More-than-coherent logic for operations on images

Paolo Bottoni[1] Anna Labella[1] Stefano Kasangian[2]
[1] Dip. di Informatica, Università di Roma "La Sapienza"
[2]Dip. di Matematica, Università di Milano

Abstract

A model of computation on multi-dimensional words, based on the overlapping operation, induces a categorical structure, to which a new type of logic corresponds, whose formulae express properties of computations in a language containing all first order formulae. While the resulting deductive system is strictly less powerful than (intuitionistic) first order logic, it is more powerful than coherent logic. The approach is illustrated through an example of an online game of map-colouring.

Keywords Categorical logics, operations on images, overlapping, online coloring.

1 Introduction

We investigate the properties of computations with images via the overlapping operation [1], as a particular case of interactive, non-deterministic, computations. To this end, we exploit a categorical structure, called $SymcatB$, which was studied in [11] to provide a formal setting for the study of concurrent processes [16] and bisimulations between them. In particular, $SymcatB$ is the category of symmetric $\mathcal{B}$-categories and $\mathcal{B}$-functors, where $\mathcal{B}$ is a suitable 2-category (see [20]).

In this paper, we observe that a new type of logic can be associated with $SymcatB$, in analogy with the association of (intuitionistic) first order logic with Heyting categories. Indeed, $SymcatB$ is a coherent category [10], a subcategory of which is a Heyting category. We introduce the notion of "more-than-coherent logic" to describe the logic associated with $SymcatB$. This exploits a full first order language, for which a logic weaker than intuitionistic logic, but stronger than a coherent logic [10], is defined. In particular, negative formulae behave as the intuitionistic ones.

By associating this logic with the categorical structure, we describe and reason on computations with images, in which agents make moves by placing copies of private images, interacting to construct a concurrent state, again defined by an image. Agents can work concurrently on different parts of the (state) image, without having to act sequentially on contiguous zones, but taking turns according to paths on a tree labeled with the names of the executed actions. Overlapping contributions can be managed exploiting partial orderings imposed on the alphabets from which the pictures are formed.

We show how forms of interactive computing can be modelled in this framework (by generalizing the original algebraic structure in [11], where $\mathcal{B}$ was obtained from a meet-semilattice monoid), as contributions can overlap in any order on any area of an image, provided that some conditions are met. This kind of computation is intrinsically non deterministic, as the same apparent result can be obtained with different actions from the involved agents. Hence, future behaviors can be influenced in different ways.

After mentioning related work in Section 2, in Section 3 we introduce the main definitions of pointed image and overlapping operation and propose the main example of collaborative game and a tentative language to speak about it. Section 4 illustrates the monoidal category of trees modeling concurrent computations, in particular computations with images. Section 5 introduces the category of symmetric $\mathcal{B}$-categories in the sense of [20]. Its coherent structure is studied and the existence of a subcategory which is a Heyting category is shown. The resulting logical system is presented in Section 6, and its instantiation to the current problem is proposed in Section 7. Section 8 draws conclusions and points to future work.

2 Related work

We consider here two research traditions, related to the study of algebraic properties of languages, notably two-dimensional ones, and to the categorical treatment of process algebras to describe the behaviour of concurrent agents, respectively.

The algebraic characterization of 2-D languages started with the seminal works of Kirsch [12] and Dacey [4], aimed at imposing some form of juxtaposition (reduced to common concatenation in the 1D case) on 2D images. Horizontal and vertical versions of concatenation were proposed, giving rise to generative models combining horizontal and vertical rewriting [19]. However, these versions of concatenation are only partial functions, being applicable only to pictures of compatible sizes. An algebraic characterization has been given as doubly ranked monoids constructed from alphabets of bidimensional elements, in which each individual operation is associative and compatible with each other [7]. Survey of these topics, with particular emphasis on the notion of recognisability of picture languages, can be found in [6].

Attachment positions for contour curves were introduced by Shaw to allow composition of curves so that the head of an element be attached to the tail of another [18]. Pointed drawings are described in [13] by a string of directions and a pair of *departure* and *arrival* positions. Connected pointed figures equipped with concatenation give rise to a finitely generated inverse monoid, the set of generators being constituted of coloured pixels with all possible positions for the arrival and departure points. While this result concerns the descriptions of figures (shapes in black-and-white images), we are interested in images, i.e. finite bidimensional structures.

The (im)possibility of recovering concatenation of images in the 2D world by simply enriching them with attachment points, or by allowing placement only over well-behaved paths, is studied in [3]. The proposal of pointed pictures [1] avoids such limitations, as they include information on attachment points, and allows translations and rotations of the original content of the pictures and of the associated information.

Modeling the behavior of concurrent agents by trees labeled on a monoid of el-

ementary moves is standard practice in concurrent computing [16] and has been interpreted in an enriched categorical context in [11]. There, several cases of the base monoidal structure are considered (free monoid, trace monoid, etc.). In this paper we consider for the first time a monoid which does not satisfy the left cancelation property.

Lawvere's *algebraic theories* [14] are the first categorical description of the models of a theory, while Lawvere and Tierney's elementary topos extends Grothendieck's (geometric) theory of toposes to the realm of logic, relating them to intuitionistic logic. Since then, many categorical structures have been investigated from a logical viewpoint, by associating with them a language and a deductive system. In particular, coherent categories are related to positive first order logic (coherent logic) [10]. Starting from a coherent category containing a Heyting subcategory, we consider a language more expressive and a deductive system stronger than the coherent ones.

The application of the language of more-than-coherent logic to interactive computations, seen as games, can be related to the computability logic proposed by Japaridze [9], who exploits a wider language, called universal language, including different types of existential and universal quantifiers. The universal language can indeed refer to individual moves, whereas we refer to whole computations. However, we can introduce a notion of factorization, to consider computations made of single moves.

3 Interactive computations on images

We consider here the problem of describing properties of interactive computations with images, based on a simplified version of the positional overlapping operation introduced in [1]. In this Section, as an instance of such a computation, we consider an interactive game based on the 4-colour problem, i.e. the problem of finding a colouring of a map using up to four colours, so that no two adjacent regions are coloured in the same way. We first introduce definitions and properties for the *overlapping operation*. In particular, overlapping is defined for (n-dimensional) images as an abstraction from many significant pixel operators as well as visual interaction phenomena, in particular in interactive construction of diagrams.

