Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 225 (2009) 421–439
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Functionally-Generalised MOQA Operations

Thierry Vallee[1,2]

Dept. of Mathematical Sciences
Georgia Southern University
Georgia, United States

Abstract

The programming language MOQA was designed by Michel Schellekens in [5,7] specifically to facilitate the average execution time analysis of its programs. MOQA is based on a special data structure and an associated suite of operations. This special data structure, said here as *labeled partial orders (lpo)*, is a pair (P, ℓ) such that $P = (X, \sqsubseteq)$ is a poset and $\ell : X \mapsto \mathcal{L}$ a strictly increasing bijection between X and a totally ordered set $\mathcal{L}$. MOQA basic operations are defined on lpo's under some restrictions. These restrictions guarantee the main property of MOQA's operations, said here as *Automatic Random Preservation*. Random Preservation (RP) is a fundamental property used to calculate the average time of programs, opening the way to a (semi-)automation of this calculation in MOQA.
In the paper, we introduce *functionally-generalised versions* of some MOQA basic operations. Those versions are defined with more natural restrictions and in a more general way than the correspondent MOQA operations. As a consequence of their greater generality, they are not RP on all their domains in contrast to MOQA operations. Nevertheless, they have two good properties. The first one is that they have the intended semantics. For instance, the generalised split actually splits any lpo into two parts using a label as a pivot. The second one is that the subdomains on which they are RP are strictly bigger than the domains where MOQA basic operations are defined and RP. A tractable characterisation of those RP subdomains is an open problem and its solution would be an important step toward an implementation of the functionally-generalised operations as Automatic RP operations.

Keywords: Posets, Average Execution Time, Program Semantics, Data Structure Operations

1 Introduction

MOQA is a new programming language, based on a kernel of basic operators, introduced by M. Schellekens in order to facilitate the Average Execution Time Analysis (A-ETA) of its programs (cf. [5,7,4,6,8]). The design of MOQA is developed in [5,7] where the semantics of the basic operators is given via operations, refered here as *basic MOQA operations*. In this article, we present *functional-generalisations* of three of these operations: the *projection*, the *unary product* and the *split*.

A first motivation for developing MOQA was that, according to the literature (see for instance [2,3,5]), A-ETA of programs is much harder than the worst case

[1] Research supported by Science Foundation Ireland under Grant 02/IN.1/181
[2] Email: vallee_th@yahoo.fr

doi:10.1016/j.entcs.2008.12.090

execution time analysis (WC-ETA). A-ETA leads to complex techniques without satisfactory generality. The second remark is that, curiously and in contrast to worst case, average time has a good property which helps the timing: *compositionality.*

In order define compositionality, we recall first that a *multiset* or *bag* is a collection which allows copies of the same element. For example, $\{a, a, b\}$ is a multiset. Multisets are used to store the outputs of programs. For instance, the output multiset of the addition on the set $I = \{3, 1\}^2$ is the set $O_+(I) = \{4, 4, 2, 6\}$. Then compositionality of average time is expressed by the following equation easily infered from the definition of average time, where $P; Q$ is the program obtained by the concatenation of P and Q, and $\bar{T}_P(I)$ is the average execution time of the program P on I: $\bar{T}_{P;Q}(I) = \bar{T}_P(I) + \bar{T}_Q(O_P(I))$.

Compositionality allows the determination of the average time of a complex program by analysing its sub-programs. It facilitates timing modularity since the analysis of $\bar{T}_Q(O_P(I))$ can be reused. So a question arises: why is A-ETA so complex? A closer look to $O_P(I)$ gives an answer. Indeed, even when I is uniformly distributed, $O_P(I)$ is generally not. For instance $\{3, 1\}^2$ contains one copy of each pair of integers while $\{4, 4, 2, 6\}$ contains two copies of 4 and only one copy of 2 and 6. Moreover, there is no general control on the distribution of the outputs. These two facts makes it difficult and hard to predict the calculation of $\bar{T}_Q(O_P(I))$. So the approach of [5,7] was to design the basic operations of MOQA in order to obtain a general control on the *multiplicities* (number of copies) of the outputs of those operations. The control relies also on the particular data structure used by MOQA, said here as *labeled partial orders (lpo's).* A lpo is a pair (P, ℓ) such that $P = (X, \sqsubseteq)$ is a poset and $\ell : X \mapsto \mathcal{L}$ is a strictly increasing bijection between X and a totally ordered set of labels $\mathcal{L}$. The bijection ℓ is said here as a *labeling.* Intuitively, a lpo is twofold: it contains a list of labels, which may be the relevant information for the programmer, and a partial order, used in particular by the system running MOQA to keep a control on the output multiplicities. A *Random Structure (RS)* is the set of all lpo's for a given P and a given $\mathcal{L}$. The essential property of MOQA basic operations is that they allow a control on the multiplicities of the outputs as soon as the inputs form a *Random Structure (RS).* That property is said here as *Random Preservation (RP).*

Nevertheless, the MOQA projection, unary product and split operations are defined under some unusual restrictions on the lpo's. Although, these restrictions guarantee the RP of MOQA's operations on their domains, they have some unwanted side effects. First, they make programming harder for the programmer who must be careful with both the semantics and the RP of his programs. Moreover, the restrictions prevent a free modularity of programs. Indeed, suppose that the programmer had written a program Q and that he wants to re-use it, for instance, in the program $P; Q$. Even if P and Q are semantically correct, that is, they compute independently the intended functions, the good conditions allowing the application of MOQA's operations of Q may have been destroyed by the previous operations of P, making the execution inside Q impossible in $P; Q$.

Thus, it seems to be of interest to separate the semantics correctness of MOQA's programs from their RP correctness. For that we introduce generalised versions of the projection, unary product, and split, which are defined for labelings under natural programming conditions. As a consequence of this generality, these operations are not RP on all their domains of definition. Nevertheless, they have two good properties. First, they behave as expected, that is, they have the same behavior as the corresponding MOQA operations. For instance, the generalised split actually splits the set of labels into two parts using the first label as a pivot. Second, they also generalise Random Preservation since they are RP on parts of their domains where the MOQA basic operations are not defined and so a fortiori not RP.

In Section 2, we present several notions, in particular the notion of Random Preservation. Note that we will introduce a distinction between Random Preservation and *Automatic Random Preservation*, a distinction which is not done in [5,7]. In Section 3, we define the notions of *isolated* subsets of a poset and *local transformation*. Isolated subsets are used in particular to express the restrictions on the applicability of MOQA operations. Then in Section 4, we introduce MOQA operations and their generalisations. Finally, Section 5 we discuss the RP property of the generalisations. We remark that this article is essentially self-contained. In particular we introduce the notions from [5,7] we need.

2 Prelimanaries

In this section we introduce some notations and operations on posets and lpo's, we then introduce posets as special case of *directed graph*. Finally, we recall the key notions of labeling, random structures and random preservation.

2.1 General notations

We will use $\mathbb{N}$ to denote the set of positive integers. We will use P to denote a poset, X_P to denote the ground set of P, or simply X when there is no ambiguity, and $\sqsubseteq_P$ or $\sqsubseteq$ to denote the partial order (p.o.) on X_P.

Definition 2.1 Let X, Y be sets, R be a binary relation on X and $P = (X, R)$:

- $X \setminus Y = \{x \in X : x \notin Y\}$
- $X \subseteq_f Y$ to express that $X \subseteq Y$ and X is finite.
- $Card(X)$ will denote the cardinal of the set X.
- $\Delta_X = \{(x, x) : x \in X\}$
- $R\restriction Y = R \cap Y^2$; $P\restriction Y = (Y,\ R\restriction Y)$
- $Dom(R) = \{x : \exists y,\ xRy\}$; $Codom(R) = \{y : \exists x,\ xRy\}$

Let X, Z two sets and f a function, we note $Codom(f)$ by $f(X)$ and:

- $f \restriction Y = \{(y, f(y)) : y \in Y\}$
- $f : X \mapsto Z$ to express that $X = Dom(f)$ and $f(X) \subseteq Z$.
- $f : X \hookrightarrow Z$ to express that $X \subseteq Dom(f)$ and $f(X) \subseteq Z$.

Definition 2.2 Let $P = (X, \sqsubseteq)$ be a poset, $y \in X$ and $Y \subseteq X$. The following sets are respectively the *ceiling, floor* of y in P:

(i) $\lceil y \rceil_P = \{x \in X \setminus \{y\} : \ y \sqsubseteq x \ \wedge \ \forall z \in X \, (y \sqsubseteq z \sqsubseteq x \ \Rightarrow \ z{=}x \vee z{=}y)\}$

(ii) $\lfloor y \rfloor_P = \{x \in X \setminus \{y\} : \ x \sqsubseteq y \ \wedge \ \forall z \in X \, (x \sqsubseteq z \sqsubseteq y \ \Rightarrow \ z = x \vee z = y)\}$

For $Y \subseteq P$, define $\lceil Y \rceil_P = \cup_{y \in Y} \lceil y \rceil_P$ and $\lfloor Y \rfloor_P = \cup_{y \in Y} \lfloor y \rfloor_P$.

