CONFER2 Workshop - Cambridge - U.K. - 14–15 September 2000

Tile Bisimilarity Congruences for Open Systems

Roberto Bruni Univ. Pisa

joint work with

David de Frutos Escrig Univ. Complutense de Madrid

Narciso Martí-Oliet Univ. Complutense de Madrid

Ugo Montanari Univ. Pisa

Research supported

by CNR Integrated Project *Progettazione e Verifica di Sistemi Eterogenei*;

by Esprit Working Groups *CONFER2* and *COORDINA*;

by CICYT Project *Desarollo Formal de Sistemas Distribuidos*;

and by MURST project *TOSCA*.

Open Systems

1. **Incomplete Systems:** e.g. repetitive, parametrized.

   $C[x]|D[x]$: same protocol used in C and D to provide a certain service.

   The two instances are instantiated with the same pattern, but can progress independently.

2. **Coordinators:** e.g. black box processes.

   $C[x]|D[x]$: agent that regulates the execution of its argument.

   The x indicates a single computational entity and replication must be avoided.

   $\mathbf{let}\ x = q\ \mathbf{in}\ C[x]|D[x]$

3. **Open Ended Systems:** e.g. dynamically reconfig.

   Dynamic bisimulation (**MS92**)

   $p_1 \sim_{\mathrm{d}} p_2$: $\forall C[_].\ C[p_1] \xrightarrow{a} q_1 \ \Rightarrow\ C[p_2] \xrightarrow{a} q_2 \ \wedge\ q_1 \sim_{\mathrm{d}} q_2$ (and vice versa).

In This Talk...

- OPEN ENDED (3) in (**BMS00**), not here.

- Here PARAMETRIZED (1) vs. COORDINATORS (2).

- BISIMILARITY via 'good formats.'

  related work: *context systems* (**LX90**); *tile model* (**GM97**); *conditional transition systems* (**Ren97**).

(pre)example 1 *Take* $P[x] = x\backslash_\alpha | x\backslash_\alpha$ *and* $Q[x] = (x|x)\backslash_\alpha$.

$P \overset{?}{\sim} Q$ *equivalent as coordinators (cannot react to* x*)*

$P \overset{?}{\sim} Q$ *but if* instance closure *is expected, they cannot be equivalent as parametrized systems:*
$P[p]$ *is not equivalent to* $Q[p]$ *for* $p = \alpha.nil + \overline{\alpha}.nil$.

Operational semantics

LTS: $p \xrightarrow{a} q$ states are closed terms; labels are actions.

Abstract semantics

Testing: $p \sim_1 q \Leftrightarrow \forall C[_].\, C[p] \equiv C[q]$.

Bisimilarity: $p \sim_2 q : p \xrightarrow{a} p' \;\Rightarrow\; q \xrightarrow{a} q' \;\wedge\; p' \sim_2 q'$ (and vice versa).

Congruence? $p \sim_2 q \overset{?}{\Rightarrow} \forall C[_].\, C[p] \sim_2 C[q]$.

For free if specification formats employed (e.g. De Simone, GSOS, tyft/tyxt)

Mixed: e.g. barbed bisimilarity.

Usually $\sim_2$ finer than $\sim_1$, but easier to prove by coinductive methods.

Operational semantics

$$\frac{x_1 \xrightarrow{a_1} y_1 \;\cdots\; x_n \xrightarrow{a_n} y_n}{C[x_1,\ldots,x_n] \xrightarrow{b} D[y_1,\ldots,y_n]} \quad \text{or} \quad C \xrightarrow[b]{\vec{a}} D \quad \text{or} \quad \begin{tikzcd} \underline{n} \arrow[r, "C"] \arrow[d, "\vec{a}"'] & \underline{1} \arrow[d, "b"] \\ \underline{n} \arrow[r, "D"'] & \underline{1} \end{tikzcd}$$

