Confluence of Graph Rewriting with Interfaces

Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski and Fabio Zanasi

IFIP WG 1.3
9-12/01/2017, Binz

Confluence of Graph Rewriting with Interfaces

Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski and Fabio Zanasi

To appear at ESOP 2017

IFIP WG 1.3
9-12/01/2017, Binz

Plan of the Talk

1) Confluence for Term Rewriting
2) Confluence for DPO Rewriting
3) Confluence for DPO Rewriting with Interfaces
4) Confluence for PROP Rewriting

Plan of the Talk

1) Confluence for Term Rewriting

2) Confluence for DPO Rewriting

3) Confluence for DPO Rewriting with Interfaces

4) Confluence for PROP Rewriting

Confluence and Termination

Confluence and Termination

Rewriting relation $\Rightarrow$

Confluence and Termination

Rewriting relation $\Longrightarrow$

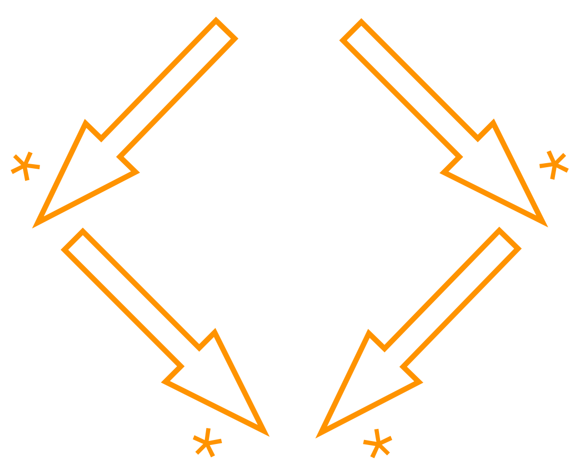
Its reflexive and transitive closure $\overset{*}{\Longrightarrow}$

Confluence and Termination

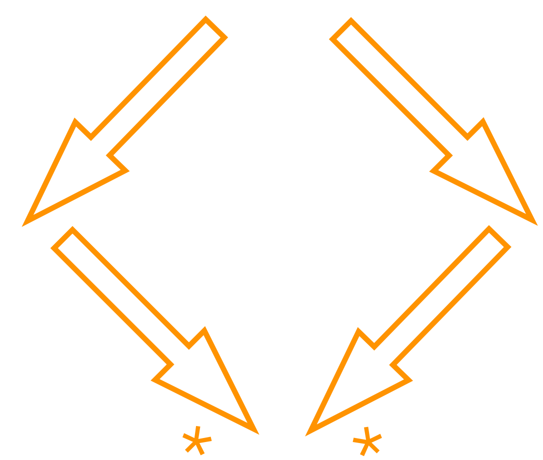
Rewriting relation $\Rightarrow$

Its reflexive and transitive closure $\overset{*}{\Rightarrow}$

Confluence



Local Confluence

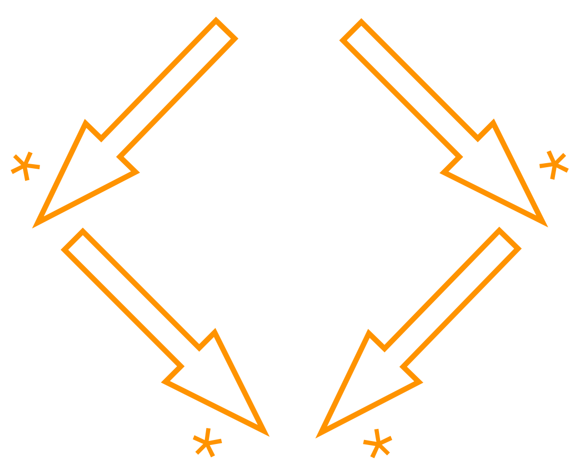


Confluence and Termination

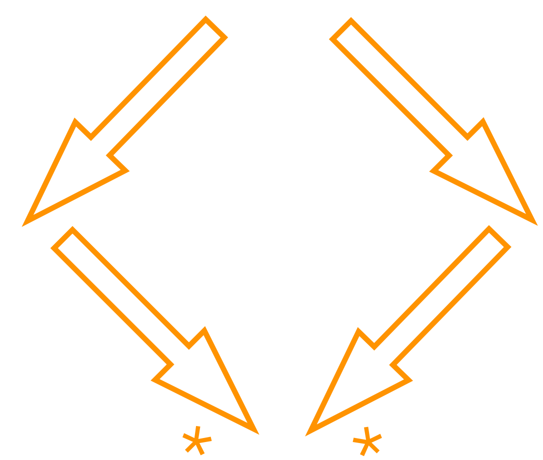
Rewriting relation $\Rightarrow$

Its reflexive and transitive closure $\overset{*}{\Rightarrow}$

Confluence



Local Confluence



Newman's Lemma

In a terminating rewriting system,
local confluence implies confluence

Confluence for Term Rewriting

Knuth-Bendix 1970

$\boxed{\text{Confluence of a terminating rewriting system is decidable}}$

Confluence for Term Rewriting

Knuth-Bendix 1970

Confluence of a terminating rewriting system is decidable

If all the critical pairs are joinable, then the system is locally confluent

Confluence for Term Rewriting

Knuth-Bendix 1970

$\boxed{\text{Confluence of a terminating rewriting system is decidable}}$

$\boxed{\begin{array}{c}\text{If all the critical pairs are joinable,} \\ \text{then the system is locally confluent}\end{array}}$

The converse implication is trivial

Confluence for Term Rewriting

Knuth-Bendix 1970

Confluence of a terminating rewriting system is decidable

If all the critical pairs are joinable,
then the system is locally confluent

The converse implication is trivial

In a terminating system,
checking joinability of critical pairs is easy

Example

Two unary symbols f,g:1-->1

$f(f(x)) \xRightarrow{(1)} f(x)$ $\quad\quad$ $f(g(x)) \xRightarrow{(2)} g(x)$

Example

Two unary symbols f,g:1-->1

$f(f(x)) \xRightarrow{(1)} f(x)$ $\qquad$ $f(g(x)) \xRightarrow{(2)} g(x)$

Trivially, the system is terminating

Example

Two unary symbols f,g:1-->1

$$f(f(x)) \xRightarrow{(1)} f(x) \qquad f(g(x)) \xRightarrow{(2)} g(x)$$

Trivially, the system is terminating

Critical pair

Example

Two unary symbols f,g:1-->1

$$f(f(x)) \xRightarrow{(1)} f(x) \qquad\qquad f(g(x)) \xRightarrow{(2)} g(x)$$

Trivially, the system is terminating

Critical pair

$$\begin{tikzcd}
 & f(f(g(x))) \arrow[dl, Rightarrow, "(1)"'] \arrow[dr, Rightarrow, "(2)"] & \\
f(g(x)) \arrow[r, Rightarrow, "*"] & {} & f(g(x)) \arrow[l, Rightarrow, "*"']
\end{tikzcd}$$

Example

Two unary symbols f,g:1-->1

$$f\,(f\,(x)\,) \xRightarrow{(1)} f\,(x) \qquad\qquad f\,(g\,(x)\,) \xRightarrow{(2)} g\,(x)$$

Trivially, the system is terminating

Critical pair

$$\begin{tikzcd}
& f\,(\,{\color{red}f}\,(g\,(x)\,)\,) \arrow[dl, Rightarrow, "(1)"'] \arrow[dr, Rightarrow, "(2)"] & \\
f\,(g\,(x)\,) \arrow[r, Rightarrow, "*"] & {} & f\,(g\,(x)\,) \arrow[l, Rightarrow, "*"']
\end{tikzcd}$$

For all terms, rewriting is confluent

Example

Two unary symbols f,g:1-->1

$$f(f(x)) \xRightarrow{(1)} f(x) \qquad\qquad f(g(x)) \xRightarrow{(2)} g(x)$$

Trivially, the system is terminating

Critical pair

$$\begin{tikzcd}
& f(f(g(x))) \arrow[dl, Rightarrow, "(1)"'] \arrow[dr, Rightarrow, "(2)"] & \\
f(g(x)) \arrow[r, Rightarrow, "*"] & {} & f(g(x)) \arrow[l, Rightarrow, "*"']
\end{tikzcd}$$

For all terms, rewriting is confluent

Every term has a unique normal form

Ground Confluence

Two unary symbols f,g:1-->1 and one constant c:0-->1

$f(g(f(x))) \Rightarrow x \qquad f(c) \Rightarrow c \qquad g(c) \Rightarrow c$

Ground Confluence

Two unary symbols f,g:1-->1 and one constant c:0-->1

$f(g(f(x))) \Rightarrow x \qquad f(c) \Rightarrow c \qquad g(c) \Rightarrow c$

The following critical pair is not joinable

Ground Confluence

Two unary symbols f,g:1-->1 and one constant c:0-->1

$f(g(f(x))) \Rightarrow x \qquad f(c) \Rightarrow c \qquad g(c) \Rightarrow c$

The following critical pair is not joinable

$f(g(f(g(f(x)))))$

$g(f(x)) \qquad\qquad f(g(x))$

Ground Confluence

Two unary symbols f,g:1-->1 and one constant c:0-->1

$f(g(f(x))) \Rightarrow x \qquad f(c) \Rightarrow c \qquad g(c) \Rightarrow c$

The following critical pair is not joinable

$$f(g(f(g(f(x)))))$$

$g(f(x)) \qquad\qquad f(g(x))$

So, the system is not confluent

Ground Confluence

Two unary symbols f,g:1-->1 and one constant c:0-->1

$f(g(f(x))) \Rightarrow x \qquad f(c) \Rightarrow c \qquad g(c) \Rightarrow c$

The following critical pair is not joinable

$$\begin{tikzcd}[column sep=small] & f(g(f(g(f(x))))) \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ g(f(x)) & & f(g(x)) \end{tikzcd}$$

So, the system is not confluent

But it is ground confluent:

Ground Confluence

Two unary symbols f,g:1-->1 and one constant c:0-->1

$f(g(f(x))) \Rightarrow x \qquad f(c) \Rightarrow c \qquad g(c) \Rightarrow c$

The following critical pair is not joinable

$f(g(f(g(f(x)))))$

$g(f(x)) \qquad\qquad f(g(x))$

So, the system is not confluent

But it is ground confluent:

it is confluent for all the ground terms

Ground Confluence

Kapur et al. 1990

For a terminating Term Rewriting System, ground confluence is not decidable

So, the system is not confluent

But it is ground confluent:

it is confluent for all the ground terms

Ground Confluence

Kapur et al. 1990

$$\boxed{\text{For a terminating Term Rewriting System, ground confluence is not decidable}}$$