We drop the index P in the notations above as soon as there is no ambiguity on the poset P. A similar convention is used whenever a subscript can be removed without loss of clarity. We define now the *Hasse Diagram* of a poset.

Definition 2.3 For every poset $P = (X, \sqsubseteq)$, the *Hasse Diagram of* P is the pair $(X, \prec)$ where the binary relation $\prec$ is defined on X^2 by: $x \prec y$ iff $x \in \lfloor y \rfloor$.

Remark 2.4 Clearly $\prec \subseteq \sqsubseteq$ and moreover: $x \sqsubseteq y$ iff there exists $x_1, \ldots x_n \in X$ such that $x = x_1 \prec x_2 \prec \ldots \prec x_n = y$.

Definition 2.5 An *atomic poset* is a poset of the form $A_Y = (Y, \Delta_Y)$.

2.2 Posets as (simple) directed graphs

It is convenient to interpret posets as particular cases of (simple) directed graphs defined here as pairs (X, R) where R is a binary relation on X. The (undirected) graph underlying a digraph (X, R) is the graph with vertices X and edges the pairs $e = \{x, y\}$ such that $(x, y) \in R$. A path linking two nodes x, y in a graph is an alternative sequence $x = x_1 e_1 x_2 \ldots x_n e_n x_{n+1} = y$ such that, for every $i \in \{1, \ldots, n\}$, $e_i = \{x_i, x_{i+1}\}$. We write $x \smile y$ when two nodes are linked by a path. We recall that a subset Y of a graph is connected if $x \smile y$ for all distinct $x, y \in Y$, and a component if it is maximal connected.

Definition 2.6 $Y \subseteq X$ is *zigzag closed* in the digraph (X, R) if it is an union of components of the underlying graph, that is, if $x \in X, y \in Y$ and $x \smile y$ then $x \in Y$.

2.3 Simply-labeled digraph and labeled partial order

Definition 2.7 A *simply-labeled digraph (sldg)* is a pair (P, ℓ) where $P = (X, R)$ is a digraph and $\ell : X \mapsto \mathcal{L}$ is a bijection where $\mathcal{L}$ is a totally ordered set. A *labeled partial order (lpo)* is a sldg where $P = (X, \sqsubseteq)$ is a poset and the bijection ℓ is increasing, that is, for every $x, y \in X$, $x \sqsubseteq y \Rightarrow \ell(x) \leq \ell(y)$.

An useful and intuitive way to represent lpo's (and sldg's) is given in the example below. The poset is represented by its *Hasse Diagram* (reflexive and transitive edges are not drawn). Since the names of nodes are not relevant, they are simply represented by anonymous circles where the corresponding labels are written.

Example 2.8 Two lpo's $F_1 = (P, \ell_1)$ and $F_2 = (P, \ell_2)$ with labels $\{1, 3, 4\}$, $P = (\{x, y, z\}, \ \{(x, z), (y, z)\})$, and $\ell_1(z) = \ell_2(z) = 4$, $\ell_1(x) = 1$, $\ell_1(y) = 3$, $\ell_2(x) = 3$

and $\ell_2(y) = 1$:

The lpo F_1

$$\begin{tikzcd}
 & 4 & \\
1 \arrow[ur] & & 3 \arrow[ul]
\end{tikzcd}$$

The lpo F_2

$$\begin{tikzcd}
 & 4 & \\
3 \arrow[ur] & & 1 \arrow[ul]
\end{tikzcd}$$

We remark that F_1 and F_2 are the only possible lpo's for P and $\{1, 3, 4\}$.

Definition 2.9 $(P, \ell) \upharpoonright Y$ denotes the pair $(P \upharpoonright Y,\ \ell \upharpoonright Y)$, for every sldg (P, ℓ).

Remark 2.10 If ℓ is a labeling on the poset $(X, \sqsubseteq)$ and $(X, \sqsubseteq')$ is a poset such that $\sqsubseteq' \subseteq \sqsubseteq$ then ℓ is a labeling on $(X, \sqsubseteq')$.

2.4 Random Structure and Random Preservation

A *Random Structure (RS)* is the set of all possible lpo's for given poset P and set of labels $\mathcal{L}$. It is denoted by $R_{\mathcal{L}}(P)$, or simply $\mathcal{R}$. For instance Example 2.8 represents the RS on $P = (\{x, y, z\},\ \{(x, z), (y, z)\})$ with $\mathcal{L} = \{1, 3, 4\}$.

An operation or a program applied on different inputs may leed to the same output. In order to keep track of the number of inputs corresponding to a given output, the notion of *output multiset* is defined below. But first, we introduce *multisets* and two of their constructors.

Definition 2.11 A *multiset* is a function $M : X \mapsto \mathbb{N} \setminus \{0\}$. For every set x, the *multiplicity* $mul_M(x)$ of x in the multiset M is defined as $M(x)$ in case $x \in X$, and as 0 otherwise.

Definition 2.12 Let M, N be multisets, Z a set, and $k \in \mathbb{N}$:

- $Z \times k = \{(x, k) : x \in Z\}$
- $M \oplus N = \{(x, k) : x \in Dom(M) \cup Dom(N)\ \wedge\ k = mul_M(x) + mul_N(x)\}$

Definition 2.13 Let Op be an operation and $Z \subseteq_f Dom(Op)$, we define respectively the *output set* and *output multiset* of Op applied to Z as:

- $Op(Z) = \{Op(x) : x \in Z\}$
- $\mathcal{O}_{Op}(Z) = \{(y, k) :\ y \in Op(Z)\ \wedge\ k = Card(Op^{-1}(y))\}$

Definition 2.14 Two RS's are *isomorphic* if their underlying posets are.

Definition 2.15 An operation Op is Random Preserving (RP) on respectively a random structure $\mathcal{R}$, a set of lpo's $\mathbb{F}$ if:

- There exist stricly positive integers $n, K_1, \ldots, K_n$ and random structures $\mathcal{R}_1, \ldots \mathcal{R}_n$ such that:
  $\mathcal{O}_{Op}(\mathcal{R}) = \mathcal{R}_1 \times K_1 \oplus \ldots \oplus \mathcal{R}_n \times K_n$.
- It is RP on every random structure $\mathcal{R} \subseteq \mathbb{F}$.

If moreover the $\mathcal{R}_i$'s are all isomorphic then the operation Op is said *Strongly random preserving (SRP)* on $\mathcal{R}$ (resp. on $\mathbb{F}$ if it is RSP on every $\mathcal{R} \subseteq \mathbb{F}$).

Note that the MOQA unary product and split are RP on their domains while the projection is SRP (cf. [7]). But, as already mentioned these operations are more than simply RP in the above sense. Indeed, for each of these operations Op, there also exists a function able to compute from every RS $\mathcal{R} \subseteq Dom(Op)$ the integer n, the K_i's and $\mathcal{R}_i$'s of Definition 2.15. Thus, we speak of *Automatic Random Preservation* concerning MOQA basic operations. We will discuss again this notion in the Conclusion.

3 Isolated subsets and Local Transformation

From now on, it is convenient to work with a fixed infinite set O. We define $\mathcal{D}$ as the set of all finite digraphs with ground set $X \subseteq O$. We define similarly the set $\mathcal{O}$ of all finite posets with ground set $X \subseteq O$. We denote by $\mathcal{S}$ the sets of all sldg's (P, ℓ) with $P \in \mathcal{D}$ and ℓ such that $Codom(\ell) \subseteq \mathbb{N}$. Finally, we denote by $\mathcal{F}$ the sets of all lpo's (P, ℓ) with $P \in \mathcal{O}$ and ℓ such that $Codom(\ell) \subseteq \mathbb{N}$.

3.1 Isolated subset of a Poset

First we recall the definition of an isolated subset in [7]. Then we give the equivalent formulation we use. We recall that an element of a poset is minimal if it has no other element below it, it is maximal if it has no other element above it.

Definition 3.1 For all posets P and $Z \subseteq Y \subseteq X_P$, we denote by $m_Y(Z)$ the set of minimal elements of Z in the poset $P \upharpoonright Y$, and by $M_Y(Z)$ the set of maximal elements of Z in the poset $P \upharpoonright Y$.

