Contents lists available at ScienceDirect

Journal of Pure and Applied Algebra

journal homepage: www.elsevier.com/locate/jpaa

# Conduché property and *Tree*-based categories[☆]

Stefano Kasangian[a], Anna Labella[b,*]

[a] Dip. di Matematica, Università di Milano, Italy

[b] Dip. di Informatica, Università di Roma "La Sapienza", Italy

ARTICLE INFO

*Article history:*
Received 11 March 2008
Received in revised form 20 November 2008
Available online xxxx
Communicated by J. Adámek

*MSC:*
18D20
68Q55

ABSTRACT

This paper focuses on a property of enriched functors reflecting the factorisation of morphisms, used in concurrency semantics. According to Lawvere [F.W. Lawvere, State categories and response functors, 1986, Unpublished manuscript], a functor strictly reflecting morphism factorisation induces a notion of state on its domain, when it is considered as a control functor. This intuition works both in case of physical and computing processes [M. Bunge, M.P. Fiore, Unique factorisation lifting functors and categories of linearly-controlled processes, Math. Structures Comput. Sci. 10 (2) 2000 137–163; M.P. Fiore, Fibered models of processes: Discrete, continuous and hybrid systems, in: Proc. of IFIP TCS 2000, in: LNCS, vol. 1872, 2000, pp. 457–473]. In this note we investigate a more general property in the family of models we proposed elsewhere for communicating processes, and we assess their bisimulation relations [S. Kasangian, A. Labella, Observational trees as models for concurrency, Math. Structures. Comput. Sci. 9 (1999) 687–718; R. De Nicola, D. Gorla, A. Labella, Tree-Functors, determinacy and bisimulations, Technical Report, 02/2006, Dip. di Informatica, Univ. di Roma "La Sapienza" (Italy), 2008 (submitted for publication), http://www.dsi.uniroma1.it/%7Egorla/papers/DGL-TR0206.pdf]. Hence, we adapt the notion of "Conduché condition" [F. Conduché, Au sujet de l'existence d'adjoints à droîte aux foncteurs image reciproque dans la catégorie des catégories, C. R. Acad. Sci. Paris 275 (1972) A891–894] to the context of enriched category theory. This notion, weaker than the original "Moebius condition" used by Lawvere, seems to be more suitable for the description of the concurrency models parametrised w.r.t. a base category via the mechanism of change of base, actually. The base category is a monoidal 2-category; a category of generalised trees, *Tree*, is obtained from it. We consider Conduché *Tree*-based categories, where enrichment reflects factorisation of objects in the base category. We prove that a form of Conduché's theorem holds for Conduché *Tree*-functors. We also show how the Conduché condition plays a crucial role in modelling concurrent processes and bisimulations between them. The notions of "state preservation" and "determinacy" [R. Milner, Communication and Concurrency, Prentice Hall International, 1989] are formally characterised.

## 1. Introduction

At the beginning of July 1982, in Trieste, the first author attended to a lecture by Bill Lawvere [1] on "Thermodynamics of deformations of continuous bodies, non homogeneous, with memory, far from equilibrium", and heard what was then called the "Moebius functor" (now called UFL, unique factorisation lifting, functor, [2,3]) in connection with the notion of "state".

[☆] To the memory of Max Kelly.
[*] Corresponding author.
*E-mail addresses:* stefano.kasangian@mat.unimi.it (S. Kasangian), labella@di.uniroma1.it (A. Labella).

0022-4049/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.jpaa.2009.05.008

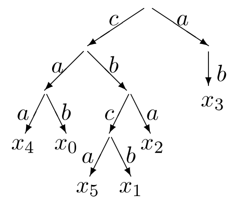

**Fig. 1.** An $A^*$-labelled tree $\mathcal{X}$ with six paths (representing computations or successful experiments).

Kasangian thought about the possibility of a weaker condition, where unicity is not required in a strict sense. Here, this possibility goes under the name of the *Conduché* condition, because it was originally used to prove Conduché's theorem [4].

In this paper we make the case that in Computer Science this weaker notion can play a relevant role. To explain this, we base our argument on our approach to concurrency, as published in [5].

Following Milner (see [6]), we considered concurrent agents as black boxes whose behaviour can be detected through a series of experiments. In order to do so, we needed to define a system of "observers" which perform "experiments". For the sake of simplicity, observers can be represented as words in a free monoid (interleaving semantics) and agents as automata whose states cannot be described. The observers "explore" the agents by making experiments on them: an *experiment on an agent consists of its interaction with a particular observer*. This interaction can be viewed as a Hoare synchronisation between observers and agents, in the sense that the observer and the agent are compatible, as long as they are *"doing the same thing"*. Given a system of observers, it is possible to explore and reconstruct the behaviour of an agent through experiments, as we will immediately see. Agents, considered as described via their behaviour detected as above, are called *processes*.

In describing the behaviour of a process as the record of all possible successful experiments, we are not far away from automata theory, where an automaton is described by the languages (sets of words on a given alphabet $A$) "operating the transition from a state $s$ to a state $s'$" (local behaviour), or the sets of words" accepted while going from $s$ to $s'$", i.e.

$$L_{s,s'} = \{w \in A^* \mid \delta(s, w) = s'\}.$$

Initial and terminal states can be specified when necessary. In this case, the language of words leading from the initial state to the terminal ones describes the global behaviour of the automaton. This set $L_{s,s'}$ can be considered as the report of a family of experiments made on the automaton in state $s$, trying to reach state $s'$, and using observers in the free monoid $A^*$. We associate the "successful" word (or string) with a computation.

Formally, $A$-automata are modelled as categories enriched in the base 2-category [7] provided by the structure of the languages [8], $\wp(A^*)$. Accordingly, comparisons between automata are defined in terms of "change of base" [9], i.e. in terms of the local structure. On the other hand, the usual (trace or language) equivalence introduced for automata is global, taking into account the global behaviour only. Thus, it "forgets" the states that one passed through.

When moving from essentially deterministic processes (like automata) to non-deterministic ones, greater attention must be spent on the notion of state, because it is necessary to be careful with the possible choices offered at any specific state. In this setting, "sets of strings" are no longer the best candidates to represent non-deterministic behaviours.

In the non-deterministic case, when we identify a state with its possible future (i.e., with the structure of its possible computations), we get a tree-shaped labelled structure[1] instead of a set of words; this is necessary to take all possible choices into account. A tree can be naturally described as a set of labelled paths suitably glued together; such a gluing provides information about the ramifications, distinguishing trees from languages (see Fig. 1).

The information (called the *extent*) about computation $p$, given by a successful experiment provided by a string $w$ is denoted by $e(p) = w$, while the relationship between two computations, $p_1$ and $p_2$, detected by two successful experiments performed on a given agent, is called *agreement* and it is denoted by $a(p_1, p_2)$. The agreement expresses *how long two computations cannot be distinguished through experiments* and it will be a prefix of both the strings associated to the two computations. Hence, the notion of agreement provides some information about the internal structure of the agents and their evolution during the experiments. In Fig. 1 we show a graphical representation of the following data, reported by experiments:

- $e(x_0) = cab$
- $e(x_1) = cbcb$
- $e(x_2) = cba$
- $e(x_3) = ab$
- $e(x_4) = caa$
- $e(x_5) = cbca$
- $a(x_0, x_1) = a(x_0, x_2) = a(x_0, x_5) = c$
- $a(x_0, x_3) = \epsilon$

[1] Algebraically, this is captured by the absence of right distributivity of concatenation over choice.

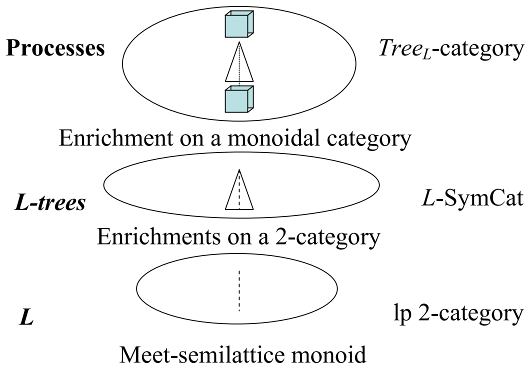

**Fig. 2.** The three levels of our construction.

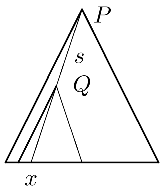

**Fig. 3.** The intermediate state $Q$ after $s$.

- $a(x_0, x_4) = ca$
- …

Back to automata theory, we start with the free monoid $A^*$, then we consider the structure of languages $\wp(A^*)$, i.e., the set of possible subsets of $A^*$ and, finally, we describe automata as sets of states with an element of $\wp(A^*)$ between any pair of them. This procedure can be formally described as an enrichment of the set of states of an automaton in the monoidal category $\wp(A^*)$, where the Frobenius product of languages is the tensor product [8]. In the case of processes, we must replace $\wp(A^*)$ with the structure of $A^*$-labelled trees, with a suitable operation of concatenation defined among them. In fact, we put a tree between two states.

We will perform a further step, because the relationship between the set of elementary labels $L$, suitably considered as a 2-category **L**, and $L$-labelled trees can be again thought in terms of an enrichment, though of a slightly different kind.[2] This 3-level construction can be annoying, but it allows us to stress the fact that all the properties we are able to detect at the top level, are essentially inherited from properties already present at the very basic level of **L** (see Fig. 2). Therefore, this level is crucial in making choices.

Formally, a system of observers can be described as a symmetric monoidal lp 2-category, whose objects are the observers (generalised words), whose 1-cells are "common prefixes" in a rather general sense [11] and whose 2-cells are usually given by the prefix relation. The set of all possible computations between two "states" of the system (or two agents, in the usual interleaving terminology) turns out to be *structured by the chosen agreement between the corresponding successful experiments*. Such a structured set is called a *structure of computations*, and is modelled as a *category based on the 2-category of observers*, in the sense of [7].

*Trees of computations* (viewed as enriched categories) are gathered in a *monoidal category Tree*, like the one described in [5], in which categories of processes are enriched.

The morphisms in the monoidal category of trees of computations can be thought of as *simulations*, because every computation in the domain is mapped into a similar one in the codomain, while agreement can increase, making the behaviour more deterministic, in correspondence with a possibly better knowledge about it. By varying the base 2-category **L**, different kinds of semantics can be modelled; our results about categorical characterisations of bisimulations apply to those different models [12].

Let us now address the problem of "finding out" states in a process when we know about its behaviour. The most intuitive idea is the following: given an instance of our semantics, whenever we can factorise a path $x$ of the (local or global) behaviour of a process $P$, that is whenever we can stop along a computation, say after performing a prefix $s$ of $e(x)$, we find a new process $Q$ (made out of the residual part of $P$ after that prefix), such that the part of computation already performed leads from $P$ to $Q$. The process $Q$ will represent an intermediate state along the given computation (see Fig. 3).

Conduché property amounts to requiring that a factorisation at the level of the base category can be properly lifted to the level of the category of trees and then to the level of the category of processes.

[2] The two kinds of enrichment are instances of the same notion of *B-category*, where $B$ is a general 2-category, as defined in [10].

We will see that in the more interesting cases of tree-semantics, the enriched categories of involved processes enjoy a sort of Conduché property with respect to the 2-category **L**. That is, factorisation of objects in the base category can be suitably lifted to the level of trees and then to the level of processes, producing a proper factorisation of the corresponding computation. This also allows us to detect a state in the process, or something quite near to a state. Moreover, if we impose equivalence relations corresponding to the more common bisimulations to these categories of trees [13], we get that they are related to the existence of enriched functors enjoying variants of a lifting factorisation property [12]. In particular, *determinacy* [6], that is the preservation of states by an equivalence associated with an enriched endofunctor, is guaranteed if and only if it enjoys the analogues of the Conduché property in this enriched context.

