- The Reconstruction Problem is indeed well-defined.
- Posets and Random Structurings are interchangeable.
- There are algorithms to map in both directions: starting from an order, to build its random structuring and, conversely, starting from a random structuring, to retrieve its associated order.

3.2 Tracing Computations in $\mathcal{MOQA}$

We now recall the main reason for our interest in the Reconstruction Problem — it concerns the applicability of $\mathcal{MOQA}$. Indeed, an interesting feature of $\mathcal{MOQA}$, compared to other programming languages, is that to each program P of the language, there corresponds an operation $\widehat{P}$ on partial orders which associates with each p.o. a sequence of p.o.'s. In particular, this correspondence means that, if the

set of inputs (of a given size) of a program P of $\mathcal{MOQA}$ is the random-structuring $R_{\mathcal{L}}(X, \sqsubseteq)$, then the outputs of P are exactly the labelings on the orders in the sequence $\widehat{P}(X, \sqsubseteq)$. Thus, $\mathcal{MOQA}$ allows for the tracking of all the computations of a program on all its possible inputs (of a given size) in a very simple way. This tracking will then be used to facilitate the average time analysis of the program.

In this context, it becomes crucial to answer the question: "For a given program P, do the sets of all the inputs (of a given size) to P form a random structuring?". The following theorem shows that the existence of a Reconstruction Algorithm enables us to answer this question.

Theorem 3.3 *Let $\mathcal{A}$ be an algorithm defined on all sets of bijections from X to $\mathcal{L}$. Then, if $\mathcal{A}$ is a reconstruction algorithm, the following algorithm $\chi_{\mathcal{A}}$ will decide, for any set $\mathcal{F}$ of bijections from X to $\mathcal{L}$, whether or not there exists a partial order $\sqsubseteq$ on X such that $\mathcal{F} = R_{\mathcal{L}}(X, \sqsubseteq)$:*

`[1]` Run algorithm $\mathcal{A}$ on input $\mathcal{F}$, and let $\sqsubseteq_{\mathcal{F}}$ be its output.

`[2]` If $\sqsubseteq_{\mathcal{F}}$ is not a partial order, then return *false*.

`[3]` Compute $R_{\mathcal{L}}(X, \sqsubseteq_{\mathcal{F}})$ (using one of the existing algorithms).

`[4]` If $R_{\mathcal{L}}(X, \sqsubseteq_{\mathcal{F}}) = \mathcal{F}$, then return *true*, else return *false*.

Proof. Clearly, for any set $\mathcal{F}$ of bijections, $\chi_{\mathcal{A}}(\mathcal{F})$ returns either *true* or *false*.

- If $\chi_{\mathcal{A}}(\mathcal{F}) =$ *true*, then step `[4]` was reached, with $\sqsubseteq_{\mathcal{F}}$ being the partial order such that $R_{\mathcal{L}}(X, \sqsubseteq_{\mathcal{F}}) = \mathcal{F}$.
- If $\chi_{\mathcal{A}}(\mathcal{F}) =$ *false*, then suppose there actually exists some partial order $\sqsubseteq$ such that $\mathcal{F} = R_{\mathcal{L}}(X, \sqsubseteq)$. However, step `[4]` will then be reached, with $\sqsubseteq_{\mathcal{F}} = \sqsubseteq$ and $R_{\mathcal{L}}(X, \sqsubseteq) = \mathcal{F}$, giving $\chi_{\mathcal{A}}(\mathcal{F}) =$ *true*, contrary to assumption.

□

We will see in the next section that the Equivalence Theorem is a corollary of the Reconstruction Theorem (Theorem 4.1).

4 Defining a hs-poset from its Labelings

4.1 Reconstruction Theorem

We show that each hs-poset $(X, \sqsubseteq)$ can be defined uniquely using the set of its order-preserving bijections (on any set of labels $\mathcal{L}$ for X).