So, the system is not confluent

But it is ground confluent:

it is confluent for all the ground terms

The trivial implication is not trivial anymore

Plan of the Talk

1) Confluence for Term Rewriting

2) Confluence for DPO Rewriting

3) Confluence for DPO Rewriting with Interfaces

4) Confluence for PROP Rewriting

Graph Rewriting (the DPO approach)

We work in an arbitrary adhesive category,
typically the category of hypergraphs and their morphisms

Graph Rewriting (the DPO approach)

We work in an arbitrary adhesive category, typically the category of hypergraphs and their morphisms

A rewriting rule is a span

$$L \longleftarrow K \longrightarrow R$$

Graph Rewriting
(the DPO approach)

We work in an arbitrary adhesive category,
typically the category of hypergraphs and their morphisms

A rewriting rule is a span

$$L \longleftarrow K \longrightarrow R$$

A rewriting step is a commuting diagram

Graph Rewriting (the DPO approach)

We work in an arbitrary adhesive category,
typically the category of hypergraphs and their morphisms

A rewriting rule is a span

$$\begin{tikzcd} L & K \arrow[l] \arrow[r] & R \end{tikzcd}$$

A rewriting step is a commuting diagram

$$\begin{tikzcd} L \arrow[d] \arrow[dr, phantom, "\urcorner"] & K \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner"] & R \arrow[d] \\ G & C \arrow[l] \arrow[r] & H \end{tikzcd}$$

Graph Rewriting
(the DPO approach)

We work in an arbitrary adhesive category,
typically the category of hypergraphs and their morphisms

A rewriting rule is a span

$$L \longleftarrow K \longrightarrow R$$

A rewriting step is a commuting diagram

$$\begin{tikzcd}
L & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G \arrow[from=1-1] \arrow[ur, phantom, "\urcorner", very near start] & C \arrow[l] \arrow[r] & H \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

where the two squares are pushouts

Graph Rewriting (the DPO approach)

We work in an arbitrary adhesive category,
typically the category of hypergraphs and their morphisms

A rewriting rule is a span

$$L \leftarrow K \rightarrow R$$

A rewriting step is a commuting diagram

$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G \arrow[ur, phantom, "\urcorner", very near start] & C \arrow[l] \arrow[r] & H \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

where the two squares are pushouts

$$G \Rightarrow H$$

Confluence for DPO rewriting

Plump 1993

For a terminating DPO Rewriting System, confluence is not decidable

Confluence for DPO rewriting

Plump 1993

For a terminating DPO Rewriting System, confluence is not decidable

Critical pair analysis is useless:
joinability of critical pairs does not entail local confluence

Confluence for DPO rewriting

Plump 1993

For a terminating DPO Rewriting System, confluence is not decidable

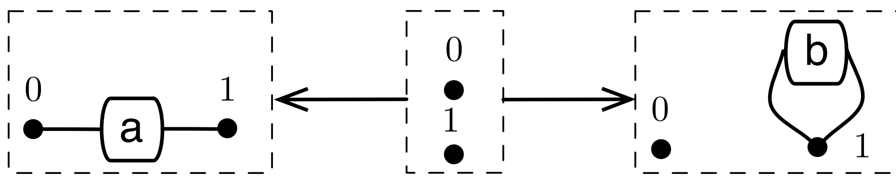
Critical pair analysis is useless:
joinability of critical pairs does not entail local confluence

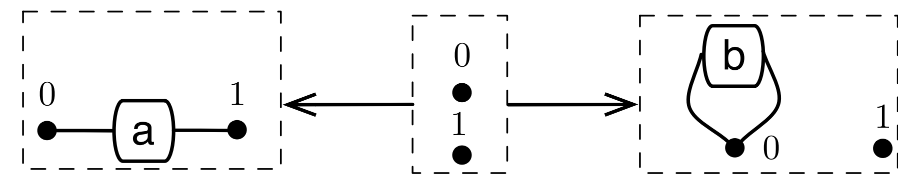
Critical Pairs

$$
\begin{tikzcd}
R_1 \arrow[d] & K_1 \arrow[l] \arrow[r] \arrow[d] & L_1 \arrow[dr, "f_1"] & & L_2 \arrow[dl, "f_2"'] & K_2 \arrow[l] \arrow[r] \arrow[d] & R_2 \arrow[d] \\
H_1 & C_1 \arrow[l] \arrow[rr] & & G & & C_2 \arrow[ll] \arrow[r] & H_2
\end{tikzcd}
$$