Coupling this result with Lawvere's position about states and determinism in physics, mentioned above, we decided to study in a systematic way the possible extension of Conduché's result to the *Tree*-enriched case. What we have called "Conduché property" has a nice consequence for an ordinary functor, namely the existence of a right adjoint to the corresponding inverse-image functor [4]. The main result of this paper is the extension of this result to *Tree*-based functors. As a consequence, it follows that a *Tree*-functor enjoying the Conduché property, and sending a behaviour to its minimal representative w.r.t. a given equivalence (as is the case for some very common bisimulations – see Section 4), induces an inverse image functor on the semantics that preserves limits and colimits. In other words, we can indifferently make canonical constructions on original behaviours or on their minimal representatives for such a kind of bisimulation.

As suggested by the construction of our semantics involving a two-level enrichment [14], we will proceed by defining and proving properties, bottom up through enrichments, starting from the base 2-category.

Section 2 will be devoted to formally describing the category of trees with its properties, while Section 3, the core of this paper, will deal with the problem of lifting factorisation, and will contain the main mathematical results. Section 4 presents results through examples coming from process semantics and bisimulations, and Section 5 will provide some final considerations on the relationships to other research about lifting factorisation properties and its possible future developments.

Our general references, besides [5], are Max Kelly's book [15] and Bob Walters paper [7].

## 2. Generalized trees

We now take from [5] the notion of a category of labelled trees, intended there to provide models for the process behaviours and some of its properties. A (generalised) tree, as the ones described in the introduction, is a symmetric **L**-category, where **L** is a meet-semilattice monoid enjoying the left-cancellation property, thought as a locally posetal 2-category. This approach allows us to define morphisms between trees as **L**-functors, which turn out to be simulations between behaviours, and to move from a kind of model to another by simply considering a 2-functor from a base category **L** to $\mathbf{L}'$, and the associated change of base. For example Milner's processes are $\mathbf{A}^*$-categories, where $\mathbf{A}^*$ is the 2-category associated with the free monoid on the alphabet of labels $A$, while $T$-categories are event structures of a certain kind, when $T$ is the 2-category associated with the monoid of Mazurkiewicz traces [16]. A typical 2-functor between base categories is DEL from $\mathbf{L}_\tau$ to **L**, which deletes silent moves (denoted by $\tau$) from processes.

We will now recall the construction in the general case, and state the properties enjoyed by the structures involved (see [5]).

**Definition 1.** A *complete meet-semilattice* $\mathbf{L} = (L, \leq, \wedge, \perp)$ is a partial order with greatest lower bound for any non-empty family of elements, $\wedge$. The bottom element is denoted by $\perp$.

A *complete meet-semilattice monoid* is a monoid $(L, \bullet, 1)$ such that the prefix relation between its elements (as usual defined as $s \leq t$ iff there exists $u \in L$ such that $s \bullet u = t$) is an order and induces a complete meet-semilattice structure.

A complete meet-semilattice monoid enjoys the *left-cancellation property* if, for every $s, t, u \in L$ such that $s \bullet t = s \bullet u$, it holds that $t = u$.

**Remark 1.** Due to the definition of the order in **L**, $\perp$ is provably equal to 1. Moreover, a complete semilattice has a join for every family that has an upper bound. From now on, we shall only consider complete meet-semilattice monoids enjoying the left-cancellation property.

Please notice that $\leq$ is a partial order – not simply a preorder – and, for this reason, the monoid $(L, \bullet, 1)$ lacks non-trivial invertible elements.

**Proposition 1.** 1. *A meet-semilattice* **L** *can be thought of as a symmetric lp* 2*-category* [7], *that we will still denote by* **L**.
2. *Composition in the meet-semilattice monoid* $(\mathbf{L}, \bullet, \mathbf{1})$ *induces a tensor product* $\odot : \mathbf{L} \times \mathbf{L} \to \mathbf{L}$ *with* 1 *as unit.* $\odot$ *is also a* 2*-functor.*

**Proof.** 1. Objects are elements of **L**, 1-cells are common prefixes composable via the meet operation. 2-cells are provided by the order relation.
2. $\odot$ is defined as $\bullet$ on objects, while on 1-cells it gives the first one of them - unless this is the identity of the first two objects. In that case it operates as $\bullet$ between them. With this definition 2-cells (order) are automatically respected. $\odot$ is associative and its neutral element is 1. □

**Remark 2.** The idea of considering a meet-semilattice as a locally posetal 2-category is due to R.F.C. Walters [7]. Here we are in the particular case where the order is induced by the particular monoid structure, and, therefore, **L** also becomes a monoidal 2-category.

Let **L**-SymCat be the category of symmetric **L**-categories and **L**-functors between them. Let us call it $Tree_L$ and call its objects (generalised) *trees*, because they turn out to be trees in the sense of the Introduction. We shall illustrate $Tree_L$ in more detail:

A symmetric **L**-category, i.e. a **L**-tree, $\mathcal{X}$ is a triple $(X, e_X, a_X)$ where $X$ is the set of *paths*, $e_X : X \to \mathbf{L}$ is the *extent map* and $a_X : X \times X \to \mathbf{L}$ is the *agreement* between paths such that, for every $x, y, z \in X$, it holds that:

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. $a_X(x, y) = a_X(y, x)$.

A **L**-functor $f : \mathcal{X} \to \mathcal{Y}$, is a function mapping paths into paths, strictly preserving labelling and non-decreasing the agreement between them, i.e.:

1. $e_X(x) = e_Y(f(x))$
2. $a_X(x, y) \leq a_Y(f(x), f(y))$.

$Tree_L$ has sums, products, initial and terminal objects. The tree $(L, id_L, \wedge)$, i.e. **L** itself, is the terminal object in $Tree_L$. Hence, there is a unique $Tree_L$-functor from every $Tree_L$-category **C** to **L**.

**Example 1.** In particular, if $A^*$ is the free monoid generated by the alphabet $A$, an $A^*$-category $\mathcal{X}$ will yield an $A^*$-labelled tree.

Let us notice that a 2-functor between meet-semilattices is actually a meet-preserving monotonic function.

If it is also a monoid homomorphism between meet-semilattice monoids, it is a monoidal 2-functor between the associated monoidal 2-categories.

**Proposition 2.** *Given two meet-semilattice monoids and a 2-functor between them $\phi : \mathbf{L}' \to \mathbf{L}$, $\phi$ induces a functor $\Phi : Tree_{L'} \to Tree_L$.*

**Proof.** Given a 2-functor $\phi : \mathbf{L}' \to \mathbf{L}$ and a $\mathcal{X}' = (X, e'_X, a'_X)$ in $Tree_{L'}$, $\Phi(\mathcal{X}') = (X, e_X, a_X)$, where

- $e_X(x) = \phi(e'_X(x))$
- $a_X(x, y) = \phi(a'_X(x, y))$

$\Phi(\mathcal{X}')$ is an object of $Tree_L$ because $\phi$, being a 2-functor, preserves meets. The map from $X$ to $Y$, defining a morphism from $\mathcal{X}' = (X, e'_X, a'_X)$ to $\mathcal{Y}' = (Y, e'_Y, a'_Y)$, defines a morphism from $\mathcal{X} = (X, e_X, a_X)$ to $\mathcal{Y} = (Y, e_Y, a_Y)$ of $Tree_L$ as well. The preservation of identities and of composition by $\Phi$ is straightforward. □

Given a meet-semilattice monoid **L** and two **L**-trees $\mathcal{X}$ and $\mathcal{Y}$, we can form the *sequential composition of $\mathcal{X}$ and $\mathcal{Y}$*, $\mathcal{X} \otimes \mathcal{Y} = (Z, e_Z, a_Z)$, as follows:

- $Z = X \times Y$
- $e_Z(x, y) = e_X(x) \bullet e_Y(y)$
- $a_Z((x, y), (x', y'))$ is $a_X(x, x')$, if $x \neq x'$, and $e_X(x) \bullet a_Y(y, y')$, otherwise.

In the sequel, we will denote the path $(x, y)$ in $\mathcal{X} \otimes \mathcal{Y}$ by $x;y$.

This operation is a functor.

**Proposition 3.** *In analogy with the tensor product $\odot : \mathbf{L} \times \mathbf{L} \to \mathbf{L}$, the operation $\otimes$ can be extended to a tensor product $\otimes : Tree_L \times Tree_L \to Tree_L$, making $(Tree_L, \otimes, \mathcal{I})$ a (non-symmetric) monoidal category where $\mathcal{I}$ is the one-path tree with trivial labelling.*

$\otimes$ corresponds to Frobenius product in language theory, and, as in language theory, we can define the derivative of $\mathcal{X}$ reached after an $s \in \mathbf{L}$ along a given path $x$.

Being $Tree_L$ a monoidal category, we can consider the category *$Tree_L$-Cat* of $Tree_L$-categories and $Tree_L$-functors. The meet semilattice monoid **L** is the terminal $Tree_L$-category, with only one object and the terminal tree $L$ as hom-object.

**Definition 2.** Let $\mathcal{X} = (X, e_X, a_X)$ be a tree. The *derivative* reached in $\mathcal{X}$ along the path $x\,(\in X)$ after $s\,(\leq e_X(x))$ – written $\mathcal{D}(\mathcal{X}, x, s)$ – is the trivial tree $(\{x\}, 1, 1)$, if $e_X(x) = s$, otherwise it is the tree $(Y, e_Y, a_Y)$ where

- $Y = \{x' \in X \mid a_X(x, x') \geq s\}$,
- $e_Y(x') = e_X(x') - s$,
- $$a_Y(x', x'') = a_X(x', x'') - s,$$

where by the symbol "−" we mean the truncation of a prefix.

Fig. 4 illustrates the terminology. The construction of the derivative yields an adjoint to the tensor product.

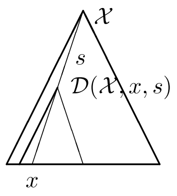

**Fig. 4.** The derivative of $\mathcal{X}$ along $x$ after $s$.

**Proposition 4** ([5]). *$(Tree_L, \otimes, \mathcal{I})$ is left-closed, i.e. the functor $- \otimes \mathcal{Y}$ has a right adjoint $Tree_L(\mathcal{Y}, -)$ for every $\mathcal{Y}$.*

**Proof.** Let us sketch the proof of the assertion. $Tree_L(\mathcal{Y}, \mathcal{X})$ consists of the part of $\mathcal{X}$ containing all the paths from the root of $\mathcal{X}$ to a homomorphic copy of $\mathcal{Y}$. Formally, it is the tree $Tree_L(\mathcal{Y}, \mathcal{X}) = (T, e, a)$, where:

- $T = \{\pi_{s,x} : \mathcal{Y} \to \mathcal{D}(\mathcal{X}, x, s)\}$ with $\pi_{s,x}$ a $Tree_L$-morphism, $s \leq e_X(x)$. $\pi_{s,x} = \pi_{s,x'}$ if $a(x, x') \geq s$ and they are equal as morphisms $\mathcal{Y} \to \mathcal{D}(\mathcal{X}, x, s) = \mathcal{D}(\mathcal{X}, x', s)$
- $e(\pi_{s,x}) = s$,
- $a(\pi_{s_1,x_1}, \pi_{s_2,x_2}) = s_1 \wedge s_2 \wedge a_X(x_1, x_2)$, for any pair of morphisms $\pi_{s_1,x_1}, \pi_{s_2,x_2}$ in $T$.
  The definition of $Tree_L(\mathcal{Y}, -)$ on morphisms is obtained via the morphism composition as usual. All the other required verifications are routine. □

**Remark 3.** We will often call elements of $Tree_L(\mathcal{Y}, \mathcal{X})$ paths from $\mathcal{X}$ to $\mathcal{Y}$. The composition of such paths – a $\pi_{s,x}$ from $\mathcal{X}$ to $\mathcal{Y}$ and a $\pi_{t,y}$ from $\mathcal{Y}$ to $\mathcal{Z}$ denoted by $\pi_{s,x}\pi_{t,y}$ – is the morphism obtained by composition $\mathcal{Z} \to \mathcal{D}(\mathcal{Y}, y, t) \to \mathcal{D}(\mathcal{X}, x, st)$, where $\pi_{s,x}(t) = y$ and the second morphism is the restriction of $\pi_{s,x}$ to $\mathcal{D}(\mathcal{Y}, y, t)$. Notice that there is always a 1-labelled path in $Tree_L(\mathcal{Y}, \mathcal{X})$, if $\mathcal{Y}$ is homomorphic to $\mathcal{X}$.