Open transitions derived by contracting proof terms

e.g. $$\frac{\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{\bar{a}} y'}{x|y \xrightarrow{\tau} x'|y'}}{(x|y)\backslash\alpha \xrightarrow{\tau} (x'|y')\backslash\alpha} \quad \text{yields} \quad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{\bar{a}} y'}{(x|y)\backslash\alpha \xrightarrow{\tau} (x'|y')\backslash\alpha}$$

using tiles: $$\begin{tikzcd} \underline{2} \arrow[r, "-1|-2"] \arrow[d, "a\otimes\bar{a}"'] & \underline{1} \arrow[r, "-1\backslash\alpha"] \arrow[d, "\tau"] & \underline{1} \arrow[d, "\tau"] \\ \underline{2} \arrow[r, "-1|-2"'] & \underline{1} \arrow[r, "-1\backslash\alpha"'] & \underline{1} \end{tikzcd} = \begin{tikzcd} \underline{2} \arrow[r, "(-1|-2)\backslash\alpha"] \arrow[d, "a\otimes\bar{a}"'] & \underline{1} \arrow[d, "\tau"] \\ \underline{2} \arrow[r, "(-1|-2)\backslash\alpha"'] & \underline{1} \end{tikzcd}$$

structural LTS specifications apt to deal with open terms

Abstract semantics

Via Ground Closure: $C \approx_1 D \Leftrightarrow \forall\sigma.\ \sigma(C) \sim \sigma(D)$

for σ ground substitution.

Bisimilarity: $C \approx_2 D : C \xrightarrow[b]{\vec{a}} C' \ \Rightarrow\ D \xrightarrow[b]{\vec{a}} D' \ \wedge\ C' \approx_2 D'$

(and vice versa).

'Instance congr.'? $C \approx_2 D \ \wedge\ p \approx_2 q \ \overset{?}{\Rightarrow}\ C[p] \approx_2 D[q]$.

On closed terms we have $\approx_2 = \approx_1= \sim$.

In general $\approx_2$ finer than $\approx_1$.

e.g. CCS with actions $\{a, b\}$

let $p = \mathbf{rec}\ z.\ az + bz + \bar{a}z + \bar{b}z + \tau z$

then $\forall q.\ q|p \approx_1 q\backslash_a|p \approx_1 p$

but $x|p \xrightarrow[a]{a} y|p$, while $x\backslash_a|p$ cannot perform $\xrightarrow[a]{a}$.

Problems and Solutions

What are the good formats?

The original tile model (De Simone extension) in (**GM97**) cannot guarantee that tile bisimilarity is an instance congruence (IC). The problem arises from shared variables in states vs. linear observations.

Proposed solutions:

1. Only **linear open systems**

2. **Variable splitting:** $f(x,x) \xrightarrow{\nabla} f(x_1,x_2)$

3. **Term graphs:** $C[p] \neq \mathbf{let}\ x = p\ \mathbf{in}\ C[x]$ preserves sharing

We define corresponding tile formats and prove the congruence result for each of them.

Categories of Substitutions

Signature Σ (one sorted) and **Variables** in X

Term algebras $\mathbb{T}_\Sigma(X)$ and $\mathbb{T}_\Sigma = \mathbb{T}_\Sigma(\varnothing)$

Substitutions $\sigma\colon X \to \mathbb{T}_\Sigma(X)$ (extend to $\sigma\colon \mathbb{T}_\Sigma(X) \to \mathbb{T}_\Sigma(X)$)

Canonical variables: $\underline{n} = \{x_1, x_2, \ldots, x_n\}$

Finite substitution $\sigma = [t_1/x_1, \ldots, t_n/x_n]$

σ' can be applied elementwise to σ yielding $[\sigma'(t_1)/x_1, \ldots, \sigma'(t_n)/x_n]$

Context $t = C[x_1, ., x_n]$ (using at most $x_1, \ldots, x_n$)

$C[t_1, \ldots, t_n]$ given by applying $[t_1/x_1, \ldots, t_n/x_n]$ to $C[x_1, ., x_n]$

