Electronic Notes in Theoretical Computer Science 18 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume18.html 19 pages

Tree Morphisms and Bisimulations

Rocco De Nicola

Dipartimento di Sistemi e Informatica, Università di Firenze
e-mail: denicola@dsi.unifi.it

Anna Labella

Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza"
e-mail: labella@dsi.uniroma1.it

Abstract

A category of (action labelled) trees is defined that can be used to model unfolding of labelled transition systems and to study behavioural relations over them. In this paper we study five different equivalences based on bisimulation for our model. One, that we called *resource bisimulation*, amounts essentially to three isomorphism. Another, its weak counterpart, permits abstracting from silent actions while preserving the tree structure. The other three are the well known strong, branching and weak bisimulation equivalence. For all bisimulations, but weak, canonical representatives are constructed and it is shown that they can be obtained via enriched functors over our categories of trees, with and without silent actions. Weak equivalence is more problematic; a canonical minimal representative for it cannot be defined by quotienting our trees. The common framework helps in understanding the relationships between the various equivalences and the results provide support to the claim that branching bisimulation is the natural generalization of strong bisimulation to systems with silent moves and that resource and weak resource have an interest of their own.

1 Introduction

Behavioural equivalences play an important rôle in the description of the operational semantics of concurrent systems. These equivalences are used to abstract away the irrelevant details introduced when describing systems as sets of states that evolve by performing actions, i.e. by means of labelled transition systems. There are various opinions about which features of a system are relevant for a given purpose, and hence various notions of equivalence for labelled transition systems have been proposed; [3] and [8,9] give a comparative accounts.

Many of the equivalences proposed in the literature are based on the notion of *bisimulation* [15] which gives rise to *strong bisimulation equivalence*. Two of the most popular generalizations of this equivalence to systems with silent moves are *branching* [10] and *weak* bisimulation [14]. Both of these equivalences ignore τ (internal or silent) actions, but they deal differently with intermediate states accessed by τ transitions and lead to different identifications, putting a different stress on the branching structure of processes. While strong bisimulation is generally regarded as the equivalence which provides the minimum abstraction from the details of behaviour for τ–free transitions systems, there is little agreement about the comparative merits of the weak and branching generalizations of strong equivalence to systems with τ actions.

In our view, category theory and their abstract constructions can be a useful tool for understanding and assessing the relative merits of different concepts. Here, we consider those categories of trees that have been used to model concurrent systems [12] and nondeterministic regular expressions [6,2]. Our aim is that of reconducting the different bisimulations to a common framework where it is easier to understand their relationships.

We shall study five different bisimulation-based equivalences for our trees. The more concrete one, that we called *resource bisimulation* [6], corresponds to tree isomorphism; it is finer than strong bisimulation in that it discriminates also according to the number of computations that can be performed to reach specific states. It will be our touchstone and will guide toward defining and assessing the other equivalences. Indeed, the second one is its weak counterpart that permits abstracting from silent action while preserving the tree structure. The other three equivalences are the well known strong, branching and weak bisimulation equivalence [15,10,14]. We shall see that branching bisimulation can be obtained by mirroring the construction for weak resource bisimulation while replacing isomorphism requirements with requirements of strong bisimilarity. For resource, strong, weak resource and branching bisimulation equivalence we shall define standard representatives of their equivalence classes. These constructions are then vindicated by the enriched categorical account that we provide in the final part of the paper. We argue that similar results cannot be obtained for weak bisimulation equivalence.

We start from a basic category of trees labelled over an actions monoid, Tree, and construct a category Der with the same objects and where the maps are paths to derivatives (so a map $f : t \to t'$ tells us how to find a copy of t' in t). Invisible actions (τs) are introduced by admitting them as labels; this generalizes Tree to Tree_τ and Der to Der_τ. To relate the two categories a functor $del\colon \text{Tree}_\tau \to \text{Tree}$ is defined which deletes τ–labelled branches.

We show that resource equivalence does not introduce any quotienting on Der and corresponds to the identity functor. After that, we introduce $F_S : Der \to Der$ which functorially map a tree to the canonical representative of its strong bisimulation equivalence class. Finally, we define F_{WR} and F_B : $Der_\tau \to Der_\tau$ which functorially maps a tree to the canonical representative of its weak resource and branching bisimulation equivalence class.

In our view, these results strengthen the claim that branching bisimulation is the natural generalization of strong bisimulation to systems with silent moves and that a suitable notion of tree is fundamental in dealing with bisimulations.

2 Trees and Transitions Systems

We begin by introducing the basic concepts of labelled transition systems, their unfoldings, and the five notions of bisimulation we will study. We then present trees as a structure of *runs* with *agreements* and the relationship with unfoldings.

We begin with standard definitions about transition systems. We suppose that a set A of actions is given, together with a distinguished action $\tau \notin A$ representing a silent move. Unless otherwise stated, we confine attention to reachable transition systems with finite unfoldings.

We write A_τ for $A \cup \{\tau\}$ and A^* for the monoid of words on A with empty word ϵ. The variables a, b etc. will range over A, and μ, ν etc. over A_τ. Words will be w, v etc.

Definition 2.1 *A (rooted)* labelled transition system *or LTS is a quadruple $\mathcal{S} = (S, E, \rightarrow, s_0)$ where S is a set of* states, *ranged over by s, u etc.; E is a set of* actions, *$E \subseteq A_\tau$; $\rightarrow \subseteq S \times E \times S$ is a relation, the* transition relation*; $s_0 \in S$ is a distinguished* starting state.

We usually write $s \xrightarrow{\mu} s'$ rather than $(s, \mu, s') \in \rightarrow$, and if s, u, s' and u' are states in S we write

i) $\overset{\epsilon}{\Longrightarrow}$ for the reflexive and transitive closure of $\xrightarrow{\tau}$;

ii) $s \overset{\mu}{\Longrightarrow} u$ if there exist s', u' such that $s \overset{\epsilon}{\Longrightarrow} s' \xrightarrow{\mu} u' \overset{\epsilon}{\Longrightarrow} u$;

iii) $s \overset{\mu}{\not\longrightarrow}$ if there exists no $s' \in S$ such that $s \xrightarrow{\mu} s'$;

iv) $s \not\longrightarrow$ if $s \overset{\mu}{\not\longrightarrow}$ for all $\mu \in A_\tau$.

We now introduce our five bisimulations, four of them are relatively well known, the other, weak resource bisimulation, is new and has been defined in collaboration with Flavio Corradini.