Left-closedness is a consequence of the left-cancellation property enjoyed by the meet-semilattice monoid **L**. Left-closedness provides an internal *hom* for $Tree_L$, so that we can now put an arrow-object of $Tree_L$ between two objects in $Tree_L$ in a canonical way. The resulting $Tree_L$-category will be denoted by $\mathbf{Tree}_L$.

$Tree_L$-categories (in particular $\mathbf{Tree}_L$) can also be thought as ordinary categories with extra structure. In fact, the hom-tree between two given objects consist in a "set of paths" that can be considered as a set of morphisms with extra information about extent and agreement. Analogously, $Tree_L$-functors are ordinary functors preserving extent and agreement of paths, i.e. their tree-structure. This can be formally seen by considering the forgetful monoidal functor $Supp : Tree_L \to Set$ mapping every tree into the set of its paths. The corresponding *change of base* turns a $Tree_L$-category into an ordinary category.

**Remark 4.** There is also a semiforgetful functor $SuppL : Tree_L \to Set|L$ mapping every tree to the set of its paths, but keeping the labelling. In this case we do not obtain monoidality w.r.t. the product functor in $Set|L$, but w.r.t. a "concatenation" functor easily definable in $Set|L$. The semiforgetful functor $SuppL : Tree_L \to Set|L$ has both a left and a right adjoint [5]. The left adjoint to a labelled set is given by the corresponding tree with trivial minimal agreement, while the right adjoint of a labelled set is given by the corresponding tree with maximal agreement.

**Proposition 5.** *Given the forgetful monoidal functor $Supp : Tree_L \to Set$*

1. *it induces a functor* $\mathbf{Supp} : Tree_L\text{–}Cat \to Cat$
2. *and for every $Tree_L$-category **B**, a functor $\mathbf{Supp}_B : Tree_L\text{–}Cat|\mathbf{B} \to Cat|B$, where $B$ is obtained from **B** by applying functor **Supp**. There are also functors $R_B : Cat|B \to Tree_L\text{–}Cat|\mathbf{B}$ right adjoint to $\mathbf{Supp}_B$ and left adjoint $L_B : Tree_L\text{–}Cat|\mathbf{B} \to Cat|B$.*

**Proof.** 1. This part is an instance of the change of base device, as observed above.
2. (sketch) The interesting part is the definition of the "reconstruction" functor $R_B$: it introduces extent and agreement on a set of morphisms of $\phi : X \to B$ by simply reflecting them along the map – due to $\phi$ – sending them into a hom-object in **B**. The functor $L_B$ reconstructs a tree by always choosing the trivial agreement. □

**Proposition 6.** 1. *A monoidal 2-functor $\phi : \mathbf{L}' \to \mathbf{L}$ induces a monoidal functor $\Phi : Tree_{L'} \to Tree_L$.*
2. *A monoidal functor $\Phi : Tree_{L'} \to Tree_L$ induces a $Tree_L$-functor $\boldsymbol{\Phi} : \mathbf{Tree}_{L'} \to \mathbf{Tree}_L$, its* effect [9].

**Proof.** 1. Given a monoidal 2-functor $\phi : \mathbf{L}' \to \mathbf{L}$, by Proposition 2, a functor $\Phi : Tree_{L'} \to Tree_L$ is also given. Monoidality of $\Phi$ is assured by the monoidality of $\phi$.
2. Due to the monoidality of $\Phi$, $Tree_{L'}$-category $\mathbf{Tree}_{L'}$ can be considered as a $Tree_L$-category by applying $\Phi$ to $\mathbf{Tree}_{L'}[\mathcal{X}', \mathcal{Y}']$ for every $\mathcal{X}', \mathcal{Y}'$. It will still be denoted by $\mathbf{Tree}_{L'}$. Now, we define a $Tree_L$-functor $\boldsymbol{\Phi} : \mathbf{Tree}_{L'} \to \mathbf{Tree}_L$ sending $\mathcal{X}' = (X, e'_X, a'_X)$ into $\Phi(\mathcal{X}') = (X, \phi(e_X), \phi(a_X))$ and $\boldsymbol{\Phi}(\mathbf{Tree}_{L'}[\mathcal{X}', \mathcal{Y}'])$ in itself as $\mathbf{Tree}_L[\Phi(\mathcal{X}'), \Phi(\mathcal{Y}')]$. □

**Example 2.** Suppose $\mathbf{L}'$ is given by the free monoid $(\mathbf{A} \uplus \{\tau\})^*$ (where $\tau$ is a special label) and **L** by the free monoid $\mathbf{A}^*$. We can define a monoidal 2-functor DEL between them by simply defining a function on words deleting $\tau$'s as follows (see Proposition 6):

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

DEL can be lifted to a monoidal functor – still called DEL – from $A \uplus \{\tau\}$-labelled trees to $A^*$-labelled trees which deletes $\tau$'s on paths. DEL can be used to transform $Tree_{(A\uplus\{\tau\})^*}$-categories into $Tree_{A^*}$-categories mapping the tree structure into a corresponding one, where $\tau$'s are deleted.

## 3. Reflecting factorisation

The original Conduché's result [4] concerned ordinary categories, i.e. $Set$-categories, and stated that a functor $F : B \to C$ has a certain lifting factorisation property iff its inverse image functor $F^* : Cat|C \to Cat|B$ has a right adjoint. The existence of an inverse image functor is due to the existence of pullbacks in $Set$. We are going to reproduce the original theory by Conduché in the $Tree_L$-enriched context, exploiting the fact that $Tree_L$ has pullbacks and its objects are sets with additional structure, as shown above (see Remark 4). For the sake of simplicity, in the sequel bold-face typed names, as **B**, **C**, **F**, . . . , will denote enriched categories and functors, while normally typed names, as $B, C, F, \ldots$, will denote ordinary categories and functors, obtained by applying the functor **Supp** (see Proposition 5.1) to the previous ones. As noticed above, every $Tree_L$-functor is also an ordinary functor preserving the tree-structure of homs, hence, in this line, it is possible to extend Conduché property to this kind of enriched functors.

In the sequel by $T$ we will mean a generic concrete monoidal category.

**Definition 3.** Given a $T$-category $\mathbf{C}$, we say that $(k, f_1, f_2)$ is a *factorisation* of $\mathbf{C}[c', c]$ for $(t_1 \otimes t_2, f)$, where $t_1$ is a singleton supported object and $f : t_1 \otimes t_2 \to \mathbf{C}[c', c]$ is a morphism in $T$, if there are an object $k$ and two arrows $f_1 : t_1 \to \mathbf{C}[k, c]$ and $f_2 : t_2 \to \mathbf{C}[c', k]$ such that the following diagram commutes ($m$ is the multiplication morphism in $\mathbf{C}$):

$$\begin{tikzcd}
t_1 \otimes t_2 \arrow[r, "f_1 \otimes f_2"] \arrow[dr, "f"'] & \mathbf{C}[k, c] \otimes \mathbf{C}[c', k] \arrow[d, "m_{c',k,c}"] \\
 & \mathbf{C}[c', c]
\end{tikzcd}$$

**Definition 4.** Given a $T$-functor $\mathbf{F} : \mathbf{B} \to \mathbf{C}$, $\mathbf{F}$ is *Conduché* iff

- given a path $\pi$ in $\mathbf{C}[\mathbf{F}(b'), \mathbf{F}(b)]$ and an inverse image of it, say $p \in (\mathbf{F}_{b',b})^{-1}(\pi)$ in $\mathbf{B}[b', b]$, for every factorisation $(k, f_1, f_2)$ of $\mathbf{C}[\mathbf{F}(b'), \mathbf{F}(b)]$ for $(t_1 \otimes t_2, f)$, with $\pi \in \mathrm{Im}(f)$, and $f'$ s. t. $\mathbf{F}_{b',b} f' = f$ and $p \in \mathrm{Im}(f')$, there is a factorisation $(h, f_1', f_2')$ of $\mathbf{B}[b', b]$, such that $\mathbf{F}(h) = k$ and the following diagram commutes

$$\begin{tikzcd}
\mathbf{B}[h, b] \otimes \mathbf{B}[b', h] \arrow[rr, "\mathbf{F}_{h,b} \otimes \mathbf{F}_{b',h}"] \arrow[dd, "m_{b',h,b}"'] & & \mathbf{C}[k, \mathbf{F}(b)] \otimes \mathbf{C}[\mathbf{F}(b'), k] \arrow[dd, "m_{\mathbf{F}(b'),k,\mathbf{F}(b)}"] \\
 & t_1 \otimes t_2 \arrow[ul, "f_1' \otimes f_2'"'] \arrow[ur, "f_1 \otimes f_2"] \arrow[dl, "f'"] \arrow[dr, "f"'] & \\
\mathbf{B}[b', b] \arrow[rr, "\mathbf{F}_{b',b}"'] & & \mathbf{C}[\mathbf{F}(b'), \mathbf{F}(b)]
\end{tikzcd}$$

- If two such factorisations $(h, f_1', f_2')$ and $(h', f_1'', f_2'')$ exist for $p$, they are *connected*, i.e. there exists a family of factorisations and a zig-zag of paths connecting the intermediate objects in $\mathbf{B}$,[3] making the resulting diagrams commute and becoming trivial when mapped by $\mathbf{F}$.

**Definition 5.** Given a $Tree_L$-category $\mathbf{C}$, we say that $\mathbf{C}$ is a *Conduché $Tree_L$-category* iff the unique $Tree_L$-functor from $\mathbf{C}$ to the terminal object $\mathbf{L}$ is Conduché.

[3] This means that there exist a family of factorisations $(h_i, f_1^i, f_2^i)$, $0 \le i \le k$, with $(h, f_1', f_2') = (h_0, f_1^0, f_2^0)$ and $(h', f_1'', f_2'') = (h_{k+1}, f_1^{k+1}, f_2^{k+1})$, and paths $p_1, \ldots, p_k$, where $p_i$ belongs to either $\mathbf{B}[h_i, h_{i+1}]$ or $\mathbf{B}[h_{i+1}, h_i]$.