Linear terms, contexts, substitutions

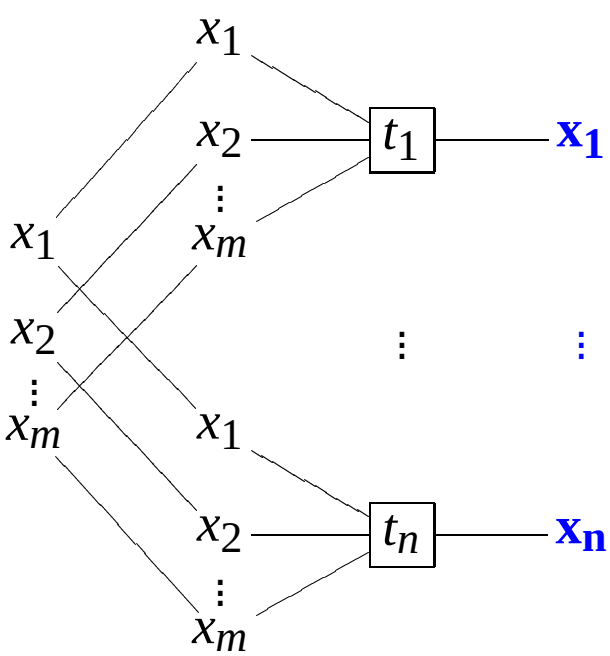
Categories of Substitutions (II)

Context as substitution from n terms to 1: $[C[x_1,\ldots,x_n]/x_1]$.

More generally: For $t_i \in \mathbb{T}_\Sigma(\underline{m})$, $\sigma\colon \underline{m} \to \underline{n}$.

given m terms as inputs, return n terms as output

$[t_1/x_1,\ldots,t_n/x_n]$ written just as $\langle t_1,\ldots,t_n\rangle$



Sequential composition is substitution application.

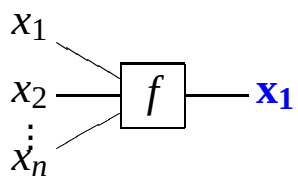
Brackets can be omitted **ex. a, b const. and f binary**

$\langle \mathbf{a},\mathbf{b}\rangle; \langle \mathbf{f}(\mathbf{x_1},\mathbf{x_2}),\mathbf{x_1}\rangle; \mathbf{f}(\mathbf{x_2},\mathbf{x_1}) = \langle \mathbf{f}(\mathbf{a},\mathbf{b}),\mathbf{a}\rangle; \mathbf{f}(\mathbf{x_2},\mathbf{x_1}) = \mathbf{f}(\mathbf{a},\mathbf{f}(\mathbf{a},\mathbf{b}))$

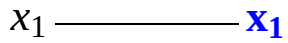
Categories of Substitutions (III)

Algebraic Theories **(Law63)**: **finite subs on Σ generated by**

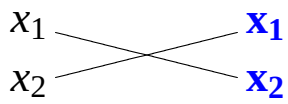
$\langle \mathbf{f} \rangle : \underline{n} \to \underline{1}$ **for** $f \in \Sigma$ **with arity** n



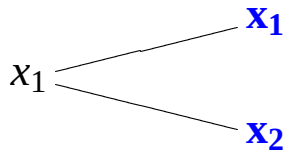
$\mathbf{id}_{\underline{1}} = \langle \mathbf{x_1} \rangle : \underline{1} \to \underline{1}$



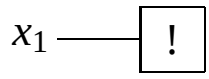
$\gamma_{\underline{1},\underline{1}} = \langle \mathbf{x_2}, \mathbf{x_1} \rangle : \underline{2} \to \underline{2}$



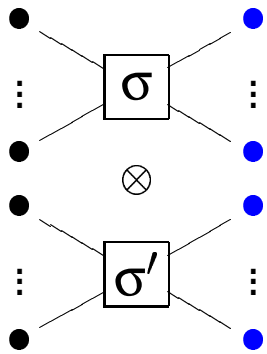
$\nabla_{\underline{1}} = \langle \mathbf{x_1}, \mathbf{x_1} \rangle : \underline{1} \to \underline{2}$

