# Labels from Reductions

## the theory of relative pushouts

Vladimiro Sassone

Department of Informatics
University of Sussex

CALCO 2005, Swansea
3 September 2005

# Motivations

Reductions model computing systems by expressing their evolution as internal 'reactions' between components:

Church's beta-reduction rule:

$$(\lambda x.M)N \longrightarrow M\{x := N\}$$

Milner's mechanics of $\pi$-calculus interaction:

$$\overline{a}\langle n\rangle.P \mid a(x).Q \longrightarrow P \mid Q\{x := n\}$$

- Pros: expressiveness, intuitiveness, conciseness, ...
- Cons: contextual semantics, lack of compositionality, ...

# Motivations

Reductions model computing systems by expressing their evolution as internal 'reactions' between components:

Church's beta-reduction rule:

$$(\lambda x.M)N \longrightarrow M\{x := N\}$$

Milner's mechanics of $\pi$-calculus interaction:

$$\overline{a}\langle n\rangle.P \mid a(x).Q \longrightarrow P \mid Q\{x := n\}$$

- Pros: expressiveness, intuitiveness, conciseness, ...
- Cons: contextual semantics, lack of compositionality, ...

# The role of labels

Labelled transitions systems model evolution indirectly, splitting actions in atomic components.

$\overline{a}\langle n\rangle.P \xrightarrow{\overline{a}n} P$: ready to evolve to $P$ by engaging in action $\overline{a}$;

$a(x).Q \xrightarrow{a(n)} Q\{x := n\}$: dual rule for partners in interactions;

Finally, an inference rule dictates

$$\frac{A \xrightarrow{\overline{a}n} A' \qquad B \xrightarrow{a(n)} B'}{A \mid B \longrightarrow A' \mid B'}$$

Advantages:

- SOS rules & compositionality
- Bisimulation / coinduction techniques
- Congruence properties well studied

# The role of labels

Labelled transitions systems model evolution indirectly, splitting actions in atomic components.

$\overline{a}\langle n\rangle.P \xrightarrow{\overline{a}n} P$: ready to evolve to $P$ by engaging in action $\overline{a}$;

$a(x).Q \xrightarrow{a(n)} Q\{x := n\}$: dual rule for partners in interactions;

Finally, an inference rule dictates

$$\frac{A \xrightarrow{\overline{a}n} A' \qquad B \xrightarrow{a(n)} B'}{A \mid B \longrightarrow A' \mid B'}$$

Advantages:

- SOS rules & compositionality
- Bisimulation / coinduction techniques
- Congruence properties well studied

# Labels from reductions: Objectives

1. Bridge the world of reductions and that of labels
2. Equip rewrite systems with compositional semantics, systematically

Technically: from a reduction system $\longrightarrow$, extract a LTS $\xrightarrow{lbl}$ such that:

1. $a \longrightarrow b$ *iff* $a \xrightarrow{} b$
2. *bisimulation on LTS is a congruence for all contexts.*

This talk: Not overly technical, but not simply a pointer to the literature.
I'd like to illustrate specific results, simply, and point to open questions.

Credits to:

Peter Sewell;
Robin Milner, James Leifer, Øle Jensen;
Pawel Sobociński, Bartek Klin

# Labels from reductions: Objectives

1. Bridge the world of reductions and that of labels
2. Equip rewrite systems with compositional semantics, systematically

Technically: from a reduction system $\longrightarrow$, extract a LTS $\xrightarrow{lbl}$ such that:

1. $a \longrightarrow b$ *iff* $a \xrightarrow{} b$
2. *bisimulation on LTS is a congruence for all contexts.*

This talk: Not overly technical, but not simply a pointer to the literature.
I'd like to illustrate specific results, simply, and point to open questions.

Credits to:

Peter Sewell;
Robin Milner, James Leifer, Øle Jensen;
Pawel Sobociński, Bartek Klin

# Labels from reductions: Objectives

1. Bridge the world of reductions and that of labels
2. Equip rewrite systems with compositional semantics, systematically

Technically: from a reduction system $\longrightarrow$, extract a LTS $\xrightarrow{lbl}$ such that:

1. $a \longrightarrow b$ *iff* $a \xrightarrow{} b$
2. *bisimulation on LTS is a congruence for all contexts.*

This talk: Not overly technical, but not simply a pointer to the literature.
I'd like to illustrate specific results, simply, and point to open questions.

Credits to:

Peter Sewell;

Robin Milner, James Leifer, Øle Jensen;

Pawel Sobociński, Bartek Klin

# Labels from reductions: Objectives

1. Bridge the world of reductions and that of labels
2. Equip rewrite systems with compositional semantics, systematically

Technically: from a reduction system $\longrightarrow$, extract a LTS $\xrightarrow{lbl}$ such that:

1. $a \longrightarrow b$ *iff* $a \xrightarrow{} b$
2. *bisimulation on LTS is a congruence for all contexts.*

This talk: Not overly technical, but not simply a pointer to the literature.
I'd like to illustrate specific results, simply, and point to open questions.

Credits to:

Peter Sewell;

Robin Milner, James Leifer, Øle Jensen;

Pawel Sobociński, Bartek Klin

# Labels from reductions: Objectives

1. Bridge the world of reductions and that of labels
2. Equip rewrite systems with compositional semantics, systematically

Technically: from a reduction system $\longrightarrow$, extract a LTS $\xrightarrow{lbl}$ such that:

1. $a \longrightarrow b$ *iff* $a \xrightarrow{lbl} b$
2. *bisimulation on LTS is a congruence for all contexts.*

This talk: Not overly technical, but not simply a pointer to the literature.
I'd like to illustrate specific results, simply, and point to open questions.

Credits to:

Peter Sewell;

Robin Milner, James Leifer, Øle Jensen;

Pawel Sobociński, Bartek Klin

# Plan of the talk

1. Basic Theory: Slice Pushouts
2. Finer Control: G-Categories
3. Examples: Graphs and Petri Nets
4. Towards a General Theory: Luxes

# The basic idea

Labels characterise terms by reflect the “outwards” actions:

- $\overline{a}\langle n\rangle.P$ is inert by itself, but willing to offer $a$ in the right context:

  e.g., $\mathscr{C}[-] = 1 \mid a(x).Q$

## Deriving labels

$$a \xrightarrow{c[-]} b \qquad \text{when} \qquad c[a] \longrightarrow b$$

So,

$$\overline{a}\langle n\rangle.P \xrightarrow{1|a(x).Q} P \mid Q\{x := n\}$$

- This idea was first expressed by Peter Sewell, who performed an ingenious syntactic analysis of relatively simple structures ($\Sigma$-terms).

# The basic idea

Labels characterise terms by reflect the “outwards” actions:

- $\overline{a}\langle n\rangle.P$ is inert by itself, but willing to offer $a$ in the right context:

  e.g., $\mathscr{C}[-] = 1 \mid a(x).Q$

## Deriving labels

$$a \xrightarrow{c[-]} b \qquad \text{when} \qquad c[a] \longrightarrow b$$

So,

$$\overline{a}\langle n\rangle.P \xrightarrow{1|a(x).Q} P \mid Q\{x := n\}$$

- This idea was first expressed by Peter Sewell, who performed an ingenious syntactic analysis of relatively simple structures ($\Sigma$-terms).

# The basic idea

Labels characterise terms by reflect the “outwards” actions:

- $\overline{a}\langle n\rangle.P$ is inert by itself, but willing to offer $a$ in the right context:

  e.g., $\mathscr{C}[-] = 1 \mid a(x).Q$

## Deriving labels

$$a \xrightarrow{c[-]} b \qquad \text{when} \qquad c[a] \longrightarrow b$$

So,

$$\overline{a}\langle n\rangle.P \xrightarrow{1|a(x).Q} P \mid Q\{x := n\}$$

- This idea was first expressed by Peter Sewell, who performed an ingenious syntactic analysis of relatively simple structures ($\Sigma$-terms).