3.1 The overlapping operation

Definition 1 [Meet-semilattice]
A *meet-semilattice* $\mathbf{L} = (L, \leq, \wedge, \bot)$ is a partial order w.r.t. $\leq$, such that $\wedge$ is the greatest lower bound function and $\bot$ denotes the bottom element.

Definition 2 [Complete meet-semilattice]
A *complete meet-semilattice* $\mathbf{L} = (L, \leq, \wedge, \bot)$ is a meet-semilattice, such that for any non-empty family of elements its greatest lower bound exists.

Remark 1 A complete meet-semilattice has a join for every family that has an upper bound, given by the meet of all upper bounds.

In the rest of the paper, we consider partially ordered finite alphabets, named V, which are complete meet-semilattices with bottom element denoted by τ. We can

interpret the order relation on V as an information about transparency of a cell. τ denotes absolute transparency as well as undefinedness of color assignment to a cell.

Definition 3 [Image and pointed images]

1. An *image* ι on an alphabet V is a function $\iota : Z^n \rightarrow V$, where Z is the set of integer numbers and ι is almost everywhere equal to τ.
2. A *pointed image* $(\iota, \overrightarrow{x_e}, \overrightarrow{x_t})$ is an image with two designated *entry* and *exit* positions.
3. PI_V is the set of pointed images on V where all the images constantly equal to τ are identified into ι_0, forgetting about their entry and exit positions. We use PI when V can be left understood.
4. A *translation* of a pointed image $(\iota, \overrightarrow{x_e}, \overrightarrow{x_t})$ by $\overrightarrow{k}$, also noted $t_{\overrightarrow{k}}$, is a pointed image $(\iota', \overrightarrow{x'_e}, \overrightarrow{x'_t})$, where $\iota'(\overrightarrow{x}) = \iota(\overrightarrow{x} + \overrightarrow{k})$, $\overrightarrow{x'_e} = \overrightarrow{x_e} + \overrightarrow{k}$, $\overrightarrow{x'_t} = \overrightarrow{x_t} + \overrightarrow{k}$.

The set of pointed images PI is partially ordered: $(\iota, \overrightarrow{x_e}, \overrightarrow{x_t}) \leq (\iota', \overrightarrow{x'_e}, \overrightarrow{x'_t})$ *iff* there exists a translation $(\iota'', \overrightarrow{x''_e}, \overrightarrow{x''_t})$ of $(\iota', \overrightarrow{x'_e}, \overrightarrow{x'_t})$ such that $\iota \leq \iota''$ as functions and $\overrightarrow{x}_e = \overrightarrow{x''_e}, \overrightarrow{x_t} \leq \overrightarrow{x''_t}$, i.e. $(\iota'', \overrightarrow{x''_e}, \overrightarrow{x''_t}) = t_{\overrightarrow{(x'_e - x_e)}}((\iota', \overrightarrow{x'_e}, \overrightarrow{x'_t}))$.

For the definitions above, each image induces a semilattice with the completely transparent image at the bottom and the original image at the top.

Proposition 1 Let $\iota_{\mathbf{0}} = (\iota_0, \vec{0}, \vec{0})$, where ι_0 is the image with all pixels transparent. Then, $(PI, \leq, \iota_{\mathbf{0}})$ is a meet-semilattice.

The family of overlapping operations $\bullet_{op}$ on PI, is parametric w.r.t. a binary associative operation $op : V \times V \rightarrow V$: $(\iota, \overrightarrow{x_e}, \overrightarrow{x_t})) \bullet_{op} (\iota', \overrightarrow{x'_e}, \overrightarrow{x'_t})) = (\iota'', \overrightarrow{x_e}, \overrightarrow{x''_t}))$ with $(\iota'', \overrightarrow{x_e}, \overrightarrow{x''_t}) = (\iota, \overrightarrow{x_e}, \overrightarrow{x_t}) \bullet_{op} (\iota''', \overrightarrow{x'''_e}, \overrightarrow{x'''_t})$, where $(\iota''', \overrightarrow{x'''_e}, \overrightarrow{x'''_t}) = t_{\overrightarrow{(x_t - x'_e)}}((\iota', \overrightarrow{x'_e}, \overrightarrow{x'_t})))$.

Figure 1 illustrates three instantiations of the operation for two pointed images, where e and t indicate the entry and exit position, respectively, and 0 the origin of the coordinates. The operation $\&_1$ preserves the value of the first argument, $\&_2$ the value of the second, and $\&_3$ combines the two values where they are both defined, and preserves the value of the defined image otherwise. Notice that in all three cases, the set of positions of the resulting image is the same, as well as the origin and the entry and exit positions.

3.2 A 4-colour based game

Let a collection of maps $\mathcal{K}$ be given to two agents, called the *Drawer* and the *Painter*. The Drawer starts by selecting a subset of regions from one map and challenging the Painter to colour them according to the 4-colour constraint, using overlapping: once a region has been painted, the colour of no cell in the region can be changed. After the Painter has done colouring, the Drawer selects a new set of regions and the game progresses. The Drawer wins if it can propose a region that the Painter is not able to

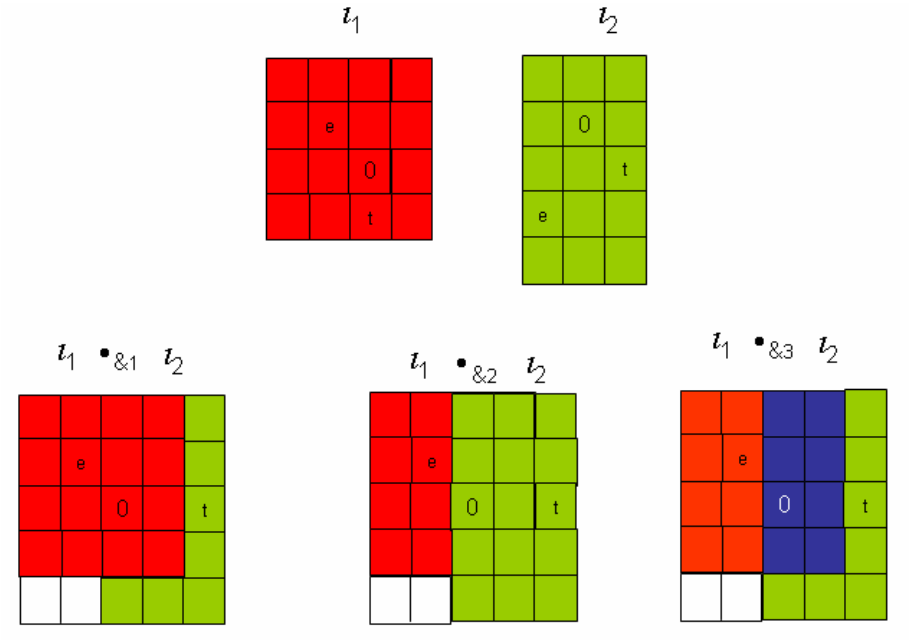

Figure 1: Instantiations of overlapping between pointed images.