f_1 and f_2 are jointly epi

Counter-example



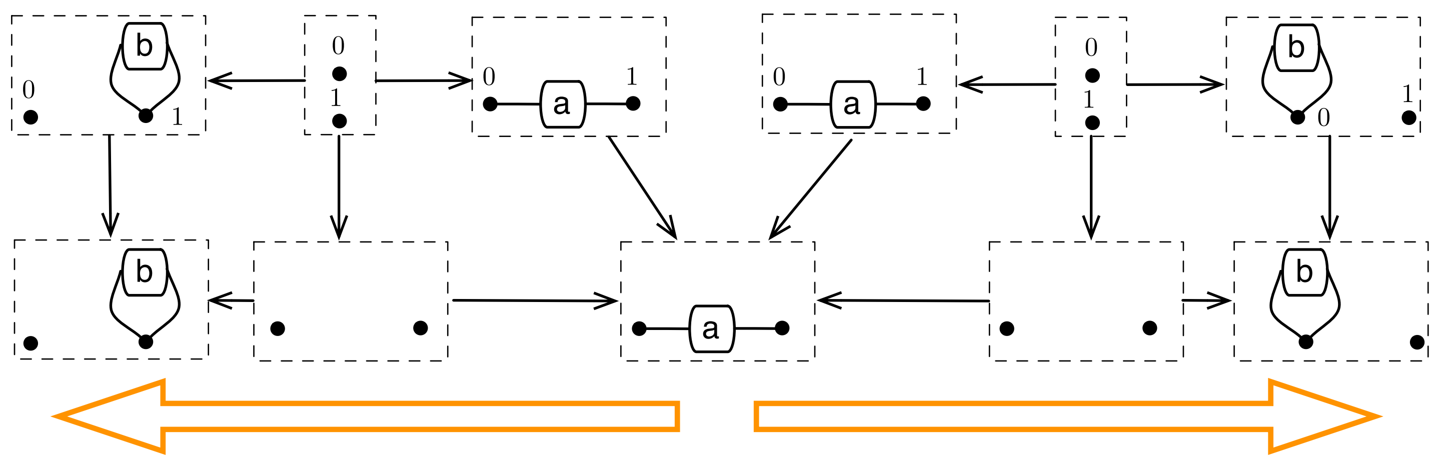


Counter-example

Only two critical pairs

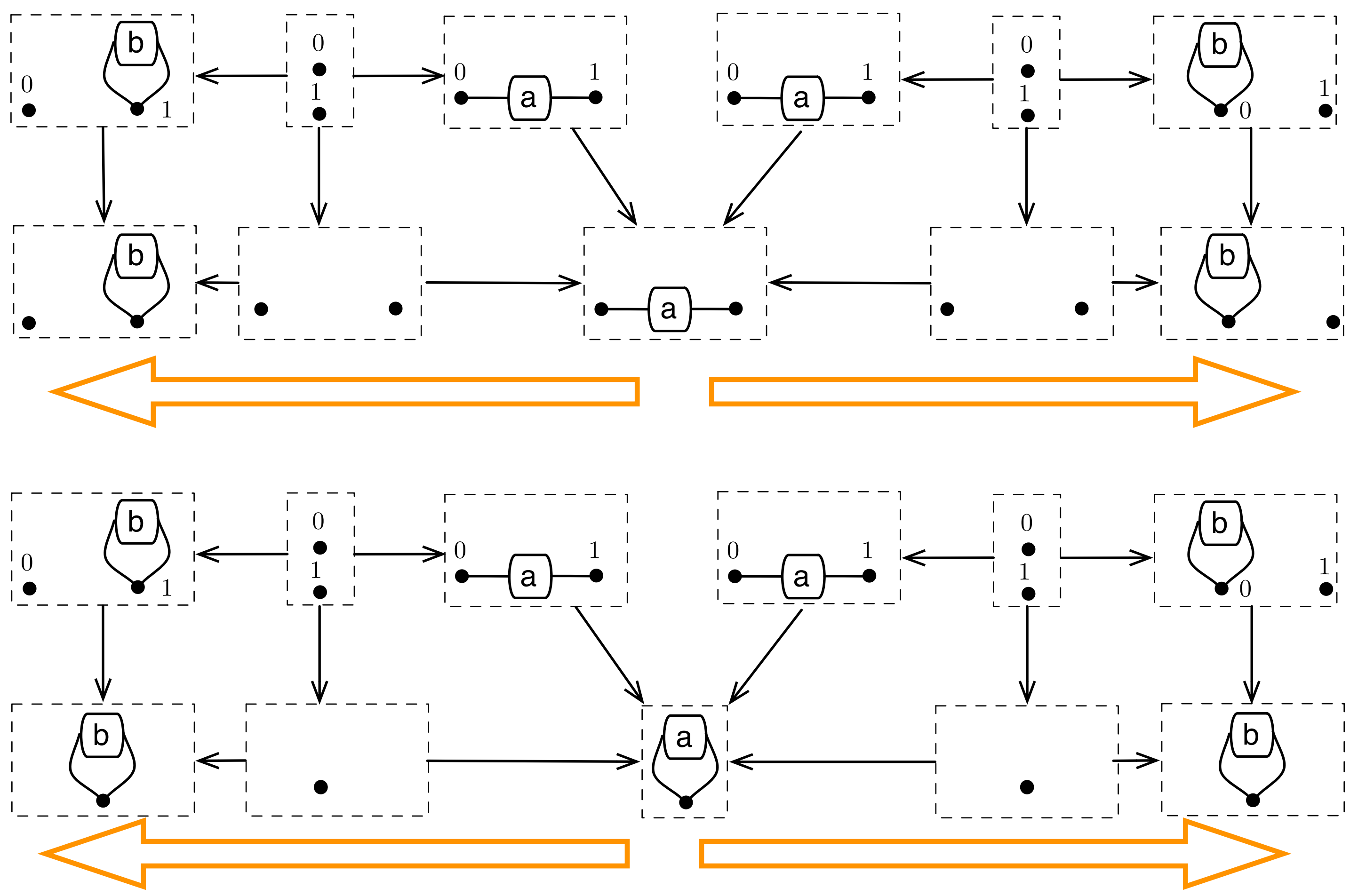
Counter-example

Only two critical pairs



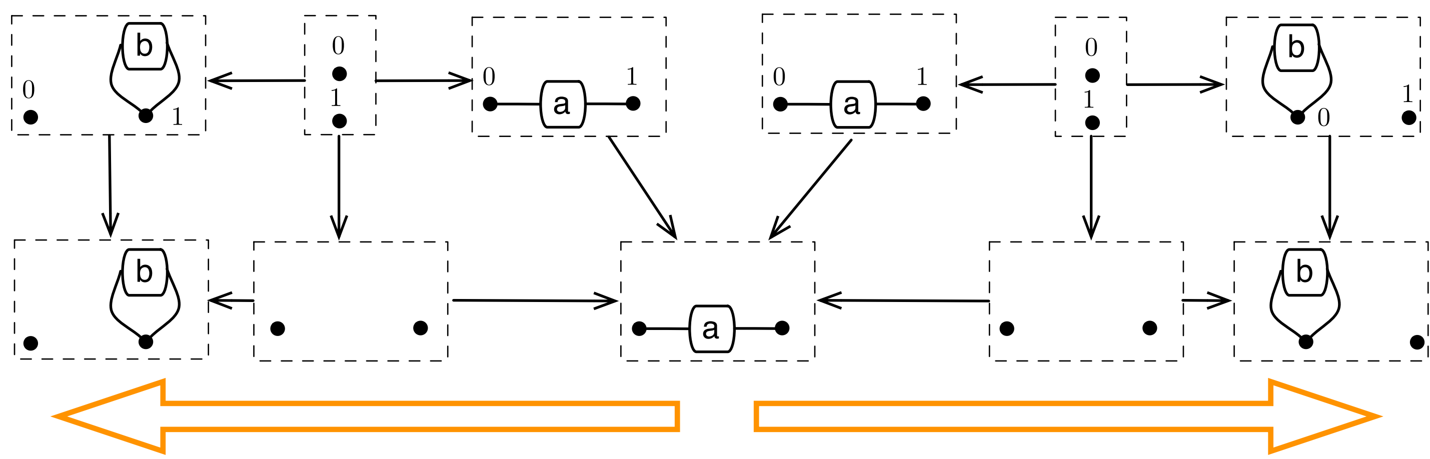
Counter-example

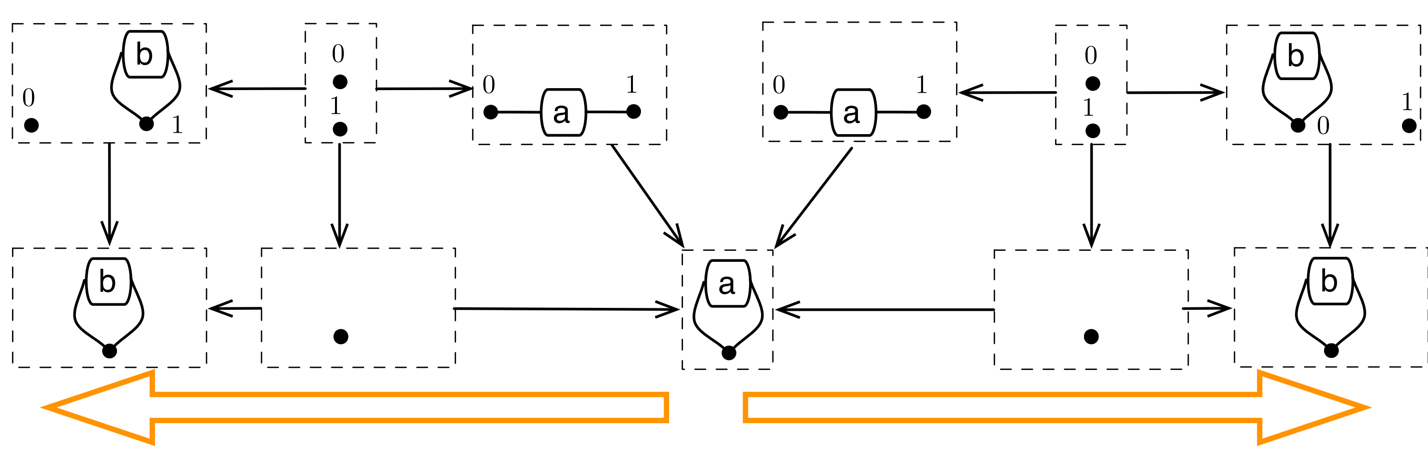
Only two critical pairs



Counter-example

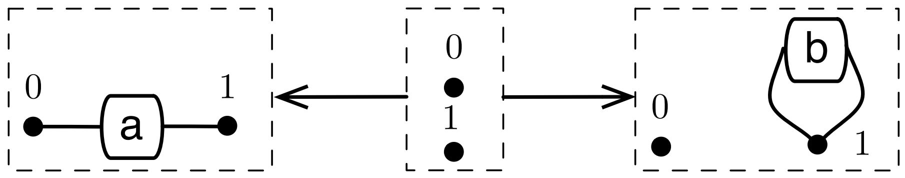
Only two critical pairs

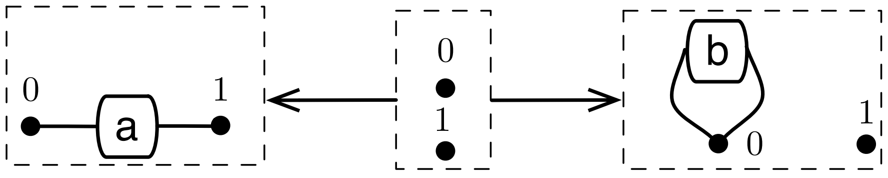




Both are trivially joinable
but the system is not confluent

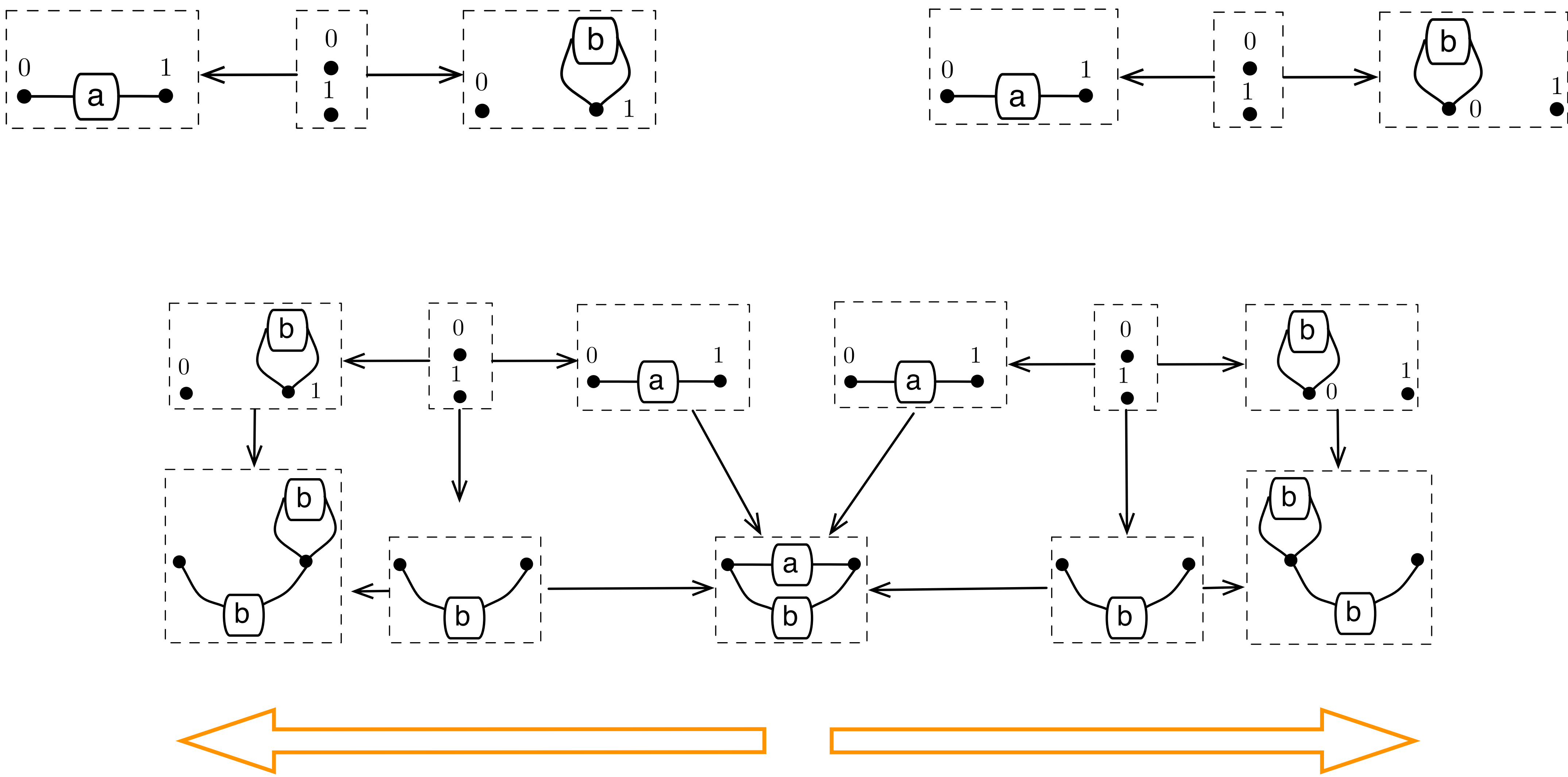
Counter-example





Both are trivially joinable
but the system is not confluent

Counter-example



Both are trivially joinable
but the system is not confluent

Plan of the Talk

1) Confluence for Term Rewriting

2) Confluence for DPO Rewriting

3) Confluence for DPO Rewriting with Interfaces

4) Confluence for PROP Rewriting

DPO rewriting
with Interfaces

DPO rewriting with Interfaces

DPO Rewriting with Borrowed Contexts:
Ehrig and Koenig 2004

DPO rewriting with Interfaces

DPO Rewriting with Borrowed Contexts:
Ehrig and Koenig 2004

Graphical Encoding of Process Calculi
Bonchi, Koenig, Gadducci 2009 - Gadducci 2007

DPO rewriting with Interfaces

DPO Rewriting with Borrowed Contexts:
Ehrig and Koenig 2004

Graphical Encoding of Process Calculi
Bonchi, Koenig, Gadducci 2009 - Gadducci 2007

Foundational studies of computads in cospans categories
Gadducci, Heckel 1997 - Sassone, Sobocinski 2005

DPO rewriting with Interfaces

DPO Rewriting with Borrowed Contexts:
Ehrig and Koenig 2004

Graphical Encoding of Process Calculi
Bonchi, Koenig, Gadducci 2009 - Gadducci 2007