# Are there bad contexts too?

Contexts must be lean! They must be:

- Small (no junk). Redundant labels mean wasting work:

$$\lambda x.x \xrightarrow{(-)y} y, \qquad \text{but not} \qquad \lambda x.x \xrightarrow{(-)yz} yz,$$

as $(-)yz$ arises as the composition of $(-)z$ and $(-)y$.

- Relevant. Irrelevant labels clutter the bisimulation:

terms: $P ::= 0 \mid a \mid \overline{a} \mid P \mid P$ rule: $a \mid \overline{a} \longrightarrow 0$

Then $\mathscr{C}[a \mid \overline{a}] \longrightarrow \mathscr{C}[0]$, for all $\mathscr{C}[-]$ and $a$. So, $a \mid \overline{a} \sim b \mid \overline{b} \sim \ldots$

# Are there bad contexts too?

Contexts must be lean! They must be:

- Small (no junk). Redundant labels mean wasting work:

$$\lambda x.x \xrightarrow{(-)y} y, \qquad \text{but not} \qquad \lambda x.x \xrightarrow{(-)yz} yz,$$

as $(-)yz$ arises as the composition of $(-)z$ and $(-)y$.

- Relevant. Irrelevant labels clutter the bisimulation:

terms: $P ::= 0 \mid a \mid \overline{a} \mid P \mid P$ rule: $a \mid \overline{a} \longrightarrow 0$

Then $\mathscr{C}[a \mid \overline{a}] \longrightarrow \mathscr{C}[0]$, for all $\mathscr{C}[-]$ and $a$. So, $a \mid \overline{a} \sim b \mid \overline{b} \sim \ldots$

# Are there bad contexts too?

Contexts must be lean! They must be:

- Small (no junk). Redundant labels mean wasting work:

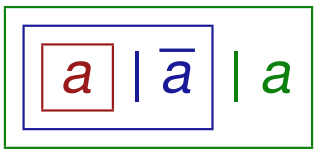
$$\lambda x.x \xrightarrow{(-)y} y, \qquad \text{but not} \qquad \lambda x.x \xrightarrow{(-)yz} yz,$$

as $(-)yz$ arises as the composition of $(-)z$ and $(-)y$.

- Relevant. Irrelevant labels clutter the bisimulation:

terms: $P ::= 0 \mid a \mid \overline{a} \mid P \mid P$ rule: $a \mid \overline{a} \longrightarrow 0$

Then $\mathscr{C}[a \mid \overline{a}] \longrightarrow \mathscr{C}[0]$, for all $\mathscr{C}[-]$ and $a$. So, $a \mid \overline{a} \sim b \mid \overline{b} \sim \ldots$



# Formalising ‘smallness’ and ‘relevance’

## Breakthrough (Leifer & Milner)

A context $c[-]$ is the ‘smallest’ to create redex $l$ in $a$, if all contexts $c'[-]$ that create $l$ in $a$ factor as $c'[-] = e[c[-]]$, for a unique context $e[-]$.

Rephrase ‘smallness’ as a problem of unique arrow factorisation in categories!




Can express minimality of $c$ as existence of a universal factorisation of all equations of the kind $c'\,a = d'\,l$.

# Formalising ‘smallness’ and ‘relevance’

## Breakthrough (Leifer & Milner)

A context $c[-]$ is the ‘smallest’ to create redex $l$ in $a$, if all contexts $c'[-]$ that create $l$ in $a$ factor as $c'[-] = e[c[-]]$, for a unique context $e[-]$.

Rephrase ‘smallness’ as a problem of unique arrow factorisation in categories!

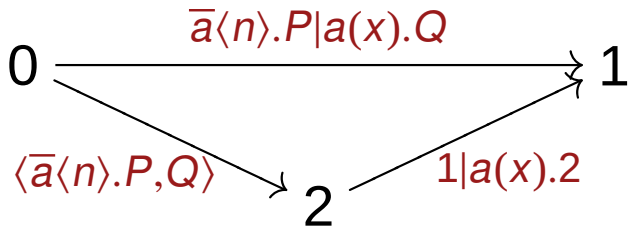



Can express minimality of $c$ as existence of a universal factorisation of all equations of the kind $c'\,a = d'\,l$.

# A categorical framework

## PROP category (MacLane)

Arrow $f : \underline{n} \to \underline{m}$: a $m$-tuple of contexts containing altogether $n$ 'holes.'

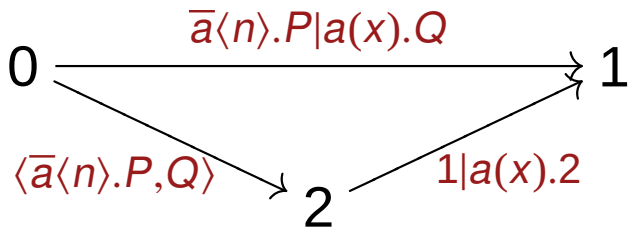



Term $a$ in context $c$ manifests a redex $l$ if there exists a suitable (e.g., reactive) context $d$ such that $ca = dl$.

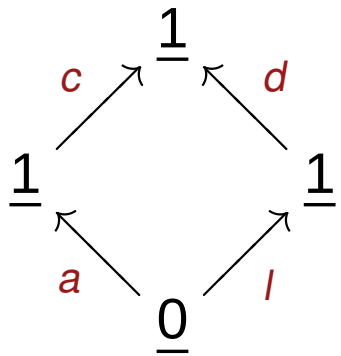
# A categorical framework

## PROP category (MacLane)

Arrow $f : \underline{n} \to \underline{m}$: a $m$-tuple of contexts containing altogether $n$ 'holes.'




Term $a$ in context $c$ manifests a redex $l$ if there exists a suitable (e.g., reactive) context $d$ such that $ca = dl$.



# Universal constructions and minimality

## Breakthrough (Leifer & Milner)

A context $c[-]$ is the 'smallest' to create redex $l$ in $a$, if all contexts $c'[-]$ that create $l$ in $a$ factor as $c'[-] = e[c[-]]$, for a unique context $e[-]$.

Rephrase 'smallness' as a problem of unique arrow factorisation in categories!



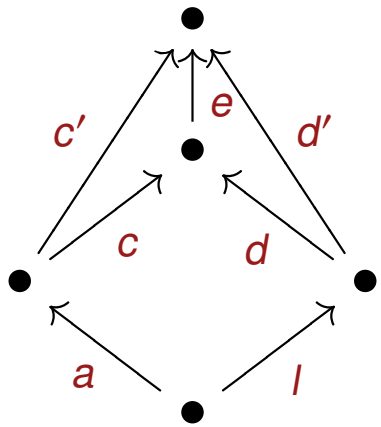

Can express minimality of $c$ as existence of a universal factorisation of all equations of the kind $c'\,a = d'\,l$.

# Universal constructions and minimality

## Breakthrough (Leifer & Milner)

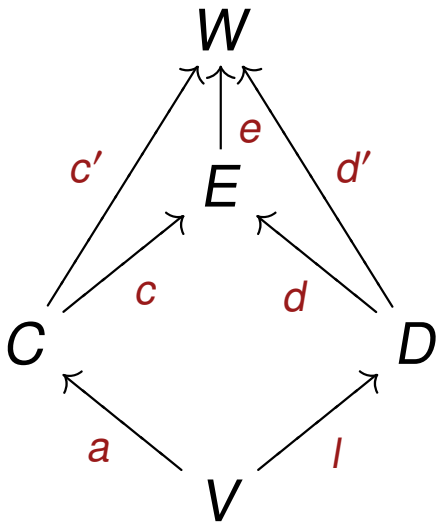
A context $c[-]$ is the 'smallest' to create redex $l$ in $a$, if all contexts $c'[-]$ that create $l$ in $a$ factor as $c'[-] = e[c[-]]$, for a unique context $e[-]$.

Rephrase 'smallness' as a problem of unique arrow factorisation in categories!




Can express minimality of $c$ as existence of a universal factorisation of all equations of the kind $c'a = d'l$.