Definition 2.2 *Let $\mathcal{S} = (S, E, \rightarrow, s_0)$ be an LTS. A symmetric relation $R \subseteq S \times S$ is said to be*

i) a resource bisimulation *if $s\ R\ u$, implies that there exists a bijection f between $\{s' | s \xrightarrow{\mu} s'\}$ and $\{u' | u \xrightarrow{\mu} u'\}$ such that $s'\ R\ f(s')$;*

ii) a strong bisimulation *if $s\ R\ u$ and $s \xrightarrow{\mu} s'$ implies that $u \xrightarrow{\mu} u'$ and $s'\ R\ u'$;*

iii) a weak resource bisimulation *if for all $\mu \in A_\tau$, if $s\ R\ u$, then there exists* *a bijection f between* $\{s' | s \overset{\mu}{\Longrightarrow} s'\} - \{s' | s' \overset{\tau}{\Longrightarrow} s'', s' \; R \; s''\}$ *and* $\{u' | u \overset{\mu}{\Longrightarrow} u'\} - \{u' | u' \overset{\tau}{\Longrightarrow} u'', u' \; R \; u''\}$ *such that* $s' \; R \; f(s')$*;*

iv) *a* branching bisimulation *if* $s \; R \; u$ *and* $s \xrightarrow{\mu} s'$ *implies that either* $\mu = \tau$ *and* $s' \; R \; u$*. or* $u \overset{\epsilon}{\Longrightarrow} u_1 \xrightarrow{\mu} u_2 \overset{\epsilon}{\Longrightarrow} u_3$ *and* $s \; R \; u_1$*,* $s' \; R \; u_2$*,* $s' \; R \; u_3$*.*

v) *a* weak bisimulation *if* $s \; R \; u$ *and* $s \overset{a}{\Longrightarrow} s'$ *implies that* $u \overset{a}{\Longrightarrow} u'$ *with* $s' \; R \; u'$*.*

Two states are said to be resource*,* strong*,* weakly resource*,* branching *or* weak bisimilar *if there exists an eponymous bisimulation relating them. We write* $\approx_R$*,* $\approx_S$*,* $\approx_{WR}$*,* $\approx_B$*, and* $\approx_W$ *for* resource*,* strong*,* weak resource*,* branching*, and* weak *bisimulation equivalence, respectively.*

It is not difficult to see that, in presence of τ-actions, the last three relations are increasingly coarser. When all actions are visible, we have instead that *resource* and *weak resource* on one hand, and *strong*, *branching* and *weak* bisimulation, on the other, do collapse.

We now introduce a category of labelled trees, Tree and some of its properties. A single tree will be modelled by specifying what runs it has, what computations are performed along each run (its *extent*), and to what extent those computations agree (the *agreement*). Thus the tree

$$\begin{tikzcd}
& {} \arrow[dl, "a"'] \arrow[dr, "a"] & \\
{} \arrow[d, "c"] & & {} \arrow[d, "b"'] \\
y & & x
\end{tikzcd}$$

will be modelled by runs, x and y, labelled with ab and ac respectively, and by stating that x and y do not agree at all. In contrast, the tree

$$\begin{tikzcd}
& {} \arrow[d, "a"] & \\
& {} \arrow[dl, "c"'] \arrow[dr, "b"] & \\
y & & x
\end{tikzcd}$$

will be modelled by giving two runs, x and y, again labelled with ab and ac, but with agreement between x and y being the initial a.

Definition 2.3 *Let* $\mathcal{A} = (A^*, \leq, \wedge, \epsilon)$ *be the meet semilattice where* $\leq$ *is the prefix order of words;* $\wedge$ *is the largest common prefix operation on words; and* ϵ *is the minimum.*

Definition 2.4 *A tree* $\mathcal{X} = (X, \varepsilon, \alpha)$ *comprises:*

- *a set* X *of* runs*;*
- *a map* $\varepsilon : X \to A^*$*, the* extent map*, giving the computation* $\varepsilon(x)$ *along a run* x*;*
- *a map* $\alpha : X \times X \to A^*$*, establishing the* agreement *between pairs of computations.*

Additionally, we require that for all $x, y, z \in X$

- $\alpha(x,x) = \varepsilon(x)$
- $\alpha(x,y) \leq \varepsilon(x) \wedge \varepsilon(y)$
- $\alpha(x,y) \wedge \alpha(y,z) \leq \alpha(x,z)$
- $\alpha(x,y) = \alpha(y,x)$

These amount to requiring that a run agrees with itself along all its length; the agreement between two runs is not bigger than their largest common prefix (runs are forced to agree on a common initial segment and they cannot join up again once split); the common agreement between x, y and z is not bigger than that between x and z; agreement is symmetrical.

We will write $\mathcal{X}$, $\mathcal{Y}$, etc. for typical trees with components $\mathcal{X} = (X, \varepsilon, \alpha)$, $\mathcal{Y} = (Y, \zeta, \beta)$. We shall use $w' - w$ to denote the word obtained from w' by deleting the prefix w from w'.

Example 2.5 *The two trees illustrated above are specified by (X, ε, α) where $X = \{x, y\}$, and $\varepsilon(x) = ab$, $\varepsilon(y) = ac$, $\alpha(x,y) = \epsilon$ and (Y, ζ, β) where $Y = \{x, y\}$, and $\zeta(x) = ab$, $\zeta(y) = ac$, $\beta(x,y) = a$, respectively.*

We can observe that A-labelled trees are symmetric $\mathcal{A}$–categories, when $\mathcal{A}$ is thought of as a posetal 2-category [16]. Therefore the appropriate notion of comparison for trees is that of $\mathcal{A}$– functor, i.e.:

Definition 2.6 *A* tree morphism *$f : \mathcal{X} \to \mathcal{Y}$ is a map $f : X \to Y$ satisfying*

i) f does not change the extent: $\zeta(f(x)) = \varepsilon(x)$;

ii) f increases the agreement: $\beta(f(x), f(y)) \geq \alpha(x,y)$.

Tree will be the category of finite A–labelled trees. For A_τ–labelled trees, we use the semilattice $\mathcal{A}_\tau = (A_\tau^*, \leq, \wedge, \epsilon)$ exactly as before, and hence obtain the category Tree$_\tau$ with extent and agreement maps valued in $\mathcal{A}_\tau$.

Definition 2.7 *Given two trees, $\mathcal{X}$ and $\mathcal{Y}$, we can form the sequential composition of $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{X} \otimes \mathcal{Y} = (Z, \eta, \gamma)$ as follows:*

i) $Z = X \times Y$ (a run in $\mathcal{X} \otimes \mathcal{Y}$ is a run in $\mathcal{X}$ followed by a run in $\mathcal{Y}$);

ii) $\eta(x,y) = \varepsilon(x).\zeta(y)$, where . is concatenation of strings (so the label of a run in $\mathcal{X} \otimes \mathcal{Y}$ is the label in $\mathcal{X}$ followed by the label in $\mathcal{Y}$);

iii) $\gamma((x,y),(x',y'))$ is $\alpha(x,x')$ if $x \neq x'$ and $\varepsilon(x)\beta(y,y')$ otherwise (so runs that are different in $\mathcal{X}$ have their $\mathcal{X}$ agreement, while runs that differ only in $\mathcal{Y}$ have their $\mathcal{X}$ agreement concatenated with their $\mathcal{Y}$ agreement).

Proposition 2.8 *Sequential composition defines the object part of an associative tensor product on Tree with unit $1 = (\{\bullet\}, \varepsilon(\bullet) = \epsilon, \alpha(\bullet, \bullet) = \epsilon)$. Tree has an initial object given by the empty tree, $0 = (\emptyset, \emptyset, \emptyset)$, and finite coproducts given by joining two trees at the root.*

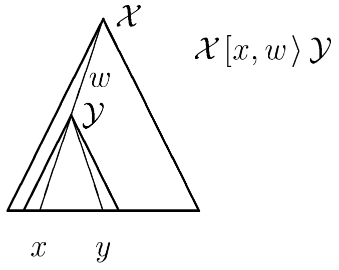
Intuitively, it is clear that the trees introduced in the last sections can be used to represent the unfoldings of finite transition systems. Here, we formally establish the correspondence and use it to motivate our equivalences and lift the bisimulations to trees.