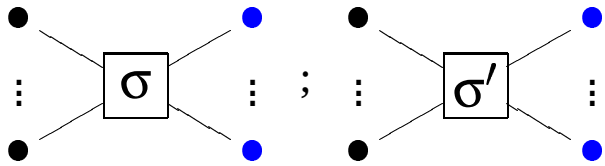


$!_{\underline{1}} = \langle \rangle : \underline{1} \to \underline{0}$



composing with tensor $_ \otimes _$ **and application** $_ ; _$





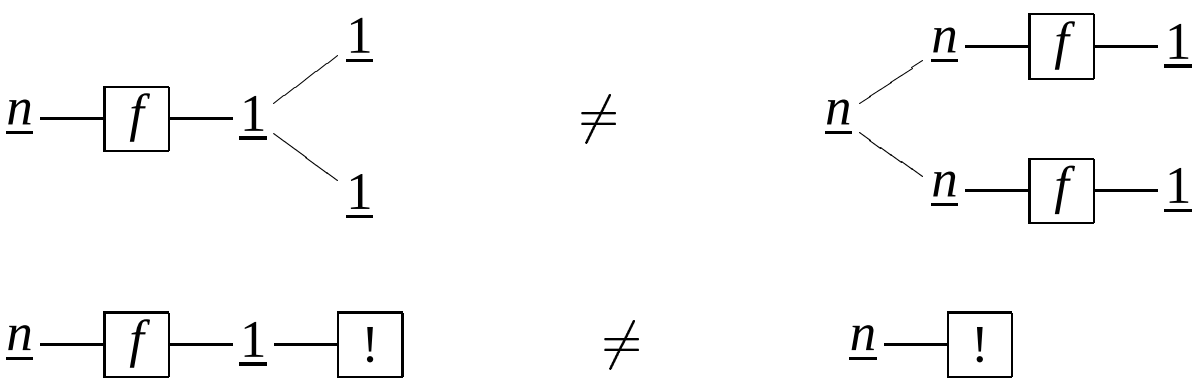
Terms

$$f;\nabla = \nabla;(f \otimes f)$$

$$f;! = !$$

Term Graphs

Abandon the naturality of ∇ and ! (two axioms above) (**CG99**)



Bisimulation and sos

Bisimulation:

$t_1 \sim t_2$ and $t_1 \xrightarrow{a} s_1$ imply $t_2 \xrightarrow{a} s_2$ with $s_1 \sim s_2$

(and vice versa)

Bisimilarity $\simeq$ is the largest bisimulation

Is $\simeq$ a congruence? $t_i \simeq s_i \overset{?}{\Rightarrow} f(t_1, \ldots, t_n) \simeq f(s_1, \ldots, s_n)$

De Simone, positive `GSOS`, `tyft/tyxt` formats

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, \ldots, x_n) \xrightarrow{a} t}$$

Conclusions $C[x_1, \ldots, x_n] \xrightarrow{a} t$ never allowed!

Main Features of Tile Logic

TL is a logic for modular descriptions of the dynamic evolution of concurrent systems

- ☐ Suited to deal with **synchronization issues**
- ☐ General notion of **configuration** and of **observation**

  horizontal structure different than terms **(linear terms, term graphs, graphs, higher-order terms)**

  vertical structure different than actions **(causality and locality dependencies, name creation, name abstraction)**
- ☐ Defines compositional models of computation in **space** and in **time**
- ☐ Allows to reason about **contexts**
- ☐ Admits suggestive **graphical presentation**
- ☐ Can model **heterogeneous systems**
- ☐ Algebra of **concurrent computation proofs**
- ☐ **auxiliary tiles** to model built-in operations

Tile Model

A ***tiles system*** is a tuple $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ where:
$\mathcal{H}$ monoidal category of ***configurations***
$\mathcal{V}$ monoidal category of ***observations***
$\mathcal{H}$ and $\mathcal{V}$ have the same objects (called ***interfaces***)
N is the set of *tile names*
$R: N \to \mathcal{H} \times \mathcal{V} \times \mathcal{V} \times \mathcal{H}$

if $R(\alpha) = \langle s, a, b, t \rangle$, written α: $s \xrightarrow[b]{a} t$, then
$$\begin{tikzcd} x \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "\alpha"] & y \arrow[d, "b"] \\ z \arrow[r, "t"'] & w \end{tikzcd}$$