# Slice pushouts




## Slice Pushout (a.k.a. Relative Pushout)

These properties spell out that the diagram above is a so-called pushout in the slice category $\mathbf{C}/W$, a slice pushout for short.

Good labels are all and only the $c$ in some slice pushout.

# The theory of slice pushouts

A reactive system $\mathbb{C}$ is a triple $\langle \mathbf{C}, \mathbf{D}, \mathcal{R} \rangle$ where:

- $\mathbf{C}$ is a category with a distinguished object $\underline{0}$;
- $\mathbf{D}$ is a subcategory of 'reactive contexts;'
- $\mathcal{R}$ is a set of reaction rules $\langle l, r \rangle$, for $l, r \in \mathbf{C}(\underline{0}, W)$.

The reaction relation is defined as

$$a \longrightarrow b \quad \text{iff} \quad a = dl,\ b = dr, \text{ for some } \langle l, r \rangle \in \mathcal{R} \text{ and } d \in \mathbf{D}$$

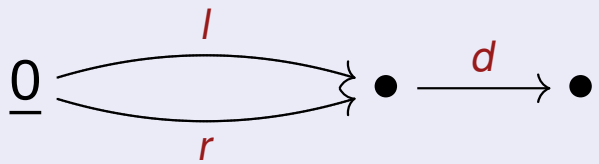
# The theory of slice pushouts

A reactive system $\mathbb{C}$ is a triple $\langle \mathbf{C}, \mathbf{D}, \mathcal{R} \rangle$ where:

- $\mathbf{C}$ is a category with a distinguished object $\underline{0}$;
- $\mathbf{D}$ is a subcategory of 'reactive contexts;'
- $\mathcal{R}$ is a set of reaction rules $\langle l, r \rangle$, for $l, r \in \mathbf{C}(\underline{0}, W)$.

The reaction relation is defined as

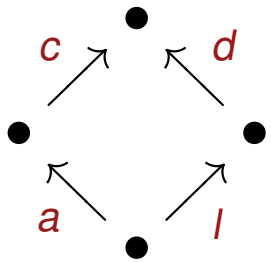
$$a \longrightarrow b \quad \text{iff} \quad a = dl,\ b = dr, \text{ for some } \langle l, r \rangle \in \mathcal{R} \text{ and } d \in \mathbf{D}$$



# Deriving bisimulation congruences

$LTS(\mathbb{C})$, the LTS associated to $\mathbb{C}$ has

- Nodes: $a : \underline{0} \to W$
- Transitions: $a \xrightarrow{c} dr$ for $\langle l, r \rangle \in \mathcal{R}$, $d \in \mathbb{D}$, and a slice pushout




**Theorem**

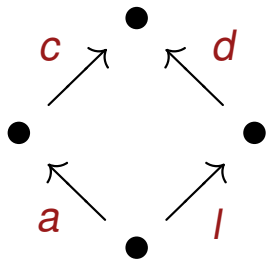
*Bisimulation on $LTS(\mathbb{C})$ is a congruence, provided* **C** *has all slice pushouts.*

Proof: simple (based on two factorisation properties of pushouts).
Showing that **C** has all relevant slice pushouts: difficult.

# Deriving bisimulation congruences

$LTS(\mathbb{C})$, the LTS associated to $\mathbb{C}$ has

- Nodes: $a : \underline{0} \to W$
- Transitions: $a \xrightarrow{c} dr$ for $\langle l, r \rangle \in \mathcal{R}$, $d \in \mathbb{D}$, and a slice pushout




**Theorem**

*Bisimulation on $LTS(\mathbb{C})$ is a congruence, provided* **C** *has all slice pushouts.*

Proof: simple (based on two factorisation properties of pushouts).
Showing that **C** has all relevant slice pushouts: difficult.

# Plan of the talk

# What's the matter with structural congruence?

This works well with 'rigid' terms, it cracks with 'fluid' ones!

Recall that context $1 \mid \overline{a}$ is never an acceptable label for $a \mid \overline{a}$.

But this is wrong for a commutative parellel composition! Because of structural congruence, $a \mid \overline{a}$ can offer $\overline{a}$ to the environment, as well as internally.




Only the left one could possibly be an SPO!

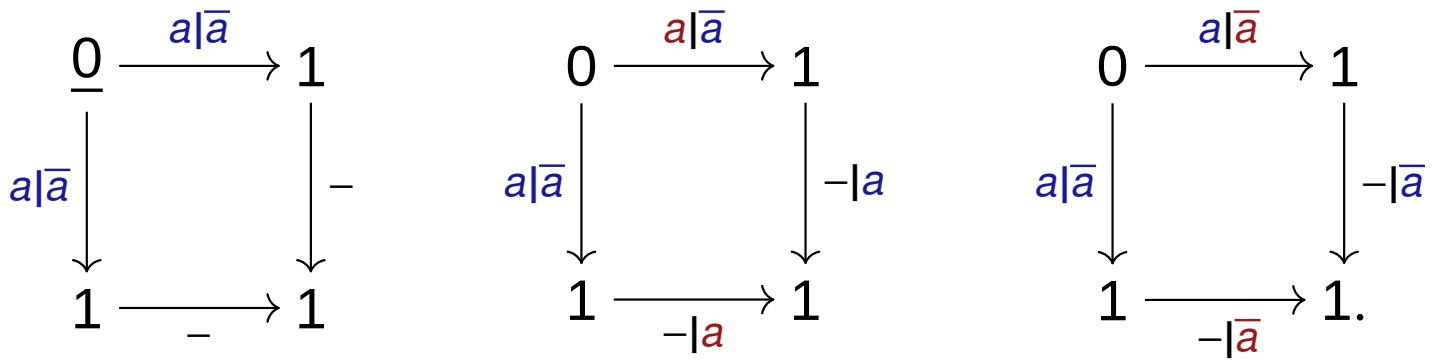
The derived LTS can only account for a $a \mid \overline{a} \longrightarrow 0$ transition.
Beware of confusion! Must eliminate ambiguity as to where the redex is.

# What's the matter with structural congruence?

This works well with 'rigid' terms, it cracks with 'fluid' ones!

Recall that context $1 \mid \overline{a}$ is never an acceptable label for $a \mid \overline{a}$.

But this is wrong for a commutative parellel composition! Because of structural congruence, $a \mid \overline{a}$ can offer $\overline{a}$ to the environment, as well as internally.




Only the left one could possibly be an SPO!

The derived LTS can only account for a $a \mid \overline{a} \longrightarrow 0$ transition.

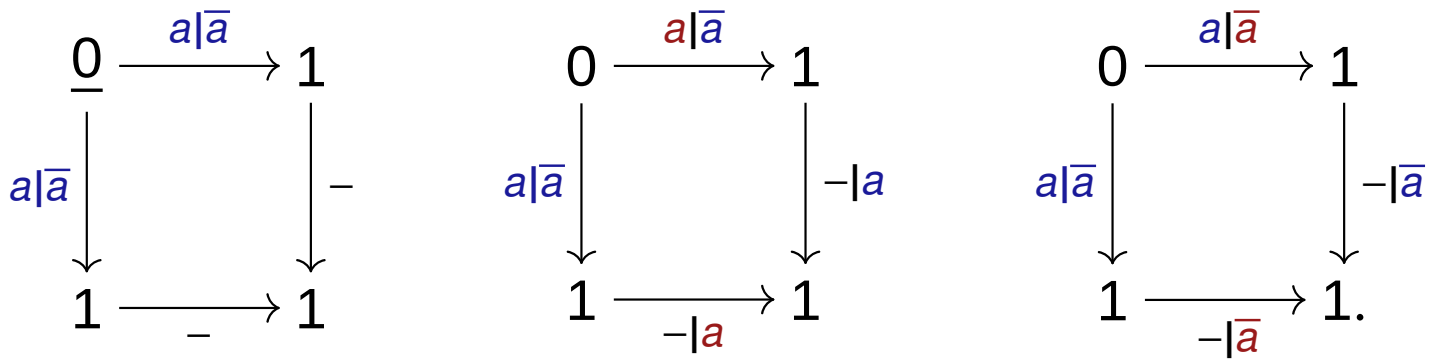
Beware of confusion! Must eliminate ambiguity as to where the redex is.