Foundational studies of computads in cospans categories
Gadducci, Heckel 1997 - Sassone, Sobocinski 2005

Rather than rewriting graphs,
we rewrite graphs with interfaces

$G \leftarrow J$

DPO rewriting with Interfaces

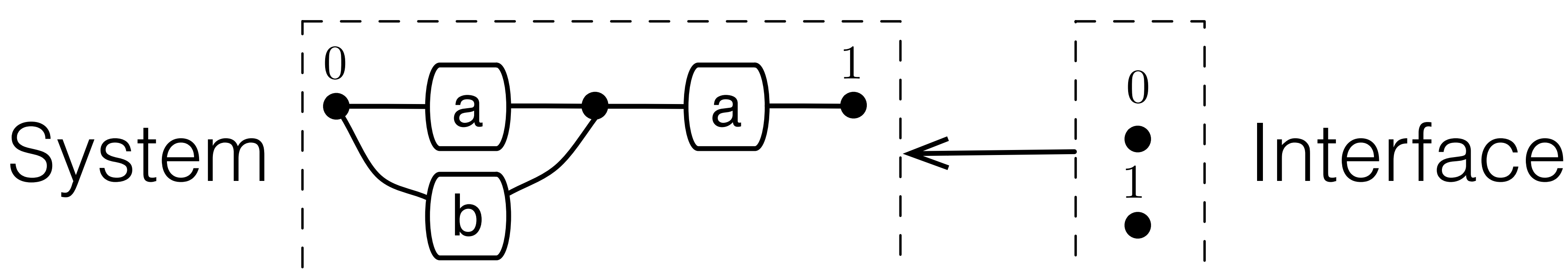
DPO Rewriting with Borrowed Contexts:
Ehrig and Koenig 2004

Graphical Encoding of Process Calculi
Bonchi, Koenig, Gadducci 2009 - Gadducci 2007

Foundational studies of computads in cospans categories
Gadducci, Heckel 1997 - Sassone, Sobocinski 2005

Rather than rewriting graphs,
we rewrite graphs with interfaces

$G \longleftarrow J$



DPO rewriting with Interfaces

DPO Rewriting with Borrowed Contexts:
Ehrig and Koenig 2004

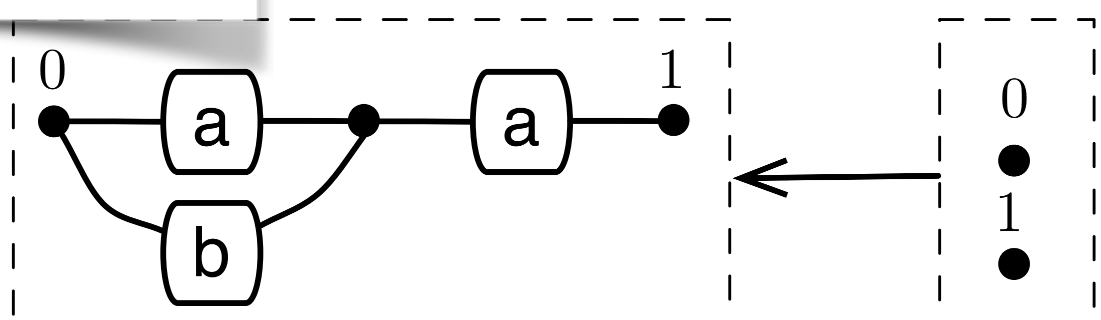
Graphical Encoding of Process Calculi
Bonchi, Koenig, Gadducci 2009 - Gadducci 2007

Foundational studies of computads in cospans categories
Gadducci, Heckel 1997 - Sassone, Sobocinski 2005

Ubiquitous in computer science:
Queries in Databases,
Kleene Algebra,
etc...

Rather than rewriting graphs,
rewrite graphs with interfaces

$$G \longleftarrow J$$

System  Interface

DPO rewriting with Interfaces

DPO rewriting with Interfaces

A rewriting rule is a span

$$L \longleftarrow K \longrightarrow R$$

DPO rewriting with Interfaces

A rewriting rule is a span

$$\begin{tikzcd}
L & K \arrow[l] \arrow[r] & R
\end{tikzcd}$$

A rewriting step is a commuting diagram

$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G \arrow[ur, phantom, "\urcorner", very near start] & C \arrow[l] \arrow[r] & H \arrow[ul, phantom, "\ulcorner", very near start] \\
& J \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

where the two squares are pushouts

DPO rewriting with Interfaces

A rewriting rule is a span

$$L \leftarrow K \rightarrow R$$

A rewriting step is a commuting diagram

$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G & C \arrow[l] \arrow[r] & H \\
 & J \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

where the two squares are pushouts

$$(G \leftarrow J) \Longrightarrow (H \leftarrow J)$$

DPO rewriting with Interfaces

A rewriting rule is a span

$$L \leftarrow K \rightarrow R$$

A rewriting step is a commuting diagram

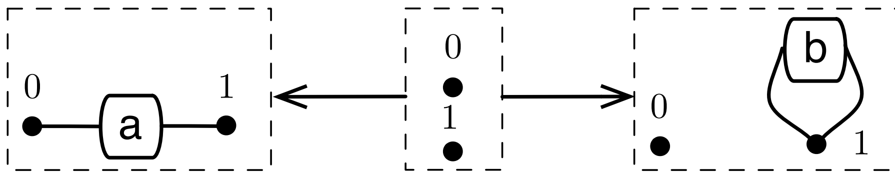
$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G \arrow[ur, phantom, "\lrcorner" very near start] & C \arrow[l] \arrow[r] & H \arrow[ul, phantom, "\llcorner" very near start] \\
& J \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

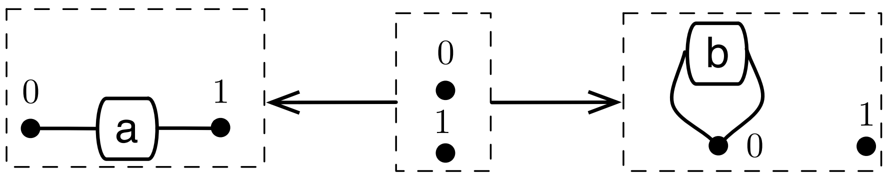
standard DPO is an instance when J is the initial object 0

where the two squares are pushouts

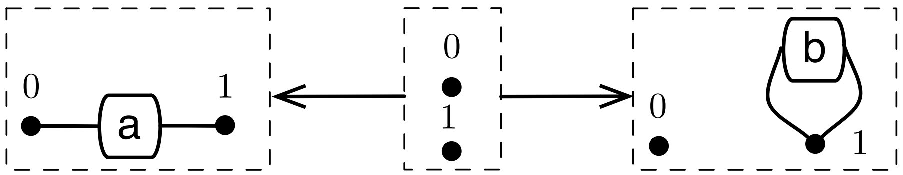
$(G \leftarrow J) \Rightarrow (H \leftarrow J)$