Definition 3.2 Let $P = (X, \sqsubseteq)$ be a poset. A subset $Y \subseteq X$ is:

- *Atomic in P* if $\sqsubseteq\upharpoonright Y = \Delta_Y$
- *Isolated* in P if it satisfies the two following *isolation* conditions:
(i) $\lfloor Y \setminus m_Y(Y) \rfloor \subseteq Y$ and $\forall x, y \in m_Y(Y),\ \lfloor x \rfloor = \lfloor y \rfloor$
(ii) $\lceil Y \setminus M_Y(Y) \rceil \subseteq Y$ and $\forall x, y \in M_Y(Y),\ \lceil x \rceil = \lceil y \rceil$
- *Atomic Isolated* if atomic and isolated.

We give now a new notation and an equivalent characterisation of isolation.

Definition 3.3 For a poset $P = (X, \sqsubseteq)$, $x \in X$ and $Y \subseteq X$, we write $x \sqsubseteq Y$ (resp. $Y \sqsubseteq x$) to express that $x \sqsubseteq y$ (resp. $y \sqsubseteq x$) for every $y \in Y$.

Lemma 3.4 *A set $Y \subseteq X$ is isolated in $P = (X, \sqsubseteq)$ iff it verifies:*

(i) *For all $x \in X \backslash Y$ and $y \in Y$: $x \sqsubseteq y$ implies $x \sqsubseteq Y$*

(ii) *For all $x \in X \backslash Y$ and $y \in Y$: $y \sqsubseteq x$ implies $Y \sqsubseteq x$.*

Proof. We prove that the isolation condition i of Definition 3.2 is equivalent the point i above. The fact that condition ii and point ii are equivalent can shown using a dual reasoning. The result follows trivially from these two equivalences.

We show first that condition i of isolation implies point i. Let $x \sqsubseteq y$, with $x \in X \setminus Y$ and $y \in Y$. Let $x_1, \ldots, x_n \in X$ be a sequence given by Remark 2.4 such that $x = x_1 \prec x_2 \prec \ldots \prec x_n = y$. Let now i be the biggest index such that $x_{i-1} \notin Y$. We have $n \geq i \geq 2$ since $x \notin Y$, and $x_i \in m_Y(Y)$ (otherwise we would have $x_{i-1} \in \lfloor Y \setminus m_Y(Y) \rfloor \subseteq Y$ contradicting $x_{i-1} \notin Y$). Let now $y' \in Y$. Clearly there is $z \in m_Y(Y)$ such that $z \sqsubseteq y'$. We get then $x_{i-1} \in \lfloor z \rfloor$ since $\prec \subseteq \sqsubseteq$ and $\lfloor z \rfloor = \lfloor x_i \rfloor$ by isolation of Y. We conclude $x \sqsubseteq y'$ by transitivity of $\sqsubseteq$.

Suppose now that Y verifies point i, we show that isolation condition i holds. We show first that i implies $\lfloor Y \setminus m_Y(Y) \rfloor \subseteq Y$. Indeed let $x \in \lfloor y \rfloor$ for some $y \in Y \setminus m_Y(Y)$, and suppose $x \notin Y$. Since y is not minimal in $P \restriction Y$, there exists $y' \in Y$ distinct from y such that $y' \sqsubseteq y$. We have then $x \sqsubseteq y'$ by point i with $x \neq y'$ since $y' \in Y$. That contradicts $x \in \lfloor y \rfloor$, and so by contradiction $x \in Y$. Take now $x, y \in m_Y(Y)$, we have to prove $\lfloor x \rfloor = \lfloor y \rfloor$. We show $\lfloor x \rfloor \subseteq \lfloor y \rfloor$, the converse inclusion can be proved by a similar reasoning. So let $z \in \lfloor x \rfloor$, we have $z \notin Y$ by minimality of x and so $z \sqsubseteq y$ by point i. Now if $z \notin \lfloor y \rfloor$, there exist $z' \neq z, y$ such that $z \sqsubseteq z' \sqsubseteq y$. Again by minimality of y, we have $z' \notin Y$, and so $z' \sqsubseteq x$ by point i. That contradicts $z \in \lfloor x \rfloor$, and so by contradiction $z \in \lfloor y \rfloor$. □

Definition 3.5 Let $I, J, Y \subseteq_f O$:

- $\mathcal{D}_Y = \{P \in \mathcal{D} : Y \subseteq X_P\}$; $\quad \mathcal{O}_Y = \{P \in \mathcal{O} : Y \subseteq X_P\}$
- $\mathcal{A}_Y = \{P \in \mathcal{O} : Y$ is atomic in P$\}$; $\quad \mathcal{I}_Y = \{P \in \mathcal{O} : Y$ is isolated in $P\}$
- $\mathcal{Z}_{IJ} = \{P \in \mathcal{O}_{I \cup J} : I, J$ are zigzag closed in $P \restriction I \cup J\}$
- $\mathcal{AI}_Y = \mathcal{A}_Y \cap \mathcal{I}_Y$; $\quad \mathcal{J}_{IJ} = \mathcal{I}_{I \cup J} \cap \mathcal{Z}_{IJ}$

3.2 Local Transformations

Definition 3.6 A function Θ is a *local transformation of Y on $\mathbb{D} \subseteq \mathcal{D}$* if for every digraph $P = (X, R) \in \mathbb{D}$:

(i) $\Theta(P)$ is a digraph with ground set X.

(ii) $R' \setminus Y^2 = R \setminus Y^2$, where R' denotes the binary relation on X in $\Theta(P)$.

If moreover, $\Theta(P)$ is a poset for every poset $P \in \mathbb{D}$, then it is said a local *poset* transformation on $\mathbb{D}$.

In other words, a local transformation of Y modifies the digraph P at most locally on Y. In particular, condition ii in the definition implies that $R' \restriction (X \setminus Y) = R \restriction (X \setminus Y)$. We link now the two notions of isolation and local transformation.

Lemma 3.7 *For every local transformation Θ of Y and every poset $P \in \mathcal{I}_Y$: $\Theta(P)$ is a poset iff $\Theta(P) \restriction Y$ is.*

Proof. Let $P = (X, \sqsubseteq)$ be a poset where Y is isolated and let $\Theta(P) = (X, \sqsubseteq')$. If $\Theta(P)$ is a poset then $\Theta(P) \restriction Y$ clearly is. Now suppose that $\Theta(P) \restriction Y$ is a poset on Y. Note first that the reflexivity and antisymmetry of $\Theta(\sqsubseteq)$ on X comes easily from its reflexivity and antisymmetry on Y and condition ii of Definition 3.6. It remains to prove the transitivity. So suppose $x \sqsubseteq' y \sqsubseteq' z$.

If $(x,y) \notin Y^2$ and $(y,z) \notin Y^2$, then by condition ii we get $x \sqsubseteq y \sqsubseteq z$, and so $x \sqsubseteq z$ since $\sqsubseteq$ is transitive by hypothesis on P. Now if $x \notin Y$ or $z \notin Y$ then condition ii gives $x \sqsubseteq' z$. We prove that the other case is impossible. Indeed, if $x, z \in Y$ then $y \notin Y$ (since $(x,y) \notin Y^2$). But $x \sqsubseteq y$ implies $Y \sqsubseteq y$ and so $z \sqsubseteq y$ by Lemma 3.4, which is impossible since $y \sqsubseteq z$.

Suppose now $(x,y) \in Y^2$ or $(y,z) \in Y^2$. Notice first that if $x,y,z \in Y$ then the result is immediate by transitivity of $\sqsubseteq'$ on Y. So suppose as a first case that $x, y \in Y$, and $z \notin Y$. We get $y \sqsubseteq z$ by condition ii, and since Y is isolated, we get $Y \sqsubseteq z$, and so $x \sqsubseteq z$, and by condition ii, $x \sqsubseteq' z$. The second possible case $x \notin Y$ and $y, z \in Y$, is similar. □

We extend now the notion of local transformation to sldg's and lpo's, and then extend Lemma 3.7 to lpo's.

Definition 3.8 A *local sldg transformation of Y on $\mathbb{S} \subseteq \mathcal{S}$* is a function which associates to every $(P,\ell) \in \mathbb{S}$ a sldg (P',ℓ') such that:

(i) The function $P \mapsto P'$ is a local transformation of Y.

(ii) $\ell(X_P) = \ell'(X_P)$

(iii) $\ell(x) \neq \ell'(x) \Rightarrow x \in Y$, for every $x \in X_P$.