colour, while the Painter wins if it achieves a complete colouring of a map. Drawn regions are assigned the colour \$, indicating that the region needs to receive its final coloring. We define the alphabets $\mathcal{C}_1 = \{a, b, c, d\}$, $\mathcal{C}_2 = \mathcal{C}_1 \cup \{\$\}$, $\mathcal{C} = \mathcal{C}_2 \cup \{\tau\}$, and the partial order on $\mathcal{C}$ induced by $< = \{(\tau, x) \mid x \in \mathcal{C}_2\} \cup \{(\$, x) \mid x \in \mathcal{C}_1\}$. All the regions are defined as pointed images with entry and exit position in $(1, 1)$, and with all symbols transparent except for those in one of the regions of one map. We denote the set of available regions in a map k as $\mathcal{R}_k = \{1, \ldots, n_k\}$. With $\mathcal{R}$, we indicate the disjoint union of all the regions in $\mathcal{K}$.

Figure 2 shows two maps, each made of six regions, and two colourings satisfying the 4-colour condition. The three central regions are defined by the same subimages in the two maps, but receive different colours due to the 4-colour condition.

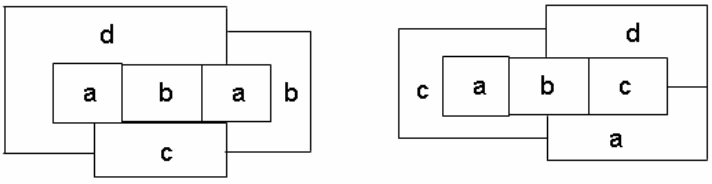

Figure 2: Two maps coloured in the correct way.

We model the colouring of a region R from a map K with a colour C as a move

$(K, R, C) \in \mathcal{K} \times \mathcal{R}_K \times \mathcal{C}$. We call $\mathcal{M}_{[\mathcal{K},\mathcal{R},\mathcal{C}]}$ the resulting alphabet of moves. Where no ambiguity arises, we omit the subscript $\mathcal{K}, \mathcal{R}, \mathcal{C}$ and only refer to $\mathcal{M}$. We partition $\mathcal{M}$ into $\mathcal{M}^d = \mathcal{M}_{\mathcal{K},\mathcal{R},\$}$ and $\mathcal{M}^p = \mathcal{M}_{\mathcal{K},\mathcal{R},\mathcal{C}_1}$, to indicate that the Drawer can only use moves with the \$ colour and the Painter only "real" colours. Moreover, a Painter can perform a move $(k, r, c) \in \mathcal{M}^p$ for some k, r, and c only if a move $(k', r, \$) \in \mathcal{M}^d$ has been previously performed by the Drawer, where either $k = k'$ or $r \in R_k \cap R_{k'}$. Note that, even if regions are taken from different maps, they are bound to agree on the already performed moves.

A colouring of a map with n regions is a process with moves of the form (y, R, x), and the resulting computation is a word on $\mathcal{M}^*_{[\mathcal{K},\mathcal{R},\mathcal{C}]}$. Actually, the languages of interest here are languages on the *trace monoid* $\mathcal{M}(E) = \mathcal{M}^*_{[\mathcal{K},\mathcal{R},\mathcal{C}]}/\equiv_E$ [15], where the *dependency relation* E is induced by the complement of an independency relation I such that $\forall k \in \mathcal{K}\ \forall r_i, r_j \in \mathcal{R}\ \forall c_h, c_l \in \mathcal{C}_1\ ((k, r_i, c_h), (k, r_j, c_l)) \in I$ and $((k, r_i, \$), (k, r_j, \$)) \in I$.

For a word $\omega \in \mathcal{M}(E)$, we call $\alpha(\omega)$ the set of moves used in ω. In general, one distinguishes Drawer and Painter computations, but we omit the distinction when no ambiguity arises. For a given map $k \in \mathcal{K}$, all computations ω satisfy $noReplication(\omega) \equiv \forall r \in \mathcal{R}_k, c_h, c_l \in \mathcal{C}[\neg\exists\omega_1, \omega_2(\omega = \omega_1 \bullet \omega_2)[((k, r, c_h) \in \alpha(\omega_1)) \wedge ((k, r, c_l) \in \alpha(\omega_2))]]$, where $\bullet$ denotes concatenation of moves.

Moreover, *admissible* computations for the Painter satisfy predicate $adm_p(\omega) \equiv \forall r, r_i, r_j \in \mathcal{R}_k\ [adj_k(r_i, r_j) \implies ((k, r_i, c) \in \alpha(\omega) \implies \neg(k, r_j, c) \in \alpha(\omega))]$, where $adj_k(r_i, r_j)$ indicates that the two regions r_i, r_j are adjacent in map k.

Each agent can perform any legal computation on a map. For the Painter, this corresponds to sequences producing 4-coloured maps, while for the Drawer these are all possible orders of presentations of regions from a map. After a Drawer's move $(k, r, \$)$, the Painter can observe only its projection $(r, \$) \in \mathcal{R} \times \mathcal{C}$ and must replicate with a move (k', r, c) from a computation on a map k' containing r. Analogously, the Drawer will observe only the projection (r, c) of such a move. Once the Drawer has selected a map, it has to go on playing with its original choice. In general, $sameMap(\omega) \equiv \forall(k, r, c), (k', r', c') \in \alpha(\omega)[k = k']]$ indicates that a player always selects moves from the same map, while $play_k$ is satisfied if a player's computation satisfies $sameMap$, selecting moves from map k.