# What's the matter with structural congruence?

This works well with 'rigid' terms, it cracks with 'fluid' ones!

Recall that context $1 \mid \overline{a}$ is never an acceptable label for $a \mid \overline{a}$.

But this is wrong for a commutative parellel composition! Because of structural congruence, $a \mid \overline{a}$ can offer $\overline{a}$ to the environment, as well as internally.




Only the left one could possibly be an SPO!

The derived LTS can only account for a $a \mid \overline{a} \longrightarrow 0$ transition.

Beware of confusion! Must eliminate ambiguity as to where the redex is.

# Two ‘competing’ approaches to ‘term colouring’

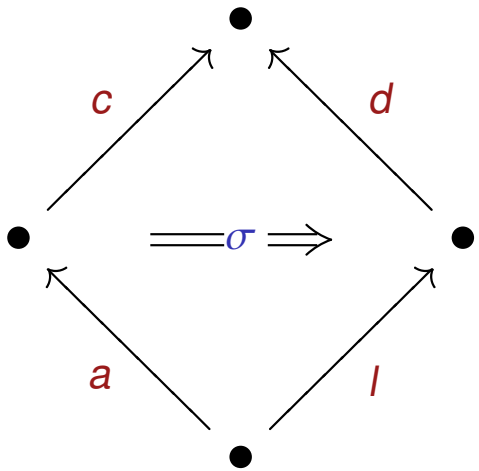
- Milner/Leifer/Jensen: Pre-categories – work with several examples, notably bigraphs, asynchronous $\pi$ and ambient calculus.
- Sassone/Sobocinski: G-categories – work with several examples, notably very general graph-like structures




Iso 2-cells are proofs of structural congruence, and ‘encode colours.’ The theory of slice pushouts lifts smoothly to G-categories in via *bipushouts* in *pseudo-slice* categories.

# Two ‘competing’ approaches to ‘term colouring’

- Milner/Leifer/Jensen: Pre-categories – work with several examples, notably bigraphs, asynchronous $\pi$ and ambient calculus.
- Sassone/Sobocinski: G-categories – work with several examples, notably very general graph-like structures




Iso 2-cells are proofs of structural congruence, and ‘encode colours.’ The theory of slice pushouts lifts smoothly to G-categories in via *bipushouts* in *pseudo-slice* categories.

# Plan of the talk

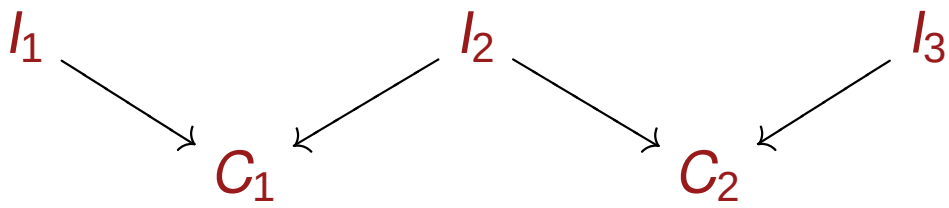
# Cospan categories

**Theorem**

*The bicategory* **Cospan(C)** *has (G-)slice pushouts, for a general class of categories* **C** *(viz., the adhesive ones).*

**Cospan(C)**:

- Objects: as in **C**;
- Arrows: $I_1 \longrightarrow C \longleftarrow I_2$, and composition: by pushout




Intuitively this gives a category of contexts over **C**: objects of **C** equipped with interfaces and composition through them.

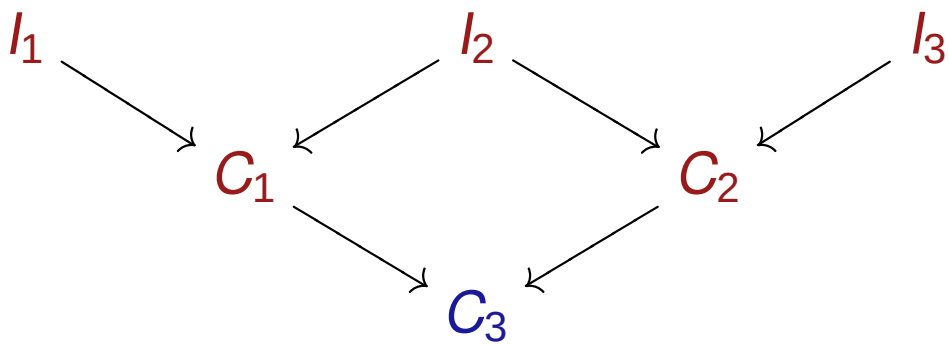
# Cospan categories

**Theorem**

*The bicategory* **Cospan(C)** *has (G-)slice pushouts, for a general class of categories* **C** *(viz., the adhesive ones).*

**Cospan(C)**:

- Objects: as in **C**;
- Arrows: $I_1 \longrightarrow C \longleftarrow I_2$, and composition: by pushout




Intuitively this gives a category of contexts over **C**: objects of **C** equipped with interfaces and composition through them.

# Graph rewriting systems

A graph rewrite system, *GR*, determines a reactive system on **Cospan(Graph)** whose reductions correspond to the standard 'double-pushout' semantics of graph rewriting.

$L \xleftarrow{l} K \xrightarrow{r} R$ translates to $\langle 0 \to L \xleftarrow{l} K, 0 \to R \xleftarrow{r} K \rangle \in \mathcal{R}$

As **Cospan(Graph)** has slice pushouts, we systematically construct a LTS *LTS(GR)* such that:

- nodes are graphs (up-to-iso) with output interfaces
- labels are smallest graph contexts (up-to-iso) that allow reaction (i.e., double-pushout rewrite)
- bisimulation on *LTS(GR)* is a congruence.

# Graph rewriting systems

A graph rewrite system, $GR$, determines a reactive system on **Cospan(Graph)** whose reductions correspond to the standard 'double-pushout' semantics of graph rewriting.

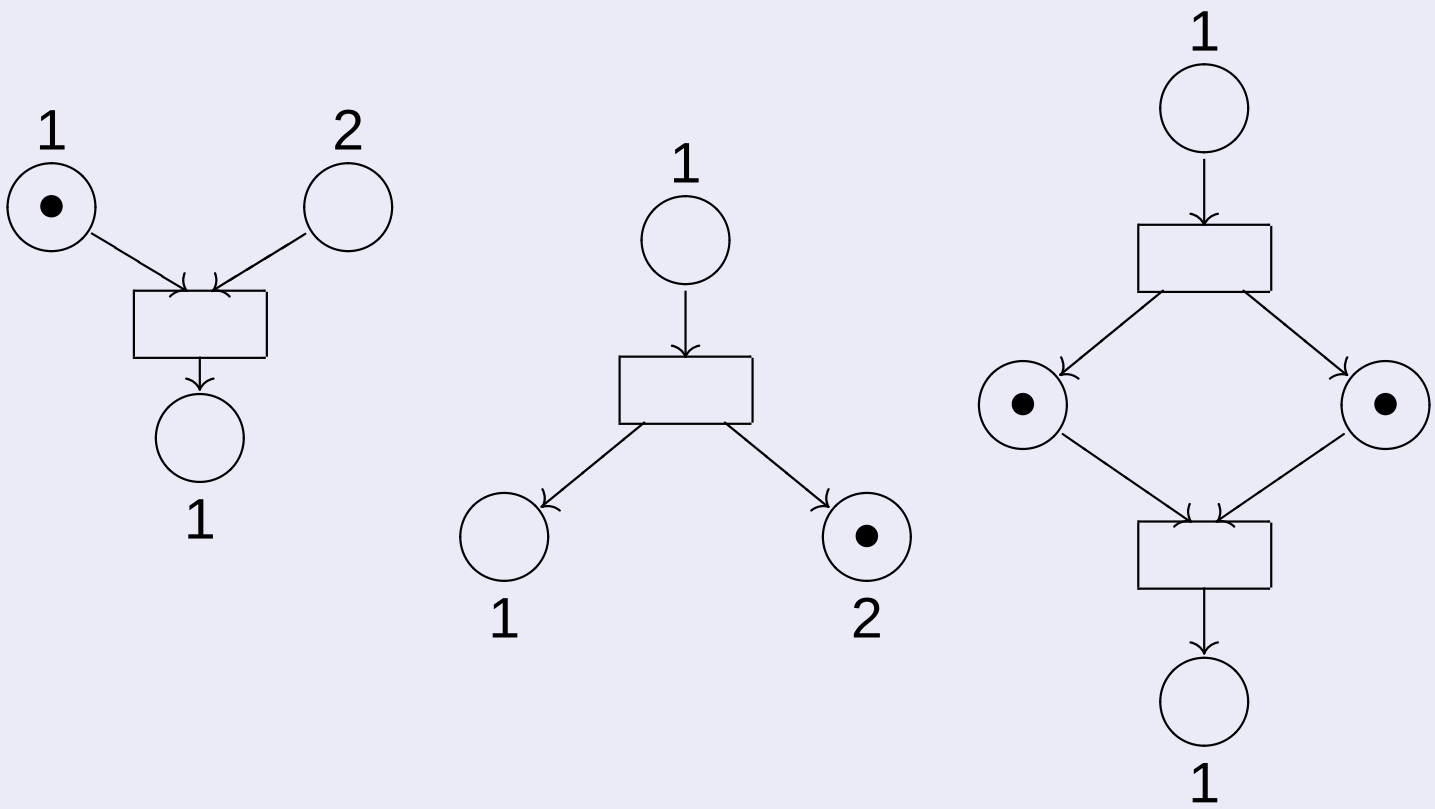
$L \xleftarrow{l} K \xrightarrow{r} R$ translates to $\langle 0 \to L \xleftarrow{l} K, 0 \to R \xleftarrow{r} K \rangle \in \mathcal{R}$