If moreover the function $P \mapsto P'$ is a poset transformation and ℓ' is a labeling of P' for every ℓ, then Θ is said a *local lpo transformation.*

Lemma 3.9 *For every local transformation Θ of Y and every lpo (P,ℓ) such that $P \in \mathcal{I}_Y$: $\Theta(P,\ell)$ is a lpo iff $\Theta(P,\ell)\restriction Y$ is.*

Proof. Let $P = (X, \sqsubseteq)$ and $\Theta(P,\ell) = (P',\ell')$. Let also $\mathcal{L} = \ell(X)$, $\mathcal{L}_Y = \ell(Y)$ and $\bar{\mathcal{L}} = \ell(X \setminus Y)$. Remark first that condition ii of Definition 3.8 implies that $\ell'(X) = \mathcal{L}$, condition iii that $\ell'(X \setminus Y) = \bar{\mathcal{L}}$, and so $\ell'(Y) = \mathcal{L}_Y$. Remark also that by condition i in Definition 3.8 and Lemma 3.7, P' is a poset iff its restriction to Y is. Hence, in particular and since ℓ' increasing on X implies ℓ' increasing on $Y \subseteq X$, we have immediatly that if $\Theta(P,\ell)$ is a lpo then $\Theta(P,\ell)\restriction Y$ is.

Suppose now that $(P',\ell')\restriction Y$ is a lpo, and so ℓ' is increasing from Y to $\mathcal{L}_Y$. Since P' is a poset, it remains to prove that ℓ' is increasing on X. So let $x, y \in X$ such that $x \sqsubseteq' y$. We have to prove $\ell'(x) \leq \ell'(y)$. We can suppose $x \notin Y$ or $y \notin Y$, otherwise the result is given by the hypothesis. Moreover, if $x, y \in X \setminus Y$, it is immediate by condition iii of Definition 3.8. There remains two cases $x \in Y \ \wedge \ y \notin Y$ or $x \notin Y \ \wedge \ y \in Y$. We show the first one, the second is similar. By $y \notin Y$, we have $\ell'(y) = \ell(y)$, and since $\{x,y\} \in \sqsubseteq' \setminus Y^2 = \sqsubseteq \setminus Y^2$ (condition i of Definition 3.8 and condition ii of Definition 3.6), $x \sqsubseteq y$. Moreover, since Y is by hypothesis isolated in P, we get that $Y \sqsubseteq y$. Since ℓ is a labeling of P, we have $\ell(Y) = \mathcal{L}_Y \leq \ell(y) = \ell'(y)$, and so in particular $\ell'(x) \leq \ell'(y)$. □

4 Functionally-generalised operations

We define in this section the generalised operations as well as the corresponding MOQA operations. For that we introduced first some auxiliary functions.

4.1 Auxiliary operations on posets and lpo's

Definition 4.1 For $(X, \sqsubseteq)$ a poset and $Y \subseteq X$, define:
$Y^{\uparrow} = \{x \in X \setminus Y : Y \sqsubseteq x\}$; $\quad Y^{\downarrow} = \{x \in X \setminus Y : x \sqsubseteq Y\}$

We remark that $Y^{\uparrow}$ is closed under upper bound and $Y^{\downarrow}$ under lower bound.

Definition 4.2 For every set $Y \subseteq_f O$, we define the operation $\imath_Y : \mathcal{O}_Y \mapsto \mathcal{D}_Y$ by setting $\imath_Y(X, \sqsubseteq) = (X, \sqsubseteq_{\imath_Y})$ where the binary relation $\sqsubseteq_{\imath_Y}$ on X is defined by: $z \sqsubseteq_{\imath_Y} z'$ iff one of the following clauses is satisfied:

(i) $z, z' \in X \setminus Y \ \wedge \ z \sqsubseteq z'$

(ii) $z, z' \in Y \ \wedge \ z \sqsubseteq z'$

(iii) $z \in Y^{\downarrow} \ \wedge z' \in Y$

(iv) $z \in Y \ \wedge z' \in Y^{\uparrow}$

The intuitive idea underlying the definition of $\sqsubseteq_{\imath_Y}$ is to remove any link between and element of Y and an element x of $X \setminus Y$ except if x is below all the elements of Y or if x is above all the elements of Y.

Lemma 4.3 *For every $Y \subseteq_f O$, we have $\imath_Y : \mathcal{O}_Y \mapsto \mathcal{I}_Y$.*

Proof. Remark first that if $\imath_Y(P)$ is a poset and $x \sqsubseteq_{\imath_Y} y$, where $x \in X \setminus Y$ and $y \in Y$, then the only clause of Definition 4.2 which can apply is clause iii taking $x = z$ and $y = z'$. From that, clearly condition i of Lemma 3.4 holds. The same remark applies in case $y \sqsubseteq_{\imath_Y} x$ for condition ii of the lemma. That shows that Y is isolated in $\imath_Y(P)$. It remains to prove that if $P = (X, \sqsubseteq) \in \mathcal{O}_Y$ then $\sqsubseteq_{\imath_Y}$ is a p.o. on X.

Note first that the reflexivity of $\sqsubseteq_{\imath_Y}$ on X is ensured by the two first clauses (taking $x = y$). Note also that if $x \in Y \ \wedge y \in Y^{\uparrow}$, we have $x \sqsubseteq y$ by definition of $Y^{\uparrow}$. The same remark hold if $x \in Y^{\downarrow} \ \wedge y \in Y$. Hence, we have $\sqsubseteq_{\imath_Y} \subseteq \sqsubseteq$. From that we get trivially the antisymetry of $\sqsubseteq_{\imath_Y}$. Then, the transitivity follows by a straightforward reasoning by case from the hypothesis $x \sqsubseteq_{\imath_Y} y \ \wedge \ y \sqsubseteq_{\imath_Y} z$ and using the different clauses of Definition 4.2. □

Lemma 4.4 *For every poset $P \in \mathcal{I}_Y$ we have $\imath_Y(P) = P$.*

Proof. Let $P = (X, \sqsubseteq)$ such that Y is isolated in P. We have immediatly $\sqsubseteq_{\imath_Y} \subseteq \sqsubseteq$. Now suppose $x \sqsubseteq y$. If $x, y \in X \setminus Y$ or $x, y \in Y$, we have immediatly $x \sqsubseteq_{\imath_Y} y$. Two cases remain, $x \in Y \ \wedge \ y \in X \setminus Y$ or $x \in X \setminus Y \ \wedge \ y \in Y$. In the first case, the isolation of Y and $x \sqsubseteq y$ give $Y \sqsubseteq y$ (Lemma 3.4), that is, $y \in Y^{\uparrow}$ and so $x \sqsubseteq_{\imath_Y} y$. In the second case, we get similarly $x \in Y^{\downarrow}$ and so again $x \sqsubseteq_{\imath_Y} y$. □

Definition 4.5 For all posets $P = (X, \sqsubseteq)$ and $I, J \subseteq X$:
$I \star_P J = \{(x, y) \in \sqsubseteq : x \in I \ \wedge \ \exists z \in J, x \sqsubseteq y \sqsubseteq z\}$

We remark that $I \times J \cap \sqsubseteq$ is a subset of $I \star_P J$ (taking $y = z$).

Definition 4.6 For all disjoint $I, J \subseteq_f O$, define the function $\wp_{IJ} : \mathcal{O}_{I\cup J} \mapsto \mathcal{D}_{I\cup J}$ by setting $\wp_{IJ}(X, \sqsubseteq) = (X, \sqsubseteq_{\wp_{IJ}})$ where: $\sqsubseteq_{\wp_{IJ}} = \sqsubseteq \setminus (I \star J \cup J \star I)$

Definition 4.7 For every $Y \subseteq_f O$, we define the operation $@_Y : \mathcal{O}_Y \mapsto \mathcal{D}_Y$ by setting $@_Y(X, \sqsubseteq) = (X, \sqsubseteq_{@_Y})$ where: $\sqsubseteq_{@_Y} = \sqsubseteq \setminus (Y \star Y \setminus \Delta_Y)$

Intuitively, $\sqsubseteq_{\wp_{IJ}}$ is obtained from $\sqsubseteq$ by deleting any path between an element of I and an element of J inside the underlying graph, creating some kind of "parallel" sub-posets in $\wp_{IJ}(X, \sqsubseteq)$. The p.o. $\sqsubseteq_{@_Y}$ is obtained by deleting any path between distinct elements of Y.

Lemma 4.8 *For all disjoint $I, J \subseteq_f O$, we have $\wp_{IJ} : \mathcal{O}_{I\cup J} \mapsto \mathcal{Z}_{IJ}$*

Proof. Let $P = (X \ \sqsubseteq) \in \mathcal{O}_{I\cup J}$. We remark that, by definition of $\wp_{IJ}$, we get easily $\sqsubseteq_{\wp_{IJ}} \cap I \times J = \emptyset = J \times I \cap \sqsubseteq_{\wp_{IJ}}$. The fact that if $\wp_{IJ}(P)$ is a poset then $\wp_{IJ}(P) \in \mathcal{Z}_{IJ}$ comes easily from that.