Of interest here are alternating games, in which the Painter has to colour regions exactly in the order in which they are proposed and progress is made only if the Painter has coloured all of them. We adopt the point of view of an external observer, which regards the game as a single computation ω^o where moves are successions of projections on $\mathcal{R} \times \mathcal{C}$ of the moves played by the Drawer and the Painter. We introduce a predicate $alt(\omega^o)$ which is satisfied if ω^o is admissible and, for any prefix ω_1 of ω^o, $\mid \omega_1 \mid_{\{\$\}} \geq \mid \omega_1 \mid_{\mathcal{C}_1}$. Moreover, for a word ω^o satisfying $alt(\omega^o)$, we introduce the projections pr^d and pr^p providing ω^d and ω^p, respectively, up to the choice of the map. If the game is played with a fixed number of drawn regions at each turn, we call *alternation step* x this number. Alternating games give rise to words of the form $\omega^o = a^d_{1,1} \dots a^d_{1,x} a^p_{1,1} \dots a^p_{1,x} \dots a^d_{r,1} \dots a^d_{r,x} a^p_{r,1} \dots a^p_{r,x}$. The predicate $alt_x(\omega^o)$ is satisfied if $alt(\omega^o)$ and ω^o is the unique word describing the alternating game of step x for which $\omega^d = pr^d(\omega^o)$ and $\omega^p = pr^p(\omega^o)$ are the Drawer's and Painter's sequence of

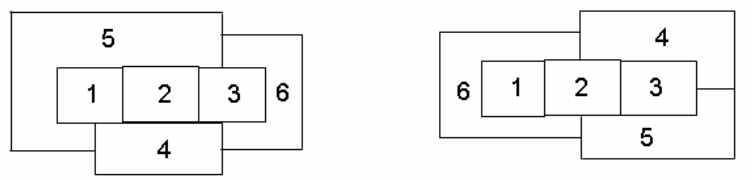

Figure 3: The Drawer's sequences for a game with two maps.

moves, respectively. This corresponds to the case of *online colouring* [8].

Two predicates can be defined on observer computations, denoting success for the Drawer or the Painter, respectively, as follows:

$succ^d(\omega^o) \equiv \exists(r,\$) \in \alpha(pr^d(\omega^o))[\neg\exists c \in \mathcal{C}_1[(r,c) \in \alpha(pr^p(\omega^o))]]$
$succ^p(\omega^o) \equiv \forall(r,\$) \in \alpha(pr^d(\omega^o))[\exists c \in \mathcal{C}_1[(r,c) \in \alpha(pr^p(\omega^o))]]$
Considering the set of maps $\mathcal{K} = \{k_1, k_2\}$ of Figure 3, one verifies Fact 1.

Fact 1 *The following hold:*

1. $\forall\omega^d \in \mathcal{M}^d(E)\forall\omega^p \in \mathcal{M}^p(E)\forall k \in \mathcal{K}[(play_k(\omega^d) \wedge play_k(\omega^p)) \implies (\exists\omega^o[\omega^d = pr^d(\omega^o) \wedge \omega^p = pr^p(\omega^o) \wedge succ^p(\omega^o)])]$

2. $\forall h,k \in \mathcal{K}[(\neg h = k) \implies (\exists\omega^d \in \mathcal{M}^d(E), \omega^p \in \mathcal{M}^p(E)[(play_h(\omega^d) \wedge play_k(\omega^p)) \implies (\exists\omega^o[\omega^d = pr^d(\omega^o) \wedge \omega^p = pr^p(\omega^o) \wedge succ^d(\omega^o)])])]$

The Drawer's strategy is to select one map in Figure 3 and present the common regions indicated with 1, 2, 3 , in this order. After the Painter has selected the third colour, the Drawer proposes the region identified with 4 in its map. The computation will result in success for the Painter if the two were moving with respect to the same map. Otherwise, the Painter will be forced to choose a colour for which no computation can be successful. From the observer's point of view, iterations of the game in which the first three moves of each player are always the same can result into different games. Such a situation occurs for alternating games of step 1 and 3, but not for any other step.

For any game, the property $succ^p(\omega_1) \wedge perm(\omega_1,\omega_2) \implies succ^p(\omega_2)$ holds, where predicate $perm(\omega_1,\omega_2)$ is satisfied if one word can be obtained from the other by permutation of the positions of the moves.

4 Computing with overlapping

We introduce here the categorical structure needed to describe computations occurring on images through the use of the overlapping operation. This structure will result into an instance of a $SymcatB$-category formally defined in Section 5. The language and the logical structure associated with $SymcatB$ in Section 6 will provide the proper solution to the problem of finding a language to speak about collaborative computing with images and characterise its logics.

4.1 The monoidal category of trees

In order to describe the possible evolution of a concurrent process from one state to another, one exploits a set of computations labeled with elements from a complete meet-semilattice **L**, representing possible observations of the behavior of an agent (in our leading example the overlapping of visible moves performed by the two players). The elements of **L** give the *extent* of the computation. Since for the observer the process is a non-deterministic one, a further piece of information is needed to identify the degree to which two given computations are indistinguishable to observation; in general, this degree, called *agreement*, will not be maximal. Such a structure gives rise to a *generalized tree* whose paths are computations, glued together via agreement. This kind of definition can produce also pathological trees as, e.g. the empty one or a tree where two paths are completely glued together. Actually, this construction can be carried for any meet-semilattice **L** [11].

Definition 4 [Trees]

1. An L-tree (or tree) is a triple (X, e_X, a_X) where X is the set of *paths*, the *extent map* $e_X : X \to \mathbf{L}$ is the labeling of the paths and the *agreement* $a_X : X \times X \to \mathbf{L}$ is the gluing between paths, such that, $\forall x, y, z \in X$, it holds that:
   (a) $a_X(x, x) = e_X(x)$
   (b) $a_X(x, y) \leq e_X(x) \wedge e_X(y)$
   (c) $a_X(x, y) \wedge a_X(y, z) \leq a_X(x, z)$
   (d) $a_X(x, y) = a_X(y, x)$
2. An L-tree-morphism, or simulation, $f : \mathcal{X} \to \mathcal{Y}$ is a function mapping paths into paths, strictly preserving labeling and non decreasing gluing between them:
   (a) $e_X(x) = e_Y(f(x))$
   (b) $a_X(x, y) \leq a_Y(f(x), f(y))$
3. L-trees with their morphisms form the category $Tree_L$, or $Tree$ for short.

Figure 4 shows an example of trees, each with two paths, with equal extent, i.e. the set of words $\{ac, ab\}$, but different agreement, as the agreement between the paths in the left tree is the empty word and that for the right tree is the word labeled a.