The Conduché condition on a $Tree_L$-category $\mathbf{C}$ amounts to say that, whenever there is in $Tree_L$ a morphism $f : t_1 \otimes t_2 \to \mathbf{C}[c', c]$ – with $t_1$ a one-path tree – then there is an object $k$ in $\mathbf{C}$ and two arrows $f_1 : t_1 \to \mathbf{C}[k, c]$ and $f_2 : t_2 \to \mathbf{C}[c', k]$ such that $(k, f_1, f_2)$ is a factorisation for $t_1 \otimes t_2$ included in $\mathbf{L}$.

If two such factorisations exist for the same $p$, say $(k, f_1, f_2)$ and $(k', f_1', f_2')$, we can connect $k$ and $k'$ by a finite zig-zag of paths labelled by 1. [4]

As we anticipated in the introduction, such a Conduché property for $Tree_L$-categories indicates the possibility of defining a good notion of state for the processes modelled thereby. Conduché property for $Tree_L$-functors between them implies preservation of this notion of state as we will see in the following sections.

**Proposition 7.** ***Tree***$_L$ *is a Conduché $Tree_L$-category.*

**Proof.** It is an immediate consequence of the definition of $\mathbf{Tree}_L[\mathcal{Y}, \mathcal{X}]$ as $Tree_L(\mathcal{Y}, \mathcal{X})$, since the $\mathbf{Tree}_L$-functor in the terminal object is the labelling function. □

Notice that in ordinary categories this condition is trivially satisfied. In fact, *Set* is a category of trees on the trivial meet-semilattice monoid $\mathcal{I}$, freely generated by the empty set [17]. There, no factorisation in the sense of Definition 5 is possible.

**Proposition 8.** *Let $T$ be a monoidal category with pullbacks. Let $\mathbf{F}$ be a $T$-functor between two $T$-categories $\mathbf{B}$ and $\mathbf{C}$, then an* inverse image *functor $\mathbf{F}^*$ from $T-Cat|\mathbf{C}$ to $T-Cat|\mathbf{B}$ is defined.*

**Proof.** Let $\psi : \mathbf{Y} \to \mathbf{C}$ be a $T$-functor, $\mathbf{F}^{-1}(\mathbf{Y})$ is the $T$-category with pairs $(b, y)$ s.t. $\mathbf{F}(b) = \psi(y)$ as objects and enriched on $T$ as follows: $\mathbf{F}^{-1}(\mathbf{Y})[(b, y), (b', y')]$ is the pullback $\mathbf{B}[b, b'] \times \mathbf{Y}[y, y']$ over $\mathbf{C}[\mathbf{F}(b) = \psi(y), \mathbf{F}(b') = \psi(y')]$. Now, $\mathbf{F}^*\psi$ is the $T$-functor from $\mathbf{F}^{-1}(\mathbf{Y})$ to $\mathbf{B}$ corresponding to the first projection. Given $\psi' : \mathbf{Z} \to \mathbf{C}$ and a $T$-functor $\alpha : \mathbf{Y} \to \mathbf{Z}$ s.t. $\psi'\alpha = \psi$, then $\mathbf{F}^*(\alpha)$ is defined from $\mathbf{F}^{-1}(\mathbf{Y})$ to $\mathbf{F}^{-1}(\mathbf{Z})$ as follows: $\mathbf{F}^*(\alpha)(b, y) = (b, \alpha(y))$ and $\mathbf{F}^*(\alpha)$ induces a morphism from $\mathbf{F}^{-1}(\mathbf{Y})[(b, y), (b', y')]$ to $\mathbf{F}^{-1}(\mathbf{Z})[(b, \alpha(y)), (b', \alpha(y'))]$, exploiting the morphism going from $\mathbf{Y}[y, y']$ to $\mathbf{Z}[\alpha(y), \alpha(y')]$ and the pullback property. □

$\mathbf{F}^*$ has always a left adjoint $\mathbf{\Sigma_F}$, given by composition. Let us now restrict to the base category $Tree_L$ and prove the analogues of Conduché's Theorem in the enriched case, i.e. we state a necessary and sufficient condition to guarantee the existence of a right adjoint $\mathbf{\Pi_F}$ to $\mathbf{F}^*$.

The general enriched form of such a result does not exist because, in general, we are not able – at the moment – to speak about factorisations in an enriched context.

The proof of the main result will amount simply to make sure that the tree structure of the set of morphisms (paths) is preserved in the construction of the right adjoint.

**Lemma 1.** *Given a Conduché $Tree_L$-functor $\mathbf{F} : \mathbf{B} \to \mathbf{C}$, between $Tree_L$-categories, given $\pi, \pi' \in \mathbf{F}_{b',b}(\mathbf{B}[b', b])$, for every $p \in \mathbf{F}^{-1}_{b',b}(\pi)$ there exists a $p' \in \mathbf{F}^{-1}_{b',b}(\pi')$ such that $a_\mathbf{B}[\mathbf{b}', \mathbf{b}](p, p') = a_\mathbf{C}[\mathbf{Fb}', \mathbf{Fb}](\pi, \pi')$.*

**Proof.** Suppose $a_\mathbf{C}[\mathbf{Fb}', \mathbf{Fb}](\pi, \pi') = s$ and let us take the subobject $d$ of $\mathbf{C}[\mathbf{F}b', \mathbf{F}b]$ made out of the set $\{\pi, \pi'\}$ with their extent and agreement $s$. Then $\mathcal{I} \otimes d$ represents a trivial factorisation for $\pi$ (that always does exist) and since $\mathbf{F}$ is a Conduché $T$-functor, there is a corresponding factorisation of $p$ in $\mathbf{B}[b', b]$. Hence, there is a $T$-morphism from $d$ to $\mathbf{B}[b', b]$ sending $\pi$ to $p$. The image of $\pi'$ in this morphism will be the one required to be $p'$. □

**Theorem 1.** *Let $\mathbf{F} : \mathbf{B} \to \mathbf{C}$ be a $Tree_L$-functor between two $Tree_L$-categories, then the* inverse image *functor $\mathbf{F}^*$ has a right adjoint $\mathbf{\Pi_F} : Tree_L-Cat|\mathbf{B} \to Tree_L-Cat|\mathbf{C}$ if $\mathbf{F}$ is Conduché.*

**Proof.** • Let us give first a definition of $\mathbf{\Pi}_F$, in analogy with the non-enriched case, taking into account that it has to be the candidate to be the right adjoint to $\mathbf{F}^*$. Given the $Tree_L$-functor $\phi : \mathbf{X} \to \mathbf{B}$, we have to construct $\mathbf{\Pi}_F(\phi) : \Pi_F(\mathbf{X}) \to \mathbf{C}$.

If it does exist, the $Tree_L$-category $\mathbf{\Pi_F}(\mathbf{X})$ must have $Tree_L$-functors in $\mathbf{X}$ from inverse images of objects of $\mathbf{C}$ as objects, $Tree_L$-functors into $\mathbf{X}$ from the inverse images of paths between objects of $\mathbf{C}$ as morphisms, taking care of the agreement structure between them; concatenations of paths must also be obtained by pulling back along $\mathbf{F}$ the ones in $\mathbf{C}$ and then mapping the result into $\mathbf{X}$.

1. More formally, an object $c$ in $\mathbf{C}$ is a $Tree_L$-functor from the trivial one-object $Tree_L$-category $\mathbf{1}$ to $\mathbf{C}$. $\mathbf{1}$ has only one object $*$ and $\mathbf{1}[*, *]$ is the unit tree $\mathcal{I}$. $\mathbf{F}^*(c)$ is a $Tree_L$-functor from $\mathbf{F}^{-1}(\mathbf{1})$ to $\mathbf{B}$. Analogously, an object in $\mathbf{\Pi}_F(\mathbf{X})$ will be a $Tree_L$-functor from $\mathbf{1}$ to $\mathbf{\Pi}_F(\mathbf{X})$. By the property of the adjunction objects in $\mathbf{\Pi}_F(\mathbf{X})$ must be $Tree_L$-functors $\theta$ from $\mathbf{F}^{-1}(\mathbf{1})$ to $\mathbf{X}$ s.t. $\phi\theta = \mathbf{F}^*(c)$ for some $c$.
2. An arrow $\pi \in \mathbf{C}[c', c]$ is a $Tree_L$-functor $\pi$ from the $Tree_L$-category $\mathbf{2}_\pi$ with two objects $*$ and $**$ and only one non-trivial path in $\mathbf{2}_\pi[*, **]$ labelled like $\pi$ is, into $\mathbf{C}$.

[4] Again, this means that there exist paths $p_1, \ldots, p_k$ and objects $k_0, \ldots, k_n$ such that $k_0 = k$, $k_n = k'$ in a family of analogous factorisations, and $p_i$ belongs to either $\mathbf{C}[k_i, k_{i+1}]$ or $\mathbf{C}[k_{i+1}, k_i]$, making the resulting diagrams commute.

$\mathbf{F}^*(\pi)$ is a $Tree_L$-functor from $\mathbf{F}^{-1}(\mathbf{2}_\pi)$ to $\mathbf{B}$. Analogously, an arrow in $\mathbf{\Pi}_F(\mathbf{X})$ will be a $Tree_L$-functor from $\mathbf{2}_\pi$ to $\mathbf{\Pi}_F(\mathbf{X})$. By the property of the adjunction, arrow-objects $\mathbf{\Pi}_F(\mathbf{X})[\theta', \theta]$ must be constructed from $Tree_L$-functors $\kappa_\pi$ from $\mathbf{F}^{-1}(\mathbf{2}_\pi)$ to $\mathbf{X}$ s.t. $\phi\kappa_\pi = \mathbf{F}^*(\pi)$ if $\theta = \mathbf{F}^{-1}(dom)\kappa_\pi$ and $\theta' = \mathbf{F}^{-1}(cod)\kappa_\pi$. *dom* and *cod* are the obvious functors from $\mathbf{1}$ to $\mathbf{2}_\pi$ sending the unique object in $\mathbf{1}$ to the first and the second object in $\mathbf{2}_\pi$, respectively. The set of such functors $\kappa_\pi, \pi \in \mathbf{C}[c', c]$, bears a tree structure:

$e(\kappa_\pi) = e(\pi)$ and $a(\kappa_\pi, \kappa_{\pi'}) = \bigwedge a(\kappa_\pi(\pi, p_i), \kappa_{\pi'}(\pi', p'_j))$ for $p_i \in \mathbf{F}^{-1}(\pi)$, $p'_j \in \mathbf{F}^{-1}(\pi')$ and $a(p_i, p'_j) = a(\pi, \pi')$. This definition makes sense, because of Lemma 1.

3. A concatenation of paths $\pi; \rho$, labelled by $s \bullet t$, in $\mathbf{C}$ is a $Tree_L$-functor from a three-object $Tree_L$-category $\mathbf{3}_{\pi,\rho}$ into $\mathbf{C}$. $\mathbf{3}_{\pi,\rho}$ has three objects $*$, $**$ and $***$ and $\mathbf{3}_{\pi,\rho}[*, **]$, $\mathbf{3}_{\pi,\rho}[**, ***]$, $\mathbf{3}_{\pi,\rho}[*, ***]$ are the one path trees labelled by $s$, $t$ and $s \bullet t$, respectively.

   If $\mathbf{\Pi}_F$ is supposed to be a right adjoint to $\mathbf{F}^*$, compositions in $\mathbf{\Pi}_F(\mathbf{X})$ must be defined as the $Tree_L$-functors $\lambda$ from $\mathbf{F}^{-1}(\mathbf{3}_{\pi,\rho})$ to $\mathbf{X}$ s.t. $\phi\lambda = \mathbf{F}^*(\pi; \rho)$ for some $\pi$ and $\rho$. Now, such a condition defines the composition, i.e. just the "commutative triangles of paths" in $\mathbf{\Pi}_F(\mathbf{X})$ making it a $Tree_L$-category, iff the "resulting" path cannot be present without its components, and those are uniquely determined up to trivially labelled subpaths.