Theorem 4.1 (Reconstruction Theorem) *Let $(X, \sqsubseteq)$ be a hs-poset, and $\mathcal{L}$ a set of labels with the same cardinality as X. Then, for all $x, y \in X$, we have:*

$$x \sqsubset y \iff \forall \ell \in R_{\mathcal{L}}(X, \sqsubseteq) : \ell(x) < \ell(y)$$

Proof. Note first that, by definition of a labeling, we have immediately that $x \sqsubset y$ implies $\forall \ell \in R_{\mathcal{L}}(X, \sqsubseteq) : \ell(x) < \ell(y)$. The converse follows by contraposition from Proposition 4.2 below. □

Proposition 4.2 *Let $(X, \sqsubseteq)$ be a hs-poset, and $\mathcal{L}$ a set of labels with the same cardinality as X. If $x, y \in X$ are such that $x \not\sqsubseteq y$, then there exists an order-preserving bijection L w.r.t. $\sqsubseteq$ such that $L(x) > L(y)$.*

We give the proof of the proposition in the next section, but we first present some of its easy corollaries.

Corollary 4.3 (Equivalence Theorem) *Let $\sqsubseteq_1$ and $\sqsubseteq_2$ be two partial orders on a set X, and let $\mathcal{L}$ be a set of labels. Then:*

$$\sqsubseteq_1 = \sqsubseteq_2 \iff R_{\mathcal{L}}(X, \sqsubseteq_1) = R_{\mathcal{L}}(X, \sqsubseteq_2)$$

Proof. If $\sqsubseteq_1 = \sqsubseteq_2$, then clearly $R_{\mathcal{L}}(X, \sqsubseteq_1) = R_{\mathcal{L}}(X, \sqsubseteq_2)$. To prove the converse, suppose $\sqsubseteq_1 \neq \sqsubseteq_2$. Then there exist $x, y \in X$ such that either $x \sqsubseteq_1 y$ and $x \not\sqsubseteq_2 y$, or $x \sqsubseteq_2 y$ and $x \not\sqsubseteq_1 y$; without loss of generality, assume the former case.

By the Reconstruction Theorem, there exists some order-preserving bijection $L \in R_{\mathcal{L}}(X, \sqsubseteq_2)$ such that $L(x) > L(y)$. Now, for each $\ell \in R_{\mathcal{L}}(X, \sqsubseteq_1)$, we have $\ell(x) < \ell(y)$, as $x \sqsubseteq_1 y$, so $L \notin R_{\mathcal{L}}(X, \sqsubseteq_1)$, and so $R_{\mathcal{L}}(X, \sqsubseteq_1) \neq R_{\mathcal{L}}(X, \sqsubseteq_2)$. □

Recall that $\Delta_X = \{(x, x) : x \in X\}$. We now present the pseudo-code for a simple Reconstruction Algorithm $\mathcal{A}_0$, referred to as the "Brute-Force Algorithm":

$\mathcal{A}_0(\mathcal{F})$:=
```
   -- INPUT  : a set F of bijections, with F = R_L(X, ⊑) for some ⊑
   -- OUTPUT : the partial order ⊑
   S := ∅
   for each pair (x, y) ∈ X × X such that x ≠ y
      if ∀ f ∈ F : f(x) < f(y) then S := S ∪ {(x, y)}
   return S ∪ Δ_X
```

Corollary 4.4 *For any $X, \sqsubseteq, \mathcal{L}$: $\mathcal{A}_0(R_{\mathcal{L}}(X, \sqsubseteq)) = \sqsubseteq$.*

Proof. Let $\mathcal{R} = R_{\mathcal{L}}(X, \sqsubseteq)$; then $(x, y) \in \mathcal{A}_0(\mathcal{R})$ iff $x = y$ or $\forall f \in \mathcal{R} : f(x) < f(y)$. The result now follows from the Reconstruction Theorem. □

Notation: Let $\mathcal{F}$ be a set of bijections from X to $\mathcal{L}$:

- $\sqsubset_{\mathcal{F}} = \{(x, y) : \forall f \in \mathcal{F},\ f(x) < f(y)\}$
- $\sqsubseteq_{\mathcal{F}} = \Delta_X \cup \sqsubset_{\mathcal{F}}$ $(= \mathcal{A}_0(\mathcal{F}))$

The above notation is justified by the following trivial corollary:

Corollary 4.5 *Let $\mathcal{F}$ be a non-empty set of bijections from X to $\mathcal{L}$. Then:*

- *$\sqsubset_{\mathcal{F}}$ is a strict partial order.*
- *$\sqsubseteq_{\mathcal{F}}$ is a partial order.*
- *$\mathcal{F} \subseteq R_{\mathcal{L}}(X, \sqsubseteq_{\mathcal{F}})$.*

Notation: The orders given in Corollary 4.5 are called the *strict order induced by $\mathcal{F}$* and the *order induced by $\mathcal{F}$*, respectively.