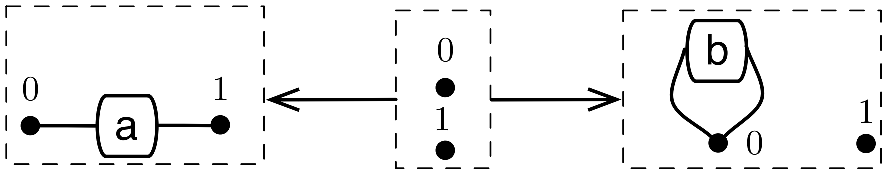
Example

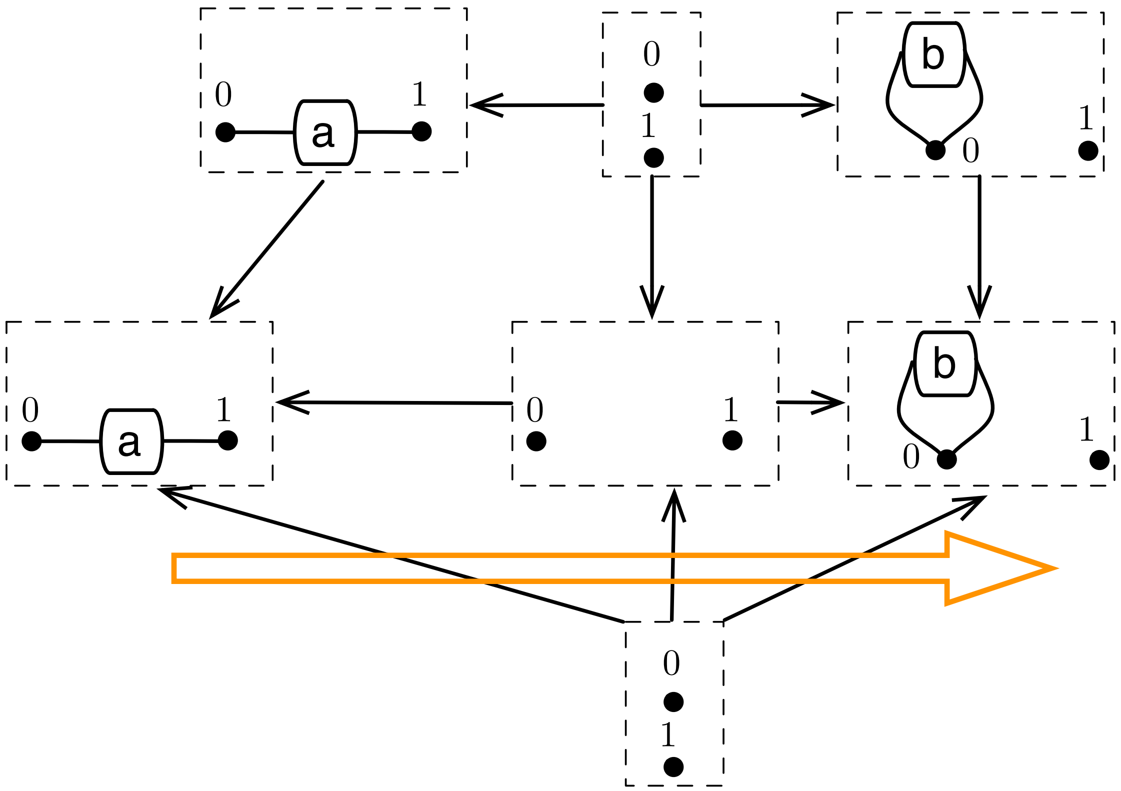




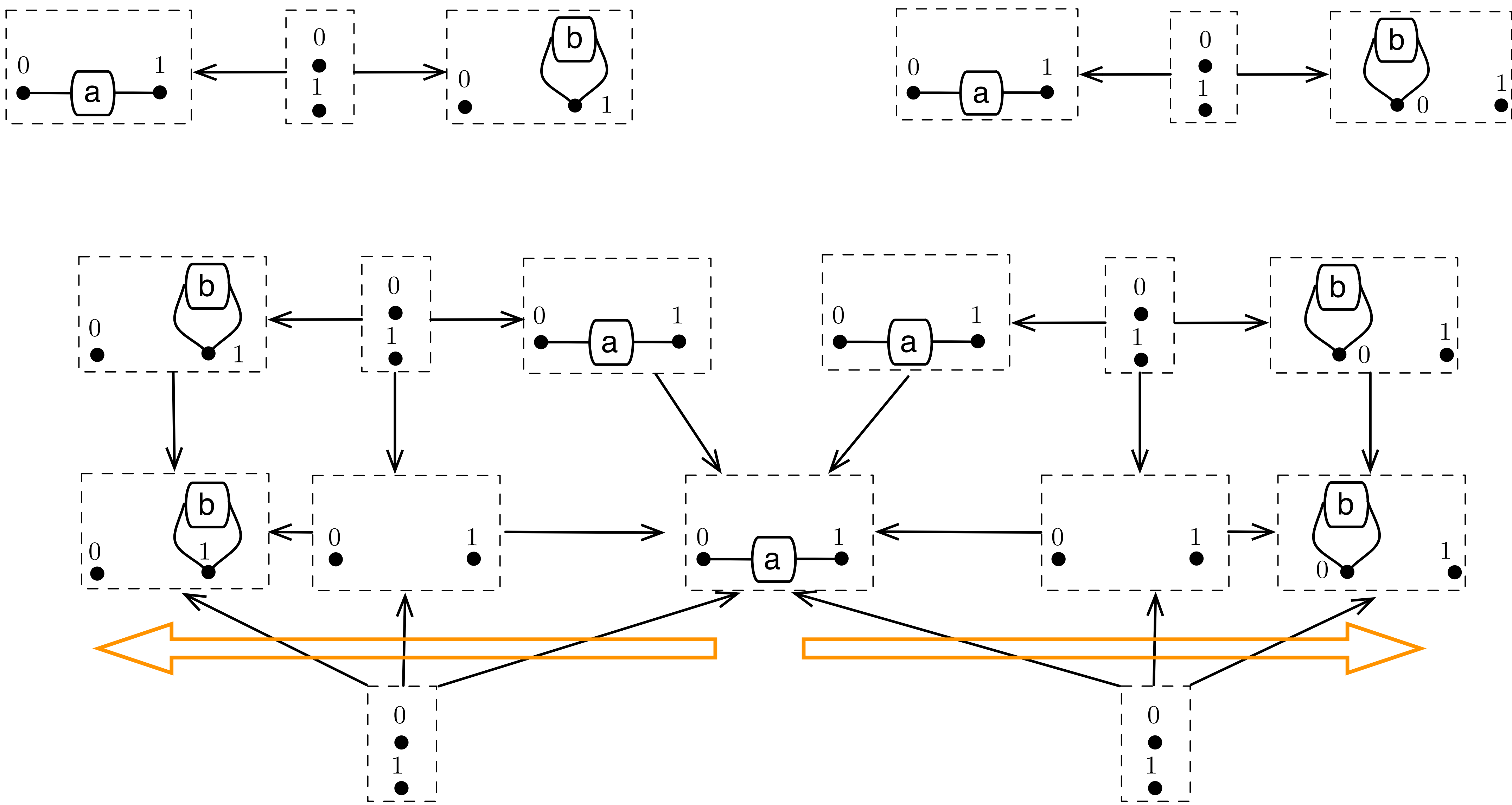
Example



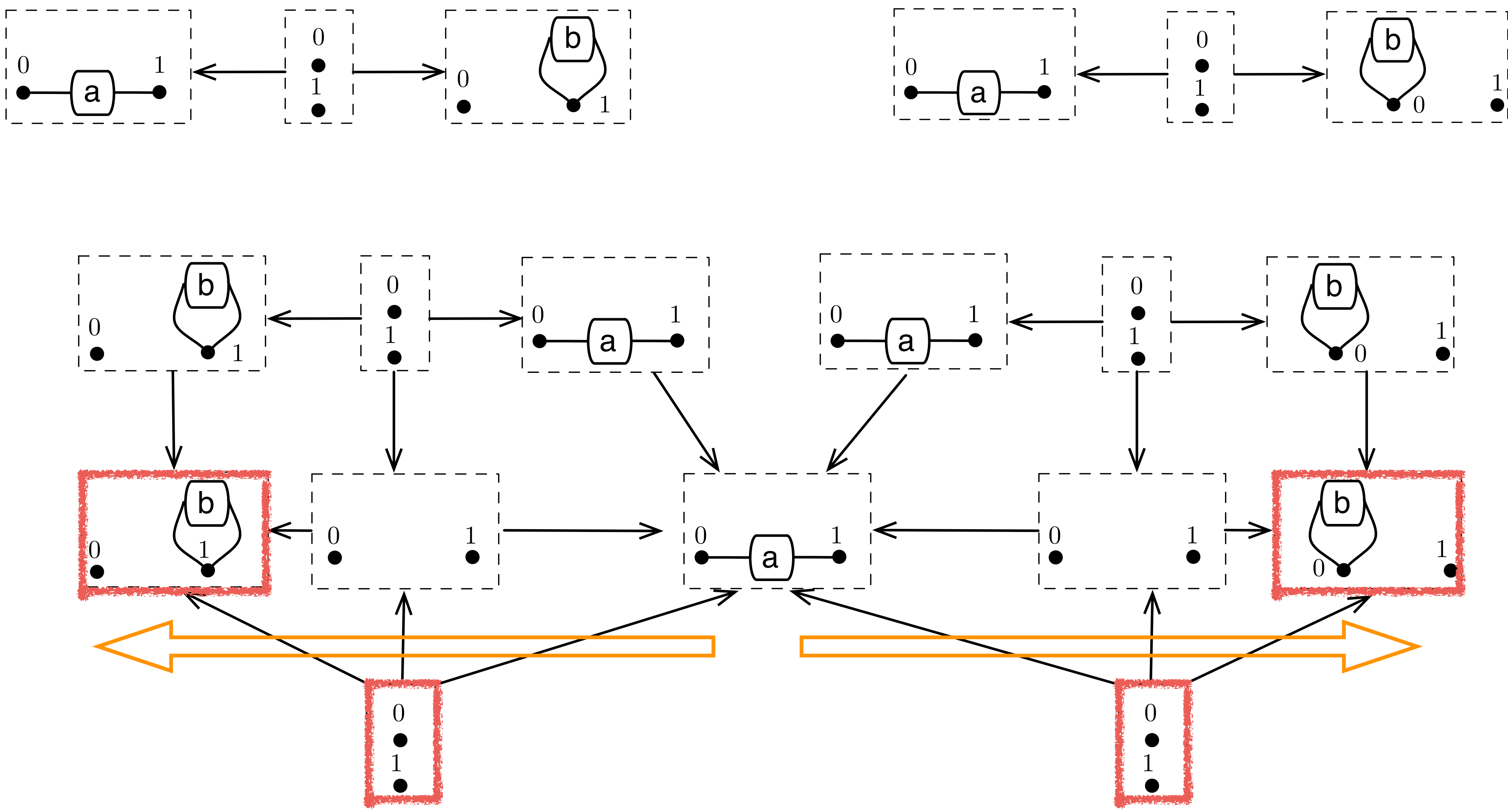




Example



Example



By adding the interface,
the arriving states are distinguished

Critical Pairs
with interfaces

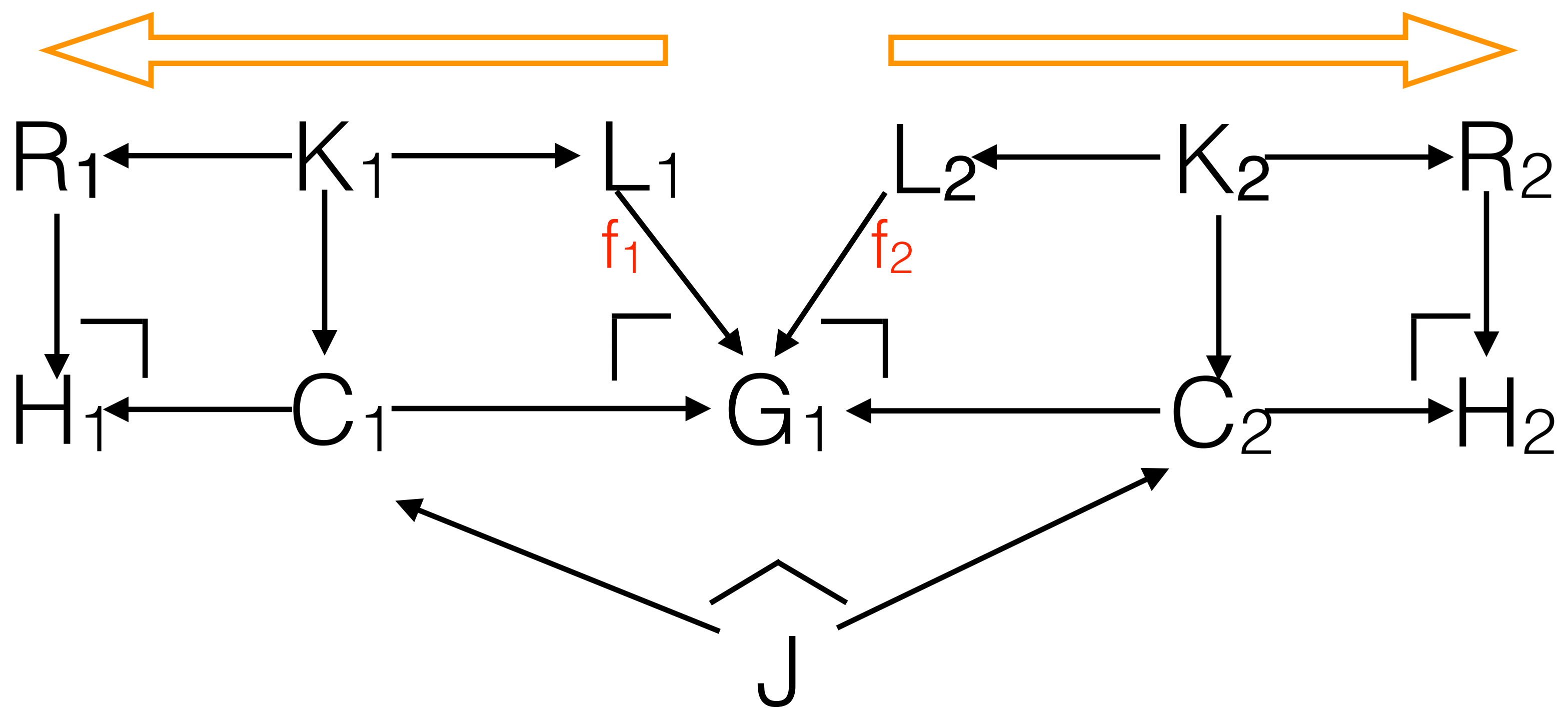
Critical Pairs with interfaces

f_1 and f_2 are jointly epi

$$
\begin{tikzcd}
R_1 \arrow[d] & K_1 \arrow[l] \arrow[r] \arrow[d] & L_1 \arrow[dr, "f_1"'] & & L_2 \arrow[dl, "f_2"] & K_2 \arrow[l] \arrow[r] \arrow[d] & R_2 \arrow[d] \\
H_1 & C_1 \arrow[l] \arrow[rr] & & G_1 & & C_2 \arrow[ll] \arrow[r] & H_2
\end{tikzcd}
$$

