# Variable Binding, Symmetric Monoidal Closed Theories, and Bigraphs

Richard Garner[1]
Tom Hirschowitz[2]
Aurélien Pardon[3]

[1]Cambridge University [2]CNRS, Université de Savoie [3]ENS Lyon

Concur '09

# Motivation

- In the paper: elementary, algebraic approach to variable binding in the presence of linearity.
- Here: additional, longer-term motivation, hoping for your feedback.

# Long term goal

Obtain

- algebraic,
- geometric, and
- modular

models of programming languages, with

- a clear separation between program and execution, e.g., 2-dimensional.

What do algebraic, geometric, and modular mean?

# Algebra

Universal algebra $<$ Lawvere theories $<$ locally presentable categories / sketches.

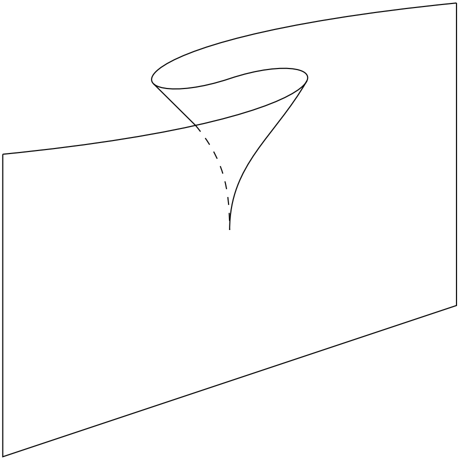
## Paradox

How algebraic is process algebra?

I here mean definition of their dynamics, not behavioural theories.

# Geometric

Geometric models of the chosen algebraic structure?
See John Baez' talk at LICS '09.



# Modular models

Montanari and colleagues (1996) and Melliès (2002) call for modular models of programming languages.

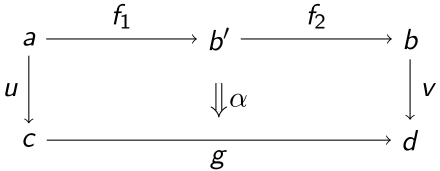
- Understanding programming languages as free double categories.
- Tiles, or cells, composing vertically and horizontally:

$$\begin{array}{ccc} a & \xrightarrow{\quad f \quad} & b \\ {\scriptstyle u}\downarrow & \Downarrow \alpha & \downarrow{\scriptstyle v} \\ c & \xrightarrow[\quad g \quad]{} & d. \end{array}$$

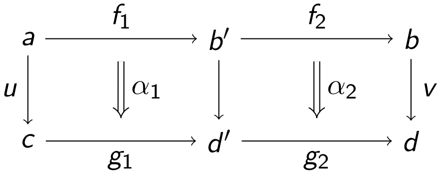
- $f$ and $g$ are programs.
- $\alpha$ is an execution or a reduction.
- $u$ and $v$ are side effects, or interactions with the environment.

# Modularity

Such a double category is modular when any execution of a composed program




decomposes as




with $g_2 g_1 = g$.

# Expected benefits

- Montanari et al., Sassone-Sobocinski: bisimulation is a automatically a congruence.
- Melliès: term tracing, rewriting.
- Hopefully compilation.

# Starting point

## Question

What should the horizontal category be?
Or: what should program composition be?

# Standard answer

Category of contexts:

- objects: typing contexts $\Gamma = (x \colon A, y \colon B)$,
- morphisms $\Delta \to \Gamma$: assignments

$$[x = e, y = f],$$

- composition by substitution:

$$\begin{array}{ccc} \Theta & \xrightarrow{\ \sigma\ } & \Delta \\ & \searrow^{[x = e[\sigma], y = f[\sigma]]} & \downarrow {\scriptstyle [x = e, y = f]} \\ & & \Gamma. \end{array}$$

# Other answers?

Do not plan to use that, for hand-waving reasons:

**Duplication belongs to the dynamics**

Does not model actual plugging of program fragments.

Besides:

**Claim, or thesis**

Duplication in composition hinders geometric intuition.

Hopefully: results will provide more.

# Linear substitution as program composition

## Proposal

Linear substitution as program composition.

I.e., the horizontal category is monoidal ($\neq$ finite products).

# Linear substitution as program composition

Who tried already?

- Linear programming languages.
  - No dynamic duplication either.
  - But a nice modular model by Melliès, a hint that linearity favours geometric intuition.
- Bigraphs (Jensen, Milner, . . . ).
  - No categorical semantics, esp. for the dynamics.
- Premonoidal or precartesian categories (Power, Robinson, Schuermann, . . . ).
  - No clear separation between program and execution.
  - Rather models program equivalence.