Observe that the above corollary suggests that each set of bijections $\mathcal{F}$ defines a partial order whose structure can be deduced from that of $\mathcal{F}$, as demonstrated in Example 4.9 below.

Definition 4.6 Let $f, g : X \mapsto \mathcal{L}$ be two bijections. Then g is *an inverse* of f if for all $x, y \in X$, we have $g(x) < g(y)$ iff $f(y) < f(x)$.

Remark 4.7 Clearly, if g is an inverse of f, then f is an inverse of g.

Remark 4.8 If X is finite and $Card(\mathcal{L}) = Card(X)$, then each bijection $f : X \mapsto \mathcal{L}$ has a unique inverse. In fact, if $(a_1, \ldots, a_n)$ are the labels in $\mathcal{L}$ in increasing order, and $(x_1, \ldots, x_n)$ are the elements of X, indexed so that $f(x_i) = a_i$, then the unique inverse g of f is given by $g(x_i) = a_{n-(i-1)}$, for each $1 \leq i \leq n$.

Example 4.9 Let $f, g : X \mapsto \mathcal{L}$ be two bijections. Then:

- The induced order $\sqsubseteq_{\{f\}}$ is a linear order.
- If f and g are inverses, then the induced order $\sqsubseteq_{\{f,g\}}$ is the discrete order.

Remark 4.10 Recall from Example 2.14 that the random structuring on a discrete order is isomorphic to the set of all the permutations on a given set of labels. Thus, the induced order $\sqsubseteq_{\{f,g\}}$ in the above example illustrates a case of an order which can be induced by a strict subset of its random structuring. We note that the task of determining how many labelings of an order are needed to induce that order is well known in the literature as the task of determining the *dimension* of the order.

Corollary 4.11 *If $\mathcal{F}_1$ and $\mathcal{F}_2$ are two sets of bijections, then:*

- $\sqsubseteq_{\mathcal{F}_1 \cup \mathcal{F}_2} = \sqsubseteq_{\mathcal{F}_1} \cap \sqsubseteq_{\mathcal{F}_2}$.
- $\sqsubseteq_{\mathcal{F}_1 \cup \mathcal{F}_2} = \sqsubseteq_{\mathcal{F}_1} \cap \sqsubseteq_{\mathcal{F}_2}$.

This observation suggests the following refinement of algorithm $\mathcal{A}_0$:

$\mathcal{A}_1(\mathcal{F})$:=
-- INPUT : a set $\mathcal{F} = \{ f_1, f_2, \ldots, f_k \}$ of bijections,
-- with $\mathcal{F} = R_{\mathcal{L}}(X, \sqsubseteq)$ for some $\sqsubseteq$
-- OUTPUT : the partial order $\sqsubseteq$
$\mathcal{S} := \sqsubseteq_{\{f_1\}}$
for each $2 \leq i \leq k$
  if $\mathcal{S} = \emptyset$ then
    return Δ_X
  else
    $\mathcal{S} := \mathcal{S} \cap \sqsubseteq_{\{f_i\}}$
return $\mathcal{S} \cup \Delta_X$

4.2 Proof of the Fundamental Proposition

We now set about the proof of Proposition 4.2. In order to do this, we introduce the notions of *pruned ascending tree* and *pruned descending tree* (for the notions of ascending and descending trees, recall Definition 2.1).