   This means that every path in $\mathbf{F}^*(\pi; \rho)$ must factorise into a path in $\mathbf{F}^*(\pi)$ concatenated with a path in $\mathbf{F}^*(\rho)$; if two such factorisations do exist, then the intermediate objects must be connected through trivially labelled paths. This is not always the case, but this is implied by the fact that $\mathbf{F}$ is Conduché.

   The fact that these compositions are morphisms in $Tree_L$, and not only set-theoretical functions, easily follows from the properties of the tensor product on $Tree_L$.

$\mathbf{\Pi}_F(\phi)$ will be defined mapping every $Tree_L$-functor above in its original object (resp. arrow) in $\mathbf{C}$.

- We will now verify the adjointness condition. As in the non-enriched case, the very definition of $\mathbf{\Pi}_F(\mathbf{X})$ implies the assertion. We have just to prove that pairs of maps corresponding to each other in the adjunction do preserve agreement if one of them does. Please, observe that, by Lemma 1, given $\psi : \mathbf{Y} \to \mathbf{C}$ and a pair of paths $\eta, \eta'$ in $\mathbf{Y}[y', y]$, there are always pairs of inverse images in $\mathbf{F}^*(\mathbf{Y}, \psi)$ such that their agreement will reach the agreement in $(\mathbf{Y}, \psi)$ between them.

  Given $\alpha : \mathbf{F}^*(\mathbf{Y}, \psi) \to (\mathbf{X}, \phi)$, we have that $a((p_i, \eta), (p'_i, \eta')) = a(\eta, \eta') \wedge a(p_i, p'_i)$.

  If $a(p_i, p'_i) = a(\pi, \pi')$, where $\pi$ is the common image of $p_i$ and $\eta$ and $\pi'$ is the common image of $p'_i$ and $\eta'$, then $a((p_i, \eta), (p'_i, \eta')) = a(\eta, \eta')$.

  Since $\alpha$ a $Tree_L$–$Cat|\mathbf{B}$-functor, $a((p_i, \eta), (p'_i, \eta', )) \leq a(\alpha(p_i, \eta), \alpha(p'_i, \eta'))$.

  We have to prove that its transpose $\alpha' : (\mathbf{Y}, \psi) \to \mathbf{\Pi}_F(\mathbf{X}, \phi)$ preserves agreement. In fact,

$$\alpha'(\eta) : \mathbf{F}^{-1}(\mathbf{2}_\pi) \to \mathbf{X}$$
$$\alpha'(\eta') : \mathbf{F}^{-1}(\mathbf{2}'_\pi) \to \mathbf{X}$$

where $\psi(\eta) = \pi$ and $\psi(\eta') = \pi'$. As usual, we define

$$\alpha'(\eta)(p_i) = \alpha(p_i, \eta)$$
$$\alpha'(\eta')(p'_j) = \alpha(p'_j, \eta').$$

We have to prove that

$$a(\eta, \eta') \leq a(\alpha'(\eta), \alpha'(\eta')).$$

By definition,

$$a(\alpha'(\eta), \alpha'(\eta')) = \bigwedge a(\alpha'(\eta)(p_i), \alpha'(\eta')(p'_j))$$

for all pairs $(p_i, p'_j)$ such that $a(p_i, p'_j) = a(\pi, \pi')$. But, for all such pairs, we know

$$a(\eta, \eta') = a((p_i, \eta), (p'_j, \eta')) \leq a(\alpha(p_i, \eta), \alpha(p'_j, \eta')).$$

Given $\beta : (\mathbf{Y}, \psi) \to \mathbf{\Pi}_F(\mathbf{X}, \phi)$ (commuting the suitable diagram), by $Tree_L$–$Cat|\mathbf{C}$ functoriality, we have $a(\eta, \eta') \leq a(\beta(\eta), \beta(\eta'))$ where

$$\beta(\eta) : \mathbf{F}^{-1}(\mathbf{2}_\pi) \to \mathbf{X}$$
$$\beta(\eta') : \mathbf{F}^{-1}(\mathbf{2}'_{\boldsymbol{\pi}}) \to \mathbf{X}.$$

Its transposed $\beta' : \mathbf{F}^*(\mathbf{Y}, \psi) \to (\mathbf{X}, \phi)$, defined as follows:

$$\beta'(p_i, \eta) = \beta(\eta)(p_i)$$
$$\beta'(p'_j, \eta') = \beta(\eta')(p'_j)$$

preserves agreement of arrow-objects in $\mathbf{F}^*\mathbf{Y}$. In fact, if $a((p_i, \eta), (p'_j, \eta'))$ does make sense, i.e. if $p_i$ and $p'_j$ have the same domain and the same codomain, we can consider two cases:

– $a(p_i, p'_j) = a(\pi, \pi')$, then

$$a((p_i, \eta), (p'_j, \eta')) = a(\eta, \eta') \leq a(\beta(\eta), \beta(\eta')) \leq a(\beta'(p_i, \eta), \beta'(p'_j, \eta'))$$

  by the definition of agreement between elements of $\mathbf{\Pi}_F(\mathbf{X}, \phi)$.

$$\begin{tikzcd}[column sep=huge, row sep=huge]
Tree_L\text{–}Cat|\mathbf{B} \arrow[r, "\mathbf{Supp}_B" description] \arrow[d, shift right=3, "\mathbf{\Pi_F}"'] \arrow[d, shift left=3, "\mathbf{\Sigma_F}"] & Cat|B \arrow[l, shift right=3, "L_B"'] \arrow[l, shift left=3, "R_B"] \arrow[d, shift right=3, "\Pi_F"'] \arrow[d, shift left=3, "\Sigma_F"] \\
Tree_L\text{–}Cat|\mathbf{C} \arrow[r, "\mathbf{Supp}_C" description] \arrow[u, "\mathbf{F}^*" description] & Cat|C \arrow[l, shift right=3, "L_C"'] \arrow[l, shift left=3, "R_C"] \arrow[u, "F^*" description]
\end{tikzcd}$$

**Fig. 5.** The doctrinal diagram.

– $a(p_i, p'_j) = s \leq a(\pi, \pi')$, then

$$a((p_i, \eta), (p'_j, \eta')) = a(\eta, \eta') \wedge a(p_i, p'_j) \leq a(\beta(\eta), \beta(\eta')) \wedge a(p_i, p'_j).$$

It is left to prove that

$$a(\beta(\eta), \beta(\eta')) \wedge a(p_i, p'_j) \leq a(\beta'(p_i, \eta), \beta'(p'_j, \eta')).$$

From Lemma 1 there exist $p'_i \in \mathbf{F}^{-1}(\pi')$ and $p_j \in \mathbf{F}^{-1}(\pi)$ such that $a(p_i, p'_i) = a(p_j, p'_j) = a(\pi, \pi')$. By the definition of a tree, it results $a(p'_i, p'_j) \geq s$, hence, being $\beta(\eta')$ a $Tree_L$–$Cat\,|\mathbf{B}$-functor,

$$a(\beta(\eta')(p'_i), \beta(\eta')(p'_j)) \geq s.$$

Now,

$$a(\beta(\eta), \beta(\eta')) \wedge a(p_i, p'_j) \leq a(\beta(\eta)(p_i), \beta(\eta')(p'_i)) \wedge a(p_i, p'_j).$$

By what we have just proved,

$$a(\beta(\eta)(p_i), \beta(\eta')(p'_i)) \wedge a(p_i, p'_j) \leq a(\beta(\eta)(p_i), \beta(\eta')(p'_i)) \wedge a(\beta(\eta')(p'_i), \beta(\eta')(p'_j))$$

and, by the definition of a tree, using point 3 in the definition of **L**-category,

$$a(\beta(\eta)(p_i), \beta(\eta')(p'_i)) \wedge a(\beta(\eta')(p'_i), \beta(\eta')(p'_j)) \leq a(\beta'(p_i, \eta), \beta'(p'_j, \eta')). \quad \square$$

**Proposition 9.** *Let* $\mathbf{F} : \mathbf{B} \to \mathbf{C}$ *be a* $Tree_L$*-functor between two* $Tree_L$*-categories and* $F : B \to C$ *the ordinary functor obtained from* **F** *by applying the functor* **Supp** *to it. If* **F** *is Conduché according to Definition 4, then* $F$ *is Conduché in the ordinary sense.*

**Proof.** It is immediate that the Conduché property for **F** implies Conduché property for $F$. $\square$

By combining the adjunctions of Proposition 5 with those for $\mathbf{F}^*$ and $F^*$, due to Propositions 8 and 9 and ordinary Conduché's theorem, we have for a Conduché $Tree_L$-functor **F** a sort of "doctrinal diagram" (see Fig. 5).

As usual in this paper, we will also see that the Conduché property can be lifted from the meet-semilattice level to the processes level.

**Definition 6.** Given a monoidal 2-functor $\phi : \mathbf{L}' \to \mathbf{L}$, $\phi$ is *Conduché* iff, when $u = s \bullet t$ in **L** and $\phi(u') = u$, then

- there are $s'$ and $t'$ in $L'$ s.t. $u' = s' \bullet t'$, $\phi(s') = s$, $\phi(t') = t$
- if there are $s''$ and $t''$ in $L'$ s.t. $u' = s'' \bullet t''$, $\phi(s'') = s$, $\phi(t'') = t$, then we can find a finite chain of analogous factorisations of $u'$, say $(s^i, t^i)$, $0 \leq i \leq k$, with $(s', t') = (s^0, t^0)$ and $(s'', t'') = (s^{k+1}, t^{k+1})$, and elements $v_1, \ldots, v_k$, s.t.
  - $s^i v_i = s^{i+1}$ and $v_i t^{i+1} = t^i$
  - or $s^{i+1} v_i = s^i$ and $v_i t^i = t^{i+1}$, with
  - and $\phi(v_i) = 1$.

**Remark 5.** Let us observe that the original *Conduché* property [4] concerned morphisms, while we are looking at objects. The reason is that we are going to consider objects in a monoidal category, which are due to become morphisms in the categories enriched therein. If $L$ and $L'$ are free monoids and $\phi$ is non-cancellative, "enjoying Conduché property" is equivalent to "being a coding", i.e. a letter to letter morphism.

**Theorem 2.** *Given a* Conduché *monoidal* 2*-functor* $\phi : \mathbf{L}' \to \mathbf{L}$ *(see Definition 6), and a factorisation in* $\mathbf{Tree}_L$*, we can recover a factorisation in* $\mathbf{Tree}_{L'}$ *thought as enriched on* $Tree_L$*. When two such corresponding factorisations exist, they are connected by a zig-zag of paths – as usual – labelled by elements of* $\mathbf{L}'$ *mapped by* $\phi$ *into the identity of* **L***.*

**Proof.** The first statement is trivial, because a $u$-labelled path $x$ from $\boldsymbol{\Phi}(\mathcal{X}')$ to $\boldsymbol{\Phi}(\mathcal{Y}')$ factorises for $(t_1 \otimes t_2, f)$ if there is a path $x \in \mathbf{Tree}_L[\mathcal{D}(\boldsymbol{\Phi}(\mathcal{X}'), x, s), \boldsymbol{\Phi}(\mathcal{X}')]$ labelled by a prefix $s$ of $u$ equal to the extent of the only path in $t_1$, and a morphism $f_2 : t_2 \to \mathbf{Tree}_L[\boldsymbol{\Phi}(\mathcal{Y}'), \mathcal{D}(\boldsymbol{\Phi}(\mathcal{X}'), x, s)]$. Given the same $x$ in $\mathcal{X}'$, by $f'$, we will have the same factorisation in $\mathbf{Tree}_{L'}$. As for the second one, when we think $\mathbf{Tree}_{L'}$ as enriched on $Tree_{L'}$, we might consider a corresponding factorisation, an element of a family of possible objects $\mathcal{D}(\mathcal{X}', x, s^i)$, with $s^i$ as in Definition 6. By Conduché condition on $\phi$, all these objects are connected via a zig-zag of paths labelled by the $v_j$s. $\square$

This last theorem means that a "state" in $\mathbf{Tree}_{L'}$ can be recovered from a state in $\mathbf{Tree}_L$, but only "up to zig-zag" of not important steps. As an instance, one can think of DEL in Example 2.