# Linear substitution as program composition: issues

- First issue: can we still handle languages with duplication?

  Hopefully yes: bound variables may be used several times, e.g., $\lambda x.xx$.

- Second issue: linearity is not stable under reduction.

$$(\lambda x.xx)y \longrightarrow yy.$$

# Linear substitution as program composition: issues

Problem: $(\lambda x.xx)y \longrightarrow yy$.

## Proposal

Give up reductions, use tiles:

$$\begin{array}{ccc} 1 & \xrightarrow{(\lambda x.xx)y} & 1 \\ {\scriptstyle c}\downarrow & \Downarrow \beta & \downarrow{\scriptstyle \tau} \\ 2 & \xrightarrow[y_1 y_2]{} & 1. \end{array}$$

The map $c$ says that $y$ is duplicated as $y_1$ and $y_2$.

# Linear substitution as program composition: issues

Problem: $(\lambda x.xx)y \longrightarrow yy$.

## Proposal

Give up reductions, use tiles:

$$\begin{array}{ccccc} n & \xrightarrow{\quad f \quad} & 1 & \xrightarrow{\quad (\lambda x.xx)y \quad} & 1 \\ {\scriptstyle c_n}\downarrow & \Downarrow {\scriptstyle c_f} & {\scriptstyle c}\downarrow & \Downarrow {\scriptstyle \beta} & \downarrow{\scriptstyle \tau} \\ 2n & \xrightarrow[\quad (f,f) \quad]{} & 2 & \xrightarrow[\quad y_1 y_2 \quad]{} & 1. \end{array}$$

The map $c$ says that $y$ is duplicated as $y_1$ and $y_2$.

# Contents

Sounds reasonable?

Now, this paper:

- Linear substitution, in the horizontal category only (no dynamics).
- Algebraic structure: symmetric monoidal closed (SMC) categories.
- Follow-up on Gadducci's GS·Λ-theories, with:
  - Use of recent, geometric presentation of the free SMC category (Hughes).
  - Application to bigraphs.

# Signatures

Formulae from intuitionistic multiplicative linear logic (IMLL):

$$A, B, \ldots \in \mathcal{F}(X) ::= x \mid I \mid A \otimes B \mid A \multimap B \qquad x \in X.$$

## Definition

A (SMC) signature is given by:

- a set $X$ of sorts, and
- a graph $\Sigma \overset{s,t}{\rightrightarrows} \mathcal{F}(X)$.

# Example 1: $\lambda$-calculus, first take

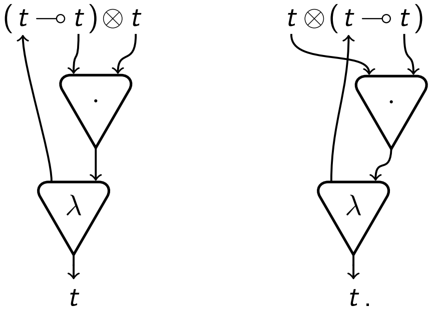
- One sort $t$.
- Two operations (edges):

$$(t \multimap t) \xrightarrow{\lambda} t \quad \text{and} \quad (t \otimes t) \xrightarrow{\cdot} t.$$

- Remember HOAS.
- Here internal to some SMC category.

# The generated SMC category

- Any $\Sigma$ freely generates an SMC category $\mathcal{S}(\Sigma)$.
  - Objects: formulae.
  - Morphisms: IMLL proof nets, modulo Trimble rewiring.
- Example, two interpretations for $\lambda x.(\Box_1 \Box_2)$:




- Combinatorial scoping condition generalising the Danos-Regnier criterion for IMLL.

# Back to Example 1

- Linearity: cannot model $\lambda x.xx$.
- Still:

## Proposition

*Morphisms* $I \longrightarrow t$ *are in bijection with closed linear* $\lambda$*-terms.*

# Example 2: $\lambda$-calculus, second take

Two sorts:

- $t$ for terms, and
- $v$ for variables.