Definition 4.12 Let $(X, \sqsubseteq)$ be a partial order and $x \in X$. Define, respectively, the *ascending tree of x pruned at height i* and the *descending tree of x pruned at height i* by:

- $\Uparrow^i x = \{ y \in \Uparrow x : height(y) \leq i \}$
- $\Downarrow_i x = \{ y \in \Downarrow x : height(y) \geq i \}$

Recall that the ascending tree of x just consists of the elements of X encountered when ascending the paths starting from x, that is, to construct $\Uparrow x$ we first take x, then the elements of $\lceil x \rceil$, plus those of $\lceil \lceil x \rceil \rceil$, and so on. In the case of the pruned ascending tree, we just stop taking the elements of an ascending path when we find an element z of this path such that $height(z) > i$. In particular, we do not include this element z itself. Note that any element above such a z will also have height greater than i, and so it is useless to further explore the ascending path after z, that is, we cannot expect to find any further elements on that path to include in $\Uparrow^i x$.

A symmetric line of reasoning applies in the case of $\Downarrow_i x$.

The next observations follow easily from Fact 2.8.

Fact 4.13 *Let $(X, \sqsubseteq)$ be a hs-poset, h its height and $x, y \in X$.*

(a) *For all $y \in \Uparrow^i x$, we have $height(x) \leq height(y) \leq i$.*

(b) *For all $y \in \Downarrow_i x$, we have $i \leq height(y) \leq height(x)$.*

(c) *$x \in \Uparrow^i x$ iff $i \geq height(x)$, and in this case, x is the unique element of $\Uparrow^i x$ whose height equals $height(x)$.*

(d) *$x \in \Downarrow_i x$ iff $i \leq height(x)$, and in this case, x is the unique element of $\Downarrow_i x$ whose height equals $height(x)$.*

(e) *If $x \not\sqsubseteq y$, then $\Uparrow x \cap \Downarrow y = \emptyset$.*

(f) *If $x \not\sqsubseteq y$, then $\Uparrow^i x \cap \Downarrow_j y = \emptyset$, for all heights $i, j \leq h$.*

To prove Proposition 4.2, and for §5 below, the following simple concept is useful:

Definition 4.14 Let f be any function defined on X, and let x, y be fixed elements of X. Define the function f_{xy} on X by:

$$f_{xy}(z) = \begin{cases} f(x), & \text{if } z = y \\ f(y), & \text{if } z = x \\ f(z), & \text{otherwise} \end{cases}$$

Notation: $X \backslash Y = \{x \in X : x \notin Y\}$.

Proof. (of **Proposition 4.2**) If $x \not\sqsubseteq y$, our task is to find a labeling L of $(X, \sqsubseteq)$ such that $L(x) > L(y)$. To accomplish this, we examine separately the three cases $height(x) > height(y)$, $height(x) = height(y)$, and $height(x) < height(y)$.

If $height(x) > height(y)$, we can take L to be any height-canonical labeling ℓ (see Example 2.18 and Fact 2.20), since $\ell(x) > \ell(y)$ for any such ℓ.

If $height(x) = height(y)$, let ℓ be any height-canonical labeling. If $\ell(x) > \ell(y)$,

then take $L = \ell$, otherwise take $L = \ell_{xy}$. Then L is still height-canonical and thus order-preserving (Remark 2.19), and of course $L(x) > L(y)$.

If $height(x) < height(y)$, then let ℓ be any height-canonical labeling on $(X, \sqsubseteq)$. Let $Z = \{z \in X : height(x) \leq height(z) \leq height(y)\}$ and $\Lambda = \ell(Z)$. In addition, let $x\uparrow y$ denote $\Uparrow^{height(y)} x$ and $y\downarrow x$ denote $\Downarrow_{height(x)} y$.

Before continuing, we make some observations:

1) Since ℓ is height-canonical, we have, for any $u \in X$:
   a) $height(u) > height(y) \;\Rightarrow\; \ell(u) > a, \forall a \in \Lambda$.
   b) $height(u) < height(x) \;\Rightarrow\; \ell(u) < a, \forall a \in \Lambda$.
2) $x\uparrow y \;\cup\; y\downarrow x \;\subseteq Z$, by definition.
3) $x\uparrow y \;\cap\; y\downarrow x \;= \emptyset$, by Fact 4.13(f).