As **Cospan(Graph)** has slice pushouts, we systematically construct a LTS $LTS(GR)$ such that:

- nodes are graphs (up-to-iso) with output interfaces
- labels are smallest graph contexts (up-to-iso) that allow reaction (i.e., double-pushout rewrite)
- bisimulation on $LTS(GR)$ is a congruence.

# The case of Petri nets

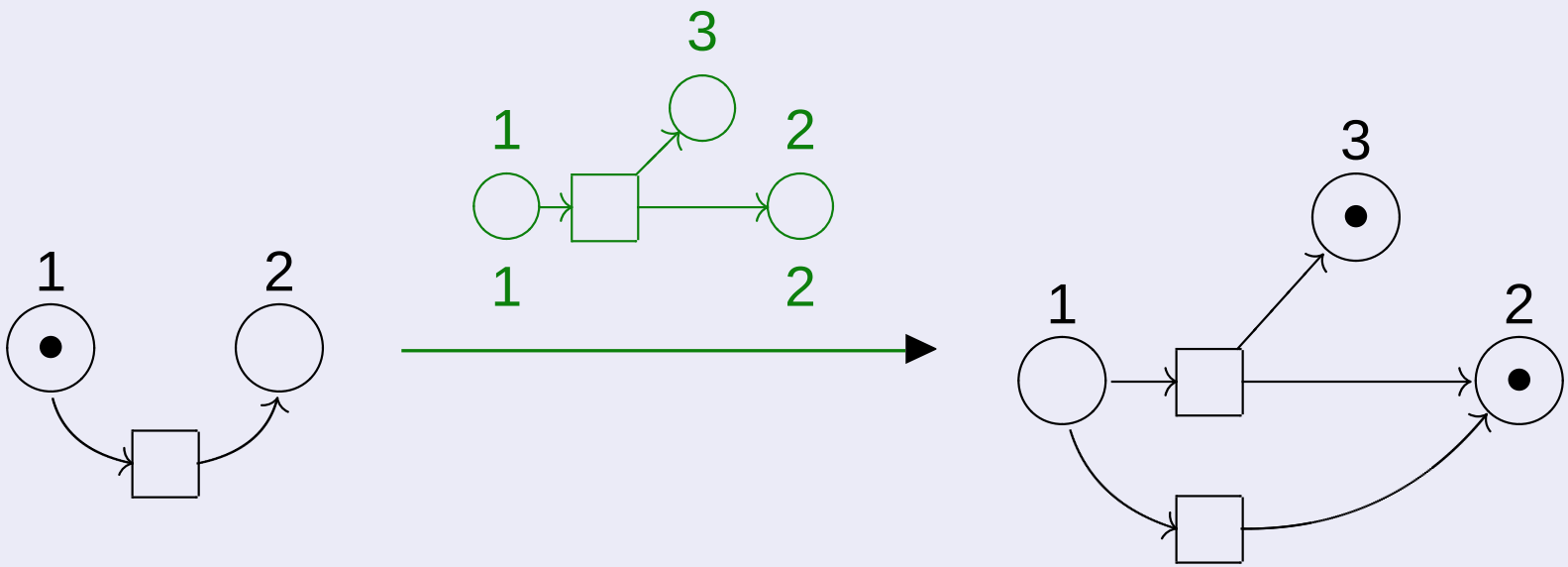
## Petri Nets with Interfaces: inner and outer faces/places



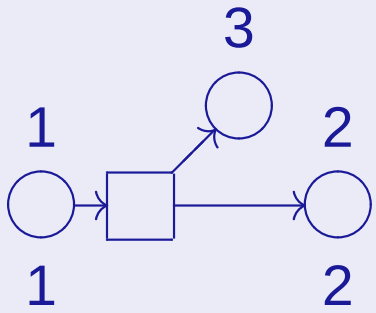

The token game can easily be decribed as a graph rewrite system and, therefore, the model fits the **Cospan(C)** framework.

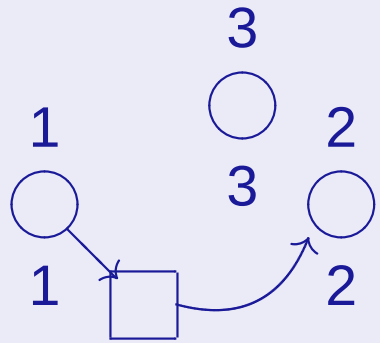
# The derived LTS

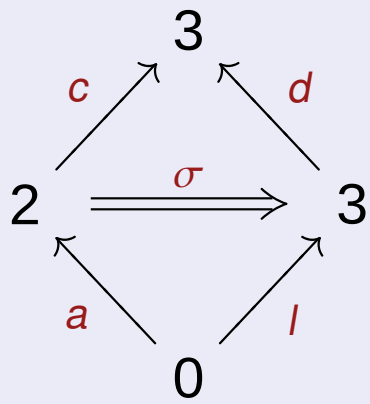
We therefore benefit ‘for free’ of a derived LTS where bisimulation is a congruence. It is instructive to look at one label.

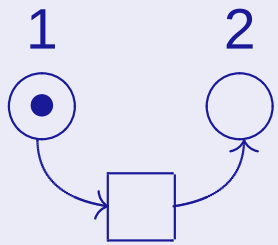


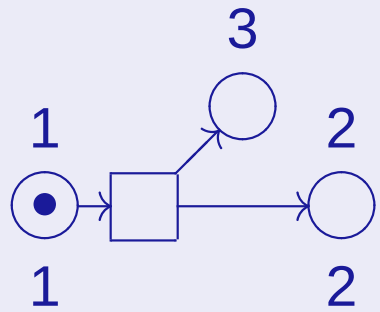
# How the slice pushout is arrived at

$c =$ 


$d =$ 


$a =$ 


$l =$ 

# A correspondence theorem

LTS $T_{\pm}$:

- Nodes: Petri terms modulo isomorphism;
- Transitions: of three types
  - $p \xrightarrow{+i} p'$ if $p'$ results from $p$ by adding a token at its $i$th outer place;
  - $p \xrightarrow{-i} p'$ if $p'$ results from $p$ by eating a token from its $i$th outer place;
  - $p \xrightarrow{\tau} p'$ if $p'$ results from $p$ through the firing of one transition.