We prove that $\sqsubseteq_{\wp_{IJ}}$ is a p.o. on X. Remark first that $\sqsubseteq_{\wp_{IJ}} \subseteq \sqsubseteq$, and so the antisymetry of $\sqsubseteq$ implies immediatly the antisymetry of $\sqsubseteq_{\wp_{IJ}}$. Now, the reflexivity comes straighforwardly from the hypothesis I and J disjoint. Then, to prove the transitivity of $\sqsubseteq_{\wp_{IJ}}$ suppose $x \sqsubseteq_{\wp_{IJ}} y \sqsubseteq_{\wp_{IJ}} z$. We have $x \sqsubseteq y \sqsubseteq z$ and so $x \sqsubseteq z$. Suppose now that $x \not\sqsubseteq_{\wp_{IJ}} z$. By definition of $\sqsubseteq_{\wp_{IJ}}$, we get $(x, z) \in I \star J$ or $(x, z) \in J \star I$. We prove that the first case is contradictory, the second case can be proved contradictory in a similar way. Indeed, if $(x, z) \in I \star J$, we have $x \in I$ and there exist $z' \in J$ such that $x \sqsubseteq z \sqsubseteq z'$. Now since $x \sqsubseteq y \sqsubseteq z'$, we have also $(x, y) \in I \star J$, and so $(x, y) \notin \sqsubseteq_{\wp_{IJ}}$. That contradicts the hypothesis $x \sqsubseteq_{\wp_{IJ}} y$. □

Lemma 4.9 *For every $Y \subseteq_f O$, we have $@_Y : \mathcal{O}_Y \mapsto \mathcal{A}_Y$.*

Proof. Let $P = (X, \sqsubseteq) \in \mathcal{O}_Y$. We remark that, by definition of $\sqsubseteq_{@_Y}$, we get easily $\sqsubseteq_{@_Y} \cap Y \times Y = \Delta_Y$. The fact that if $@_Y(P)$ is a poset then Y is atomic in $@_Y(P)$ comes then straightforwardly. In order to prove that $\sqsubseteq_{@_Y}$ is a p.o. on X, remark first that $\sqsubseteq_{@_Y} \subseteq \sqsubseteq$, and so we have immediatly the antisymetry of $\sqsubseteq_{@_Y}$ from the antisymetry of $\sqsubseteq$. The reflexivity is obvious from the definition. Finally the transitivity is proved in the same way than for $\sqsubseteq_{\wp_{IJ}}$ in Lemma 4.8. □

Lemma 4.10 *For all $I, J, Y \subseteq_f O$ such that I, J are disjoint:*

(i) *$\wp_{IJ}$ is a local poset transformation of $I \cup J$ on $\mathcal{I}_{I\cup J}$.*

(ii) *$@_Y$ is a local poset transformation of Y on $\mathcal{I}_Y$.*

Proof. We have to prove condition ii of Definition 3.6 for a poset $P = (X, \sqsubseteq)$ where $I \cup J$ and Y are isolated. We show the result for $\wp_{IJ}$, the proof is similar for $@_Y$.

Let $Z = I \cup J$. We have $\sqsubseteq_{\wp_{IJ}} \subseteq \sqsubseteq$ and so immediatly $\sqsubseteq_{\wp_{IJ}} \setminus Z^2 \subseteq \sqsubseteq \setminus Z^2$. We reason now by contradiction. So let $(x, y) \in \sqsubseteq \setminus Z^2$ such that $(x, y) \notin \sqsubseteq_{\wp_{IJ}} \setminus Z^2$. By the first hypothesis we have $x \sqsubseteq y$, and also $x \notin I \cup J$ or $y \notin I \cup J$. Moreover, by the second hypothesis, we have $(x, y) \in I \star J$ or $(x, y) \in J \star I$ since $\sqsubseteq_{\wp_{IJ}} = \sqsubseteq \setminus (I \star J \cup J \star I)$. Note that the case $x \notin I \cup J$ is then impossible since the definition of $\star$ would implies $x \in I$ or $x \in J$. So suppose now $y \notin I \cup J$. We show that the case $(x, y) \in I \star J$ is impossible, a similar reasoning holds for the case $(x, y) \in J \star I$. Indeed, suppose the first case, we have $x \in I$ and there exists $z \in J$ such that $y \sqsubseteq z$. Since $x \sqsubseteq y \Rightarrow I \cup J \sqsubseteq y$ by weak-isolation, we get also $z \sqsubseteq y$. Hence we have $y = z \in J$ which is contradictory with $y \notin I \cup J$. □

Lemma 4.11 *Let $I, J, Y \subseteq_f O$ such that I and J are disjoint, and $P \in \mathcal{O}$:*

(i) *If $P \in \mathcal{J}_{IJ}$ then $\wp_{IJ}(P) = P$.*

(ii) *If $P \in \mathcal{AI}_Y$ then $@_Y(P) = P$.*

Proof. Suppose I, J, Y and $P = (X, \sqsubseteq)$ satisfying the hypothesis of the lemma:

(i) We have to prove $\sqsubseteq_{\wp_{IJ}} = \sqsubseteq$. We remark that $\sqsubseteq_{\wp_{IJ}} \subseteq \sqsubseteq$, and so it remains to show $\sqsubseteq \subseteq \sqsubseteq_{\wp_{IJ}}$, which is done by contradiction. Suppose $x \sqsubseteq y$ and $x \not\sqsubseteq_{\wp_{IJ}} y$. We get $(x, y) \in I \star J$ or $(x, y) \in J \star I$. We show that the first case is contradictory, the second case is similar. Indeed, if $(x, y) \in I \star J$ we get $x \in I$ and there exist $z \in J$ such that $x \sqsubseteq y \sqsubseteq z$. So we get $x \sqsubseteq z$ and I is not zigzag closed in $P \restriction I \cup J$ (since $z \in J$ and $I \cap J = \emptyset$ by hypothesis).

(ii) We have to prove $\sqsubseteq_{@_Y} = \sqsubseteq$. We remark that $\sqsubseteq_{@_Y} \subseteq \sqsubseteq$, and so it remains to prove the converse inclusion which is done by contradiction. So suppose $x \sqsubseteq y$ and $x \not\sqsubseteq_{@_Y} y$. We get $(x, y) \in Y \star Y$ with $x \neq y$. We have $x \in Y$ and there exists $z \in Y$ such that $x \sqsubseteq y \sqsubseteq z$. We have $y \notin Y$, otherwise Y would be not atomic, and so by isolation we get $Y \sqsubseteq y$. So in particular $z \sqsubseteq y$, and then $z = y$, which is impossible.

□

Definition 4.12 For all $I, J, Y \subseteq_f O$ where I, J are disjoint:
(i) $\jmath_{IJ} = \wp_{IJ} \circ \imath_{I \cup J}$ *(ii)* $@\imath_Y = @_Y \circ \imath_{I \cup J}$

Lemma 4.13 *For all $I, J, Y \subseteq_f O$ where I, J are disjoint:*

(i) *If $P \in \mathcal{J}_{IJ}$ then $\jmath_{IJ}(P) = P$ and $\jmath_{IJ} : \mathcal{O}_{I \cup J} \mapsto \mathcal{J}_{IJ}$.*

(ii) *If $P \in \mathcal{AI}_Y$ then $@\imath_Y(P) = P$ and $@\imath_Y : \mathcal{O}_Y \mapsto \mathcal{AI}_Y$.*

Proof. The first point is a direct application of Lemmas 4.4, 4.11 and Lemmas 4.3, 4.8; the second of Lemmas 4.4, 4.11 and Lemmas 4.3, 4.9. □

We extend now Definition 3.5 to $\mathcal{F}$ as well as $\jmath_{IJ}$ and $@\imath_Y$ to lpo's.

Definition 4.14 $\mathcal{F}_Y$ (resp. $\mathcal{FAI}_Y$ and $\mathcal{FJ}_{IJ}$) is the set of lpos of $\mathcal{F}$ with posets in $\mathcal{O}_Y$ (resp. in $\mathcal{AI}_Y$ and $\mathcal{J}_{IJ}$), for all $I, J, Y \subseteq_f O$.

Definition 4.15 For all $I, J, Y \subseteq_f O$ with I, J disjoint and (P, ℓ) respectively in $\mathcal{FJ}_{IJ}$ and $\mathcal{FAI}_Y$, define: $\jmath_{IJ}(P, \ell) = (\jmath_Y(P), \ell)$ and $@\imath_Y(P, \ell) = (@\imath_Y(P), \ell)$.

Since $\sqsubseteq_{\imath_Y}$, $\sqsubseteq_{\wp_{IJ}}$ and $\sqsubseteq_{@_Y}$ are subsets of $\sqsubseteq$, the following fact comes immediatly from Remark 2.10 and Lemma 4.13.