Let $Z^- = Z \setminus (x\uparrow y \;\cup\; y\downarrow x)$ (see observation 2). Let Λ_x and Λ_y denote the sets of the $Card(x\uparrow y)$ largest values and the $Card(y\downarrow x)$ smallest values in the set Λ, respectively, noting from observation 3 that Λ_x and λ_y must be disjoint. Finally, let $\Lambda^- = \Lambda \setminus (\Lambda_x \cup \Lambda_y)$. With this notation, we complete our observations with:

4) $a < b < c$, $\forall a \in \Lambda_y$, $\forall b \in \Lambda^-$, $\forall c \in \Lambda_x$

We now define the required labeling L. The general idea is to label the elements of $X\setminus Z$ in the same manner as ℓ does, setting $L(u) = \ell(u)$ for all $u \in X\setminus Z$. It then remains to define L on Z, and we do so by separately labeling the elements of $x\uparrow y$ with the labels of Λ_x, those of $y\downarrow x$ with the labels of Λ_y, and finally, the remaining elements of Z, i.e., Z^-, with those of Λ^-, and we do so in a "height-canonical" way; in other words, we define L as a *locally height-canonical* labeling.

We present the detailed definition of L on $x\uparrow y$; its definition on $y\downarrow x$ and Z^- are done in a similar way. Let $u \in x\uparrow y$, so that $height(x) \leq height(u) \leq height(y)$; we proceed inductively on $height(u)$:

- As the base case, note that by Fact 4.13(c), if $height(u) = height(x)$ then $u = x$; define $L(x)$ to be the smallest label in Λ_x.
- Having already defined L on all elements of height k ($height(x) \leq k < height(y)$), take successively each $u \in x\uparrow y$ of height $k+1$, and define $L(u)$ to be the smallest label in Λ_x which is not yet used by L.

Note that $L(Z) = \Lambda$. Thus, the labeling functions L and ℓ are identical on $X\setminus Z$, while their values on Z are merely permuted. Note also that the restrictions of L to the sets $x\uparrow y$, $y\downarrow x$ and Z^- are each height-canonical labelings relative to the corresponding restrictions on $\sqsubseteq$.

Since $L(x) \in \Lambda_x$ and $L(y) \in \Lambda_y$, observation 4 gives $L(x) > L(y)$, as desired.

It now remains for us to show that L is an order-preserving bijection w.r.t. $\sqsubseteq$. To do so, it suffices to show that for all $u \in X$ and $u' \in \lceil u \rceil$, we have $L(u) < L(u')$. The possible cases are each examined in turn:

- $u \notin Z$, $u' \notin Z$: Then $L(u) = \ell(u) < \ell(u') = L(u')$.
- $u \in Z$, $u' \notin Z$: As $u' \notin Z$, either $height(u') < height(x)$ or $height(u') > height(y)$.

But $height(u') > height(u) \geq height(x)$ (since $u \in Z$), so we must in fact have $height(u') > height(y)$. Using observation 1.a (with u' in place of u) and the fact that $L(u) \in \Lambda$ now gives $\ell(u') > L(u)$, and as $L(u') = \ell(u')$, we have $L(u) < L(u')$.

- $u \notin Z$, $u' \in Z$: By a symmetric argument to the previous one, but this time using observation 1.b, we conclude that $L(u) < L(u')$ in this case also.
- $u \in Z$, $u' \in Z$: Since Z is the union of the three disjoint sets $x \uparrow y$, $y \downarrow x$, and Z^-, we consider separately the cases of u belonging to each of these three sets:
  - $u \in x \uparrow y$: Then $u' \in x \uparrow y$ (by definition of $x \uparrow y$), and as L is height-canonical on $x \uparrow y$, we have $L(u) < L(u')$.
  - $u \in y \downarrow x$: Then $L(u) \in \Lambda_y$. We examine the three possible sub-cases:
    - $u' \in x \uparrow y$: Then $L(u') \in \Lambda_x$ and by observation 4, $L(u) < L(u')$.
    - $u' \in y \downarrow x$: As L is height-canonical on $y \downarrow x$, we have $L(u) < L(u')$.
    - $u' \in Z^-$: Then $L(u') \in \Lambda^-$ and by observation 4, $L(u) < L(u')$.
  - $u \in Z^-$: Then $L(u) \in \Lambda^-$. We examine the three possible sub-cases:
    - $u' \in x \uparrow y$: Then $L(u') \in \Lambda_x$ and by observation 4, $L(u) < L(u')$.
    - $u' \in y \downarrow x$: This sub-case cannot arise, for if $u' \in y \downarrow x$, then since $u \in \lfloor u' \rfloor$ and $height(u) \geq height(x)$, we would have $u \in y \downarrow x$, contradicting $u \in Z^-$.
    - $u' \in Z^-$: As L is height-canonical on Z^-, we have $L(u) < L(u')$.