Definition 2.9 *Let $\mathcal{X} = (X, \varepsilon, \alpha)$ be a tree. A* prefix of a run *in $\mathcal{X}$ is a pair (x, w) consisting of a run $x \in X$ and a word $w \in A^*$ with $w \leq \varepsilon(x)$.*

A path *in $\mathcal{X}$ is an equivalence class $[x, w]$ of prefixes quotiented by*
$(x, w) \equiv (y, v) \qquad$ *iff* $\qquad w = v \quad \leq \quad \alpha(x, y)$*;*

The derivative *reached along x after w, for a path $[x, w]$ in $\mathcal{X}$, is the tree (Y, ζ, β) where $Y = \{x' | x' \in X, \alpha(x, x') \geq w\}$; $\zeta(x') = \varepsilon(x') - w$; $\beta(x', x'') = \alpha(x', x'') - w$.*

The figure below illustrates the terminology.



We will write paths($\mathcal{X}$) for the paths of a tree $\mathcal{X}$ and $\mathcal{X}\,[x, w\rangle\,\mathcal{Y}$ if $\mathcal{Y}$ is the derivative reached along x after w in $\mathcal{X}$ We will write $\mathcal{X}\,[x, v\rangle$ for the unique tree $\mathcal{Y}$ such that $\mathcal{X}\,[x, v\rangle\,\mathcal{Y}$.

Notice that in the definition above, we did not mention nodes explicitly: they are in an obvious bijective correspondence with paths. Sometimes, we will refer to paths as nodes.

Given a transition system $\mathcal{S}$ with a finite unfolding, we now construct a tree unf($\mathcal{S}$) representing that unfolding.

Definition 2.10 *The* unfolding *unf($\mathcal{S}$) $=$ (runs($\mathcal{S}$), $\varepsilon_{\mathcal{S}}, \alpha_{\mathcal{S}}$) of a transition system $\mathcal{S} = (S, E, \rightarrow, s_0)$ is a tree given by*

i) *runs($\mathcal{S}$) $= \{s_0\mu_1 s_1 \mu_2 \ldots \mu_n s_n \mid s_0 \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} \ldots \xrightarrow{\mu_n} s_n \not\rightarrow\}$;*

ii) *$\varepsilon_{\mathcal{S}}(s_0\mu_1 s_1 \ldots \mu_n s_n) = \mu_1 \ldots \mu_n$;*

iii) *$\alpha_{\mathcal{S}}(s_0\mu_1 s_1 \ldots \mu_n s_n, s_0\nu_1 u_1 \ldots \nu_m u_m) = \mu_1 \ldots \mu_l$ where for all $k \leq l$, $\nu_k = \mu_k$ and $s_k = u_k$, and $\mu_{l+1} \neq \nu_{l+1}$ or $s_{l+1} \neq u_{l+1}$.*

It is not difficult to see that, if we use unfold($\mathcal{S}$) to indicate the standard unfolding of a transition system $\mathcal{S}$ and use tran($\mathcal{X}$) to refer to the transition system associated to a tree $\mathcal{X}$ defined as follows:

$$\text{tran}(\mathcal{X}) = (\text{paths}(\mathcal{X}), \text{im}(\varepsilon), \rightarrow, [x, \epsilon])$$

where $\text{im}(\varepsilon) = \{\mu \mid \mu \text{ appears somewhere in } \varepsilon(x), x \in X\}$ and $[x, w] \xrightarrow{\mu} [y, v]$ if and only if $(y, v) \in [x, w\mu]$, then we have: unfold($\mathcal{S}$) $\cong$ tran(unf($\mathcal{S}$)).

3 Bisimulation for Trees

We shall work directly on trees and provide a concrete definition of resource and strong equivalence directly over them. We will also consider weak resource equivalence, branching and weak equivalence. We will however prove that our definitions are in full agreement with the corresponding ones introduced in the previous section for labelled transition systems.

Definition 3.1 *Two trees, $\mathcal{X}$ and $\mathcal{Y}$, are resource bisimilar, written $\mathcal{X} \approx_R \mathcal{Y}$ if and only if there exists a bijective function $f : X \longrightarrow Y$ such that $\varepsilon(x) = \zeta(f(x))$ and $\forall w \leq \varepsilon(x)$, with $w \neq \epsilon$, $\mathcal{X}[x, w\rangle \approx_R \mathcal{Y}[f(x), w\rangle$.*

Proposition 3.2 *Two (finite) trees are resource bisimilar if and only if they are isomorphic.*

Proposition 3.3 *Two transition systems with finite unfoldings are resource bisimilar, $\mathcal{S} \approx_R \mathcal{S}'$, if and only if there is a resource bisimulation between their unfoldings as trees, i.e., if and only if unf$(\mathcal{S}) \approx_R$ unf$(\mathcal{S}')$.*

Definition 3.4 *Two trees, $\mathcal{X}$ and $\mathcal{Y}$, are strongly bisimilar, written $\mathcal{X} \approx_S \mathcal{Y}$ if and only if*

i) $\forall x \in X\, \exists y \in Y$: $\varepsilon(x) = \zeta(y)$ and for all $w \leq \varepsilon(x)$, with $w \neq \epsilon$, $\mathcal{X}[x, w\rangle \approx_S \mathcal{Y}[y, w\rangle$.

ii) $\forall y \in Y\, \exists x \in X$: $\varepsilon(x) = \zeta(y)$ and for all $w \leq \zeta(y)$, with $w \neq \epsilon$, $\mathcal{Y}[y, w\rangle \approx_S \mathcal{X}[x, w\rangle$.

Proposition 3.5 *Two transition systems with finite unfoldings are strongly bisimilar, $\mathcal{S} \approx_S \mathcal{S}'$, if and only if there is a strong bisimulation between their unfoldings as trees, i.e., if and only if unf$(\mathcal{S}) \approx_S$ unf$(\mathcal{S}')$.*

We introduce a function, *del*, which deletes τs, and transforms a tree with τ moves in Tree$_\tau$ into a tree in Tree obtained by ignoring all τ moves. Below, we overload notation, and call *del* the obvious deletion on words, $del(\epsilon) = \epsilon$ and $del(\mu w)$ as $\mu\, del(w)$ if $\mu \neq \tau$ and $del(w)$ otherwise. Please notice that images of glued runs in $del(\mathcal{X})$ are glued in $del(\mathcal{Y})$.

Definition 3.6 *Function del : Tree$_\tau$ $\rightarrow$ Tree is defined as $del(X, \varepsilon, \alpha) = (Y, \zeta, \beta)$ where $Y = X$; $\zeta(x) = del(\varepsilon(x))$; $\beta(x, y) = del(\alpha(x, y))$.*

It can be immediately seen that del extends to a functor del : Tree$_\tau$ $\rightarrow$ Tree; indeed morphisms from $\mathcal{X}$ to $\mathcal{Y}$ induce morphisms from $del(\mathcal{X})$ to $del(\mathcal{Y})$.

Once we ignore τs, a derivative is not uniquely determined by its access path any longer. To see this, examine Figure 1: the same run, x, leads to both the derivative $t + \tau t'$ and t' along $del(w)$ or $del(w\tau)$. Thus, in general, $(x, del(w))$ specifies the path to a set of derivatives.