Critical Pairs with interfaces

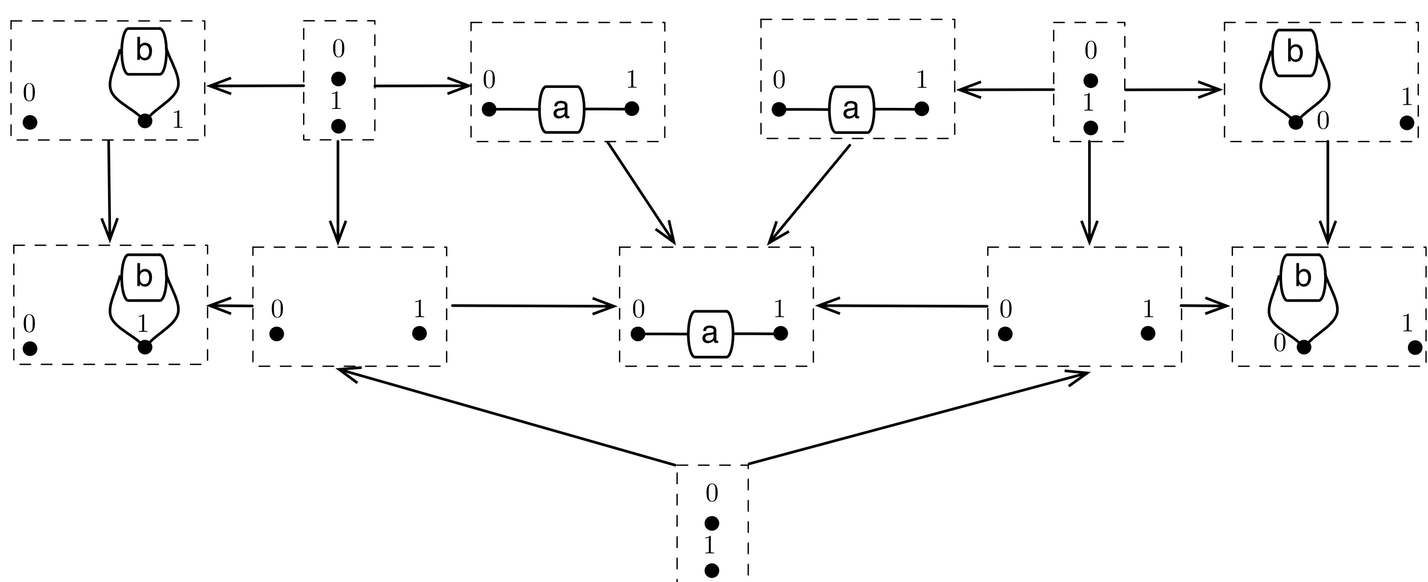
f_1 and f_2 are jointly epi



Critical Pairs with interfaces

f_1 and f_2 are jointly epi

$$\begin{tikzcd}
R_1 \arrow[d] & K_1 \arrow[l] \arrow[r] \arrow[d] & L_1 \arrow[dr, "f_1"'] & & L_2 \arrow[dl, "f_2"] & K_2 \arrow[l] \arrow[r] \arrow[d] & R_2 \arrow[d] \\
H_1 & C_1 \arrow[l] \arrow[rr] & & G_1 & & C_2 \arrow[ll] \arrow[r] & H_2 \\
& & & J \arrow[ull] \arrow[urr] & & &
\end{tikzcd}$$



Confluence for
DPO with Interfaces

Confluence for DPO with Interfaces

Theorem

In a DPO rewriting system with interfaces,
if all critical pairs are joinable,
then the system is locally confluent

Confluence for DPO with Interfaces

Theorem

In a DPO rewriting system with interfaces,
if all critical pairs are joinable,
then the system is locally confluent

Corollary

In a terminating DPO rewriting system
with interfaces,
confluence is decidable

Confluence for DPO with Interfaces

Theorem

In a DPO rewriting system with interfaces,
if all critical pairs are joinable,
then the system is locally confluent

Corollary

In a terminating DPO rewriting system
with interfaces,
confluence is decidable

Confluence for all graphs with interfaces G <- J

Confluence for DPO with Interfaces

Theorem

In a DPO rewriting system with interfaces,
if all critical pairs are joinable,
then the system is locally confluent

Corollary

In a terminating DPO rewriting system
with interfaces,
confluence is decidable

Confluence for all graphs with interfaces G <- J

Plump's result concerns all graphs with interfaces G <- 0

Confluence for DPO with Interfaces

Theorem

In a DPO rewriting system with interfaces,
if all critical pairs are joinable,
then the system is locally confluent

Corollary

In a terminating DPO rewriting system
with interfaces,
confluence is decidable

G
R
O
U
N
D

Confluence for all graphs with interfaces G <- J

Plump's result concerns all graphs with interfaces G <- 0

A nice analogy

	Terminating Term Rewriting	Terminating DPO Rewriting
Ground Confluence	Undecidable (Kapur et al. 1990)	Undecidable (Plump 1993)
Confluence	Decidable (Knuth-Bendix 1970)	Decidable (This talk)

A nice analogy

	Terminating Term Rewriting	Terminating DPO Rewriting
Ground Confluence	Undecidable (Kapur et al. 1990)	Undecidable (Plump 1993)
Confluence	Decidable (Knuth-Bendix 1970)	Decidable (This talk)

Plump's notion of "strongly joinable"

A nice analogy

	Terminating Term Rewriting	Terminating DPO Rewriting
Ground Confluence	Undecidable (Kapur et al. 1990)	Undecidable (Plump 1993)
Confluence	Decidable (Knuth-Bendix 1970)	Decidable (This talk)

Plump's notion of "strongly joinable"

Koenig et al. 2011

Plan of the Talk

1) Confluence for Term Rewriting

2) Confluence for DPO Rewriting

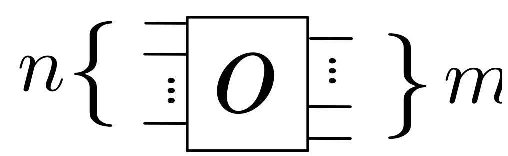
3) Confluence for DPO Rewriting with Interfaces