□

5 Some Further Results on Random Structures

5.1 A Characterisation of Atomic Isolated Subsets

In this section, we recall two significant concepts from $\mathcal{MOQA}$: the concept of an *Atomic Isolated Subset (AIS)* and that of a *free pair* of labels for a given labeling. Then we show that the AIS's can be characterised using only the labelings on the given poset. This result is a consequence of the Reconstruction Theorem.

Definition 5.1 A set $Y \subseteq X$ is an *Atomic Isolated Subset* in $(X, \sqsubseteq)$ if, for all distinct elements $x, y \in Y$:

- $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$ (Y is an *antichain* in $(X, \sqsubseteq)$); and
- $\lceil x \rceil = \lceil y \rceil$ and $\lfloor x \rfloor = \lfloor y \rfloor$.

Definition 5.2 Let $a, b \in \mathcal{L}$ be two labels, let $\ell \in R_{\mathcal{L}}(X, \sqsubseteq)$, and let $x = \ell^{-1}(a)$ and $y = \ell^{-1}(b)$. Then $\{a, b\}$ is called a *free pair of labels for* ℓ if $\ell_{xy} \in R_{\mathcal{L}}(X, \sqsubseteq)$.

If $x \sqsubset y$, then we must have $\ell(x) < \ell(y)$ for all labelings ℓ, and thus the set $\{\ell(x), \ell(y)\}$ cannot be a free pair.

Lemma 5.3 *Let $(X, \sqsubseteq)$ be a hs-poset, and let $Y \subseteq X$. Then Y is an AIS iff for every $x, y \in Y$ and every $\ell \in R_{\mathcal{L}}(X, \sqsubseteq)$, the set $\{\ell(x), \ell(y)\}$ is a free pair.*

Proof. Firstly, suppose Y is an AIS, $x, y \in Y$, and $\ell \in (X, \sqsubseteq)$. We show that $\{\ell(x), \ell(y)\}$ is a free pair by showing that $\forall u, v \in X : u \sqsubset v \Rightarrow \ell_{xy}(u) < \ell_{xy}(v)$.

- Since $u \sqsubset v$, but Y is an antichain, at most one of u, v can belong to Y.

- If neither u nor v equals x or y, then $\ell_{xy}(u) = \ell(u) < \ell(v) = \ell_{xy}(v)$.
- If $u \in \{x, y\}$, then assume, without loss of generality, that $u = x$. Then we have $v \in \lceil x \rceil = \lceil y \rceil$, so $\ell(y) < \ell(v)$. Thus $\ell_{xy}(u) = \ell_{xy}(x) = \ell(y) < \ell(v) = \ell_{xy}(v)$.
- The case $v \in \{x, y\}$, taking $v = x$ and thus $u \in \lfloor x \rfloor$, is handled similarly.

Secondly, suppose that for all $x, y \in Y$ and all $\ell \in R_{\mathcal{L}}(X, \sqsubseteq)$, the set $\{\ell(x), \ell(y)\}$ is a free pair. It is clear that for all distinct $x, y \in Y$, we have $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$ (otherwise, $\{\ell(x), \ell(y)\}$ would not be a free pair for *any* labeling ℓ).

We now show that, for all distinct $x, y \in Y$, we have $\lceil x \rceil = \lceil y \rceil$ and $\lfloor x \rfloor = \lfloor y \rfloor$. We establish the first equality by contradiction; the second is handled similarly.