It is important to notice that there is always a largest tree ($t + \tau t'$ here) to which $[x, del(w)]$ leads, and any other tree so accessed (like t') is a τ–summand of this one.

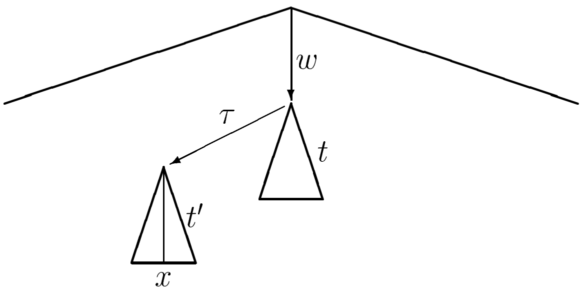


Fig. 1. Derivatives accessed along $[x, del(w)]$ in the presence of τs.

Definition 3.7 *Given a tree $\mathcal{X} = (X, \varepsilon, \alpha)$ in $Tree_\tau$, a run $x \in X$, and a prefix $v \leq del(\varepsilon(x))$, let $w \in A_\tau^*$ be the shortest word such that $w \leq \varepsilon(x)$ and $del(w) = v$, $\mathcal{X}\,[x, w\rangle\,\mathcal{Y}$.*

(i) $\mathcal{R}_\tau(\mathcal{X}, x, v) \; = \; \{\mathcal{Z} \mid \mathcal{X}\,[x, w\rangle\,\mathcal{Y}\;[x, \tau^n\rangle\,\mathcal{Z}, n \geq 0\}$,
the family of derivatives reachable along x by v.

(ii) $\mathcal{R}(\mathcal{X}, x, v) \; = \; \{del(\mathcal{Z}) \mid \mathcal{X}\,[x, w\rangle\,\mathcal{Y}\;[x, \tau^n\rangle\,\mathcal{Z}, n \geq 0\}$,
the family of derivatives reachable along x by v but pruned using del.

Within this setting, we write $\mathcal{Z} \leq_+ \mathcal{Y}$ if $\mathcal{Z}$ is a τ–summand of $\mathcal{Y}$, and note that $\mathcal{R}(\mathcal{X}, x, v)$ is linearly ordered by the relation induced by $\leq_+$, that with abuse of notation we will write $\leq_+$ as well.

Definition 3.8 *Two trees $\mathcal{X}$ and $\mathcal{Y}$ are weak resource bisimilar, written $\mathcal{X} \approx_{WR} \mathcal{Y}$, if and only if there exists a bijection $f : X \longrightarrow Y$, such that $del(\varepsilon(x)) = del(\zeta(f(x)))$ and*

a) For all $v = del(w)$, $w \leq \varepsilon(x)$ and $w \neq \epsilon$, if $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ then there exists $t' \in \mathcal{R}_\tau(\mathcal{Y}, f(x), v)$ and $t \approx_{WR} t'$;

b) for all $v = del(w)$, $w \leq \varepsilon(y)$ and $w \neq \epsilon$, if $t \in \mathcal{R}_\tau(\mathcal{X}, y, v)$ then there exists $t' \in \mathcal{R}_\tau(\mathcal{Y}, f(y), v)$ and $t \approx_{WR} t'$.

Proposition 3.9 *Two transition systems are weak resource bisimilar, $\mathcal{S} \approx_{WR} \mathcal{S}'$, if and only if there is a weak resource bisimulation between their unfoldings as trees, i.e., if and only if $unf(\mathcal{S}) \approx_{WR} unf(\mathcal{S}')$.*

Definition 3.10 *Two trees $\mathcal{X}$ and $\mathcal{Y}$ are branching bisimilar, written $\mathcal{X} \approx_B \mathcal{Y}$,if and only if*

i. $\forall x \in X\; \exists y \in Y$ such that $del(\varepsilon(x)) = del(\zeta(y))$ and

a) for all $v = del(w)$, with $w \leq \varepsilon(x)$ and $w \neq \epsilon$, if $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ then there exists $t' \in \mathcal{R}_\tau(\mathcal{Y}, y, v)$ such that $t \approx_B t'$;

b) for all $v = del(w)$, with $w \leq \zeta(y)$ and $w \neq \epsilon$, if $t' \in \mathcal{R}_\tau(\mathcal{Y}, y, v)$ then there exists $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ such that $t \approx_B t'$.

ii. $\forall y \in Y\; \exists x \in X$ such that $del(\varepsilon(x)) = del(\zeta(y))$ and

a) for all $v = del(w)$, with $w \leq \zeta(y)$ and $w \neq \epsilon$, if $t' \in \mathcal{R}_\tau(\mathcal{Y}, y, v)$ then there exists $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ such that $t \approx_B t'$;

b) for all $v = del(w)$, with $w \leq \varepsilon(x)$ and $w \neq \epsilon$, if $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ then there exists $t' \in \mathcal{R}_\tau(\mathcal{Y}, y, v)$ such that $t \approx_B t'$.

Proposition 3.11 *Two transition systems are branching bisimilar, $\mathcal{S} \approx_B \mathcal{S}'$, if and only if there is a branching bisimulation between their unfoldings as trees, i.e., if and only if unf($\mathcal{S}$) $\approx_B$ unf($\mathcal{S}'$).*

Definition 3.12 *Two trees $\mathcal{X}$ and $\mathcal{Y}$ are* weakly bisimilar, *written $\mathcal{X} \approx_W \mathcal{Y}$,if and only if*

i. $\forall x \in X\ \exists y \in Y$ such that $del(\varepsilon(x)) = del(\zeta(y))$ and

a) for all $v = del(w)$, with $w \leq \varepsilon(x)$ and $w \neq \epsilon$, if $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ then there exists $t' \in \mathcal{R}_\tau(\mathcal{Y}, y, v)$ such that $t \approx_W t'$.

ii. $\forall y \in Y\ \exists x \in X$ such that $del(\varepsilon(x)) = del(\zeta(y))$ and

a) for all $v = del(w)$, with $w \leq \zeta(y)$ and $w \neq \epsilon$, if $t' \in \mathcal{R}_\tau(\mathcal{Y}, y, v)$ then there exists $t \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ such that $t \approx_W t'$.

Proposition 3.13 *Two transition systems are weakly bisimilar, $\mathcal{S} \approx_W \mathcal{S}'$, if and only if there is a weak bisimulation between their unfoldings as trees, i.e., if and only if unf($\mathcal{S}$) $\approx_W$ unf($\mathcal{S}'$).*

It is interesting to note the essential difference between definitions 3.10 and 3.12 is one of symmetry: definition 3.12 is missing cases i.b and ii.b of definition 3.10. This will turn out to have important consequences; the symmetrical form of definition 3.10 means that we can define a branching bisimulation as an equivalence relation between runs (in the style of the 'back–and–forth' approach [7]), whereas no such definition will be possible for weak bisimulation.

4 Canonical Representatives for Bisimulation

In the previous section, we rephrased the definition of weak and branching bisimulation by only relying on τ–less structures. The information about silent step transitions is collected in what we called the family of derivatives reached along run x via label w, $\mathcal{R}(\mathcal{X}, x, w)$. In this section we will exploit this intuition, and the construction of standard representatives for strong equivalence classes of trees, to build standard representatives for branching equivalence classes of trees. We will show that, given a rigid (τ–less) tree whose nodes are labelled by $\mathcal{R}$–sets, it is possible to obtain a (non–rigid) "minimal" tree that is branching equivalent to the original one. The same procedure will be used for weak resource equivalence, taking into account that, in that case, the corresponding "rigid" equivalence is isomorphism. We remind the reader that we use $\cong$ to denote tree isomorphism.