Tiles can be composed

$$\begin{tikzcd} \circ \arrow[r] \arrow[d] \arrow[dr, phantom, "\alpha"] & \circ \arrow[r] \arrow[d] \arrow[dr, phantom, "\beta"] & \circ \arrow[d] \\ \circ \arrow[r] & \circ \arrow[r] & \circ \end{tikzcd}$$

$$\begin{tikzcd} \circ \arrow[r] \arrow[d] & \circ \arrow[d] \\ \circ \arrow[r] \arrow[d] \arrow[dr, phantom, "\alpha"] & \circ \arrow[d] \\ \circ \arrow[r] & \circ \end{tikzcd}$$

$$\begin{tikzcd} \circ \arrow[r] \arrow[d] \arrow[dr, phantom, "\alpha"] & \circ \arrow[d] \\ \circ \arrow[r] \arrow[d] \arrow[dr, phantom, "\beta"] & \circ \arrow[d] \\ \circ \arrow[r] & \circ \end{tikzcd}$$

Identities

$$\begin{tikzcd} x \arrow[r, "s"] \arrow[d, "id_x"'] \arrow[dr, phantom, "id_s"] & y \arrow[d, "id_x"] \\ x \arrow[r, "s"'] & y \end{tikzcd} \qquad \begin{tikzcd} x \arrow[r, "id_x"] \arrow[d, "a"'] \arrow[dr, phantom, "id_a"] & x \arrow[d, "a"] \\ z \arrow[r, "id_z"'] & z \end{tikzcd}$$

(possibly auxiliary tiles)

Tile Logic and Auxiliary Tiles

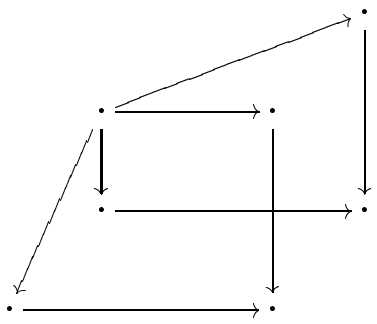
Given $\mathcal{R}$, the associated *tile logic* is obtained by adding some canonical 'auxiliary' tiles and then closing by composition both auxiliary and basic tiles.

$$\mathcal{R} \vdash s \xrightarrow[b]{a} t$$

aux. tiles can e.g. model isomorphic transf. of interfaces

e.g. identity tiles allow for reduction in any context $\dfrac{x \xrightarrow{id} y}{f(x) \xrightarrow{id} f(y)}$

e.g. variable splitting



e.g. function application

$$\begin{tikzcd} \cdot \arrow[r, "\Lambda t \otimes id"] \arrow[d, "id"'] & \cdot \arrow[d, "eval"] \\ \cdot \arrow[r, "t"'] & \cdot \end{tikzcd}$$

Tile Bisimulation and Tile Decomposition

Tiles allow for a very general notion of bisimulation (note that t_1 and t_2 can be open terms)

$\mathbf{t_1} \sim_{\mathrm{t}} \mathbf{t_2}$ and $\mathcal{R} \vdash t_1 \xrightarrow[b]{a} s_1$ imply $\mathcal{R} \vdash t_2 \xrightarrow[b]{a} s_2$ with $\mathbf{s_1} \sim_{\mathrm{t}} \mathbf{s_2}$ (and vice versa)

Tile bisimilarity $\simeq_{\mathrm{t}}$ is the largest tile bisimulation

Definition 1 *A tile system $\mathcal{R}$ enjoys the* **decomposition property** *if for all $\mathcal{R} \vdash s \xrightarrow[b]{a} t$*

1. *if $s = s_1; s_2$ then*
   $\mathcal{R} \vdash s_1 \xrightarrow[c]{a} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b]{c} t_2$ *with* $t = t_1; t_2$

2. *if $s = s_1 \otimes s_2$ then*
   $\mathcal{R} \vdash s_1 \xrightarrow[b_1]{a_1} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b_2]{a_2} t_2$,
   with $a = a_1 \otimes a_2$, $b = b_1 \otimes b_2$ *and* $t = t_1 \otimes t_2$

Proposition 1 *If $\mathcal{R}$ enjoys the decomposition property, then tile bisimilarity is a congruence.*