**Theorem**

*The bisimulation on the derived LTS coincide with that on $T_{\pm}$*

Remarkable that our labels are very many and complex also in a simple case like this. Is this intrinsic? Can we do better?

# A correspondence theorem

LTS $T_{\pm}$:

- Nodes: Petri terms modulo isomorphism;
- Transitions: of three types
  - $p \xrightarrow{+i} p'$ if $p'$ results from $p$ by adding a token at its $i$th outer place;
  - $p \xrightarrow{-i} p'$ if $p'$ results from $p$ by eating a token from its $i$th outer place;
  - $p \xrightarrow{\tau} p'$ if $p'$ results from $p$ through the firing of one transition.

**Theorem**

*The bisimulation on the derived LTS coincide with that on $T_{\pm}$*

Remarkable that our labels are very many and complex also in a simple case like this. Is this intrinsic? Can we do better?

# Plan of the talk

# Open terms and parametric rules

A collection of ground rules $\overline{a}.P \mid a.Q \longrightarrow P \mid Q$ is not satisfactory:
it gives infinitely many rules, infinitely many higher-order labels.

We would like a calculus like:

$$\textit{terms}: \quad P ::= 0 \mid a.P \mid \overline{a}.P \mid P \mid P, \qquad \textit{rule}: \quad a.1 \mid \overline{a}.2 \longrightarrow 1 \mid 2$$

where the labels are derivable for open terms via universal properties imposed both on the contexts and the parameters.

$$\langle a.P, 1\rangle \xrightarrow{1\mid\overline{a}.2} P \mid 1 \qquad \text{and} \qquad a.P \mid 1 \xrightarrow[\overline{a}.1]{} P \mid 1,$$

- Labels: smallest context and the most general parameter that make a reduction possible.

# Open terms and parametric rules

A collection of ground rules $\overline{a}.P \mid a.Q \longrightarrow P \mid Q$ is not satisfactory:
it gives infinitely many rules, infinitely many higher-order labels.

We would like a calculus like:

$$\textit{terms}: \quad P ::= 0 \mid a.P \mid \overline{a}.P \mid P \mid P, \qquad \textit{rule}: \quad a.1 \mid \overline{a}.2 \longrightarrow 1 \mid 2$$

where the labels are derivable for open terms via universal properties imposed both on the contexts and the parameters.

$$\langle a.P, 1\rangle \xrightarrow{1 \mid \overline{a}.2} P \mid 1 \qquad \text{and} \qquad a.P \mid 1 \xrightarrow[\overline{a}.1]{} P \mid 1,$$

- Labels: smallest context and the most general parameter that make a reduction possible.

# Open terms and parametric rules

A collection of ground rules $\overline{a}.P \mid a.Q \longrightarrow P \mid Q$ is not satisfactory:
it gives infinitely many rules, infinitely many higher-order labels.

We would like a calculus like:

$$\textit{terms}: \quad P ::= 0 \mid a.P \mid \overline{a}.P \mid P \mid P, \qquad \textit{rule}: \quad a.1 \mid \overline{a}.2 \longrightarrow 1 \mid 2$$

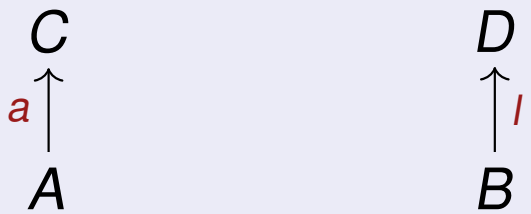
where the labels are derivable for open terms via universal properties imposed both on the contexts and the parameters.

$$\langle a.P, 1\rangle \xrightarrow{1\mid\overline{a}.2} P \mid 1 \qquad \text{and} \qquad a.P \mid 1 \xrightarrow[\overline{a}.1]{} P \mid 1,$$

- Labels: smallest context and the most general parameter that make a reduction possible.

# Locally universal hexagons

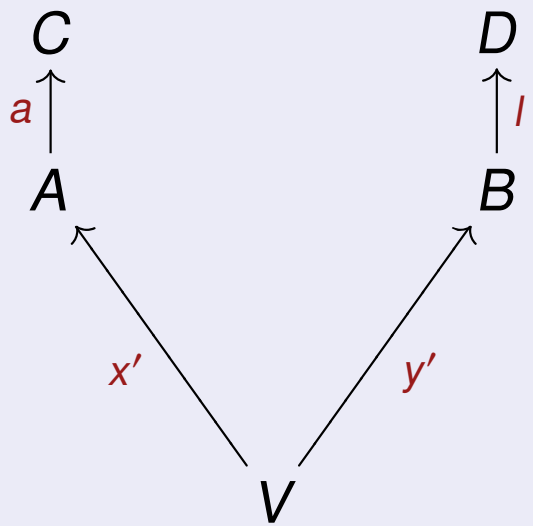
## Luxes




Hmmm!?!?