We begin by characterizing strong bisimulation via a canonical representative. The canonical representative for the $\approx_S$–equivalence class will be obtained by merging those runs that have the same extent and equivalent relationships with other runs in the same tree.

Definition 4.1 *Let $\equiv$ be the equivalence relation on runs defined by $x \equiv x'$ if and only if $\varepsilon(x) = \varepsilon(x')$ and for every $v \leq \varepsilon(x)$, $\mathcal{X}[x, v\rangle \approx_S \mathcal{X}[x', v\rangle$, and let $|x|$ denote the $\equiv$–equivalence class of x. The* canonical S–reduction *of a tree $\mathcal{X} = (X, \varepsilon, \alpha)$, $S\mathcal{X}$, is the tree (Y, ζ, β) where*

i) $Y = \{|x| \mid x \in X\}$;

ii) $\zeta(|x|) = \varepsilon(x)$;

iii) $\beta(|x|, |y|) = \max\{\alpha(x', y') \mid x' \in |x|, y' \in |y|\}$.

It is worth noticing that the maximum above exists. Indeed, $x \equiv x'$ implies $\varepsilon(x) = \varepsilon(x')$; hence $\alpha(x', y)$ for $x' \in |x|$ is always a prefix of $\varepsilon(x)$. Thus $\alpha(x', y)$, as x' varies, is linearly ordered.

We approach the proof that S$\mathcal{X}$ is the canonical representative of the $\approx_S$–equivalence class of $\mathcal{X}$ (and hence that strong equivalence coincides with $\approx_S$) by showing that $\mathcal{X}$ and S$\mathcal{X}$ have the same transitions:

Lemma 4.2 *$\mathcal{X}$ and $S\mathcal{X}$ can perform the same labelled transitions:*

i) if $\mathcal{X}\,[x, w\rangle\, \mathcal{Y}$ *then* $(S\mathcal{X})\,[|x|, w\rangle\,(S\mathcal{Y})$;

ii) if $(S\mathcal{X})\,[|x|, w\rangle\, \mathcal{Z}$ *and* $\mathcal{X}\,[x, w\rangle\, \mathcal{Y}$, *then* $S\mathcal{Y} \cong \mathcal{Z}$.

Theorem 4.3 *Given trees, $\mathcal{X}$ and $\mathcal{Y}$, we have $S\mathcal{X} \cong S\mathcal{Y}$ if and only if $\mathcal{X} \approx_S \mathcal{Y}$.*

In order to define a canonical representative for equivalences involving silent moves, we need a procedure of reconstruction of a non rigid tree starting from data given in terms of rigid trees. In the appendix we report an example reconstruction, here we provide a general procedure that given a collection of (x, v)–indexed families of trees, under some conditions on the collection, yields a reconstructed tree.

Definition 4.4 *Fix a tree $\mathcal{X}$ in Tree and suppose that a finite collection $\mathcal{R}(x, v)$ of sets of trees is given, one set for each pair (x, v), $x \in X$, $v \leq \varepsilon(x)$ We will call this collection $\mathcal{X}$–*reconstructible *if it satisfies the following conditions:*

i) $(\mathcal{R}(x, v), \leq_+)$ *is a finite chain,* $R(x, v)_i \leq_+ R(x, v)_{i-1} \leq_+ \ldots \leq_+ R(x, v)_1$, *in Tree;*

ii) There exists a surjective morphism (epimorphism) $f_{x,v}$ from $\mathcal{X}\,[x, v\rangle$ to the maximal element in the chain $R(x, v)_1$ and for every i, $f_{x,v}(x) \in R(x, v)_i$;

iii) If $f_{x,v}(y) \in R(x, v)_i$ then $\mathcal{R}(x, s) = \mathcal{R}(y, s)$ for all $s < v$, and, if $s = v$, for all $j \leq i$ $R(x, s)_j = R(y, s)_j$, furthermore, for all $s \leq v$, $f_{x,s}$ and $f_{y,s}$ coincide on common domains.

Condition i) means that the derivative after v along x is considered as standing for i different states that are bigger and bigger, but ii) guarantees that the biggest of such states is covered by the original one; iii) deals with coherence of the derivatives associated with each run.

The properties of reconstructible families are sufficient to ensure a well–behaved reconstruction.

Definition 4.5 *Given a tree $\mathcal{X}$ in Tree, consider a $\mathcal{X}$–reconstructible family $\mathcal{R}(x,v)$, the* reconstruction $\int \mathcal{X}, \mathcal{R}(x,v) = (\mathcal{X}', \varepsilon', \alpha')$ *of $\mathcal{X}$ is given by:*

i) $\mathcal{X}' = \mathcal{X}$;

ii) $\varepsilon'(x) = \tau^{i_1}a_1\tau^{i_2}a_2 \dots \tau^{i_n}a_n\tau^{i_{n+1}}$, *given* $\varepsilon(x) = a_1a_2\dots a_n$ *and* $i_k = |\mathcal{R}(x, a_1 \dots a_{k-1})| - 1$ *for* $1 \le k \le n+1$;

iii) $\alpha'(x,y) = \tau^{i_1}a_1\tau^{i_2}a_2 \dots \tau^{i_m}a_m\tau^{i_{m+1}}$ *where* $\alpha(x,y) = a_1a_2\dots a_m$, *for* $1 \le k \le m$, $i_k = |\mathcal{R}(x, a_1 \dots a_{k-1})| - 1$ *and* $i_{m+1} = |\mathcal{R}(\mathcal{X}, x, a_1a_2\dots a_m) \cap \mathcal{R}(y, a_1a_2\dots a_m)| - 1$.

Proposition 4.6 *Let $\int \mathcal{X}, \mathcal{R}(x,v)$ be as in definition 4.5, then it is a tree in Tree$_\tau$.*

Lemma 4.7 *Given a tree $\mathcal{X}$ in Tree, let w be a word different from τ. There is an epimorphism between del$(\int \mathcal{X}, \mathcal{R}(x,v)\left[x, w\tau^h\right\rangle)$ and $R(x, del(w))_{h+1}$.*

Proposition 4.8

i) *There is a bijection between $\mathcal{R}_\tau(\int \mathcal{X}, \mathcal{R}(x,v))$ and $\mathcal{R}(x,v)$.*

ii) $\int(\mathcal{X}[x', w\rangle), \mathcal{R}(x,v)) = (\int \mathcal{X}, \mathcal{R}(x,v))\,[x', w\rangle$.

Let us now consider reconstruction in the cases of interest for us. We will start with a tree in Tree$_\tau$, we apply deletion on it and reconstruct it with families obtained from the original tree. Two kinds of families will be considered, to obtain trees that are weakly resource and branching bisimilar to the original tree. As in the general case (Definition 4.5), reconstruction will be carried run by run and two trees will be identified only if they are isomorphic and reachable along the same run.