Decomposition Illustrated

$$\begin{tikzcd} \bullet \arrow[r, "s"] \arrow[d, "a"'] & \bullet \arrow[d, "b"] \\ \bullet \arrow[r, "t"'] & \bullet \end{tikzcd}$$

$$\begin{tikzcd} \bullet \arrow[r, "s_1"] \arrow[d, "a"'] & \bullet \arrow[r, "s_2"] & \bullet \arrow[d, "b"] \\ \bullet \arrow[rr, "t"'] & & \bullet \end{tikzcd}$$

implies

$$\begin{tikzcd} \bullet \arrow[r, "s_1"] \arrow[d, "a"'] & \bullet \arrow[r, "s_2"] \arrow[d, "c"'] & \bullet \arrow[d, "b"] \\ \bullet \arrow[r, "t_1"'] & \bullet \arrow[r, "t_2"'] & \bullet \end{tikzcd}$$

$$\begin{tikzcd} \bullet \arrow[r, Rightarrow, "s_1 \otimes s_2"] \arrow[d, "a"'] & \bullet \arrow[d, "b"] \\ \bullet \arrow[r, "t"'] & \bullet \end{tikzcd}$$

implies

$$\begin{tikzcd} \bullet \arrow[r, "s_1"] \arrow[dd, "a_1"'] & \bullet \arrow[dd, "b_1"] \\ & \\ \bullet \arrow[r, "t_1"'] & \bullet \end{tikzcd} \qquad \begin{tikzcd} \bullet \arrow[r, "s_2"] \arrow[d, "a_2"'] & \bullet \arrow[d, "b_2"] \\ \bullet \arrow[r, "t_2"'] & \bullet \end{tikzcd}$$

Tile Formats

Starting from signatures (generating $\mathcal{H}$ and $\mathcal{V}$)

$$\frac{\{x_i \xrightarrow{a_i} y_i\}}{C[x_1,\ldots,x_n] \xrightarrow{a} D[y_1,\ldots,y_n]}$$

$$\begin{tikzcd} \underline{n} \arrow[r, "C[x_1,\ldots,x_n]"] \arrow[d, "a_1\otimes\cdots\otimes a_n"'] & \underline{1} \arrow[d, "a"] \\ \underline{n} \arrow[r, "D[x_1,\ldots,x_n]"'] & \underline{1} \end{tikzcd}$$

note the canonical renaming of the y's

Basic Source (BS): C consists of one operator.

Main Results

Format	hor.	vert.	BS $\Rightarrow$ IC?
Algebraic (GM97)	term	linear	no
monoidal (MM97)	linear	linear	yes
term (BMM98)	term	term	yes
term graph	term graph	linear	yes

Tiles for CCS

$$\frac{}{\mu.x_1 \xrightarrow{\mu} x_1}$$

$$\begin{tikzcd} \underline{1} \arrow[r, "\mu"] \arrow[d, "id"'] & \underline{1} \arrow[d, "\mu"] \\ \underline{1} \arrow[r, "id"'] & \underline{1} \end{tikzcd}$$

$$\frac{x_1 \xrightarrow{\mu} y_1}{x_1 \backslash \alpha \xrightarrow{\mu} y_1 \backslash \alpha} \qquad (\mu \neq \alpha, \overline{\alpha})$$

$$\begin{tikzcd} \underline{1} \arrow[r, "\backslash\alpha"] \arrow[d, "\mu"'] & \underline{1} \arrow[d, "\mu"] \\ \underline{1} \arrow[r, "\backslash\alpha"'] & \underline{1} \end{tikzcd}$$

$$\frac{x_1 \xrightarrow{\mu} y_1}{x_1 + x_2 \xrightarrow{\mu} y_1} \qquad \text{(and symmetric)}$$

$$\begin{tikzcd} \underline{2} \arrow[r, "+"] \arrow[d, "\mu\otimes id"'] & \underline{1} \arrow[d, "\mu"] \\ \underline{2} \arrow[r, "id\otimes !"'] & \underline{1} \end{tikzcd}$$

$$\frac{x_1 \xrightarrow{\mu} y_1}{x_1 | x_2 \xrightarrow{\mu} y_1 | x_2} \qquad \text{(and symmetric)}$$