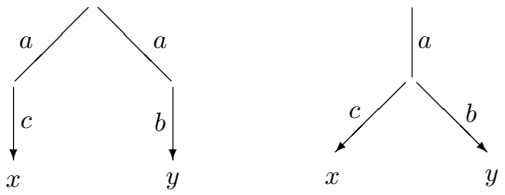

Figure 4: Two Trees.

Example 1 Let A^* be the free monoid generated by an alphabet A. An A^*-category $\mathcal{X}$ results in an A-labeled tree, with the set of labels ordered according to the prefix relation.

From a monoid on which the prefix relation induces a complete meet-semilattice structure (as for a free monoid), one obtains an instance of the theory developed so far. The same construction can be performed for a trace monoid.

Due to their relevance in our context, we now introduce some further operations on $Tree$, in particular cases. We now consider a meet-semilattice which has also a monoidal structure, according to Definition 5.

Definition 5 [Meet-semilattice with monoidal structure]
A meet-semilattice $\mathbf{L} = (L, \leq, \wedge, \bot)$ has a monoidal structure *iff* it is a monoid $(L, \bullet, 1)$ such that the following hold:

- $h \leq h'$ implies $k \bullet h \leq k \bullet h'$ *(right monotonicity)*
- $k \bullet (h \wedge h') = (k \bullet h) \wedge (k \bullet h')$ *(right semidistributivity)*
- $k \leq k \bullet h$ *(non decreasing property)*

Proposition 2 If $\mathbf{L} = (L, \leq, \wedge, \bot)$ has a monoidal structure $(L, \bullet, 1)$, then $\bot = 1$.

Now we can lift the monoidal structure of L to the level of $Tree_L$.

Proposition 3 Let $\mathbf{L}$ be a meet-semilattice with a monoidal structure $(L, \bullet, 1)$. Then a tensor product $\otimes : Tree_L \times Tree_L \to Tree_L$ exists, producing a (non symmetric) monoidal category $(Tree_L, \otimes, \mathcal{I})$, where $\mathcal{I}$ is the one-path tree with trivial labeling 1.

Remark 2 [11] If $\bullet$ is left-cancellative, the tensor $(Tree, \otimes, \mathcal{I})$ is left-closed, i.e. $- \otimes \mathcal{Y}$ has a right adjoint $Tree(\mathcal{Y}, -)$ for every $\mathcal{Y}$. Monoidal left-closedness would allow speaking of the "tree leading from one state to another state", so that, going back in time along a behavior, one could recover uniquely the remaining part of it from a given state. However, this property is very strong and not verified for most examples here.

Proposition 4 A homomorphism of meet-semilattice monoids $\phi : \mathbf{L}' \to \mathbf{L}$ induces two monoidal functors $\Phi : Tree_{L'} \to Tree_L$ and $\Phi' : Tree_L \to Tree_{L'}$.

Example 2 Given a set A, let $A_\dagger = (A \cup \{\dagger\})$. We now let the free monoids $A_\dagger^*$ and A^* play the roles of $\mathcal{L}'$ and $\mathcal{L}$, respectively, and define a monoidal functor between them by introducing a function on words in $A_\dagger^*$, deleting instances of $\dagger$ as follows:

$$\text{DEL}(s) = \begin{cases} \epsilon & \text{if } s = \epsilon \\ \mu \bullet \text{DEL}(s') & \text{if } s = \mu \bullet s' \text{ and } \mu \neq \dagger \\ \text{DEL}(s') & \text{if } s = \dagger \bullet s' \end{cases}$$

DEL can be extended to a monoidal functor Δ from trees labelled with $A \cup \{\dagger\}$ to A-labelled trees, deleting $\dagger$'s on paths. On the other hand, according to Proposition 4, given the inclusion homomorphism $i : A^* \to A_\dagger^*$ we obtain also a pair of functors, namely the obvious inclusion functor $INC : Tree_{A^*} \to Tree_{A_\dagger^*}$ and the functor $RES : Tree_{A_\dagger^*} \to Tree_{A^*}$. The latter, given a tree $\mathcal{X}$, erases from it all the paths with labels containing $\dagger$. This operation corresponds to what in concurrent process algebra is called *restriction*. There is a (strict) mono $res : INC(RES(\mathcal{X})) \rightarrowtail \mathcal{X}$.

4.2 The case of overlapping operation

For the overlapping operations defined in Section 3 one proves the following:

Proposition 5 Let $op : V \times V \rightarrow V$ be an associative monotonic operation with τ as unit and satisfying Definition 5; then $(PI, \leq, \bullet_{op}, \iota_{\mathbf{0}})$ with $\bullet_{op}$ defined point-wise and coordinate-wise as above, is a meet-semilattice with a monoidal structure.

One can thus define generalized trees on $\mathbf{PI}$ as illustrated in Section 4.1.

5 The category *SymcatB*

We expose here for the sake of completeness the general categorical-theoretic arguments which allow the introduction of the notion of more-than-coherent logic in next section. The reader more interested in applications than in general theories can skip this section or, better, substitute everywhere the locally posetal 2-category $\mathcal{B}$ with a meet-semilattice $\mathbf{L}$ and $SymcatB$ with $Tree_L$. In Section 7 we will see that for our purposes we need this particular instance only.

Given a suitable locally posetal 2-category $\mathcal{B}$ ([20]), one obtains the category $SymcatB$ where for every pair of objects b and b' the hom-set $hom[b, b']$ of morphisms from b to b' is assumed to be cocomplete and *sups* are preserved by composition. We also assume the existence of an operation on hom-sets, behaving as a *meet* w.r.t. the order. If $\mathcal{B}$ is generated as the category of relations on a regular category $\mathbf{B}$, the meet operation is given by the pullback.

Definition 6 [Symmetric categories and functors]

1. [20] A symmetric $\mathcal{B}$-category $\mathcal{X} =< X, e_X, a_X >$ is a set X equipped with an *extent* function $e : X \rightarrow B$ and an *agreement* function $a : X \times X \rightarrow Mor(\mathcal{B})$ satisfying: $\forall x, y, z \in X$: 1) $a_X(x, x) = e_X(x)$; 2) $a_X(x, y) \leq e_X(x) \wedge e_X(y)$; 3) $a_X(x, y) \wedge a_X(y, z) \leq a_X(x, z)$; 4) $a_X(x, y) = a_X(y, x)$ where $\wedge$ denotes the meet operation.