If $x, y \in Y$ with $\lceil x \rceil \neq \lceil y \rceil$, we exhibit a labeling ℓ for which $\{\ell(x), \ell(y)\}$ is not a free pair. By hypothesis, there exists $z \in \lceil x \rceil$ and $z \notin \lceil y \rceil$, or $z \in \lceil y \rceil$ and $z \notin \lceil x \rceil$. Assume the first case; the second is handled similarly. There are two possibilities:

- $y \not\sqsubseteq z$: By Proposition 4.2, there is a labeling ℓ such that $\ell(y) > \ell(z)$. But then ℓ_{xy} is not a labeling, since $\ell_{xy}(x) = \ell(y) > \ell(z) = \ell_{xy}(z)$, although $z \in \lceil x \rceil$.
- $y \sqsubseteq z$: Clearly, $z \neq y$, since otherwise $y \in \lceil x \rceil$, contradicting $x \not\sqsubseteq y$. So $y \sqsubset z$, and as $z \notin \lceil y \rceil$, there exists a $z' \in \lceil y \rceil$ with $z' \sqsubset z$. Moreover, we have $x \not\sqsubseteq z'$, since $z \in \lceil x \rceil$. Thus, we are now in the same situation as in the previous case, but with the roles of x and y reversed and z' in place of z: $x \not\sqsubseteq z'$, $z' \in \lceil y \rceil$ and $z' \notin \lceil x \rceil$. We then conclude that $\{\ell(x), \ell(y)\}$ is not a free pair.

□

Note that if Y is an AIS, then all its elements have the same *ranking probabilities* in the sense of [12,13].

5.2 Cardinality of Finite Random Structurings

In this section, we give some results about the cardinality of random structurings on finite (and so height-stratifiable) posets.

Let $(X, \sqsubseteq)$ be a finite poset with $n = Card(X)$, and let $\mathcal{L}$ be a set of labels, with $Card(\mathcal{L}) = n$. Denote $R_{\mathcal{L}}(X, \sqsubseteq)$ by $\mathcal{R}$, and let $c = Card(\mathcal{R})$.

If $\sqsubseteq$ is a linear order, that is, if there is a bijection $f : X \mapsto \mathcal{L}$ with $\sqsubseteq = \sqsubseteq_f$, then we have $\mathcal{R} = \{f\}$, and so $c = 1$.

If $\sqsubseteq$ is the discrete order, then $\mathcal{R}$ contains all possible bijections from X to $\mathcal{L}$, and so $c = n!$. The result for non-discrete orders is given by:

Lemma 5.4 *If $(X, \sqsubseteq)$ is a non-discrete finite poset, then $Card(R_{\mathcal{L}}(X, \sqsubseteq)) \leq \frac{n!}{2}$, where $n = Card(X)$.*

Proof. Suppose the contrary. Then $R_{\mathcal{L}}(X, \sqsubseteq)$ must contain a labeling, but also its inverse (as an easy consequence of the second part of Remark 4.8), and so $\sqsubseteq$ must be the discrete order (Example 4.9). This contradicts the hypothesis. □

We sum up the results of this section by the following:

Theorem 5.5 *If $(X, \sqsubseteq)$ is a finite poset, with $n = Card(X)$, $\mathcal{L}$ is a set of labels with $Card(\mathcal{L}) = n$, then the possible values of $c = Card(R_{\mathcal{L}}(X, \sqsubseteq))$ are:*

- $c = 1 \quad \Leftrightarrow \ \sqsubseteq$ *is a linear order*
- $c = n! \ \Leftrightarrow \ \sqsubseteq$ *is the discrete order*
- $1 < c \leq \frac{n!}{2} \ \Leftrightarrow \ \sqsubseteq$ *is neither linear nor discrete.*

Conclusion

We showed that every height-stratifiable poset is uniquely determined by the set of all labelings on it using a technique, in the proof of Proposition 4.2, based on the existence of height-canonical labelings. That technique consists of a direct modification of a labeling by definition of local height-canonical labelings to get a new labeling (or equivalently a new linear extension) satisfying a required property. It can be used, for instance, to prove that the possible values for $\ell(x)$, where $x \in X$ and $\ell : X \mapsto \mathcal{L}$ is a labeling on the poset $(X, \sqsubseteq)$ with n elements, are precisely $a_{i+1}, \ldots, a_{n-j}$, where $i = Card(\Downarrow x) - 1$, $j = Card(\Uparrow x) - 1$, and $a_1, \ldots, a_n$ are the labels, arranged in increasing order.