$$\begin{tikzcd} \underline{2} \arrow[r, "|"] \arrow[d, "\mu\otimes id"'] & \underline{1} \arrow[d, "\mu"] \\ \underline{2} \arrow[r, "|"'] & \underline{1} \end{tikzcd}$$

$$\frac{x_1 \xrightarrow{\lambda} y_1 \quad x_2 \xrightarrow{\overline{\lambda}} y_2}{x_1 | x_2 \xrightarrow{\tau} y_1 | y_2}$$

$$\begin{tikzcd} \underline{2} \arrow[r, "|"] \arrow[d, "\lambda\otimes\overline{\lambda}"'] & \underline{1} \arrow[d, "\tau"] \\ \underline{2} \arrow[r, "|"'] & \underline{1} \end{tikzcd}$$

Counterexample Algebraic Format

Horizontal: **term** and Vertical: **linear**

$$\underline{1} \xrightarrow{(x|x)\backslash a} \underline{1} \qquad \sim_{\mathrm{t}} \qquad \underline{1} \xrightarrow{(x\backslash a)|(x\backslash a)} \underline{1}$$

In fact they can do the same, e.g.

$$\begin{tikzcd}
\underline{1} \arrow[r, "(x|x)\backslash a"] \arrow[d, "\tau"'] & \underline{1} \arrow[d, "\tau;\tau"] \\
\underline{1} \arrow[r, "(x|x)\backslash a"'] & \underline{1}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "(x\backslash a)|(x\backslash a)"] \arrow[d, "\tau"'] & \underline{1} \arrow[d, "\tau;\tau"] \\
\underline{1} \arrow[r, "(x\backslash a)|(x\backslash a)"'] & \underline{1}
\end{tikzcd}$$

However

$$\underline{0} \xrightarrow{a.nil+\bar{a}.nil} \underline{1} \xrightarrow{(x|x)\backslash a} \underline{1} \qquad \not\sim_{\mathrm{t}} \qquad \underline{0} \xrightarrow{a.nil+\bar{a}.nil} \underline{1} \xrightarrow{(x\backslash a)|(x\backslash a)} \underline{1}$$

Since

$$\begin{tikzcd}
\underline{0} \arrow[r, "((a.nil+\bar{a}.nil)|(a.nil+\bar{a}.nil))\backslash_a"] \arrow[d, "id"'] & \underline{1} \arrow[d, "\tau"] \\
\underline{0} \arrow[r, "(nil|nil)\backslash_a"'] & \underline{1}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\underline{0} \arrow[r, "((a.nil+\bar{a}.nil)\backslash_a)|((a.nil+\bar{a}.nil)\backslash_a)"] \arrow[d, "id"'] \arrow[dr, phantom, "\mathbf{NO}"] & \underline{1} \arrow[d, "\tau"] \\
\underline{0} \arrow[r, "?"'] & \underline{1}
\end{tikzcd}$$

Example Monoidal Format

Horizontal: **linear** and Vertical: **linear**

$$\underline{2} \xrightarrow{(x_1|x_2)\backslash a} \underline{1} \qquad \not\sim_{\mathrm{t}} \qquad \underline{2} \xrightarrow{(x_1\backslash a)|(x_2\backslash a)} \underline{1}$$

In fact

$$\begin{tikzcd} \underline{2} \arrow[r, "(x_1|x_2)\backslash a"] \arrow[d, "a\otimes\bar{a}"'] & \underline{1} \arrow[d, "\tau"] \\ \underline{2} \arrow[r, "(x_1|x_2)\backslash a"'] & \underline{1} \end{tikzcd} \qquad \begin{tikzcd} \underline{2} \arrow[r, "(x_1\backslash a)|(x_2\backslash a)"] \arrow[d, "a\otimes\bar{a}"'] \arrow[dr, phantom, "\mathbf{NO}"] & \underline{1} \arrow[d, "\tau"] \\ \underline{2} \arrow[r, "?"'] & \underline{1} \end{tikzcd}$$