# Locally universal hexagons

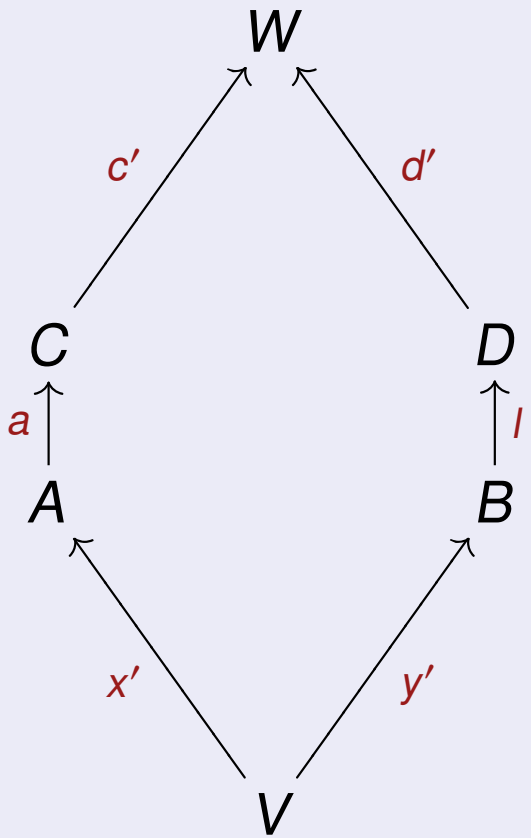
## Luxes




Get span

# Locally universal hexagons

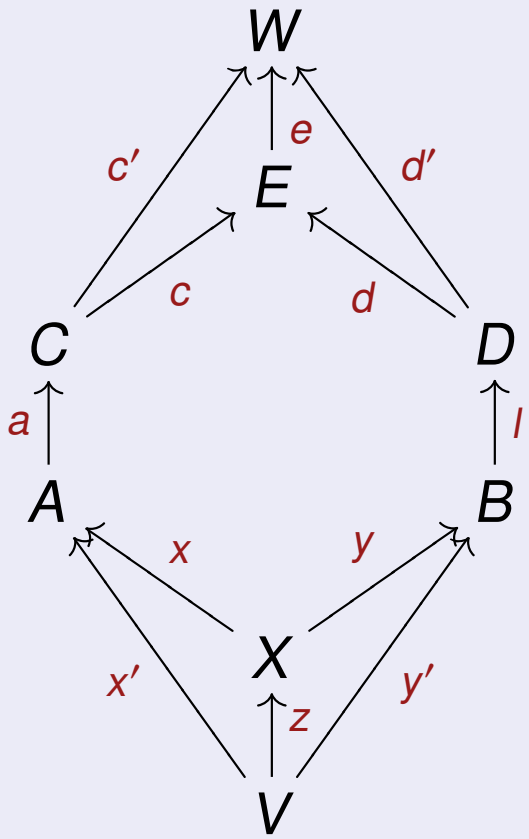
## Luxes




Get a square

# Locally universal hexagons

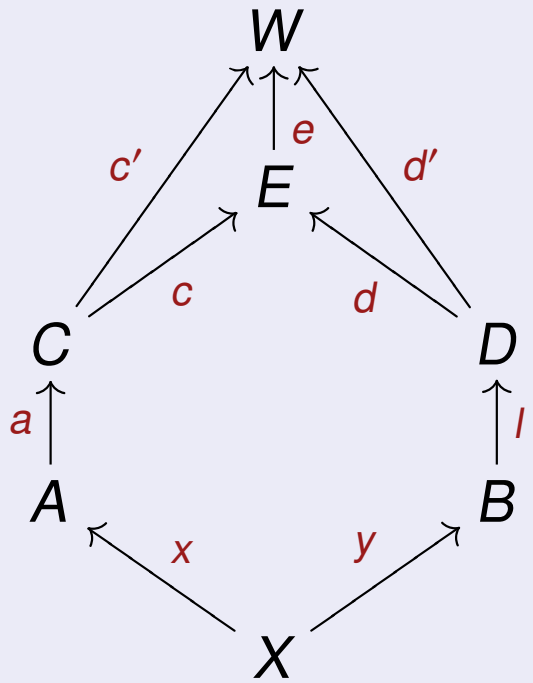
## Luxes




It's a lux!

# Locally universal hexagons

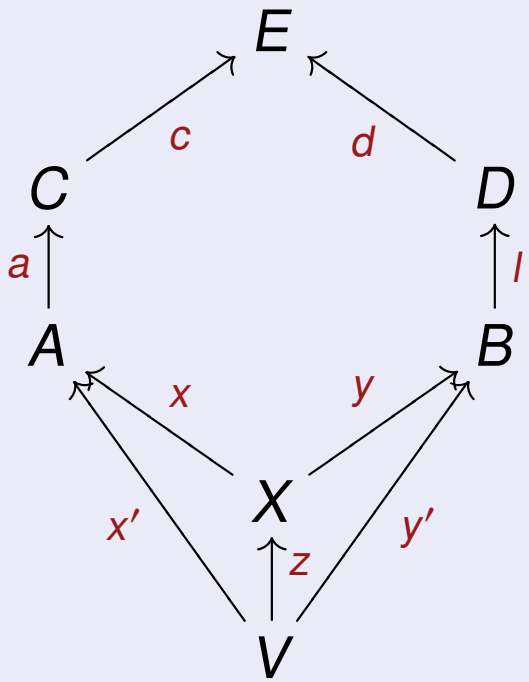
## Luxes




## A Slice Pushout?

# Locally universal hexagons

## Luxes

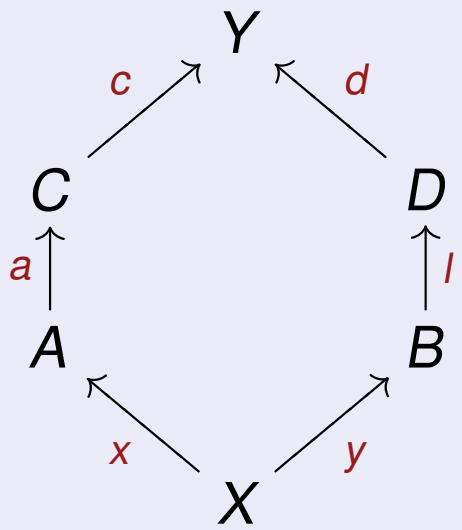



## A Coslice Pullback?

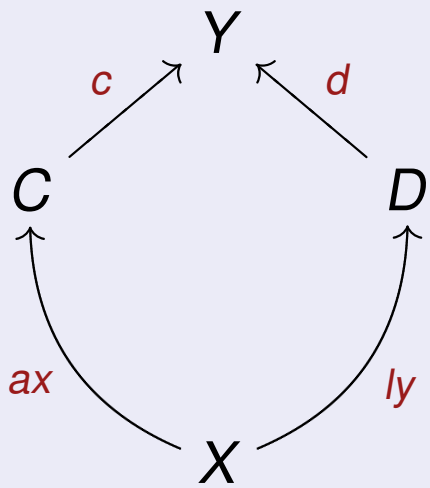
# Characterising categories with luxes

## Theorem

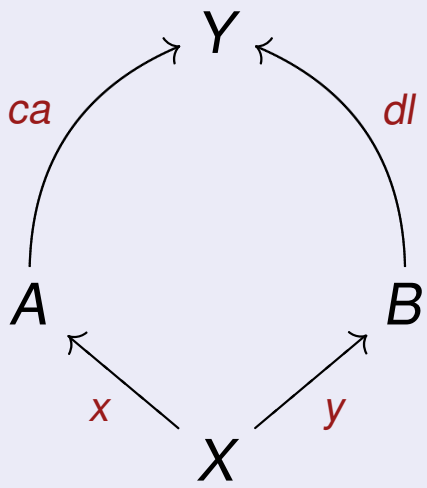
*A commutative diagram (i) is a lux iff diagram (ii) is a SPO and diagram (iii) is an CPB.*




(i)




(ii)




(iii)

# Characterising categories with luxes (ctd)

**Theorem**

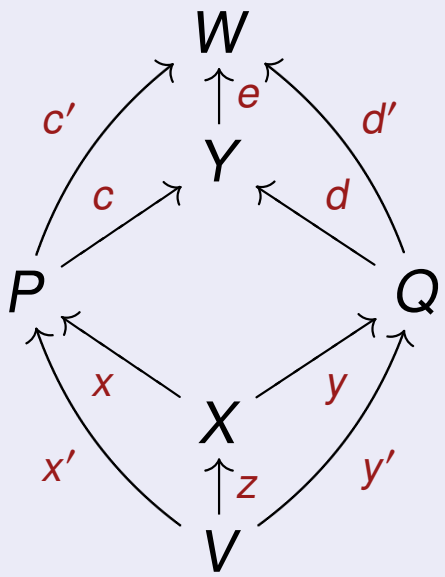
*The following conditions are both sufficient for the existence of luxes in* **C**.

1. **C** *has SPO, CPB and all arrows are mono;*
2. **C** *has SPO, CPB and all arrows are epi.*

# Characterising categories with luxes (ctd)

## Theorem

*A category has luxes if it has SPO and CPB, and these commute*



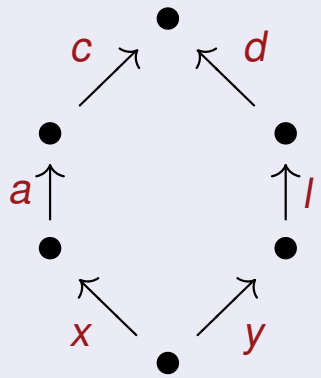

Commutativity of slice pushouts and coslice pullbacks

# Open LTS

## Definition (LTS from an Open Reactive System)

Given an open reactive system $\mathbb{C}$, $LTS(\mathbb{C})$ is:

- Nodes: arbitrary arrows $a : A \to C$ – i.e., terms are possibly open;
- Transitions: $a \xrightarrow[x]{c} b$ whenever there exist $\langle l, r \rangle \in \mathcal{R}$ and a lux




such that $b = dry$.

# The congruence theorem for luxes

**Theorem (Congruence I)**

*Let $\mathbb{C}$ be an open reactive system over* **C**.
*If* **C** *has luxes and all arrows of* **C** *are mono, then $p \sim q$ implies $cp \sim cq$ for all contexts $c$ in* **C**.

**Theorem (Congruence II)**

*If* **C** *has luxes and all arrows of* **C** *are epi, then $p \sim q$ implies $px \sim qx$ for all parameters $x$ in* **C**.

- Mono: no two different terms can be made equal by insertion in the same context.
- Epi: no two different terms can be made equal by filling with the same parameters.

# The congruence theorem for luxes

**Theorem (Congruence I)**

*Let $\mathbb{C}$ be an open reactive system over* **C**.
*If* **C** *has luxes and all arrows of* **C** *are mono, then $p \sim q$ implies $cp \sim cq$ for all contexts $c$ in* **C**.