Fact 4.16 *For all $I, J, Y \subseteq_f O$ such that I, J are disjoint:*

(i) *If $(P, \ell) \in \mathcal{FJ}_{IJ}$ then $\jmath_{IJ}(P, \ell) = (P, \ell)$ and $\jmath_{IJ} : \mathcal{F}_{I \cup J} \mapsto \mathcal{FJ}_{IJ}$.*

(ii) *If $(P, \ell) \in \mathcal{FAI}_Y$ then $@\imath_Y(P, \ell) = (P, \ell)$ and $@\imath_Y : \mathcal{F}_Y \mapsto \mathcal{FAI}_Y$.*

4.2 The generalised projection

The language MOQA has a projection operator $Proj$ which takes two arguments: a lpo ℓ and a set Y. Its semantics are given in [7] by a class of projections $Proj_Y$ (the first argument is used as a parameter). These projections operations are defined on lpo's only when Y is isolated in the underlying poset. A trivial way to extend them would be to set $Proj_Y(P, \ell)$ to return a special value "Error" when $P \notin \mathcal{I}_Y$. We introduce below more interesting extensions where $Proj_Y(P, \ell)$ is defined without an error message as soon as $Y \subseteq X_P$. As already mentioned, this condition is more natural and easy to control than the isolation one. The operations are defined in two steps: on the poset first and then on the lpo. We remark that it is done in the same way in [7] and so Proposition 4.19 is immediate.

Definition 4.17 Let $P = (X, \sqsubseteq) \in \mathcal{O}$ a poset and $Y \subseteq_f O$, define:

$$proj^G_Y(P) = \begin{cases} (Y, \sqsubseteq\restriction Y) \text{ if } & Y \subseteq X \\ \text{“}Error'' & \text{else} \end{cases}$$

Definition 4.18 Let $(P, \ell) \in \mathcal{F}$ and $Y \subseteq_f O$, define:

$$Proj^G_Y(P, \ell) = \begin{cases} (proj_Y(P), \ \ell \restriction Y) \text{ if } & Y \subseteq X_P \\ \text{“}Error'' & \text{else} \end{cases}$$

Proposition 4.19 *For all $Y \subseteq_f O$ and $(P, \ell) \in \mathcal{F}_Y$, $Proj^G_Y(P, \ell)$ is a lpo on $P \restriction Y$. Moreover, if $P \in \mathcal{I}_Y$ then $proj^G_Y(P) = proj_Y(P)$ and $Proj^G_Y(P, \ell) = Proj_Y(P, \ell)$.*

4.3 The generalised (unary) product

The language MOQA has an operator $\bigotimes$ which takes three arguments: two sets I and J, and a lpo. Its semantics are given in [7]. We present this semantics and our generalisation below. It is done in two steps, starting with the operation $\otimes$ on posets.

4.3.1 The unary product on posets

The semantics of $\bigotimes$ described here is slightly different from the original one of [7]. Nevertheless, the equivalence of the two formulations is straightforward. Remark also that we generalise Definition 4.20 to the case where I and J are zigzag closed in $I \cup J$ (but not necessary components as in the original definition).

Definition 4.20 For all $I, J \subseteq_f O$ disjoint and posets $P = (X, \sqsubseteq) \in \mathcal{J}_{IJ}$, we define the *unary product of P relatively to I, J* by: $\otimes_{IJ}(P) = (X, \sqsubseteq \cup I \times J)$.

We remark that $\otimes_{IJ}(P)$ is a poset on X. Indeed, the reflexivity of $\sqsubseteq \cup I \times J$ is immediate, the transitivity follows easily from isolation and the antisymetry from zigzag closure. It is easy to check that Definition 4.20 is equivalent to the one in [7] when I, J are components. Another way to prove the result is to use Lemma 3.7 and the easy fact that $\otimes_{IJ}$ is a local transformation of $I \cup J$ on $\mathcal{J}_{IJ}$. The operation was called "unary" product because it is a partial unary operation on the class of finite posets, and "product" because it performs a kind of (binary!) product of I and J inside X. A trivial extension of $\otimes_{IJ}$ is define by setting $\otimes_{IJ}(P) =$ "*Error*" whenever $P \notin \mathcal{J}_{IJ}$. A more interesting one is introduced below.

Definition 4.21 For all posets $P = (X, \sqsubseteq)$ and $I, J \subseteq_f O$ disjoint, define:

$$\otimes^G_{IJ}(P) = \begin{cases} \otimes_{IJ} \circ \jmath_{IJ}(P) & \text{if} \quad I, J \subseteq X \\ \text{“}Error\text{''} & \text{else} \end{cases}$$

A direct application of point i of Lemma 4.13 gives the following result.

Lemma 4.22 *For every poset $P \in \mathcal{O}_{I \cup J}$:*

(i) *$\otimes^G_{IJ}(P)$ is a poset with ground set X_P.*

(ii) *$\otimes^G_{IJ}(P) = \otimes_{IJ}(P)$, if moreover $P \in \mathcal{J}_{IJ}$.*

4.3.2 Generalised unary product extended to lpo's

We extend now $\otimes^G_{IJ}$ to a function $\bigotimes^G_{IJ}$ on lpo's. That extension generalises the operations $\bigotimes_{IJ}$ of [7]. The latter is defined from two other operations defined respectively on pairs of posets and pairs of lpo's, and said *binary products*. We introduce first these binary operations and then we define $\bigotimes_{IJ}$ and $\bigotimes^G_{IJ}$.

The binary products

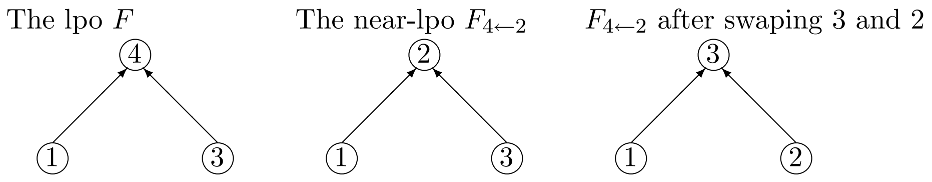
Notice that the presentation of the binary products is here slightly different that the one in [7]. In particular, the binary product $F \bigotimes F'$ of two disjoint lpo's $F = (P, \ell)$ and $F' = (P', \ell')$ is defined as a labeling of the binary product $P \otimes P'$ instead of the corresponding lpo's (with poset $P \otimes P'$ and the labeling defined here as $F \bigotimes F'$). This technical modification allows a simplification of the presentation. The binary product $\otimes$ on disjoint posets is given now. We remark that the fact that I, J are disjoint in the definition implies in particular that $\sqsubseteq \cup \sqsubseteq'$ is a p.o. on $I \cup J$. Hence $P \otimes P'$ is a poset for the same reasons that $\otimes_{IJ}(P)$ in Definition 4.20.

Definition 4.23 For all posets $P = (I, \sqsubseteq)$ and $P' = (J, \sqsubseteq')$ such that I, J are disjoint we define $P \otimes P'$ as the poset with ground set $I \cup J$ and p.o. $\sqsubseteq \cup \sqsubseteq' \cup I \times J$.

The extension $\bigotimes$ of $\otimes$ to pairs of disjoint lpo's in defined in [7] using a pseudo code involving two operations *PushUp* and *PushDown*. These operations generalise to *near-lpo's* the William's *PushDown* and *PushUp* which transform a nearheap into a heap (cf. [1,9]). We give first an intuitive description on the behavior of $PushDown$ (the $PushUp$ operation being simply its dual).

A *near-lpo* is a lpo where the biggest label has been replaced by a fresh label (a label not appearing in the lpo). The behavior of $PushDown$ is illustrated using Example 4.24 below. The first step is obtained by replacing the biggest label, here 4, by a fresh one, here 2. Then, $PushDown$ first checks if the replacing label is bigger than all labels below it. If it is true then $PushDown$ stops. If not, which is the case in our example, it swaps the replacing label with the biggest label below it, that is, in our example, it swaps 2 and 3. Then it iterates the same operations, continuing to push down the replacing label, until either the replacing label is bigger that all the labels below it or it is placed on a minimal element of the poset. In our example, after the first swap 2 has reached a minimal node and so $PushDown$ stops. It is straightforward to verify that, when $PushDown$ stops, the results of its swaps on the near-lpo is a lpo (in particular from the fact that the swaps always involve the biggest label below the label to be swapped).