2. [20] A $\mathcal{B}$-functor $f : \mathcal{X} \rightarrow \mathcal{Y}$ between two $\mathcal{B}$-categories is a function $f : X \rightarrow Y$, satisfying $\forall x, y \in X$: $e_X(x) = e_Y(f(x))$; $a_X(x, y) \leq a_Y(f(x), f(y))$

3. A $\mathcal{B}$-functor f is called *strict* if the following equation holds: $a_X(x, y) = a_Y(f(x), f(y))$

Definition 7 [Cartesian and coherent categories][10]

1. A cartesian category C is *regular* if it has stable images under pullbacks.

2. A regular category C is *coherent* if it has stable unions under pullbacks.

In the following, we give a series of results needed to introduce more-than-coherent logic in Section 6. As the focus of the paper is on logic rather than on categorical constructions, we omit the proofs.

Proposition 6 If $\mathcal{B}$ is locally cocomplete, then $SymcatB$ is a coherent category.

Corollary 1 In $SymcatB$, with every object $\mathcal{X}$, a distributive lattice $(Sub(\mathcal{X}), \cup, \cap)$ is associated, with $\cup$ and $\cap$ the *join* and *meet*, respectively. Given a morphism $f : \mathcal{X} \to \mathcal{Y}$, an image operator $\Sigma_f : Sub(\mathcal{X}) \to Sub(\mathcal{Y})$ exists, which is left adjoint to the inverse image operator f^*. The whole structure is stable under pullbacks.

We now consider *strict* morphisms in $SymcatB$: strictness is preserved by identity, composition and pullbacks; images are strict if the original morphisms are. We call $sSymcatB$ the subcategory of $SymcatB$ where morphisms are strict. $\mathcal{B}$ can be considered as the terminal object in $SymcatB$ as well as in $sSymcatB$.

Proposition 7 If $\mathcal{B}$ is locally cocomplete, then $sSymcatB$ is a Heyting category [10].

In other words, every object $\mathcal{X}$ in $SymcatB$ is associated with a pair of categories $Sub(\mathcal{X})$ and $sSub(\mathcal{X})$: the first one with a coherent structure, the second one, using only strict monos, with a Heyting structure. In particular, for any morphism $f : \mathcal{X} \to \mathcal{Y}$ and any strict mono $m : \mathcal{X}' \to \mathcal{X}$, we define $\Pi_f X' \equiv \{y \in Y \mid \forall x \in X(f(x) = y \implies x \in X')\}$, which will play the role of universal quantifier (right adjoint to f^*). From the existence of a universal quantifier one derives the "negative" operators: $\mathcal{X}' \implies \mathcal{X}'' \equiv \Pi_m(\mathcal{X}' \cap \mathcal{X}'' \rightarrowtail \mathcal{X}')$, where $m : \mathcal{X}' \rightarrowtail \mathcal{X}$ and $\neg\mathcal{X}' \equiv \mathcal{X}' \implies \bot$.

Proposition 8 For $\mathcal{X}$ in $SymcatB$ we have functors $i : sSub(\mathcal{X}) \to Sub(\mathcal{X})$ (*inclusion*) and $s : Sub(\mathcal{X}) \to sSub(\mathcal{X})$, left-inverse left-adjoint to i. s has also a left adjoint j.

For every sub-object structure in $SymcatB$, one defines all the usual set-theoretical operators including the "negative" ones: negation, implication and universal quantifier. However, these will have the expected adjunction properties for the strict subobjects only. In fact, if we apply the definition of $\Pi_f \mathcal{X}'$ to non strict monos, the result will be a strict one, while the correspondence necessary for the adjunction will work only one-way, i.e., for $f^*(\mathcal{Y}') \to \mathcal{X}'$, there is a unique $\mathcal{Y}' \to \Pi_f \mathcal{X}'$, but the opposite is not always true. The same happens for negation and implication. As for the positive operators, they will be the same for both strict and non strict subobjects; only in the case of strict ones will they coincide with those obtained from connectives/quantifiers defined from the subobject classifier.

Proposition 9 The following hold: 1) A mono (an epi) in $SymcatB$ is regular if and only if it is strict. 2) There is an object Ω in $SymcatB$ which classifies strict monos.

Remark 3 In set-theoretical terms, a non-strict mono corresponds to a subobject containing "elements" of an object with both an evaluated membership and an evaluated equality, in both cases possibly non maximal w.r.t. the object. Ω will classify only strict subobjects, i.e. it will classify subobjects w.r.t. their membership. However, since it ignores non-strict monos, it will be completely indifferent to the actual equality. On the other hand, we are interested in non-strict monos in order to take into account seriously non-determinism in computations; hence we need to consider a variation of both membership and equality.

Actually, under a technical condition, which is satisfied in our case, we can prove that $SymcatB$ contains a topos, composed of skeletal Cauchy complete (see [21]) symmetric $\mathcal{B}$-categories with strict $\mathcal{B}$-functors between them.

6 More-than-coherent-logic

Given the categorical structure presented in Section 5 we can now introduce, in a standard way, a new logical system, intermediate between coherent and first order logics.

Definition 8 [Logics]

1. A *coherent* logic [10] is a sorted language with formulae built on constants $\top$, $\bot$, connectives $\wedge$ and $\vee$, the existential quantifier $\exists$, the "equality predicate" and whose deductive system comprehends rules α), β), γ), δ), ζ) and θ) in Table 1.

2. A *first order* logic [10] is a sorted language containing all formulae of the language of coherent logic, plus those built on connectives $\neg$ and $\implies$ and the universal quantifier $\forall$, and whose deductive system comprehends all the inference rules for a coherent logic plus the rules ϵ) and η) in Table 1 (in this case, rules in θ) are a consequence of the others, in particular of ϵ).

3. A *more-than-coherent* logic is a sorted language containing all the first order logic formulae, and ewhose deductive system comprehends all the inference rules of first order logic, except for the second parts of the ϵ) and η) rules.

We can interpret terms and formulae of the language in the usual way (see [10]), by fixing a "context", i.e. a finite set of variables containing those appearing as free ones; a term is interpreted as a morphism from the product of the interpretations of the types of the variables in the context to the interpretation of the type of the term, while a formula is interpreted as a subobject of the product of the interpretations of the types of the variables in the context. A sequent between two formulae is satisfied *iff* there is an instance of the order between the corresponding subobjects.

Theorem 1 If $\mathcal{B}$ is locally cocomplete, $SymcatB$ is a model for a more-than-coherent logic.