Operations:

$$(v \multimap t) \xrightarrow{\lambda} t \qquad (t \otimes t) \xrightarrow{\cdot} t$$

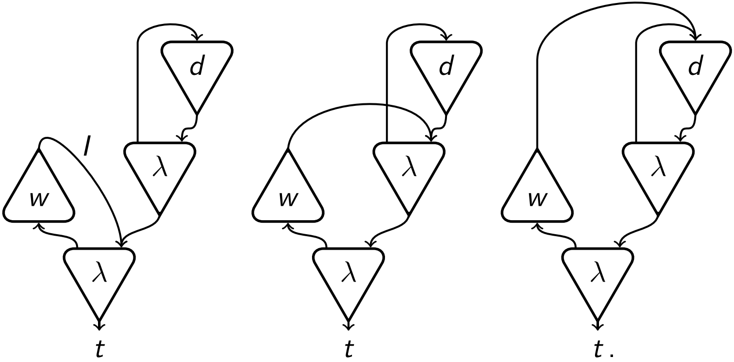
$$v \xrightarrow{d} t \qquad v \xrightarrow{c} v \otimes v \qquad v \xrightarrow{w} I.$$

Reminiscent from weak HOAS and Montanari et al. (independently).

This signature uses the unit $I$!

# Introducing Trimble rewiring

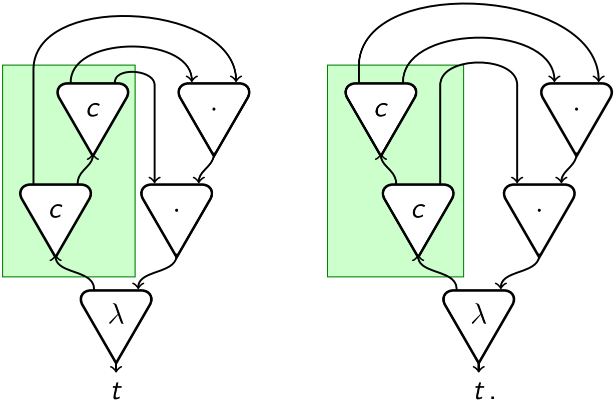
Identify the proof nets




A weakened variable is linked to anywhere inside its scope, indifferently.

# The need for equations

No bijection with $\lambda$-terms, e.g., $\lambda x.xxx$:



# SMC theories

- Specify equations between morphisms of $\mathcal{S}(\Sigma)$.
- For the $\lambda$-calculus: equations making $(v, c, w)$ into a commutative comonoid object yield:

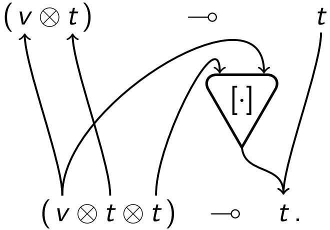
**Theorem**

*Morphisms $I \longrightarrow t$ are in bijection with closed $\lambda$-terms.*

Not modulo $\beta$, $\eta$, or any equation apart from $\alpha$-equivalence.

# Higher-order

- Theory for *Mobile Ambients*.
- Ex 1: $(a, P) \mapsto \mathtt{in}\, a.P$ becomes $\mathtt{in}\colon v \otimes t \to t$.
- Ex 2: $a[P \mid Q]\colon v \otimes t \otimes t \to t$.
- Second-order transition: label $\lambda Q.(a[Q] \mid \Box)$.
- Expressible as:

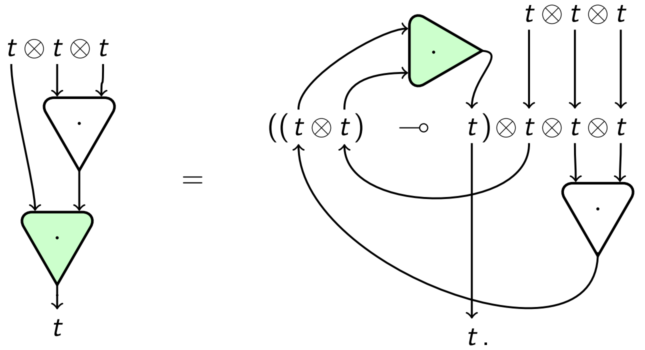


# A curiosity: factorisation

## Proposition

*For any (representative of a) proof net* $A \xrightarrow{f} B$ *in* $\mathcal{S}(\mathcal{T})$ *with a set* $C$ *of cells, and any partition of* $C$ *into* $C_1$ *and* $C_2$, $f$ *decomposes as* $f_2 \circ f_1$, *where each* $f_i$ *contains exactly the cells in* $C_i$.