Example 4.24 Application of the $PushDown$ operation on a near-lpo obtained from lpo F by replacing 4 by 2 :


The lpo F
The near-lpo $F_{4\leftarrow 2}$
$F_{4\leftarrow 2}$ after swaping 3 and 2
4
1
3
2
1
3
3
1
2

We describe the behavior of the $\bigotimes$ algorithm on two disjoint lpo's. Let $F = (P, \ell)$ and $F' = (P', \ell')$ be lpo's where P and P' have disjoint ground sets I and J, and where $\ell(I) = \mathcal{L}$ and $\ell'(J) = \mathcal{L}'$ are disjoint. Note that since ℓ is increasing the maximum label of $\mathcal{L}$ is necessary on a maximal node of I, otherwise it would have some node above it in $\sqsubseteq$ with a label bigger than a, which is impossible. Dual reasoning ensures that b, the minimum label of ℓ', is necessary on a minimal node of J. Then we have two cases: either a is below b or it is not. In the first case, all the labels of $\mathcal{L}$ are necessary below all the labels of $\mathcal{L}'$, and it is straightforward, by defining $F \bigotimes F' = \ell \cup \ell'$, to check that $F \bigotimes F'$ is a labeling on $P \otimes P'$. In the second case, a and b are swapped, and then a is pushed up in $(P', \ell'_{b\leftarrow a})$ and b is pushed down in $(P, \ell_{a\leftarrow b})$ using $PushDown$ and $PushUp$. As already noticed, the results of these two operations will be lpo's on respectively P and P'. Hence, we are back into the initial situation with labelings on P, P', and set of labels $\mathcal{L}'_{a\leftarrow b}$ and $\mathcal{L}_{b\leftarrow a}$ clearly disjoint. Then the algorithm iterates the same process, checks if the maximum label on P is below the minimum label on P', swaps these labels or not, and so on. Since the biggest label on P is always swapped with the smallest label on P', clearly, after some iterations of the process, the maximum label on P will be below the minimum one on P' (in the worst case when all the labels of $\mathcal{L}$ or $\mathcal{L}'$ will have been swapped) and the algorithm will stop. We then defined $F \bigotimes F'$ as the union of the two labelings on P and P' obtained from the iterations. We recall the following result from [7] and then define $\bigotimes_{IJ}$ and $\bigotimes^{G}_{IJ}$.

Lemma 4.25 *If $F = (P, \ell)$ and $F' = (P', \ell')$ are disjoint lpo's then $F \bigotimes F'$ is a labeling on $P \otimes P'$.*

Definition 4.26 Let $I, J \subseteq_f O$ be disjoint sets, and $F = (P, \ell) \in \mathcal{F}\mathcal{J}_{IJ}$:

$\bigotimes_{IJ}(P, \ell) = (\otimes_{IJ}(P),\ \otimes_{IJ}(\ell))$

where $\otimes_{IJ}(\ell) = \ell \upharpoonright (X_P \setminus I \cup J)\ \cup\ F \upharpoonright I \bigotimes F \upharpoonright J.$

It is easy to check that $\bigotimes_{IJ}$ is local lpo transformation on $\mathcal{F}\mathcal{J}_{IJ}$, and so by Lemmas 3.9 and 4.25 we get immediatly.

Lemma 4.27 *For all $I, J \subseteq_f O$ disjoint: $\bigotimes_{IJ} : \mathcal{F}\mathcal{J}_{IJ} \mapsto \mathcal{F}$.*

Definition 4.28 Let $I, J \subseteq_f O$ disjoint and $(P, \ell) \in \mathcal{F}$:

$$\bigotimes^G_{IJ}(P, \ell) = \begin{cases} \bigotimes_{IJ} \circ \jmath_{IJ}(P, \ell) \text{ if } & I \cup J \subseteq X_P \\ \text{“}Error\text{''} & \text{else} \end{cases}$$

As a direct application of point i of Fact 4.16 and Lemma 4.27, we get.

Proposition 4.29 *For all $I, J \subseteq_f O$ disjoint, $\bigotimes^G_{IJ} : \mathcal{F}_{I \cup J} \hookrightarrow \mathcal{F}$. If moreover $P \in \mathcal{J}_{IJ}$ then $\bigotimes_{IJ}(P, \ell) = \bigotimes^G_{IJ}(P, \ell)$.*

4.4 The generalised split

The language MOQA has an operator *Split* which takes two arguments: a set Y and a lpo. Its semantics are summarized here and then we introduce our generalisation in two steps.

4.4.1 The generalised split on posets

We present the semantics of the *Split* operator in a slightly different way than it was done in [7]. In particular, we suppose from now on that O is totally ordered.

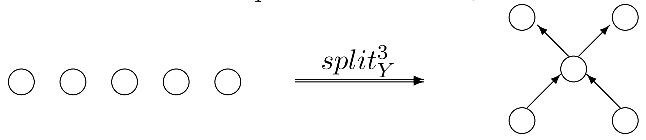
Definition 4.30 For all $Y \subseteq_f O$ and positive integer $m \in \{1, \ldots, Card(Y)\}$, define $Y^{<m^{th}}$ (resp. $Y^{\leq m^{th}}$) as the subset containing the $m - 1$ (resp. the m) smallest elements of Y (relatively to the order on O). The subset $Y^{>m^{th}}$ (resp. $Y^{\geq m^{th}}$) contains the elements of Y with ranks strictly above (resp. above or equal) m.

Definition 4.31 Let $P = (X, \sqsubseteq) \in \mathcal{A}\mathcal{I}_Y$, $m \in \{1, \ldots, Card(Y)\}$ and y be the m^{th} smallest element of Y. We define the poset $split^m_Y(P) = (X, \sqsubseteq_m)$ by:

$\sqsubseteq_m = \sqsubseteq\ \cup\ (Y^{<m^{th}} \times Y^{\geq m^{th}})\ \cup\ (\{y\} \times Y^{>m^{th}})$

Clearly $split^m_Y$ is a local transformation on $\mathcal{A}\mathcal{I}_Y$, and so $split^m_Y$ is a poset if its restriction to Y is (Lemma 3.7). Note that the reflexivity of $\sqsubseteq_m$ is immediate by reflexivity of $\sqsubseteq$, while the antisymmetry and the transitivity on Y are straightforward using in particular the fact that, for every distinct $x, x' \in Y$, $x \sqsubseteq_m x' \Rightarrow x' \in Y^{\geq m^{th}}$. We give now an example.

Example 4.32 $split^3_Y$ applied to A_Y (cf. Definition 2.5), Y contains five nodes:



Definition 4.33 Let $Y \subseteq_f O$, $P \in \mathcal{O}$ and m a positive integer:

$$split^G_Y(m,P) = \begin{cases} split^m_Y \circ @\imath_Y(P) \text{ if } & Y \subseteq X_P \wedge m \in \{1, \dots Card(Y)\} \\ \text{“}Error\text{''} & \text{else} \end{cases}$$

A direct application of the point ii of Lemma 4.13, we get the following.

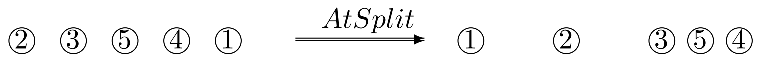
Lemma 4.34 *For all $P \in \mathcal{O}_Y$ and $m \in \{1, \dots, Card(Y)\}$, $split^G_Y(m,P)$ is a poset. If moreover $P \in \mathcal{AI}_Y$, then $split^G_Y(m,P) = split^m_Y P$.*

4.4.2 The generalised split extended to lpo's

We define the extension $Split^G_Y$ of $split^G_Y$ to lpo's and the corresponding operation $Split_Y$. The latter operation is defined first on lpo's with atomic posets (cf. Definition 2.5) using an algorithm written in pseudo-code. It is then generalised to every lpo where Y is atomic isolated. We present first that algorithm. For that, we introduce here a special notation *AtSplit* for the operation implemented by the algorithm. Moreover, as for $\bigotimes$, we prefer to define $AtSplit(A_Y, \ell)$ as a labeling instead of a lpo as it is done in [7].

Let $Y \subseteq_f O$ and $F = (A_Y, \ell)$ be an atomic lpo, and let $\mathcal{L} = \ell(Y)$. The algorithm processes as follow. First, it picks up the label a on the first element of Y and determines its rank m in $\mathcal{L}$ by comparing that label to other labels. Now, the algorithm defines a bijection $\ell' : Y \mapsto \mathcal{L}$ as follow, where y is the m^{th} smallest element of Y: it puts the $m-1$ smallest labels on the elements of $Y^{<m^{th}}$, the pivot label a on y and finally the rest of the labels on $Y^{>m^{th}}$. The example below gives a simple application of $Split_Y$. Note that Fact 4.36 is quite obvious.

Example 4.35 *AtSplit* applied to (A_Y, ℓ), where Y contains five points written from left to right according to their ranks in O:



Fact 4.36 *For every atomic lpo (A_Y, ℓ), $AtSplit(A_Y, \ell)$ is a labeling on $split^m_Y(P)$, where m is the rank of $\ell(y)$ in $\ell(Y)$ and y is the smallest element of Y.*

We can now introduce $Split_Y$ and $Split^G_Y$.