Theorem 4.9 *Given a tree $\mathcal{X}$ in Tree$_\tau$, let us consider two del$(\mathcal{X})$–reconstructible families:*

i) $\mathcal{R}_1(x,v) = \mathcal{R}(\mathcal{X}, x, v)$, *epimorphisms are given by identity;*

ii) $\mathcal{R}_2(x,v) = S\,(\mathcal{R}(\mathcal{X}, x, v))$, *epimorphisms are functions induced by S.*

Then

i) $\mathcal{X} \approx_{WR} \int \mathcal{X}, \mathcal{R}_1(x,v)$ *and*

ii) $\mathcal{X} \approx_B \int \mathcal{X}, \mathcal{R}_2(x,v) \approx_B S\int \mathcal{X}, \mathcal{R}_2(x,v)$.

Definition 4.10

i) *The* canonical WR–reduction *of a tree $\mathcal{X} = (X, \varepsilon, \alpha)$ in Tree$_\tau$, written $WR\mathcal{X}$, is $\int \mathcal{X}, \mathcal{R}_1(x,v)$.*

ii) *The* canonical B–reduction *of a tree $\mathcal{X} = (X, \varepsilon, \alpha)$ in Tree$_\tau$, written $B\mathcal{X}$, is $S\int \mathcal{X}, \mathcal{R}_2(x,v)$.*

Lemma 4.11 *Given a tree $\mathcal{X}$ in $Tree_\tau$, two derivatives $t, t' \in \mathcal{R}_\tau(\mathcal{X}, x, v)$ are*

i) weak resource bisimilar if and only if $del(t) = del(t')$;

ii) branching bisimilar if and only if $Sdel(t) \cong Sdel(t')$.

Theorem 4.12 *If $\mathcal{X}$ and $\mathcal{Y}$ are two trees, then we have*

i) $\mathcal{X} \approx_{WR} \mathcal{Y}$ if and only if $WR\mathcal{X} \cong WR\mathcal{Y}$.

ii) $\mathcal{X} \approx_B \mathcal{Y}$ if and only if $B\mathcal{X} \cong B\mathcal{Y}$.

Due to theorem 4.12 WR$\mathcal{X}$ can be thought of as the minimal weak resource representative for $\mathcal{X}$, while B$\mathcal{X}$ as its minimal branching representative.

We have not been able to provide a standard minimal representation for weak bisimulation by following the pattern of the construction for the other two weak equivalences. The reason for this idiosyncrasy is the impossibility of building the standard representation via quotienting: weak bisimulation does not enforce a direct correspondence between the *runs* of equivalent trees, and hence we cannot build a canonical representative as a quotient over the set of runs of a tree. To see this, consider the two weakly equivalent trees corresponding to the two terms

$$a(\tau b + c) + ab + a(\tau b + d) \qquad \textit{and} \qquad a(\tau b + c) + a(\tau b + d).$$

Now the tree corresponding to the second term is a good candidate for a minimal standard representative. However, the composition of the equivalence class of runs is unclear: the run ab of the first tree can either be absorbed by $a\tau b$ in the first or the third summands, and there is clearly no reason to prefer one choice over the other. Moreover, we cannot put it in both equivalence classes, for that would leave us with $a(\tau b + c + d)$ as the representative, and this is not bisimilar to the original.

5 An Enriched–Categorical Account

In this section we rephrase our account using more explicitly categorical machinery. We will show that the construction of minimal representative is "functorial" w.r.t. tree structure in all the cases. Furthermore a characterization of the resulting functors is given, that emphasizes the fact that the only difference between the resource-weak resource cases and strong-branching cases is the difference between the existence of a bijective function and the existence of relation epimorphic on both sides.

The notion that $\mathcal{Y}$ is a derivative of $\mathcal{X}$ accessed by a word w along a run x, $\mathcal{X}\,[x, w\rangle\,\mathcal{Y}$, naturally leads to a notion of map between trees different from our morphism. Clearly we could define the set of maps between $\mathcal{X}$ and $\mathcal{Y}$, as

$$\{[x, w] \mid \mathcal{X}\,[x, w\rangle\,\mathcal{Y}\}$$

and this would lead to a category of trees where a map from $\mathcal{X}$ to $\mathcal{Y}$ is a way of finding the derivative $\mathcal{Y}$ in $\mathcal{X}$. However, a moment's reflection shows that these arrows from $\mathcal{X}$ to $\mathcal{Y}$ are not just a set: they naturally bear a tree structure. Indeed, there are not just two paths from $\mathcal{X} \otimes \mathcal{Y}$ to $\mathcal{Y}$ of example $\otimes$, there is a tree, consisting of two runs; see Figure 2.

Definition 5.1 *The category Der, has trees as objects, arrows $f : \mathcal{X} \to \mathcal{Y}$ in Der$[\mathcal{X}, \mathcal{Y}]$ are paths $[x, w]$ such that $\mathcal{X}[x, w\rangle\, \mathcal{Y}'$ where $\mathcal{Y}'$ is an isomorphic copy of $\mathcal{Y}$. Given the tree of paths Der$[\mathcal{X}, \mathcal{Y}]$ and Der$[\mathcal{Y}, \mathcal{Z}]$, their composition in Der is given by concatenation in Tree.*

Der is properly seen not just as a category, but as a category enriched over Tree equipped with the monoidal structure $\otimes$ [13].

Similarly, Der$_\tau$ is the category of paths to derivatives with τs defined over Tree$_\tau$ in the same way as Der is defined over Tree.

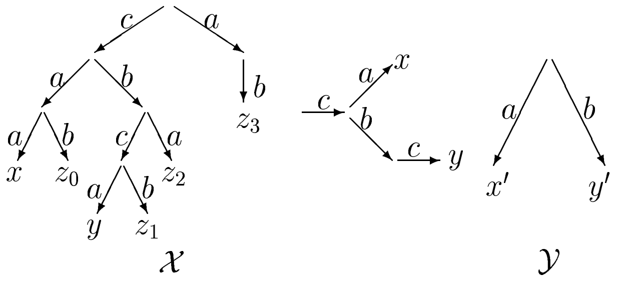


Fig. 2. Maps in *Der*: $\mathcal{X}[x, ca\rangle\, \mathcal{Y}$, $\mathcal{X}[y, cbc\rangle\, \mathcal{Y}$.

We have also that Der_τ is, as well, a Tree–category due to the effect of the functor del : Tree$_\tau \to$ Tree obtained by applying del to homs. In the sequel, Der_τ will always denote this Tree–category.

Of course the identity functor on Tree induces the identity functor on Der, but we will show now that the reduction maps S, B and WR, though not being endofunctors, do induce Tree–functors from Der (Der_τ) to itself.

Lemma 5.2 *Given any two trees, $\mathcal{Z}$ and $\mathcal{Z}'$, there is a Tree–map between the trees:*

i) Der$[\mathcal{Z}, \mathcal{Z}']$ and Der$[S\mathcal{Z}, S\mathcal{Z}']$.

ii) del(Der$_\tau[\mathcal{Z}, \mathcal{Z}']$) and del(Der$_\tau[WR\mathcal{Z}, WR\mathcal{Z}']$).

iii) del(Der$_\tau[\mathcal{Z}, \mathcal{Z}']$) and del(Der$_\tau[B\mathcal{Z}, B\mathcal{Z}']$).

Theorem 5.3 *The endomap S on Tree induces a Tree–functor, F_S, on Der. The endomaps WR and B on Tree$_\tau$ induce Tree–functors, F_{WR}, and F_B on Der$_\tau$.*

We now go on to examine the nice property that allows us to characterize the Tree–functors above.