Proof: As $SymcatB$ is coherent, we interpret its positive logical operators according to the rules for a coherent logic. As for the negative ones, we also interpret them using the Π_π operator for the universal quantifier, Σ_π for the existential quantifier, for some suitable projection π, and the obvious correspondences for the other operators. We are left to prove the validity of the first part of ϵ) and of η). To this end, one observes that formulae involving negative operators are interpreted in strict subobjects. Hence, if ϕ is interpreted in $\mathcal{X}_1$ and ψ in $\mathcal{X}_2$, for the first part of η) we have: $\mathcal{X}_1 \to \mathcal{X}_2$ implies $s(\mathcal{X}_1) \to s(\mathcal{X}_2)$, because s is a functor. The latter implies $s(\mathcal{X}_1) \to \Pi_\pi(s(\mathcal{X}_2))$, because the first part of η) holds in a Heyting category; then composing with $\mathcal{X}_1 \to s(\mathcal{X}_1)$ and using the fact that $\Pi_\pi(s(\mathcal{X}_2)) = \Pi_\pi(\mathcal{X}_2)$, we have $\mathcal{X}_1 \to \Pi_\pi(\mathcal{X}_2)$. In the same way one can prove the first part of ϵ). □

α) identity $\frac{}{\phi \vdash_{\vec{x}} \phi}$ substitution $\frac{\phi \vdash_{\vec{x}} \psi}{\phi[s/x] \vdash_{\vec{y}} \psi[s/x]}$ cut $\frac{\phi \vdash_{\vec{x}} \psi \ \psi \vdash_{\vec{x}} \chi}{\phi \vdash_{\vec{x}} \chi}$

β) equality $\frac{}{\top \vdash_{\vec{x}} x{=}x}$ $\frac{}{x{=}y \wedge \phi \vdash_{\vec{x}} \phi[y/x]}$

γ) conjunction $\frac{}{\phi \vdash_{\vec{x}} \top}$ $\frac{}{\phi \wedge \psi \vdash_{\vec{x}} \phi}$ $\frac{}{\phi \wedge \psi \vdash_{\vec{x}} \psi}$ $\frac{\phi \vdash_{\vec{x}} \psi \ \phi \vdash_{\vec{x}} \chi}{\phi \vdash_{\vec{x}} \psi \wedge \chi}$

δ) disjunction $\frac{}{\bot \vdash_{\vec{x}} \phi}$ $\frac{}{\phi \vdash_{\vec{x}} \phi \vee \psi}$ $\frac{}{\psi \vdash_{\vec{x}} \phi \vee \psi}$ $\frac{\phi \vdash_{\vec{x}} \chi \ \psi \vdash_{\vec{x}} \chi}{\phi \vee \psi \vdash_{\vec{x}} \chi}$

ϵ) implication $\frac{\phi \wedge \psi \vdash_{\vec{x}} \chi}{\psi \vdash_{\vec{x}} \phi \implies \chi}$ $\frac{\psi \vdash_{\vec{x}} \phi \implies \chi}{\phi \wedge \psi \vdash_{\vec{x}} \chi}$

ζ) existential quantifier $\frac{\phi \vdash_{\vec{x},y} \psi}{\exists y \phi \vdash_{\vec{x}} \psi}$ $\frac{\exists y \phi \vdash_{\vec{x}} \psi}{\phi \vdash_{\vec{x},y} \psi}$

η) universal quantifier $\frac{\phi \vdash_{\vec{x},y} \psi}{\phi \vdash_{\vec{x}} \forall y \psi}$ $\frac{\phi \vdash_{\vec{x}} \forall y \psi}{\phi \vdash_{\vec{x},y} \psi}$

θ) distributivity $\frac{}{\phi \wedge (\psi \vee \chi) \vdash_{\vec{x}} (\phi \wedge \psi) \vee (\phi \wedge \chi))}$ Frobenius $\frac{}{\phi \wedge \exists y \psi \vdash_{\vec{x}} \exists y (\phi \wedge \psi)}$

Table 1: Logical rules

A first order language with all the usual connectives and quantifiers can thus be associated with $SymcatB$. Formulae corresponding to strict subobjects (in particular the negative ones) will enjoy all the rules of a full first order (intuitionistic) logic, while all the other formulae will enjoy all the rules of a coherent logic plus the first parts of ϵ) and η).

7 Formulae on trees

We now give some examples illustrating the use of the language associated with $SymcatB$ with reference to the category $Tree$. In fact, we are able to define a first order language to speak about paths on these trees, i.e. computations in a concurrent framework or, as seen in Section 3, *interactive computations* on images.

A complete meet-semilattice $\mathbf{L}$ gives rise to a locally-posetal, locally-cocomplete 2-category $\mathcal{L}$, in the way described in Section 5. The category $Tree_L$, of L-labeled structured computations and simulations between them, coincides with the category whose objects are symmetric $\mathcal{L}$-categories and whose morphisms are the $\mathcal{L}$-functors between them. The terminal object in $Tree$ is given by $\mathbf{L}$ itself, thought of as a tree $(L, id_L, \wedge)$. Namely, it has all the elements as paths and the agreement between all of them is the meet.

Operations from Section 4.1 provide some predicates. For example, the mono $res : INC(RES(\mathcal{X})) \to \mathcal{X}$ is an interpretation of $\exists x[P_\dagger(x)]$, where $P_\dagger(x)$ means that the computation x does not contain an occurrence of the elementary move $\dagger$. Using negation, one can also express the occurrence of a given elementary move in every computation. Analogously, using the tensor product, one can express the factorization of a computation. In particular, if the tensor product has a right adjoint we are also able to say that "p is a computation from a state s to a state s'" using the object $[s', s]$ and the image of its unique morphism in the terminal tree. In this case the corresponding mono is not in general strict.

Strict monomorphisms in $Tree_L$ are injective simulations that strictly preserve agreement. In other words, a strict subtree $\mathcal{X}'$ of a given tree $\mathcal{X}$ contains some of its paths with the same extent and the same agreement as they have in $\mathcal{X}$. If a formula is interpreted in such a kind of subobject, it will behave as a first order formula.