**Theorem (Congruence II)**

*If* **C** *has luxes and all arrows of* **C** *are epi, then $p \sim q$ implies $px \sim qx$ for all parameters $x$ in* **C**.

- Mono: no two different terms can be made equal by insertion in the same context.
- Epi: no two different terms can be made equal by filling with the same parameters.

# The congruence theorem for luxes

**Theorem (Congruence I)**

*Let $\mathbb{C}$ be an open reactive system over* **C**.
*If* **C** *has luxes and all arrows of* **C** *are mono, then $p \sim q$ implies $cp \sim cq$ for all contexts $c$ in* **C**.

**Theorem (Congruence II)**

*If* **C** *has luxes and all arrows of* **C** *are epi, then $p \sim q$ implies $px \sim qx$ for all parameters $x$ in* **C**.

- Mono: no two different terms can be made equal by insertion in the same context.
- Epi: no two different terms can be made equal by filling with the same parameters.

# The parametric CCS case

**Theorem**

*All free PROPs have luxes*

Application to the example of parametric CCS nicely gives expected labels

$$\langle a.P, 1\rangle \xrightarrow{1|\overline{a}.2} P \mid 1$$

$$a.P \mid 1 \xrightarrow[\overline{a}.1]{} P \mid 1$$

$$a.P \mid \overline{a}.Q \longrightarrow P \mid Q$$

$$\langle a.P \mid \overline{a}.Q, 1\rangle \xrightarrow{1|\overline{a}.2} P \mid \overline{a}.Q \mid 1$$

But problems too: we have $a.P \xrightarrow{1|\overline{a}.Q} P \mid Q$, where $Q$ is irrelevant.

Even worse, $a.1 \xrightarrow[X]{1|a.P|\overline{a}.Q} a.X \mid P \mid Q$, where the term is not involved in the reduction.

# The parametric CCS case

**Theorem**

*All free PROPs have luxes*

Application to the example of parametric CCS nicely gives expected labels

$$\langle a.P, 1\rangle \xrightarrow{1|\overline{a}.2} P \mid 1$$

$$a.P \mid 1 \xrightarrow[\overline{a}.1]{} P \mid 1$$

$$a.P \mid \overline{a}.Q \longrightarrow P \mid Q$$

$$\langle a.P \mid \overline{a}.Q, 1\rangle \xrightarrow{1|\overline{a}.2} P \mid \overline{a}.Q \mid 1$$

But problems too: we have $a.P \xrightarrow{1|\overline{a}.Q} P \mid Q$, where $Q$ is irrelevant.

Even worse, $a.1 \xrightarrow[X]{1|a.P|\overline{a}.Q} a.X \mid P \mid Q$, where the term is not involved in the reduction.

# Irredundant luxes

- SPO and CPB take care very well of ‘smallness’ and ‘relevance’;
- But there can still be unwanted overlaps between parameters and contexts.

Refine the notion of lux to rule out overlap between SPO and CPB.

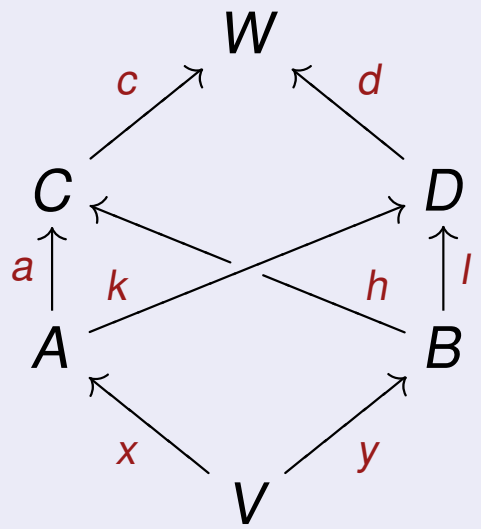



- Irredundant luxes rule out all the bad labels, and feel right.
- Alas, the congruence theorem fails: need a smarted definition of LTS?

# Irredundant luxes

- SPO and CPB take care very well of ‘smallness’ and ‘relevance’;
- But there can still be unwanted overlaps between parameters and contexts.

Refine the notion of lux to rule out overlap between SPO and CPB.

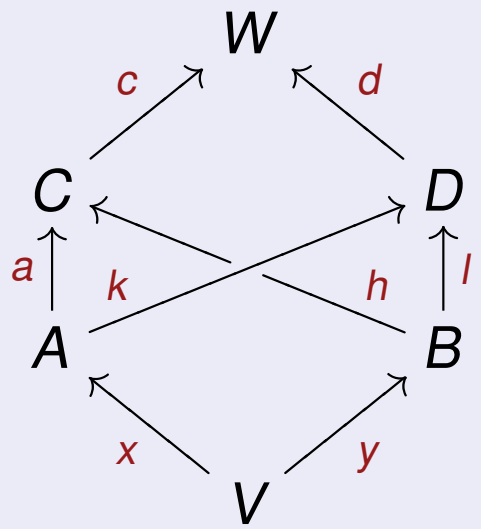



- Irredundant luxes rule out all the bad labels, and feel right.
- Alas, the congruence theorem fails: need a smarted definition of LTS?

# Irredundant luxes

- SPO and CPB take care very well of ‘smallness’ and ‘relevance’;
- But there can still be unwanted overlaps between parameters and contexts.

Refine the notion of lux to rule out overlap between SPO and CPB.




- Irredundant luxes rule out all the bad labels, and feel right.
- Alas, the congruence theorem fails: need a smarted definition of LTS?

# Summary and future work

- Stage 1: formalise ‘smallness’ and derive labels ✓
- Stage 2: understand structural congruence ✓
- Stage 3: test and adapt theory on complex structures ✓
- Stage 4: metatheorems for wide classes of models ✓, open-ended
- Stage 5: parameters and open terms ×, open problem

- Investigate new definition of LTS;
- Investigate variations of irredundance for luxes;
- Apply luxes to known cases.

# Summary and future work

- Stage 1: formalise ‘smallness’ and derive labels ✓
- Stage 2: understand structural congruence ✓
- Stage 3: test and adapt theory on complex structures ✓
- Stage 4: metatheorems for wide classes of models ✓, open-ended
- Stage 5: parameters and open terms ×, open problem

- Investigate new definition of LTS;
- Investigate variations of irredundance for luxes;
- Apply luxes to known cases.

# Summary and future work

- Stage 1: formalise ‘smallness’ and derive labels ✓
- Stage 2: understand structural congruence ✓
- Stage 3: test and adapt theory on complex structures ✓
- Stage 4: metatheorems for wide classes of models ✓, open-ended
- Stage 5: parameters and open terms ×, open problem

- Investigate new definition of LTS;
- Investigate variations of irredundance for luxes;
- Apply luxes to known cases.

# Summary and future work

- Stage 1: formalise ‘smallness’ and derive labels ✓
- Stage 2: understand structural congruence ✓
- Stage 3: test and adapt theory on complex structures ✓
- Stage 4: metatheorems for wide classes of models ✓, open-ended
- Stage 5: parameters and open terms ×, open problem

- Investigate new definition of LTS;
- Investigate variations of irredundance for luxes;
- Apply luxes to known cases.

# Summary and future work

- Stage 1: formalise ‘smallness’ and derive labels ✓
- Stage 2: understand structural congruence ✓
- Stage 3: test and adapt theory on complex structures ✓
- Stage 4: metatheorems for wide classes of models ✓, open-ended
- Stage 5: parameters and open terms ×, open problem

- Investigate new definition of LTS;
- Investigate variations of irredundance for luxes;
- Apply luxes to known cases.

# Summary and future work

- Stage 1: formalise ‘smallness’ and derive labels ✓
- Stage 2: understand structural congruence ✓
- Stage 3: test and adapt theory on complex structures ✓
- Stage 4: metatheorems for wide classes of models ✓, open-ended
- Stage 5: parameters and open terms ×, open problem

- Investigate new definition of LTS;
- Investigate variations of irredundance for luxes;
- Apply luxes to known cases.