Example Term Format

Horizontal: **term** and Vertical: **term**

Open systems seen as **parametrized**

$$\underline{1} \xrightarrow{(x|x)\backslash a} \underline{1} \qquad \not\sim_{\mathrm{t}} \qquad \underline{1} \xrightarrow{(x\backslash a)|(x\backslash a)} \underline{1}$$

In fact

$$\begin{tikzcd}
\underline{1} \arrow[r, "(x|x)\backslash a"] \arrow[d, "\nabla_{\underline{1}}"'] & \underline{1} \arrow[d, "id"] \\
\underline{2} \arrow[r, "(x_1|x_2)\backslash a"] \arrow[d, "a\otimes\overline{a}"'] & \underline{1} \arrow[d, "\tau"] \\
\underline{2} \arrow[r, "(x_1|x_2)\backslash a"'] & \underline{1}
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{1} \arrow[r, "(x\backslash a)|(x\backslash a)"] \arrow[d, "\nabla_{\underline{1}}"'] & \underline{1} \arrow[d, "id"] \\
\underline{2} \arrow[r, "(x_1\backslash a)|(x_2\backslash a)"] \arrow[d, "a\otimes\overline{a}"'] \arrow[dr, phantom, "\mathbf{NO}"] & \underline{1} \arrow[d, "\tau"] \\
\underline{2} \arrow[r, "?"'] & \underline{1}
\end{tikzcd}$$

We remind that e.g. $(x|x)\backslash a = \nabla_{\underline{1}}; \langle (x_1|x_2)\backslash a \rangle$ and hence we can use the auxiliary tile

$$\begin{tikzcd}
\underline{1} \arrow[r, "\nabla_{\underline{1}}"] \arrow[d, "\nabla_{\underline{1}}"'] & \underline{2} \arrow[d, "id"] \\
\underline{2} \arrow[r, "id"'] & \underline{2}
\end{tikzcd}$$

Example Term Graph Format

Horizontal: **term graph** and Vertical: **linear**

Open systems seen as **coordinators**

$$\underline{1} \xrightarrow{(x|x)\backslash a} \underline{1} \qquad \sim_{\mathrm{t}} \qquad \underline{1} \xrightarrow{(x\backslash a)|(x\backslash a)} \underline{1}$$

In fact they can do the same, e.g.

$$\begin{tikzcd} \underline{1} \arrow[r, "(x|x)\backslash a"] \arrow[d, "\tau"'] & \underline{1} \arrow[d, "\tau;\tau"] \\ \underline{1} \arrow[r, "(x|x)\backslash a"'] & \underline{1} \end{tikzcd} \qquad \begin{tikzcd} \underline{1} \arrow[r, "(x\backslash a)|(x\backslash a)"] \arrow[d, "\tau"'] & \underline{1} \arrow[d, "\tau;\tau"] \\ \underline{1} \arrow[r, "(x\backslash a)|(x\backslash a)"'] & \underline{1} \end{tikzcd}$$

Furthermore

$$\underline{0} \xrightarrow{a.nil+\bar{a}.nil} \underline{1} \xrightarrow{(x|x)\backslash a} \underline{1} \qquad \sim_{\mathrm{t}} \qquad \underline{0} \xrightarrow{a.nil+\bar{a}.nil} \underline{1} \xrightarrow{(x\backslash a)|(x\backslash a)} \underline{1}$$

Because both

$$\underline{0} \xrightarrow{\textbf{let } x{=}a.nil+\bar{a}.nil \textbf{ in } (x|x)\backslash_a} \underline{1} \quad \text{and} \quad \underline{0} \xrightarrow{\textbf{let } x{=}a.nil+\bar{a}.nil \textbf{ in } (x\backslash_a)|(x\backslash_a)} \underline{1}$$

cannot move.

Concluding Remarks

Tile logic as an **abstract, compositional framework** suitable to deal in a uniform way with

- ⋆ parametrized systems
- ⋆ coordinators
- ⋆ open ended systems (**BMS00**)

Part of a research on extending concepts like

actions, synchronization, restriction, transaction

LTS, SOS, formats

bisimulation, reduction

concurrency, causality, locality, mobility

to structures different than terms

graphs, higher-order graphs, UML-like diagrams