## 4. Bisimulations: An application

We now consider some instances of our theory. In the sequel we will deal with a *Tree*$_L$-subcategory of **Tree**$_L$, namely **Beh**$_L$, where the "set of paths" going from the second one to an isomorphic copy of the first one (instead of a homomorphic one, as it is implied by Proposition 4) is put between two objects of *Tree*$_L$. Our interest in **Beh**$_L$ is due to its capability of modelling process behaviours, because its hom objects correspond to operational semantics. It will be denoted by **Beh**, when the subscript $L$ will be obvious.

**Definition 7.** **Beh**$_L$ is the *Tree*$_L$-category with the same objects as **Tree**$_L$ and with $\mathbf{Beh}_L[\mathcal{Y}, \mathcal{X}] =< T, e, a >$, where:

- $T = \{\pi_{s,x} : \mathcal{Y} \to \mathcal{D}(\mathcal{X}, x, s)\}$ with $\pi_{s,x}$ a *Tree*$_L$-isomorphism, $s \le e_X(x)$. $\pi_{s,x} = \pi_{s,x'}$ if $a(x, x') \ge s$.
- $e(\pi_{s,x}) = s$,
- $a(\pi_{s_1,x_1}, \pi_{s_2,x_2}) = s_1 \wedge s_2 \wedge a_X(x_1, x_2)$, for any pair of isomorphisms $\pi_{s_1,x_1}, \pi_{s_2,x_2}$ in $T$.

**Proposition 10.** ***Beh***$_L$ *is a Conduché Tree*$_L$*-category.*

**Proof.** Also in this case the *Tree*$_L$-functor into the terminal *Tree*$_L$-category is given by the labelling function. Given $f : t_1 \otimes t_2 \to \mathbf{Beh}_L[\mathcal{Y}, \mathcal{X}]$, with $t_1$ a one-path tree, then we put $\mathcal{Z}$ as $\mathcal{D}(\mathcal{X}, x, s)$ if $x \in \mathrm{Im}(f)$. $\mathcal{Z}$ is determined up to isomorphisms and it makes the appropriate diagram commute. □

This fact means that, given a factorisation of a path in the behaviour of a process, we can find a unique "state" reached after the first part of the path.

**Proposition 11.** *Let* $\mathbf{F}$ *be a Tree*$_L$*-functor from* ***Beh*** *to* ***Beh****, then it is Conduché. In fact, it enjoys a stronger property (UFL), because the factorisation is, in this case, unique (up to isos).*

**Proof.** Given a path $\pi$ in $\mathbf{Beh}[\mathbf{F}(\mathcal{Y}), \mathbf{F}(\mathcal{X})]$ and an inverse image of it, say $p \in (\mathbf{F}_{\mathcal{Y},\mathcal{X}})^{-1}(\pi)$ in $\mathbf{Beh}[\mathcal{Y}, \mathcal{X}]$, given a factorisation $(\mathcal{K}, f_1, f_2)$ of $\mathbf{Beh}[\mathbf{F}(\mathcal{Y}), \mathbf{F}(\mathcal{X})]$, and $f'$ s. t. $\mathbf{F}_{\mathcal{Y},\mathcal{X}} f' = f$, and $\pi \in \mathrm{Im}(f)$, there is a unique (up to isos) factorisation $(\mathcal{Z}, f_1', f_2')$ of $\mathbf{B}[\mathcal{Y}, \mathcal{X}]$, where $p \in (f_1' \otimes f_2') m_{\mathcal{Y},\mathcal{Z},\mathcal{X}}$, such that $\mathbf{F}(\mathcal{Z}) = \mathcal{K}$ and the proper diagram commutes. In fact, by Proposition 10, $\mathcal{Z}$ is $\mathcal{D}(\mathcal{X}, p, s)$ if $s$ is the extent of the only element in $t_1$, hence it is unique. □

**Definition 8.** Given a *Tree*$_L$-functor $\mathbf{F} : \mathbf{B} \to \mathbf{C}$, $\mathbf{F}$ *reflects paths* iff for every $b \in Ob(\mathbf{B})$ and for every path $\pi \in \mathbf{C}[c, \mathbf{F}(b)]$ there are $b'$ and $p \in \mathbf{B}[b', b]$ s.t. $\mathbf{F}(b') = c$, $\mathbf{F}(p) = \pi$.

**Remark 6.** Let $\mathbf{F} : \mathbf{B} \to \mathbf{C}$ be a path reflecting, Conduché *Tree*$_L$-functor between two *Tree*$_L$-categories. Then, if $a(\pi_1, \pi_2) = s$ in $\mathbf{C}[c', c]$, for every $p_{i_1} \in \mathbf{F}^*(\pi_1)$, we can find $p_{i_2} \in \mathbf{F}^*(\pi_2)$ s.t. $a(p_{i_1}, p_{i_2}) = s$. (See Lemma 1).

From now on, for the sake of simplicity, we will restrict ourselves to the case where $\mathbf{L}$ is the free monoid $A^*$. In this case, our structure **Beh** strictly corresponds to operational semantics for labelled transition systems (LTS).

**Definition 9.** A symmetric relation $\mathfrak{R}$ between trees is a *strong bisimulation* (see [6]) if, for every $(\mathcal{X}, \mathcal{Y}) \in \mathfrak{R}$, it holds that $\forall x \in X \exists y \in Y$ such that $e_Y(y) = e_X(x)$ and $\forall s \le e_X(x) \forall \mathcal{X}' \cong \mathcal{D}(\mathcal{X}, x, s) \exists \mathcal{Y}' \cong \mathcal{D}(\mathcal{Y}, y, s)$ such that $(\mathcal{X}', \mathcal{Y}') \in \mathfrak{R}$.

Two trees are *strongly bisimilar*, written $\mathcal{X} \simeq_S \mathcal{Y}$, if and only if it exists a strong bisimulation relating them.

**Proposition 12** (*See [12]*). *Let* $\mathbf{F}$ : ***Beh*** $\to$ ***Beh*** *be a path reflecting Tree*$_L$*-functor, then it induces a strong bisimulation relation on* ***Beh****.*

The proof depends on path reflection and Conduché property.

In order to model a stronger non-determinism in computing processes, a special element $\tau$, mimicking a "silent move", was introduced by Milner in the labelling alphabet. Hence, we will consider the free monoid generated by the alphabet $A$ extended with the new label $\tau$. Let us call it $A^*{}_\tau$.

As we have seen in Example 2, the function $\delta$ can be canonically extended to a morphism of semilattice monoids between $\mathbf{A}^*_\tau$ and $\mathbf{A}^*$. Again, this function can be considered as a monoidal 2-functor DEL from $\mathbf{A}^*_\tau$ to $\mathbf{A}^*$, and can be lifted to a functor from $Tree_{A^*{}_\tau}$ to $Tree_{A^*}$. Simplifying the notation, we will denote by $Tree_\tau$ the category of trees with silent actions, i.e. $\mathbf{A}^*_\tau$-trees, and by *Tree* the category $Tree_{A^*}$.

DEL : $Tree_\tau \to Tree$ is the monoidal functor such that DEL $(X, e_X, a_X) = (Y, e_Y, a_Y)$ where

- $Y = X$,
- $e_Y(x) =$ DEL $(e_X(x))$,
- $a_Y(x, y) =$ DEL $(a_X(x, y))$.

Morphisms remain unchanged under DEL (functions defining morphisms from $\mathcal{X}$ to $\mathcal{Y}$ define morphisms from DEL $(\mathcal{X})$ to DEL $(\mathcal{Y})$ as well, indeed).

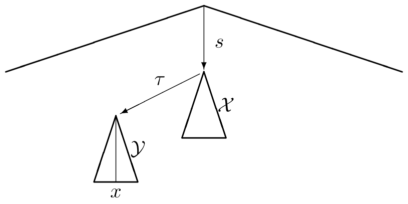

**Fig. 6.** Two derivatives ($\mathcal{X}$ and $\mathcal{Y}$) along $x$ in the presence of $\tau$'s.

Accordingly, we could define a $Tree_\tau$-category $\mathbf{Beh}_\tau$ in the same way as **Beh** was defined over *Tree*, but we can deal with it also as a *Tree*-category. The functor DEL is monoidal; hence, we can consider the effect on $\mathbf{Beh}_\tau$ of the change of base obtained by applying DEL to its hom-objects, while the objects remain unchanged. In the sequel we will call $\mathbf{Beh}_\tau$ this *Tree*-category.

**Notation.** By $s \preccurlyeq e_X(x)$ we mean that $s = \text{DEL}\,(t)$, for some $t \leq e_X(x)$ (where '$\leq$' is the prefix relation in the meet-semilattice monoid $A^*{}_\tau$).

Having ignored $\tau$'s in $Tree_\tau$, as we can do using the functor DEL above, a derivative is no longer uniquely determined by its access path after a given visible prefix $s$. To see this, examine Fig. 6: the same path, viz. $x$, leads to both the derivative $\mathcal{X} + \tau\mathcal{Y}$ and to $\mathcal{Y}$ along $\text{DEL}\,(s)$ or $\text{DEL}\,(s \bullet \tau)$. It is however important to notice that there is always the largest of such trees ($\mathcal{X} + \tau\mathcal{Y}$ here) and any other tree accessed this (like $\mathcal{Y}$) is a $\tau$-summand of this one. Hence, $\mathcal{D}_\tau(\mathcal{X}, x, s) = \{\mathcal{Z} \in \mathcal{D}(\mathcal{X}, x, t)$, for every $t$ s.t. $\text{DEL}\,(t) = s\}$ is a finite chain w.r.t. the order induced by $+$.

**Proposition 13.** *$\mathbf{Beh}_\tau$ is a Conduché* Tree*-category.*

**Proof** (*Sketch*). $\mathbf{Beh}_\tau$ is a Conduché Tree-category in its general form, because we can proceed in the proof as we did for **Beh**, but, in analogy with Theorem 2, now $\mathcal{D}_\tau(\mathcal{X}, p, s)$ is no longer a unique tree, i.e. in $\mathbf{Beh}_\tau$ we get a chain of trees connected by the originally $\tau$-labelled paths. $\tau$'s have disappeared in the morphisms because of the effect of functor DEL . Hence we can go between two such trees via a zig-zag of trivial paths. □

**Definition 10.** A symmetric relation $\mathfrak{R}$ on trees is a *branching bisimulation* (see [18]) if, for every $(\mathcal{X}, \mathcal{Y}) \in \mathfrak{R}$, it holds that $\forall x \in X \exists y \in Y$: $\text{DEL}\,(e_Y(y)) = \text{DEL}\,(e_X(x))$ and

1. $\forall s \preccurlyeq e_X(x)\ \forall \mathcal{X}' \in \mathcal{D}_\tau(\mathcal{X}, x, s)\ \exists \mathcal{Y}' \in \mathcal{D}_\tau(\mathcal{Y}, y, s)$ such that $(\mathcal{X}', \mathcal{Y}') \in \mathfrak{R}$;
2. $\forall s' \preccurlyeq e_Y(y)\ \forall \mathcal{Y}' \in \mathcal{D}_\tau(\mathcal{Y}, y, s')\ \exists \mathcal{X}' \in \mathcal{D}_\tau(\mathcal{X}, x, s')$ such that $(\mathcal{X}', \mathcal{Y}') \in \mathfrak{R}$.