4) Confluence for PROP Rewriting

Freely Generated PROPs

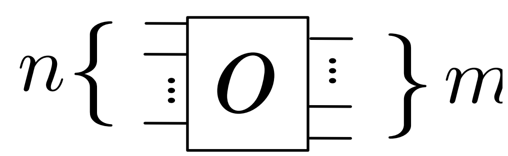
Freely Generated PROPs

A signature Σ is a set of gates with arity and coarity



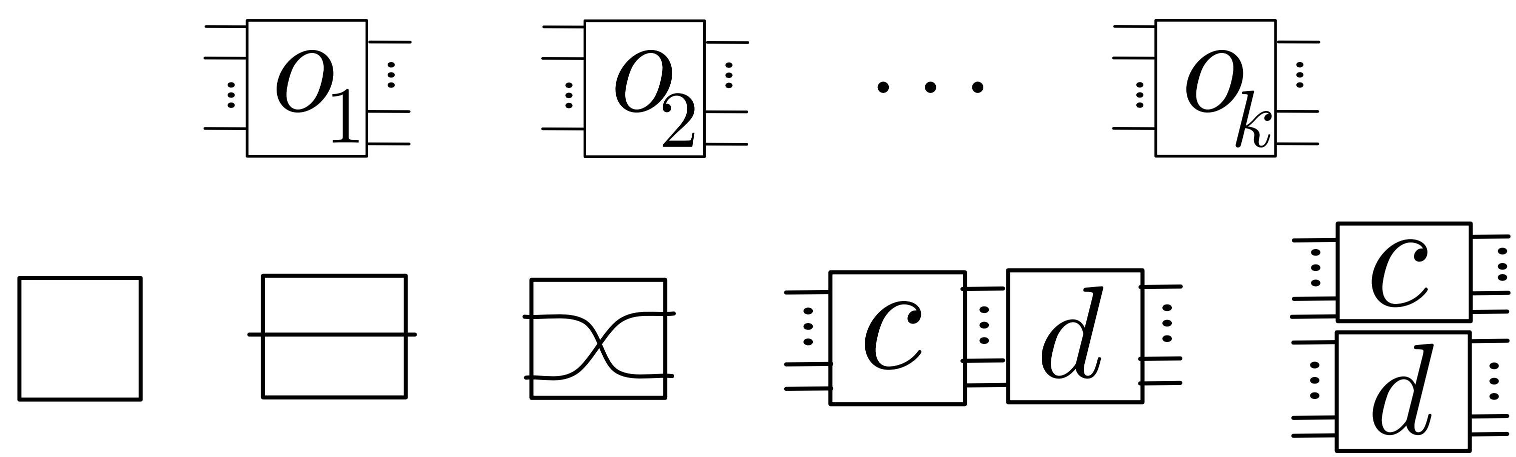
Freely Generated PROPs

A signature Σ is a set of gates with arity and coarity



The set of Σ-diagrams is generated by the following grammar

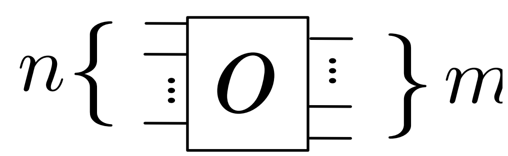
$c, d ::=$



$o_i \in \Sigma$

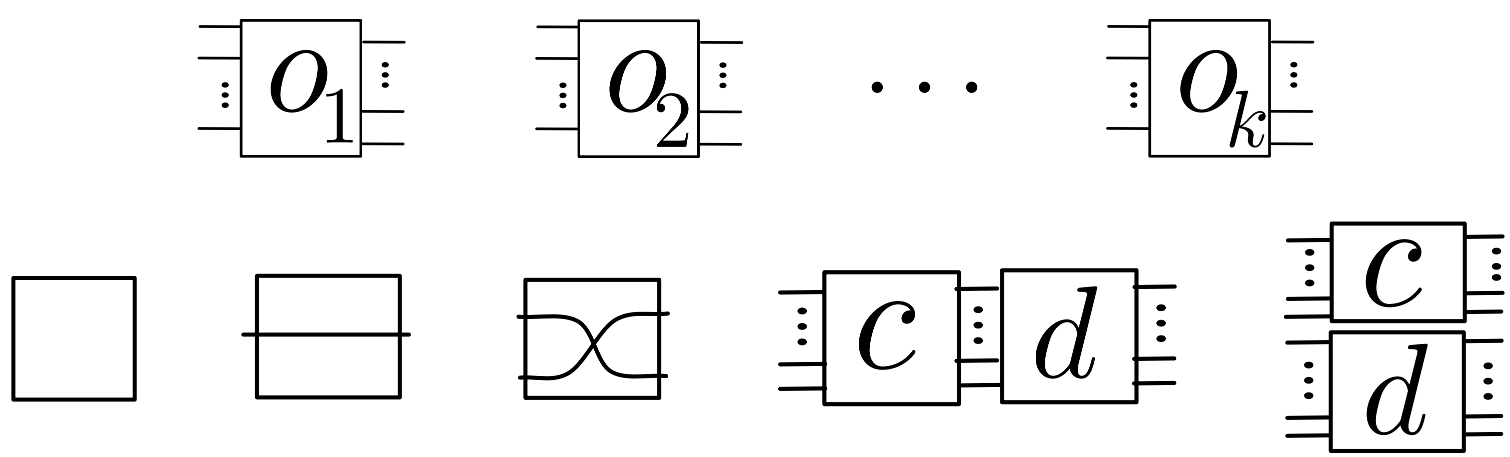
Freely Generated PROPs

A signature Σ is a set of gates with arity and coarity



The set of Σ-diagrams is generated by the following grammar

$c, d ::=$



$o_i \in \Sigma$

The PROP freely generated by Σ, $\mathbf{T}_\Sigma$, has
as arrows the Σ-diagrams
modulo the laws of strict symmetric monoidal categories

axioms for PROPs

axioms for PROPs

$$(t_1\,;t_2)\,;t_3 = t_1\,;(t_2\,;t_3) \qquad id_n\,;c = c = c\,;id_m$$

axioms for PROPs

$$(t_1 \,;\, t_2) \,;\, t_3 = t_1 \,;\, (t_2 \,;\, t_3) \qquad id_n \,;\, c = c = c \,;\, id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

axioms for PROPs

$$(t_1\,;t_2)\,;t_3 = t_1\,;(t_2\,;t_3) \qquad id_n\,;c = c = c\,;id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

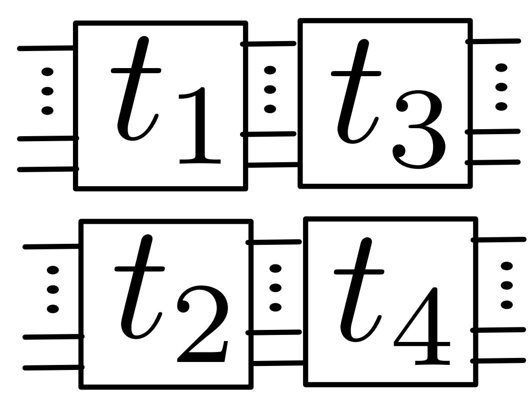
$$(t_1\,;t_3) \oplus (t_2\,;t_4) = (t_1 \oplus t_2)\,;(t_3 \oplus t_4)$$

axioms for PROPs

$$(t_1 \,;\, t_2) \,;\, t_3 = t_1 \,;\, (t_2 \,;\, t_3) \qquad id_n \,;\, c = c = c \,;\, id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

$$(t_1 \,;\, t_3) \oplus (t_2 \,;\, t_4) = (t_1 \oplus t_2) \,;\, (t_3 \oplus t_4)$$



axioms for PROPs

$$(t_1\,;t_2)\,;t_3 = t_1\,;(t_2\,;t_3) \qquad id_n\,;c = c = c\,;id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

$$(t_1\,;t_3) \oplus (t_2\,;t_4) = (t_1 \oplus t_2)\,;(t_3 \oplus t_4)$$

axioms for PROPs

$$(t_1\,;t_2)\,;t_3 = t_1\,;(t_2\,;t_3) \qquad id_n\,;c = c = c\,;id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

$$(t_1\,;t_3) \oplus (t_2\,;t_4) = (t_1 \oplus t_2)\,;(t_3 \oplus t_4)$$

$$(t \oplus id_z)\,;\sigma_{m,z} = \sigma_{n,z}\,;(id_z \oplus t)$$

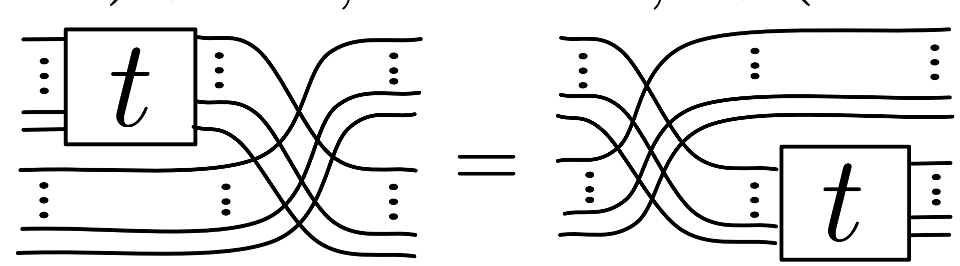
axioms for PROPs

$$(t_1 \,; t_2) \,; t_3 = t_1 \,; (t_2 \,; t_3) \qquad id_n \,; c = c = c \,; id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

$$(t_1 \,; t_3) \oplus (t_2 \,; t_4) = (t_1 \oplus t_2) \,; (t_3 \oplus t_4)$$

$$(t \oplus id_z) \,; \sigma_{m,z} = \sigma_{n,z} \,; (id_z \oplus t)$$



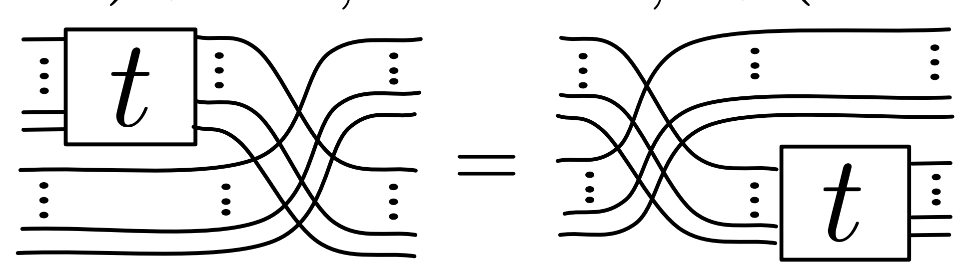
axioms for PROPs

$$(t_1 \,;\, t_2) \,;\, t_3 = t_1 \,;\, (t_2 \,;\, t_3) \qquad id_n \,;\, c = c = c \,;\, id_m$$

$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

$$(t_1 \,;\, t_3) \oplus (t_2 \,;\, t_4) = (t_1 \oplus t_2) \,;\, (t_3 \oplus t_4)$$

$$(t \oplus id_z) \,;\, \sigma_{m,z} = \sigma_{n,z} \,;\, (id_z \oplus t)$$



$$\sigma_{1,1} \,;\, \sigma_{1,1} = id_2$$

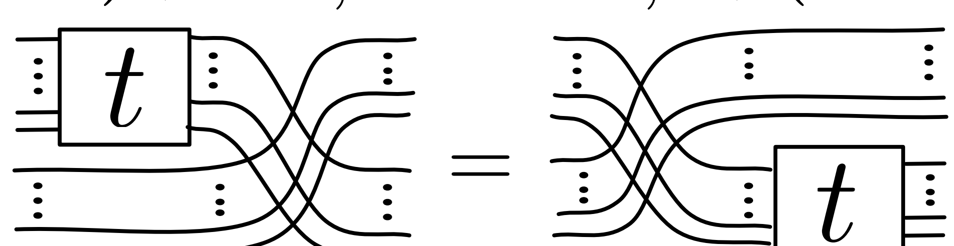
axioms for PROPs

$$(t_1\,;t_2)\,;t_3 = t_1\,;(t_2\,;t_3) \qquad id_n\,;c = c = c\,;id_m$$

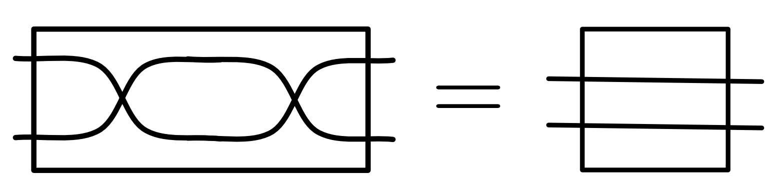
$$(t_1 \oplus t_2) \oplus t_3 = t_1 \oplus (t_2 \oplus t_3) \qquad id_0 \oplus t = t = t \oplus id_0$$

$$(t_1\,;t_3) \oplus (t_2\,;t_4) = (t_1 \oplus t_2)\,;(t_3 \oplus t_4)$$

$$(t \oplus id_z)\,;\sigma_{m,z} = \sigma_{n,z}\,;(id_z \oplus t)$$



$$\sigma_{1,1}\,;\sigma_{1,1} = id_2$$



Symmetric Monoidal Theories

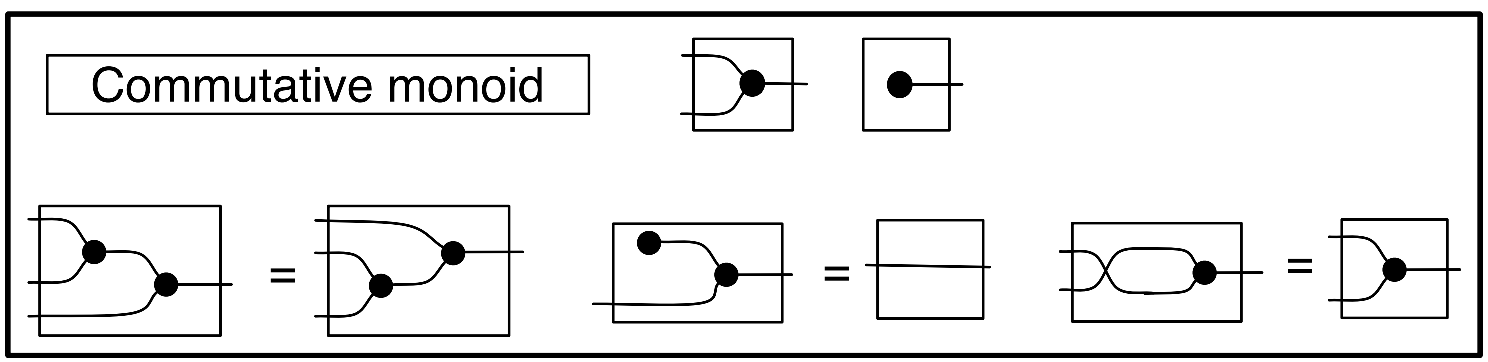
An SMT is a pair (Σ,E) where

- Σ is a signature and
- E is a set of equations $l=r$, for Σ-diagrams $l,r:n \dashrightarrow m$