Definition 4.37 For every $Y \subseteq_f O$ and $(P, \ell) \in \mathcal{FAI}_Y$, define:
$Split_Y(P, \ell) = (split_Y(P), Split_Y(\ell))$

where $Split_Y(\ell) = \ \ell \upharpoonright (X \setminus Y) \ \cup \ AtSplit((P, \ell) \upharpoonright Y)$

It is easy to check that $Split_Y$ is local lpo transformation on $\mathcal{FAI}_Y$, and so by Lemmas 3.9 and Fact 4.36, we get immediatly.

Lemma 4.38 *For every $Y \subseteq_f O$: $Split_Y : \mathcal{FAI}_Y \mapsto \mathcal{F}$.*

Definition 4.39 For every $Y \subseteq_f O$ we define the operation $Split^G_Y$ on $\mathcal{F}$ by:

$$Split^G_Y(P, \ell) = \begin{cases} Split_Y \circ @\imath_Y(P, \ell) & \text{if} \quad Y \subseteq X_P \\ \text{“}Error\text{''} & \text{else} \end{cases}$$

A direct application of point ii of Fact 4.16 and Lemma 4.38 gives.

Proposition 4.40 *For every $Y \subseteq_f O$, we have $Split^G_Y : \mathcal{F}_Y \hookrightarrow \mathcal{F}$. If moreover $P \in \mathcal{AI}_Y$ then $Split^G_Y(P, \ell) = Split_Y(P, \ell)$.*

5 RP-Domains of the Generalised Operations

We first introduce formally the notions of *Domain of Definition (Def-Domain)* and *Domain of Random Preservation (RP-Domain)* of a function acting on lpo's. We recall that the definition of Random Preservation is given in Section 2.4.

Definition 5.1 For an operation Op with domain a subset of $\mathcal{F}$, we call respectively Domain of Definition and RP-Domain the following sets:

(i) $Dom_{def}(Op) = \{F \in Dom(Op) : Op(F) \in \mathcal{F}\}$

(ii) $Dom_{rp}(Op) = \cup\{\mathcal{R} : \mathcal{R}$ is a Random Structure and Op is RP on $\mathcal{R}\}$

As already mentioned, RP-domains and Def-domains of MOQA operations are identical. It is not the case for the generalised ones and it would be easy to exhibit random structures where they are defined but not RP. Nevertheless, Propositions 4.19, 4.29 and 4.40 express in particular that the generalised operations, when restricted to the domains of the corresponding MOQA operations, coincide with these operations. That fact implies in particular that the RP-domains of generalised operations contain the RP-domains of the corresponding MOQA operations. In fact we have a stronger result.

Theorem 5.2 *For all $I, J, Y \subseteq_f O$ such that I, J disjoint:*

(i) $Dom_{rp}(Split^G_Y) \supset Dom_{rp}(Split_Y)$

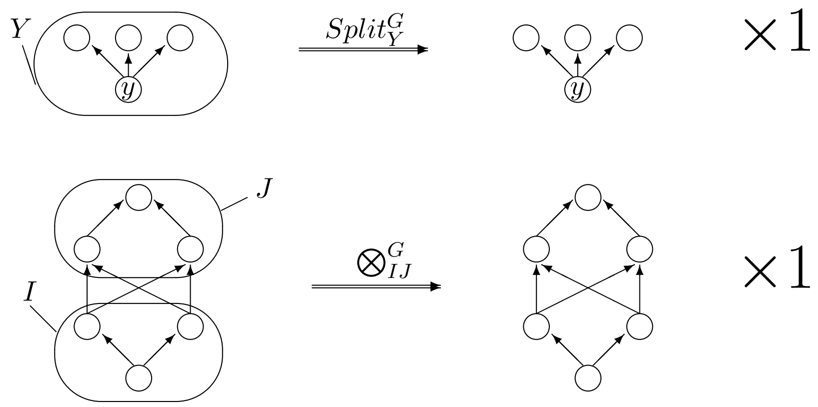
(ii) $Dom_{rp}(\bigotimes^G_{IJ}) \supset Dom_{rp}(\bigotimes_{IJ})$

(iii) $Dom_{rp}(Proj^G_Y) \supset Dom_{rp}(Proj_Y)$

We prove now Theorem 5.2 by giving examples where the MOQA operators are not defined (and so a fortiori not RP) while the generalised operations are both defined and RP. Notes that random structures are represented by their underlying posets; the choice of the set of labels $\mathcal{L}$ being not relevant here. We write $\times K$ to express that the output multiset $O_{Op}(\mathcal{R})$ of the operation Op applied to $\mathcal{R}$ is equal to $Op(\mathcal{R}) \times K$.

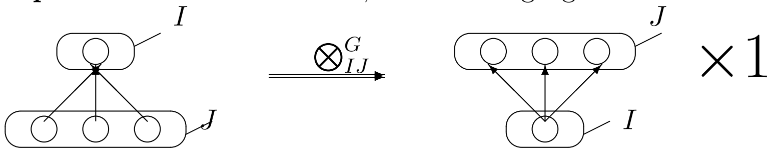
In the first example below, the shape of the underlying poset is not changed by the operation. Indeed, one can verifies that all the links between nodes of the underlying posets which had been deleted by application of the operations $@\imath_Y$ and $\jmath_{IJ}$ are finally put back in place by application of operations $split_Y$ and $\otimes_{IJ}$. Then the operations $Split_Y^G$ and $\bigotimes_Y^G$ work as the identity on lpos of the RS of the example. Note finally that the result of random preservation expressed by that example can be generalised to random structures where input and output posets are isomorphic.

Example 5.3 Let y be the smallest element of Y w.r.t the order on O. We remark that Y is not atomic and I, J are not zigzag closed:



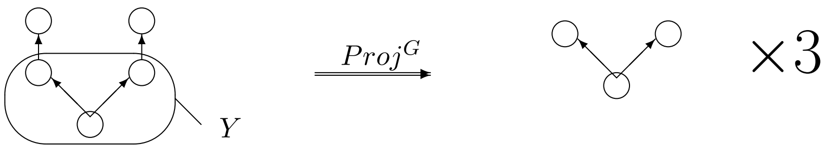
In the next example the operation $\otimes_{IJ}^G$ reverse a poset denoted by $K_{3,1}$ to give a poset $K_{1,3}$. The application of $\bigotimes_{IJ}^G$ to the RS on $K_{3,1}$ give then one copy of the RS on $K_{1,3}$. Remark, that the RP result expressed by the example can be generalised to every integers n, m.

Example 5.4 We remark that I, J are not zigzag free in $I \cup J$.



We recall that $Proj_Y$ is strongly random preserving (cf. Section 2.4). The operation $Proj_Y^G$ has the same property on the random structure in the example below, where 3 isomorphic copies of the output set are generated.

Example 5.5 We remark that Y is not isolated.



6 Conclusion and Future Work

In this article, we introduced three generalised MOQA operations. These operations have several advantages compared to their correspondent MOQA operations. First, they extend the domains of definition of the latter in a non trivial and still meaningful way, using more natural restrictions. Secondly, as shown in the previous section they extend also the domains of random preservation of the corresponding MOQA operations. The fact that the examples given in Section 5 can be generalised indicates that this extension is non-trivial and quite rich.

Nevertheless, and in contrast to MOQA operations, there is no known characterization of the RP-domains of generalised operations. That implies in particular that there is no known computable function able to calculate in full generality the output multiplicities as it can be done in MOQA (cf. Section 2.4). The semi-automatisation of average time analysis deriving from the existence of that function is then not possible either.

Thus, the next step toward a full generalisation of MOQA operations including the semi-automation of average time analysis would be a tractable characterization of RP-domains of the functionally-generalised operations, or at least non trivial subsets of these domains. Note also the generalisations we gave here may be not the only possible ones.

References

[1] Aho A., J. Hopcroft and J. Ullma: “Data structures and algorihms”, Addison-Wesley Series in Computer Science and Information Processing, Addison-Wesley (1987).

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

[3] Knuth D.: “The art of computer programming vol.3”, Addison-Wesley (1973).

[4] Manning J.: *Reconstruction of Partial Orders*, MFCSIT’06 Proceedings, Cork, Ireland (Aug 2006), 296-299.

[5] Schellekens M.: *Compositional Average-Case Analysis*, Preprint, submitted to the Journal of ACM (2006), 78p.

[6] Schellekens M.: Modular Timing, An overview of CEOL research. MFCSIT’06 Proceedings, Cork, Ireland (Aug 2006), 300-303.

[7] M. P. Schellekens, A Modular Calculus for the Average Cost of Data Structuring, Springer book to appear (2008), 250 pp.

[8] Schellekens M., Hickey D., Bollella G.: ACETT, a Linearly-Compositional Programming Language for (semi-)automated Average-Case analysis. IEEE Real-Time Systems Symposium - WIP session Proceedings (2004).

[9] Williams J.W.J: Algorithm 232, Communication of ACM **7**(6), 347-348, 1964.