Two trees $\mathcal{X}$ and $\mathcal{Y}$ are *branching bisimilar*, written $\mathcal{X} \simeq_B \mathcal{Y}$, if and only if there is a branching bisimulation relating them.

**Proposition 14** (*See [12]*). *If* **F** *be a path reflecting Conduché* Tree*-functor from $\mathbf{Beh}_\tau$ to $\mathbf{Beh}_\tau$, then it induces a branching bisimulation relation on $\mathbf{Beh}_\tau$.*

The paper [12] characterising bisimulations provides an example of a *Tree*-endofunctor on $\mathbf{Beh}_\tau$ not enjoying the Conduché property, but only a weak form of it.

**Definition 11.** A $Tree_L$-functor $\mathbf{F} : \mathbf{B} \to \mathbf{C}$, **F** is *weak-Conduché* iff

- given a path $\pi$ in $\mathbf{C}[\mathbf{F}(b'), \mathbf{F}(b)]$, for every factorisation $(k, f_1, f_2)$ of $\mathbf{C}[\mathbf{F}(b'), \mathbf{F}(b)]$ for $(t_1 \otimes t_2, f)$, with $\pi \in \text{Im}(f)$, and $f'$ s. t. $\mathbf{F}_{b',b} f' = f$, there is an inverse image of $\pi$, say $p \in (\mathbf{F}_{b',b})^{-1}(\pi)$ in $\mathbf{B}[b', b]$ and a factorisation $(h, f_1', f_2')$ of $\mathbf{B}[b', b]$, where $p \in \text{Im}(f')$, such that $\mathbf{F}(h) = k$ and the same diagram of Definition 4 commutes.
- If two such factorisations $(h, f_1', f_2')$ and $(h', f_1'', f_2'')$ exist, then they are connected in **B** via a zig-zag of $\tau$-labelled paths commuting with the given factorisations.

Though weaker than Conduché property – in the presence of path reflection – the weak-Conduché property implies the reflection of paths, because of the different position of quantifiers.

**Definition 12.** A symmetric relation $\mathfrak{R}$ on trees is a *weak bisimulation* (see [6]) if, for every $(\mathcal{X}, \mathcal{Y}) \in \mathfrak{R}$, it holds that $\forall x \in X \exists y \in Y$ s.t. $\text{DEL}\,(e_Y(y)) = \text{DEL}\,(e_X(x))$ and $\forall s \preccurlyeq e_X(x)\ \forall \mathcal{X}' \in \mathcal{D}_\tau(\mathcal{X}, x, s)\ \exists \mathcal{Y}' \in \mathcal{D}_\tau(\mathcal{Y}, y, s)$ such that $(\mathcal{X}', \mathcal{Y}') \in \mathfrak{R}$.

Two trees $\mathcal{X}$ and $\mathcal{Y}$ are *weakly bisimilar*, written $\mathcal{X} \simeq_W \mathcal{Y}$, if and only if a weak bisimulation exists relating them.

It is interesting to observe that Definition 10 differs from Definition 12 only for an extra symmetry requirement.

**Proposition 15.** *If F is a weak Conduché* Tree*-functor from $\mathbf{Beh}_\tau$ to $\mathbf{Beh}_\tau$, then it induces a weak bisimulation relation on $\mathbf{Beh}_\tau$.*

Summing up, Conduché property is enjoyed by all *Tree*-functors from **Beh** to **Beh**. This is not the case for $\mathbf{Beh}_\tau$, due to the presence of $\tau$. Nevertheless, the *Tree*-functor characterising branching equivalence still enjoys the property, while the *Tree*-functor characterising weak equivalence does not.

A *Tree*-functor $F$ enjoying weak Conduché property also satisfies that $F(\mathcal{I}) \cong \mathcal{I}$, therefore all our functors satisfy this second property.

In this way we are able to get a true characterisation of our *Tree*-functors (at least in the case of regular trees), because it is possible to prove an inverse result: in fact, we can find a *Tree*-functor inducing any of the three equivalences mentioned and enjoying the required property. The proof of this proposition is achieved through the definition of standard representatives, in the three cases (see [12]).

**Definition 13.** 1. Let $\equiv_S$ be the equivalence relation on paths of a tree $\mathcal{X}$ defined by $x \equiv_S x'$ if and only if $e_X(x) = e_X(x')$ and, for every $s \leq e_X(x)$, $\mathcal{D}(\mathcal{X}, x, s)$ is strongly bisimilar to $\mathcal{D}(\mathcal{X}, x', s)$. Let $|x|_S$ denote the $\equiv_S$-class of $x$. The *standard strong representative* of a tree $\mathcal{X} = (X, e_X, a_X)$ is the tree $\mathbf{S}\mathcal{X} = (SX, e_{SX}, a_{SX})$, where
- $SX = \{|x|_S \mid x \in X\}$;
- $e_{SX}(|x|_S) = e_X(x)$;
- $a_{SX}(|x|_S, |y|_S) = \bigvee_{x' \in |x|_S, y' \in |y|_S} a_X(x', y')$.

2. Let $\equiv_B$ be the equivalence relation on paths of a given tree $\mathcal{X}$ defined by $x \equiv_B x'$ if and only if DEL $(e_X(x)) =$ DEL $(e_X(x'))$ and, for every $s \preccurlyeq e_X(x)$, $\mathcal{D}_S(\mathcal{X}, x, s)$ is in a bijective correspondence with $\mathcal{D}_S(\mathcal{X}, x', s)$, such that corresponding trees are isomorphic. The *standard branching representative* of a tree $\mathcal{X} = (X, e_X, a_X)$ is the tree $\mathbf{B}\mathcal{X} = (BX, e_{BX}, a_{BX})$, where
- $BX = \{|x|_B \mid x \in X\}$;
- $e_{BX}(|x|_B) = \tau^{i_1}s_1\tau^{i_2}s_2 \ldots \tau^{i_n}s_n\tau^{i_{n+1}}$, with $i_k = |\mathcal{D}_{\mathcal{S}}(\mathcal{X}, x, s_1 \ldots s_{k-1})| - 1$, for $1 \leq k \leq n+1$ and $e_X(x) = \tau^{k_1}s_1\tau^{k_2}s_2 \ldots \tau^{k_n}s_n\tau^{k_{n+1}}$;
- $a_{BX}(|x|_B, |y|_B) = \tau^{i_1}s_1\tau^{i_2}s_2 \ldots s_m\tau^{i_{m+1}}$, with
  * $i_k = |\mathcal{D}_{\mathcal{S}}(\mathcal{X}, x, s_1 \ldots s_{k-1})| - 1$, for $1 \leq k \leq m$, and
  * $i_{m+1} = |\mathcal{D}_{\mathcal{S}}(\mathcal{X}, x, s_1s_2 \ldots s_m) \wedge \mathcal{D}_{\mathcal{S}}(\mathcal{X}, y, s_1s_2 \ldots s_m)| - 1$,[5]

  whenever $\bigvee_{x' \in |x|_B, y' \in |y|_B} a_X(x', y') = \tau^{k_1}s_1\tau^{k_2}s_2 \ldots \tau^{k_m}s_m\tau^{k_{m+1}}$.

($\mathcal{D}_{\mathcal{S}}(\mathcal{X}, x, s)$ denotes the family $\mathcal{D}_\tau(\mathcal{X}, x, s)$ where all the trees are replaced by their strong standard representatives.)

3. Let $\leq_B$ be the preorder relation on paths of a given tree $\mathcal{X}$ defined by $x \leq_B x'$ if and only if DEL $(e_X(x)) =$ DEL $(e_X(x'))$ and, for every $s \preccurlyeq e_X(x)$, there is a monotonic injective function from $\mathcal{D}_S(\mathcal{X}, x, s)$ to $\mathcal{D}_S(\mathcal{X}, x', s)$ such that the corresponding trees are isomorphic.

   When considered on $\equiv_B$-classes of paths, $\leq_B$ becomes a partial order with finite chains (recall that finiteness is guaranteed by the fact that we restricted ourselves to regular trees), hence with maximal elements; representatives of maximal classes will be called *maximal paths* in the original tree. In the sequel, $|x|_W$ will denote the equivalence class $|x'|_B$ of a maximal path $x'$ such that $|x|_B \leq_B |x'|_B$. The *standard weak representative* of a tree $\mathcal{X} = (X, e_X, a_X)$ is the tree $\mathbf{W}\mathcal{X} = (WX, e_{WX}, a_{WX})$, where
- $WX = \{|x|_W, \mid x \in X, x \textit{ maximal}\}$;
- $e_{WX}(|x|_W) = e_{BX}(|x|_B)$;
- $a_{WX}(|x|_W, |y|_W) = a_{BX}(|x|_B, |y|_B)$.

We can extend functions $S$, $B$, $W$ to *Tree*-functors. Let us fix a choice of the standard representative in the three cases.

**Proposition 16.** • *there is a path reflecting* Tree-*functor* $\mathbf{S}$ *such that, if* $\mathcal{X} \simeq_S \mathcal{Y}$, *then* $\mathbf{S}(\mathcal{X}) = \mathbf{S}(\mathcal{Y})$
- *there is a path reflecting, Conduché* Tree-*functor* $\mathbf{B}$ *such that, if* $\mathcal{X} \simeq_B \mathcal{Y}$, *then* $\mathbf{B}(\mathcal{X}) = \mathbf{B}(\mathcal{Y})$
- *there is a weak Conduché* $Tree_L$-*functor* $\mathbf{W}$ *such that, if* $\mathcal{X} \simeq_W \mathcal{Y}$, *then* $\mathbf{W}(\mathcal{X}) = \mathbf{W}(\mathcal{Y})$

*(see* [12]*).*

**Corollary 1.** *Inverse image functors of* $\mathbf{S}$ *and* $\mathbf{B}$, *(and of course* $\mathbf{S}^*$ *and* $\mathbf{B}^*$*), from* Tree-*Cat* |***Beh*** *to* $Tree_L$-*Cat* |***Beh***, *from* $Tree_L$-*Cat* |$\boldsymbol{Beh}_\tau$ *to* $Tree_L$-*Cat* |$\boldsymbol{Beh}_\tau$, *respectively, do have a right adjoint.*

This fact is an immediate consequence of the enriched form of Conduché property enjoyed by $\mathbf{S}$ and $\mathbf{B}$.

One could try to describe these right adjoints, but it seems more interesting to observe that, due to these adjunctions, $\mathbf{S}^*$ and $\mathbf{B}^*$ preserve both limits and colimits. These functors map a $Tree_L$-Cat over **Beh** or $\mathbf{Beh}_\tau$ into a similar category. These categories are possible models for concurrent systems, since their hom-objects are tree-like local behaviours. Preservation of limits and colimits in this framework means that universal constructions among models using only standard representatives w.r.t. strong and branching bisimulations can be transferred to models using non-standard local behaviours, still maintaining their properties. This is not the case for weak bisimulation.