To begin with, consider the notion of a $\mathcal{V}$–functor $F : \mathcal{C} \to \mathcal{C}$ over some $\mathcal{V}$–category $\mathcal{C}$ being *full* [13]. For this to be the case, we require that for each pair of objects, A, B, the induced function $F_{A,B}$ from $\mathcal{C}[A, B]$ to $\mathcal{C}[FA, FB]$ is an epimorphism. Fullness condition in our case would amount to asking that all paths from FA to FB arise via paths from A to B. This is both too naïve and too demanding.

We want to require that all paths from Ft to any u are obtained via some u_i such that $Fu_i \cong u'$; this is the notion that will allow us to capture functors like ours.

Definition 5.4 *A $\mathcal{V}$–functor $F : \mathcal{C} \to \mathcal{C}$ is said to be* hereditarily full *if and only if for any objects A, B of $\mathcal{C}$, there exists a family $\{B_i\}$ such that $FB_i \cong B$ and $\{F_{A,B_i}\}$ covers $\mathcal{C}[FA, B]$.*

Proposition 5.5

i) F_S : Der $\to$ Der is an hereditarily full Tree–functor.

ii) F_{WR} : Der$_\tau$ $\to$ Der$_\tau$ is an hereditarily full Tree–functor preserving 1 and sums.

iii) F_B : Der$_\tau$ $\to$ Der$_\tau$ is an hereditarily full Tree–functor.

The only trees in Tree that have none but trivial derivatives are finite, nonempty, sums of 1s. Let us call them *quasi terminals*. If $F(t)$ is a quasi terminal, so is t. If F is hereditarily full, one has the viceversa, i.e., quasi terminals are preserved as a class.

Naturally, identity (the Tree–functor *Der* $\to$ *Der* induced by resource bisimulation) is hereditarily full. It is the only one, up to isomorphism, enjoying this property and preserving 1 and sums, hence preserving quasi terminals as individuals. This fact can be easily proved by induction on the depth of the tree. Next theorem will show that the other Tree–functors considered in this paper enjoy a similar feature, because they are in some sense universal with respect to the class of Tree–functors with the same properties. The statement corresponds to the minimality of the canonical representative.

Theorem 5.6 *i) For all hereditarily full Tree–functors F : Der $\to$ Der, $F_S F \cong F_S$.*

ii) For all hereditarily full, preserving 1 and sums, Tree–functors F : Der$_\tau$ $\to$ Der$_\tau$, $F_{WR} F \cong F_{WR}$.

iii) For all hereditarily full Tree–functors F : Der$_\tau$ $\to$ Der$_\tau$, $F_B F \cong F_B$.

A direct consequence of this theorem is that all hereditarily full Tree–functors preserving 1 and sums, preserve weak resource bisimulation equivalence, while all hereditarily full Tree– functors preserve branching bisimulation equivalence.

6 Conclusions

We have studied labelled trees as unfoldings of transitions systems and characterized different bisimulations as special functors between categories of trees, enjoying universal properties. We have thus devised criteria for comparing and assessing different equivalences: branching bisimulation appears as the natural generalization of strong bisimulation just like weak resource bisimulation is the natural generalization of isomorphism of trees.

The definition of the functors has required, as an intermediate step, the construction of a canonical representative of the considered equivalence classes. The construction of canonical representatives for weak bisimulation equivalence turned out to be problematic; we could not define a quotient that preserved the structure of the runs.

Our approach to bisimulations characterizations is related to that introduced in [11] and used in [1], only they start from a different view of the same "topological" structure. Our trees have originally been defined as categories enriched over a locally–posetal 2–category $\mathcal{A}$ namely that associated with the free monoid A^* [12]. Similarly, morphisms between trees are $\mathcal{A}$–functors. It is well known that our trees, as categories enriched on a posetal 2-category, can be thought as presentations of sheaves on the topology where elements of A^* constitute a base. To obtain the corresponding sheaves we would roughly need to complete runs with all their prefixes. To recover the approach followed in [11], elements of A^* could be considered as a subcategory P of paths in Tree and we could characterize strong and branching bisimulation as in [11] via spans of P–open maps. The Tree–functoriality corresponds to preservation of "path logic", but our construction, provides also minimal representatives that cannot be obtained via spans. As in our case, characterization of weak equivalence in [11] is problematic, see [1]; it requires introducing an "ad hoc" selection of morphisms or a weakening of the logic to be preserved. This weaker characterization is not reproducible in our context that is more demanding on structural properties.

The two new equivalences that we have considered, and that are not considered in the above mentioned papers, have proved very useful. *Resource bisimulation* has been used to obtain a complete axiomatization of a tree-based interpretation of regular expressions [6] and to provide alternative operational semantics of process algebras [2]. *Weak resource bisimulation* can be fully axiomatized by simply adding to the axioms for resource bisimulation the following law:

$$\alpha; \tau; X \ = \ \alpha; X$$

that essentially says that all and only the “irrelevant” τs are ignored.

Besides this line of investigation, let us mention two promising topics for further work. In this paper we have only considered action–labelled finite trees. There are two obvious generalizations.

Firstly, like it has been done for the open-maps approach, we could export our characterizations to different semantics, those that admit an enriched categorical presentation, i.e., we could consider richer labels that would enable us to rely on the same bisimulations also for non-interleaving models of concurrency [17] and capture, e.g., causal dependence, maximal concurrency, locality–based properties in the same vein of [4,5]. Secondly, we could consider finite state transition systems with cycles (and hence infinite unfoldings).

However, while the generalization to richer labels is direct, the adjustments needed for dealing with infinity are not minor. Indeed, a key point of our approach is that unfoldings of systems are described as sets of runs from an initial to a final state. Now, while in the case of finite LTSs we immediately have final states, in the cyclic case, we would need to single out specific states as final and ensure that all of them are equivalent. One possibility is to 'massage' systems to include sink states in correspondence with each final state, for instance via the (standard automata–theoretic) construction of introducing epsilon moves. The set of runs of an LTS would then be the set of all finite runs with the obvious labeling; the agreement of any two runs would be the string associated with their initial common run. A run x is considered an approximation of a run y whenever $\alpha(x, y) = \varepsilon(x) < \varepsilon(y)$.

But this will be the subject of future research.

7 APPENDIX: Rebuilding Trees

In this appendix we provide an example of the reconstruction procedure formally defined in Section 4.

First of all, we show how to obtain decorated rigid trees from those with silent actions.

In Figure 3 we have represented the tree $\mathcal{X}$ and the tree del($\mathcal{X}$) obtained by deleting all silent moves from $\mathcal{X}$. For the sake of readability, we name y_i the runs of *del*($\mathcal{X}$) corresponding to the x_i of $\mathcal{X}$.

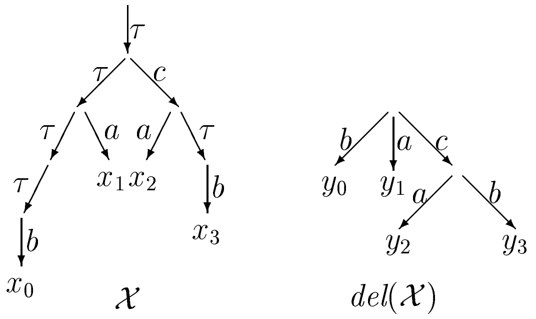


Fig. 3. A tree $\mathcal{X}$ in Tree$_\tau$ and its deletion.