Example:



# Bigraphs in one slide

- A bigraphical signature is an SMC signature $\Sigma$ containing at least:
  - two sorts $t$ and $v$,
  - operations

$$t \otimes t \xrightarrow{\quad \mid \quad} t \qquad I \xrightarrow{\quad 0 \quad} t$$

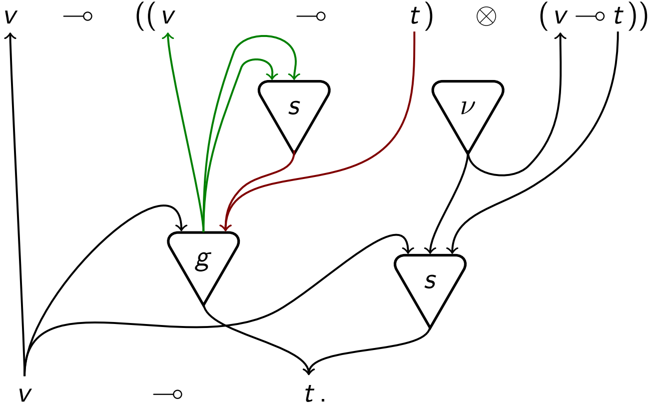
$$v \xrightarrow{\quad c \quad} v \otimes v \qquad v \xrightarrow{\quad w \quad} I$$

$$I \xrightarrow{\quad \nu \quad} v.$$

- The corresponding category of abstract bigraphs is $\mathcal{S}(\mathcal{T}_\Sigma)$, where $\mathcal{T}_\Sigma$ extends $\Sigma$ with
  - equations making $t$ into a commutative monoid,
  - equations making $v$ into a commutative comonoid, and
  - an equation for $\nu a.0 \cong 0$.

# Optimising the representation

E.g., with $g\colon v \otimes (v \multimap t) \to t$ and $s\colon v \otimes v \otimes t \to t$:



# Corrections

Of course, that is not exactly true nor fair, we:

- Forget the precategory stuff.
- Relax from 2nd order signatures.
- Add objects ($t \multimap t$), but identify others (names vs. indices).
- Relax the scoping discipline.

Garner, Hirschowitz, Pardon



# Result 1

But:

- for any bigraphical signature $\Sigma$,
- letting $\mathcal{M}(\Sigma)$ denote Jensen-Milner's category of bigraphs:

**Theorem**

*There is a faithful, essentially injective on objects functor* $\mathsf{T}\colon \mathcal{M}(\Sigma) \longrightarrow \mathcal{S}(\mathcal{T}_\Sigma)$.

- Which is neither full, nor surjective on objects (see corrections above).

# Result 2

And the overall scoping discipline is maintained:

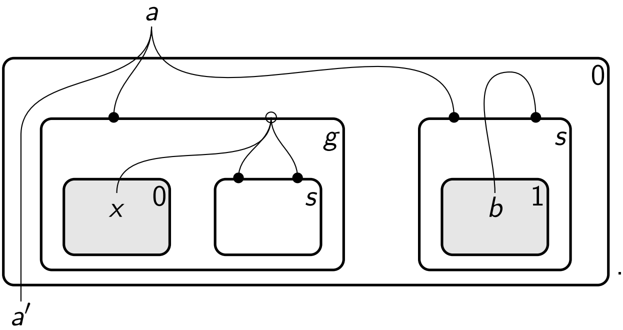
## Theorem

T *is full on whole programs, i.e., ground bigraphs.*

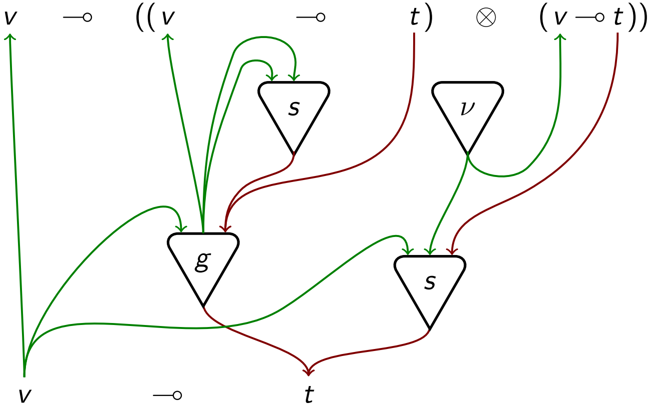
Technically, for any object $U \in \mathcal{M}(\Sigma)$, we have

$$\mathcal{S}(\mathcal{T}_\Sigma)(I, \mathsf{T}(U)) \cong \mathcal{M}(\Sigma)(I, U).$$

# Illustration



# Illustration




(place graph vs. link graph).

## Conclusion

- SMC theories adequately model syntax with variable binding.
- Let's move on to the dynamics.

Thanks for your attention.