Theorem 5.5 gives a first characterisation of the cardinality of random structures. A more precise and complete characterisation is expected. Moreover, the theorem shows that many recursive sets of lists cannot be captured as random structures. Nevertheless, it opens the way to a more complex question, that is, how recursive sets can be captured by sequences of random structures in the most efficient way.

We may also expect that the new framework in the which linear extensions are used via their interpretations as labelings and random structures in [15,16,14] will open the way to interesting new questions about linear extensions.

Acknowledgement

The authors wish to thank the anonymous referees for their helpful suggestions.

References

[1] Brightwell G.: Linear extensions of infinite sets. Discrete Math **70** (1988) 113–136.

[2] Brightwell G., Prömel H.-J., Steger A.: The average number of linear extensions of a partial order. J. Combinatorial Theory (A) **73** (1996) 193–206.

[3] Brightwell G., Tetali P.: The number of linear extensions of the boolean lattice. Order **20** (2003) 333–345.

[4] Brightwell, G., Winkler P.: Counting Linear Extensions. Order **8** (1991) 225–242.

[5] Brualdi, R. A.: Introductory Combinatorics, 4th ed. New York: Elsevier, 1997.

[6] Bubley, R., Dyer, M.: Faster Random Generation of Linear Extensions. In *Proc. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, Calif. (1998) 350–354.

[7] Carlsen L., Lerche D., Sorensen P.B.: Improving the predicting power of partial order based QSARs through linear extensions. J. Chem. Inf. Comput. Sci. **42**(4) (Jul–Aug 2002) 806–811.

[8] Dushnik B., Miller E.W.: Partially ordered sets. American Journal of Mathematics **63** (1941) 600–610.

[9] Flajolet P., Vitter, J.S.: Average-Case Analysis of Algorithms and Data Structures. Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity. Elsevier (1990) 431–524.

[10] Kelly, D.: The 3-irreducible partially ordered sets. Canad. J. Math. **29** (1977) 367–383.

[11] Knuth D.: The Art of Computer Programming Vol. 3. Addison-Wesley (1973).

[12] Lerche D., Sorensen P.B.: Evaluation of the ranking probabilities for partial orders based on random linear extensions. Chemosphere **53**(8) (Dec 2003) 981–92.

[13] Lerche D., Sorensen P.B., Bruggeman R.: Improved estimation of the ranking probabilities in partial orders using random linear extensions by approximation of the mutual ranking probability. J. Chem. Inf. Comput. Sci. **43**(5) (Sep–Oct 2003) 1471–1480.

[14] Schellekens M.: Modular Timing, An overview of CEOL research. Proceeding of MFCSIT'06, Cork, Ireland (Aug 2006), pp. 300–303.

[15] Schellekens M.: A Modular Calculus for the Average Cost of Data Structuring. Springer (2008). See: `www.springer.com/computers/foundations/`

[16] Schellekens M.: MOQA: Unlocking the Potential of Compositional Static Average-Case Analysis. Journal of Logic and Algebraic Programming (2008) (in press).

[17] Schellekens M., Hickey D., Bollella G.: ACETT, a Linearly-Compositional Programming Language for (semi-)automated Average-Case analysis. IEEE Real-Time Systems Symposium - Work In Progress Session (2004).

[18] Szpilrajn E.: Sur l'extension de l'ordre partiel. Fundamenta Mathematicae **16** (1930).

[19] Vallée T.: Functionally-Generalised $\mathcal{MOQA}$ Operations. Proceedings of MFCSIT'06, Cork, Ireland (Aug 2006), pp 308–311.

[20] Varol, Y., Rotem, D.: An Algorithm to Generate All Topological Sorting Arrangements. Comput. J. **24** (1981) 83–84.