The first step of our reconstruction consists of decorating each node $[y_i, u]$ in *del*($\mathcal{X}$) with the derivatives in $\mathcal{R}(\mathcal{X}, x_i, v)$. For instance, we decorate the nodes of *del*($\mathcal{X}$) in this way:

i) the root is decorated with four sets of derivatives, one for each y_i,

$$
\begin{aligned}
\mathcal{R}(\mathcal{X}, x_0, \epsilon) &= \{b + a + c(a + b),\ b + a,\ b\} \\
\mathcal{R}(\mathcal{X}, x_1, \epsilon) &= \{b + a + c(a + b),\ b + a\} \\
\mathcal{R}(\mathcal{X}, x_2, \epsilon) &= \{b + a + c(a + b)\} \\
\mathcal{R}(\mathcal{X}, x_3, \epsilon) &= \{b + a + c(a + b)\}
\end{aligned}
$$

ii) the leaf node of the y_0 branch, $[y_0, b]$, is decorated with:

$$\mathcal{R}(\mathcal{X}, x_0, b) = \{1\}$$

iii) the leaf node of the y_1 branch is decorated with:

$$\mathcal{R}(\mathcal{X}, x_1, a) = \{1\}$$

iv) the node $[y_2, c] = [y_3, c]$ is decorated with:

$$
\begin{aligned}
\mathcal{R}(\mathcal{X}, x_2, c) &= \{a + b\} \\
\mathcal{R}(\mathcal{X}, x_3, c) &= \{a + b,\ b\}.
\end{aligned}
$$

v) the leaf node of the y_2 branch is decorated with:

$$\mathcal{R}(\mathcal{X}, x_2, ca) = \{1\}$$

vi) the leaf node of the y_3 branch is decorated with:

$$\mathcal{R}(\mathcal{X}, x_3, cb) = \{1\}$$

The reconstruction of a tree in Tree$_\tau$ from the decorated version of $del(\mathcal{X})$ proceeds along the following lines.

Given the set of runs x_0, x_1, x_2, x_3, first, their new extent is defined. To do this we rely on the fact that each tree in $\mathcal{R}(\mathcal{X}, x, wa_i)$ represents a derivative accessible by a $\overset{a_i}{\Longrightarrow}$–step from $[x, w]$, and reconstruct the extent by introducing after each a_i a number of τs equal to $|\mathcal{R}(\mathcal{X}, x, wa_i)| - 1$ in order to guarantee the necessary branching points. To see this, consider Figure 4, where it is assumed that:

$$\mathcal{R}(\mathcal{X}, z_1, del(wa)) \ = \ \{t4 + t3 + t2 + t1, t3 + t2 + t1, t2 + t1, t1\}$$

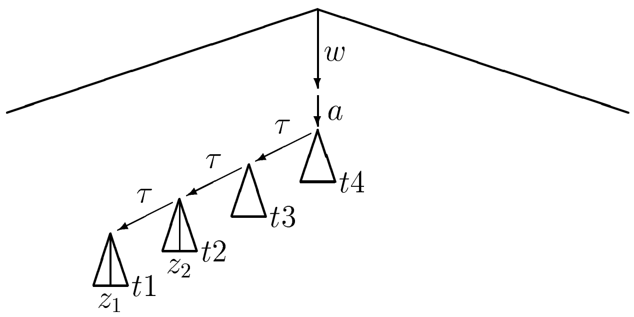


Fig. 4. A step in the reconstruction.

In our specific example the suggested construction amounts to defining:

$$\varepsilon(x_0) = \tau^2 b, \qquad \varepsilon(x_1) = \tau a, \qquad \varepsilon(x_2) = ca, \qquad \varepsilon(x_3) = c\tau b$$

The agreement between two given runs is then obtained again by adding after each a_i a number of τs equal to

$$|\mathcal{R}(\mathcal{X}, x_i, wa_i) \cap \mathcal{R}(\mathcal{X}, x_j, wa_i)| - 1$$

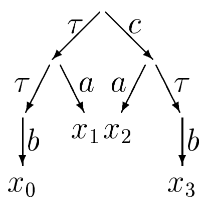


Fig. 5. The reconstructed tree: $\int \mathcal{R}(\mathcal{X}, x, v)$.

Thus, the complete reconstruction of the tree of Figure 3, which will be written $\int \mathcal{R}(\mathcal{X}, x, v)$, is shown in Figure 5. The reader may like to check that the reconstruction is weak resource bisimilar to the original tree.

References

[1] A. Cheng, M. Nielsen. Open Maps (at) Work. *BRICS Report Series*,n. 23, 1995.

[2] Corradini,F De Nicola,R. and Labella,A. Fully Abstract Models for Nondeterministic Regular Expressions. In Proc. *Concur '95.* LNCS 962, Springer Verlag, pp. 130-144, 1995.

[3] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24, pp. 211–237, 1987.

[4] P. Degano, R. De Nicola and U. Montanari. Universal Axioms for Bisimulation, *Theoretical Computer Science* volume 14, North Holland, pp. 63–91, 1993.

[5] P. Degano, R. De Nicola and U. Montanari. Observation Trees, In *Proc. North American Conference on Process Algebras*, (Purushothaman,S. and Zwarico, A. eds.) Springer-Verlag, Workshops in Computing series, pp. 103-118, 1993.

[6] De Nicola,R. and Labella,A. A Completeness theorem for Nondeterministic Kleene Algebras. in Proc. *MFCS '94.* LNCS 841, Springer Verlag, pp. 536-45, 1994.

[7] R. De Nicola, U. Montanari, and F. Vaandrager. Back and forth bisimulations. In *CONCUR'90* (J. Baeten and J. Klop eds.) volume 458, Springer-Verlag LNCS, pp. 152–165, 1990.

[8] R. van Glabbeek. The linear time - branching time spectrum. In *CONCUR'90* (J. Baeten and J. Klop, eds.), volume 458, Springer-Verlag LNCS, pp. 278–297, 1990.

[9] R. van Glabbeek. The linear time - branching time spectrum II (The Semantics of sequential systems with silent moves). In *CONCUR'93* (E. Best, ed.), volume 715, Springer-Verlag LNCS, pp. 66–81, 1993.

[10] R. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics *Journal of the ACM*, 43(3), pp. 555–600, 1996.

[11] A. Joyal, M. Nielsen and G. Winskel. Bisimulations and Open Maps. In *Logic in Computer Science*, pp. 418–427, 1993.

[12] S. Kasangian, A. Labella, and A. Pettorossi. Observers, Experiments, and Agents: A comprehensive approach to parallelism. In *Semantics of Concurrency*, (Guessarian,I. ed.) volume 469. Springer-Verlag LNCS, pp. 375-406, 1990.

[13] G. Kelly. *Basic Concepts of Enriched Category Theory.* Cambridge University Press, 1982.

[14] R. Milner. *Communication and concurrency.* International series on computer science, Prentice Hall International, 1989.

[15] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of Theoretical Computer Science 1981*, volume 104. Springer-Verlag LNCS, pp. 167–183, 1981.

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

[17] G. Winskel and M. Nielsen. Categories of models of concurrency. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science.* Vol. 4, Oxford Science Publications, pp. 2–148, 1995.