As example of a non 'well behaved" formula, take $\exists x \exists x'[e(f(x)) = del(e(f(x')))]$, where the type of both x and x' has been interpreted in $\mathcal{X}$ and f is a functional symbol interpreted in a non-strict morphism $f : \mathcal{X} \to \mathcal{Y}$[1]. In concurrent process parlance, this means that we have a simulation of the process represented by $\mathcal{X}$ via the process represented by $\mathcal{Y}$, which is more deterministic than $\mathcal{X}$, and we state the existence of two computations in $\mathcal{X}$ which are simulated by computations in $\mathcal{Y}$ with the same extent, up to some extra labels in the second one. Due to non determinism, the subobject of $\mathcal{Y}$ corresponding to this formula is not strict, i.e. we can find two computations satisfying the condition, but their agreement could be smaller than the one they have when they are simulated in $\mathcal{Y}$.

In order to show the limitations of the deductive system of more-than-coherent logic with respect to first-order logic, consider the formula $\forall y[\exists x \exists x'[e(f(x)) = del(e(f(x')))) \wedge a(f(x), y) = \bot]]$, with x typed in $\mathcal{X}$ and y typed in $\mathcal{Y}$, suppose it to be true for the given $\mathcal{X}$ and $\mathcal{Y}$, and that we interpret it w.r.t. the context $\mathcal{Y}$. Then there will be a mono from $\mathcal{Y}$ to its interpretation. If we now remove the universal quantification, the new formula will be still interpreted as subobject of $\mathcal{Y}$, but since it corresponds to a non-strict one there will be no mono from $\mathcal{Y}$ to such a subobject. This fact falsifies the second part of rule η).

Coming back to our main example in Section 3, we can easily prove:

Proposition 10 *The following hold:*

1. *PI is a meet-semilattice with monoidal structure on V.*
2. *An intrinsic first order language exists, equipped with a more-than-coherent logic, to speak about non deterministic computations with images.*

The same happens in particular for the trace monoid $\mathcal{M}(E)$. As a consequence, once one defines all the terms and predicates mentioned in Section 3.2 using the operations defined on $\mathcal{M}(E)$, all the formulae appearing there are formulae of a first order language equipped with a more-than-coherent logic. This fact shows the use of the proposed language for computing with images, in particular in cooperative or interactive processes, with reference to online coloring.

[1] We will abuse notation a bit here and identify syntactical and semantical symbols.

8 Conclusions

Computation on multidimensional words is becoming standard practice for multimedia applications, as well as for representing evolution of distributed states. Ad hoc methods are usually devised for different numbers of dimensions or for modelling semantics.

We have proposed a categorical setting for describing such computations based on the ubiquitous operation of overlapping. This gives an interesting enriched categorical structure, accommodating a new type of logical system, called more-than-coherent logic, with the expressive power of first-order logic, but a weaker deductive system.

This supports reasoning on composition of different contributions where each intermediate state is "more defined" than the previous one. In particular, such a logical structure can be used to filter out moves which cannot contribute to reaching a desired final state. In [2], filters were used to explore the power of the overlapping operation, allowing the simulation of several rewriting mechanisms. Since more-than-coherent logic naturally emerges from the properties of the overlapping operation, this logic seems to provide a good setting for reasoning about different interactive phenomena involving images, and we plan to investigate its properties more deeply.

The overlapping operation can be parameterized to any number of dimensions and different types of value composition, preserving the required properties for a monoidal structure. Moreover, pointed words introduce a natural notion of synchronization where agents can cooperate in the construction of words defining the result of a computation only on designated positions. The overlapping operation exploited here is a point-wise one. An exploration of the structure underlying other types of operations may uncover different types of logic. As an example, grey-scale image morphology has been related to computations on complete lattices [17] and to fuzzy logic [5]. Moreover, it would be interesting to combine reasoning on computations with reasoning on their results, for example composing filtering on computations and on results.

References

[1] Bottoni, P. and A. Labella, *Pointed pictures*, JVLC **18** (2007), pp. 523–536.

[2] Bottoni, P. and A. Labella, *Cooperative construction of pointed pictures*, Rom. J. of IST (to appear).

[3] Bottoni, P., G. Mauri and P. Mussio, *From strings to pictures and back*, Rom. J. of IST **6** (2003), pp. 87–104.

[4] Dacey, M. F., *The syntax of a triangle and some other figures.*, Pattern Recognition **2** (1970), pp. 11–31.

[5] Deng, T.-Q. and H. J. A. M. Heijmans, *Grey-scale morphology based on fuzzy logic*, J. of Math. Im. and Vis. **16** (2002), pp. 155–171.

[6] Giammarresi, D. and A. Restivo, *Two-dimensional languages*, , **III**, Springer, 1997 pp. 215–267.

[7] Grammatikopoulou, A., *Prefix picture sets and picture codes*, web.auth.gr/cai05/papers/21.pdf.

[8] Halldórsson, M. M. and M. Szegedy, *Lower bounds for on-line graph coloring*, TCS **130** (1994), pp. 163–174.

[9] Japaridze, G., *Introduction to computability logic*, Annals of Pure and Applied Logic **123** (2003), pp. 1 – 99.

[10] Johnstone, P., "Sketches of an elephant," Oxford Science Publications, 2002.

[11] Kasangian, S. and A. Labella, *Observational trees as models for concurrency*, MSCS **9** (1999), pp. 687–718.

[12] Kirsch, R., *Computer interpretation of english text and picture patterns*, IEEE Trans. EC **13** (1964), pp. 363–376.

[13] Latteux, M., D. Robilliard and D. Simplot, *Figures composées de pixels et monoïde inversif*, Bull. Belg. Math. Soc. **4** (1997), pp. 89–111.

[14] Lawvere, F., *Functorial semantics of algebraic theories*, Proc. Nat. Acad. Sci. U.S.A. **50** (1963), pp. 869–872.

[15] Mazurkiewicz, A. W., *Trace theory*, in: *Advances in Petri Nets*, LNCS **255**, 1986, pp. 279–324.

[16] Milner, R., "Communication and concurrency," Prentice Hall International, 1989.

[17] Ronse, C., *Why mathematical morphology needs complete lattices*, Signal Processing **21** (1990), pp. 129 – 154.

[18] Shaw, A., *The formal picture description scheme as a basis for picture processing systems*, Inf. and Cont. **14** (1969), pp. 9–52.

[19] Siromoney, R. and K. Krithivasan, *Parallel context-free grammars*, Inf. and Con. **24** (1974), pp. 155–162.

[20] Walters, R., *Sheaves and Cauchy-complete categories*, Cahiers de Topologie et Geometrie Diff. **22** (1981), pp. 283–286.

[21] Walters, R., *Sheaves on sites as Cauchy-complete categories*, J. Pure Appl. Algebra **24** (1982), pp. 95–102.