Our *Tree*-functors $\mathbf{S}$, $\mathbf{B}$ and $\mathbf{W}$ are characterisable also as those ones which produce the "maximal quotient" among those which enjoy their properties.

## 5. Conclusions and related work

We have investigated the world of $Tree_L$-categories, where $\mathbf{L}$ is a meet-semilattice monoid, with particular attention to lifting factorisation properties for $Tree_L$-functors.

[5] By $\wedge$ we mean the maximal common prefix of the two chains.

Our method consisted of lifting constructions as far as possible through the different levels of our structures, namely: the lp 2-category **L**, the monoidal category $Tree_L$ of symmetric **L**-categories, and $Tree_L$-Cat. We applied the same method to the factorisation reflection property in order to extend Conduché's theorem to this enriched setting. In this context, we proved that the factorisation reflection property guarantees the existence of a right adjoint to the inverse image functor.

$Tree_L$-categories are particularly useful in modelling non-deterministic concurrent processes; in particular $\mathbf{Beh}_L$ is a powerful tool to represent operational semantics, in the case of both interleaving and true concurrent approaches.

As we saw before, $\mathbf{Beh}_L$ is a Conduché $Tree_L$-category where all $Tree_L$-endofunctors are Conduché or, better, UFL (see Proposition 11), because all the "states" of its objects are completely determined via their access path and the label. In this context the UFL property together with the path reflection completely characterises (strong) bisimulation.

When we move to $\mathbf{Beh}_\tau$ the "states" of its objects are no longer completely determined via their access path and the label. Hence, if we want to characterise some kind of bisimulation, we have to add a condition about preservation of states. This condition may be, as we saw, more or less cogent for **B** and **W**, respectively (see Propositions 14 and 15).

Lawvere's statement [19,1] about the preservation of states by functors enjoying the Conduché property in the physical control theory can be formally extended to Computer Science, because the latter corresponds to determinism in physical systems. In fact, Lawvere's remark deals with a property – that he named the *Moebius* property – stronger than Conduché's. We proved that this is the property enjoyed by the $Tree_L$-functors inducing strong equivalence, i.e, UFL. $Tree_L$-functors inducing weak equivalence enjoy only a weak form of the Conduché property, according to the fact that they partially forget about states. These two facts perfectly correspond to Milners' formulation of the problem [6]: strong bisimulation preserves determinacy, while weak bisimulation does not. On the other hand, we found the discriminant point in the branching bisimulation [12]: $Tree_L$-functors inducing such an equivalence still enjoy the Conduché property. Indeed, they preserve states, though in a non-deterministic context, because in the presence of a silent move $\tau$ mimicking non-determinism in the behaviours, $Tree_L$-functors keep track of the states, though forgetting the redundant ones.

Summing it all up, we can say that Lawvere's determinism is strictly related to unique factorisation lifting, while the weaker Milners determinacy is strictly related to the Conduché property.

Notice that, in the case of automata theory, states are completely neglected when trace (language) equivalence is imposed on their behaviour: the minimalisation theorem is a proof of this fact.

Let us now consider more closely how the Conduché property is also somewhat implicitly assumed in the construction of our models. A language $M$, i.e., a subset of $A^*$, is a sheaf over the space defined by its monoid $A^*$ as follows:

$$M : A^* \to Set\,.$$

It sends a word $w$ into the set of words $u \in M$ such that $w \leq u$ (definition of restrictions is immediate).

In the same way, a tree $\mathcal{X}$ could also be considered as a sheaf

$$\mathcal{X} : A^* \to Set$$

sending a word $w$ into the set of paths $x \in X$ such that $w \leq e(x)$, but in this way, every information about bifurcation points, i.e., states of our non-deterministic processes, would be lost. $\mathcal{X}$ would be simply considered as a multilanguage.

On the contrary, our definition of a tree amounts to say that it is a function

$$\mathcal{X} : A^* \to Set$$

sending a word $w$ into the set of paths $x \in X$ such that $w = e(x)$. In this way we get only a presentation of a sheaf over the space defined by its labelling monoid. Anyway, we are able to complete it by defining:

$$\mathcal{X} : A^* \to Set$$

as the map sending a word $w$ into the set of derivatives along the paths $x \in X$ after $w$, $\mathcal{D}(\mathcal{X}, x, s)$, such that $w \leq e(x)$. This corresponds to the fact that the labelling function $e$, i.e. the $Tree_L$-functor in the terminal $Tree_L$-category **L**, is UFL. Restrictions are defined by sending a derivative $\mathcal{D}(\mathcal{X}, x, s')$ to its ancestor $\mathcal{D}(\mathcal{X}, x, s)$ for $s \leq s'$.

In completing our presentations of sheaves we used the fact that in **L**-SymCat we can define derivatives, or, equivalently, states. This is due to the left-cancellation property that did reconstruct operational semantics as the internal structure. A single tree $\mathcal{X}$ with its operational semantics is a full subcategory of **Beh** together with a functor into **L** (its control) which assigns to every derivative of $\mathcal{X}$, $\mathcal{D}(\mathcal{X}, x, s)$ the unique object in the monoid **L** and, to a $w$-transition between the derivatives, the word $w$.

Analogously, trees with silent moves are presentations of sheaves with the Conduché property only. This corresponds to the fact that, while there is a UFL *Tree*-functor from **Beh** to **L**, there is only a Conduché *Tree*-functor from $\mathbf{Beh}_\tau$ to **L**. In this perspective, the fact that bisimulations can be characterised in terms of UFL *Tree*-functors and Conduché *Tree*-functors, respectively, seems particularly meaningful. In fact our result could be rephrased as: the "good" bisimulations preserve the intrinsic reflection properties of the models. From this point of view, weak bisimulation seems to be not so good.

The strong form of the Conduché property (UFL) has already been used by Bunge and Fiore [2] to define sheaf models for processes, exploiting Lawvere's point of view about its relationship with a determinacy on states. Bunge and Fiore considered processes as categories of states equipped with a control functor on a category of paths (a free monoid actually), over which the UFL property is imposed. Hence, the two approaches, though using different mathematical devices, can be easily translated into each other. On the other hand, the two approaches are also quite different in defining bisimulations and, in a sense, they are dual. Fiore et al. [20] define bisimulations in terms of spans of open maps [21]; moreover, they capture weak bisimulation by starting from the $\tau$-less case, i.e., by saturating their processes via the unit of a monad (that introduces as many $\tau$ s as possible) and by comparing the resulting models using open maps. Nonetheless, the path reflection condition on our *Tree*-functors on $\mathbf{Beh}_L$ also corresponds to an openness property for the maps when it is stated in the form (see Proposition 1 in [21]):

$$\text{if } \sigma(s) \xrightarrow{a} s' \text{ in } T' \text{ then } s \xrightarrow{a} u \text{ in } T \text{ and } \sigma(u) = s' \text{ for some } u \text{ of } T.$$

This can be rephrased in our context: if the image through **S** of an object $\mathcal{X}$ can perform an action along a path and go into another object $\mathcal{Y}$, than the original object can perform the same action and go into an object $\mathcal{Z}$ in the fibre of $\mathcal{Y}$. In the case of **Beh** every path reflective functor is also Conduché and UFL (see Remark 6).

On the other hand, differently from [20], we consider strong bisimulation as a quotient induced on the model by a *Tree*-functor enjoying UFL and construct the "minimal" strong representative for every process. Moving to the $\tau$-insensitive case, we make a change of base to obtain the model, where we induce branching and weak bisimulation via **Tree**-functors enjoying different reflecting factorisation properties. This catches the different degrees of nondeterminacy underlying such equivalences. Also, in this case we produce "minimal" representatives.

The use of the same categorical notions both for the construction of models and for defining equivalences, paves the way to fascinating directions for future work.

## Acknowledgments

We want to thank Anastasia Pagnoni, Andrea Montoli and Paolo Bottoni for their careful reading of the preliminary version of the manuscript.

## References

[1] F.W. Lawvere, Thermodynamics of deformations of continuous bodies, non homogeneous, with memory, far from equilibrium, Handwritten notes from a seminar held in Trieste, July 1982.
[2] M. Bunge, M.P. Fiore, Unique factorization lifting functors and categories of linearly-controlled processes, Math. Structures in Comput. Sci. 10 (2) (2000) 137–163.
[3] M.P. Fiore, Fibered models of processes: Discrete, continuous and hybrid systems, in: Proc. of IFIP TCS 2000, in: LNCS, vol. 1872, 2000, pp. 457–473.
[4] F. Conduché, Au sujet de l'existence d'adjoints à droîte aux foncteurs image reciproque dans la catégorie des catégories, C. R. Acad. Sci. Paris 275 (1972) A891–894.
[5] S. Kasangian, A. Labella, Observational trees as models for concurrency, Math. Structures Comput. Sci. 9 (1999) 687–718.
[6] R. Milner, Communication and Concurrency, Prentice Hall International, 1989.
[7] R. Walters, Sheaves and Cauchy-complete categories, Cah. Topol. Geom. Differ. 22 (1981) 283–286.
[8] R. Betti, S. Kasangian, A quasi-universal realization of automata, Rend. Ist. Mat. Univ. Trieste 14 (1982) 41–48.
[9] S. Eilenberg, G. Kelly, Closed Categories, in: Proc. of the Conference on Categorical Algebra, La Jolla 1965, Springer, 1966, pp. 421–562.
[10] R. Betti, A. Carboni, R. Street, R. Walters, Variation through enrichment, J. Pure Appl. Algebra 29 (1983) 109–127.
[11] S. Kasangian, A. Labella, On continuous time agents, in: Mathematical Foundations of Programming Semantics, in: LNCS, vol. 598, Springer, Berlin, 1992, pp. 403–425.
[12] R. De Nicola, D. Gorla, A. Labella, Tree-Functors, determinacy and bisimulations, Technical Report, 02/2006, Dip. di Informatica, Univ. di Roma "La Sapienza" (Italy), 2008 (submitted for publication), http://www.dsi.uniroma1.it/%7Egorla/papers/DGL-TR0206.pdf.
[13] D. Park, Concurrency and automata on infinite sequences, in: Proc. of Theoret. Comput. Sci., in: LNCS, vol. 104, Springer, 1981, pp. 167–183.
[14] S. Kasangian, A. Labella, Enriched categorical semantics for distributed calculi, J. Pure Appl. Math. 83 (1992) 295–321.
[15] G. Kelly, Basic Concepts of Enriched Category Theory, Cambridge University Press, 1982.
[16] A. Mazurkiewicz, Trace theory, in: LNCS, vol. 255, Springer, 1987, pp. 279–324.
[17] A. Labella, Categories with sums and right distributive tensor product, J. Pure Appl. Algebra 178 (3) (2003) 273–296.
[18] R. van Glabbeek, W. Weijland, Branching time and abstraction in bisimulation semantics, J. ACM 43 (3) (1996) 555–600.
[19] F.W. Lawvere, State categories and response functors, Unpublished manuscript, 1986.
[20] M. Fiore, G.L. Cattani, G. Winskel, Weak Bisimulation and Open Maps, in: Proc. of LICS, IEEE, 1999, pp. 67–76.
[21] A. Joyal, M. Nielsen, G. Winskel, Bisimulations and open maps, in: Proc. of LICS, IEEE, 1993, pp. 418–427.