Symmetric Monoidal Theories

An SMT is a pair (Σ,E) where

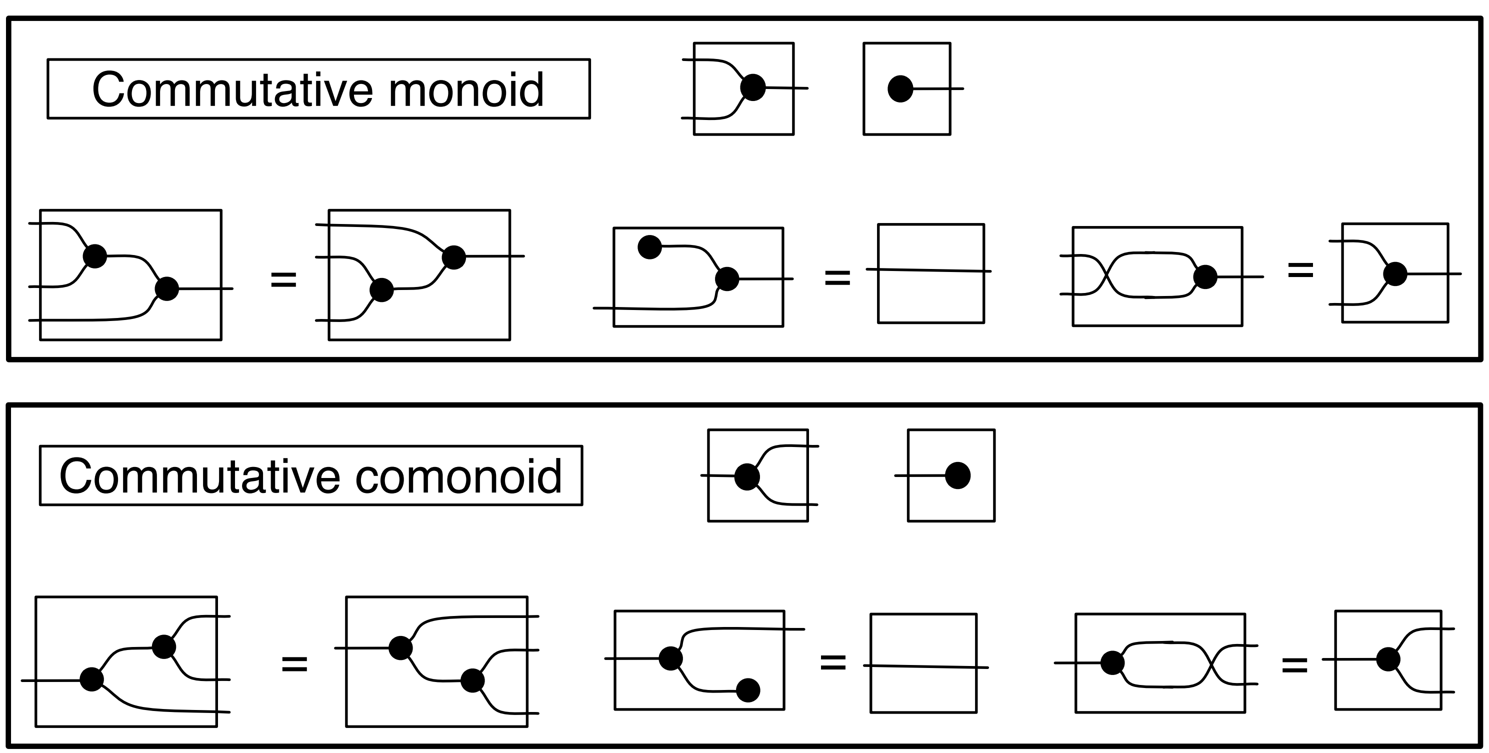
- Σ is a signature and
- E is a set of equations l=r, for Σ-diagrams l,r:n-->m



Symmetric Monoidal Theories

An SMT is a pair (Σ,E) where

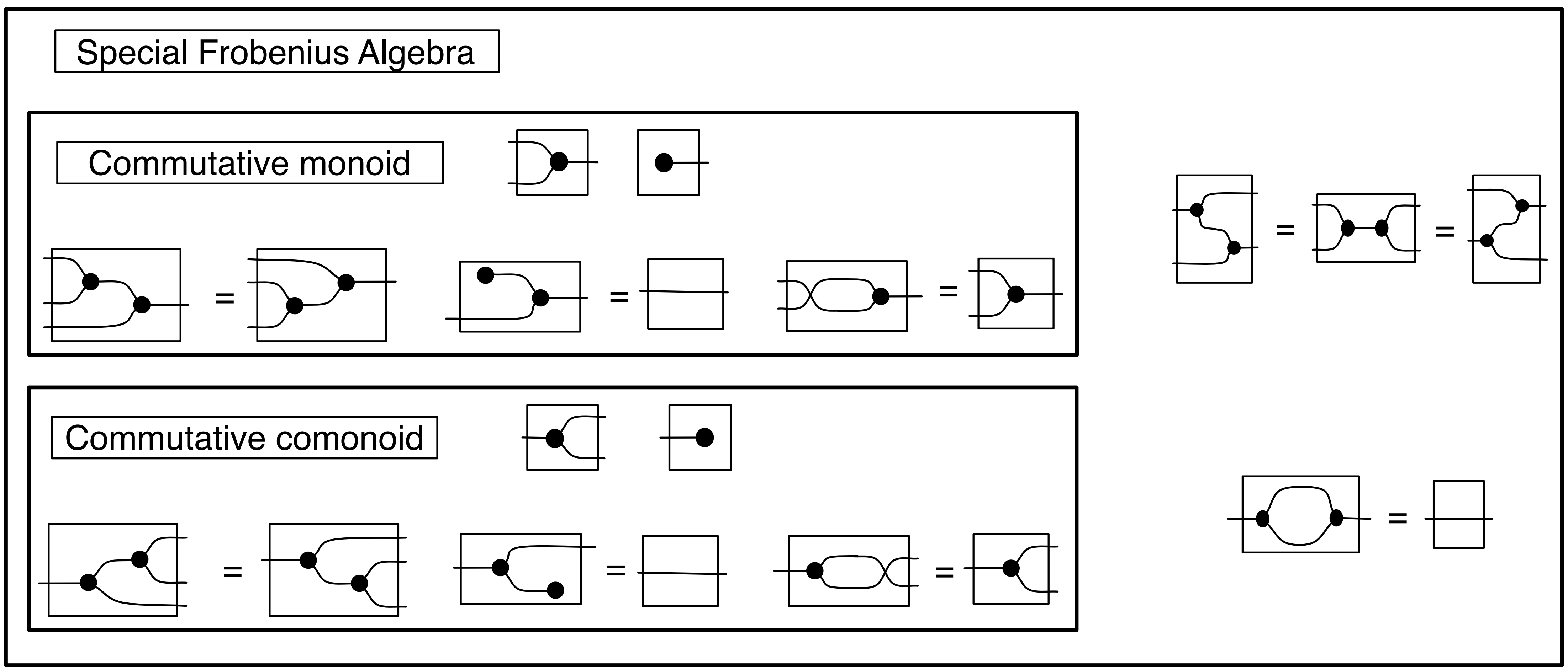
- Σ is a signature and
- E is a set of equations l=r, for Σ-diagrams l,r:n-->m



Symmetric Monoidal Theories

An SMT is a pair (Σ,E) where

- Σ is a signature and
- E is a set of equations l=r, for Σ-diagrams l,r:n-->m



Symmetric Monoidal Theories

Symmetric Monoidal Theories

SMTs can be thought as Algebraic Theories but

1) terms are DAGs rather than Trees
2) variables are linear (cannot be copied or discarded)

Symmetric Monoidal Theories

SMTs can be thought as Algebraic Theories but

1) terms are DAGs rather than Trees
2) variables are linear (cannot be copied or discarded)

These features make SMTs fundamental for
Quantum Informations,
Concurrency Theory,
Linear Logics and
Control Theory.

Symmetric Monoidal Theories

SMTs can be thought as Algebraic Theories but

1) terms are DAGs rather than Trees
2) variables are linear (cannot be copied or discarded)

These features make SMTs fundamental for
Quantum Informations,
Concurrency Theory,
Linear Logics and
Control Theory.

More and more interest on SMTs: an entire workshop at Simons Institute (Berkley)

Symmetric Monoidal Theories

SMTs can be thought as Algebraic Theories but

1) terms are DAGs rather than Trees
2) variables are linear (cannot be copied or discarded)

These features make SMTs fundamental for
Quantum Informations,
Concurrency Theory,
Linear Logics and
Control Theory.

More and more interest on SMTs: an entire workshop at Simons Institute (Berkley)

The celebrated theoretical physicist John Baez "reinvented" DPO rewriting

PROP Rewriting

By orienting the equations of an SMTs,
one obtains a rewriting system

PROP Rewriting

By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

PROP Rewriting

By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

More generally, rewriting is important for completeness proofs
that often rely on normal forms

PROP Rewriting

By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

More generally, rewriting is important for completeness proofs
that often rely on normal forms

But rewriting modulo the axioms of PROPs is tough...

PROP Rewriting

By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

More generally, rewriting is important for completeness proofs
that often rely on normal forms

But rewriting modulo the axioms of PROPs is tough...

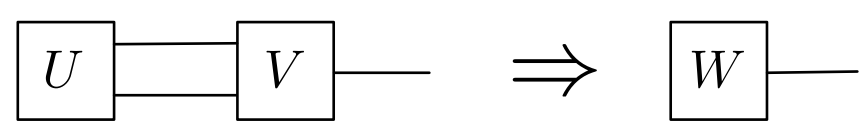
PROP Rewriting

By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

More generally, rewriting is important for completeness proofs
that often rely on normal forms

But rewriting modulo the axioms of PROPs is tough...



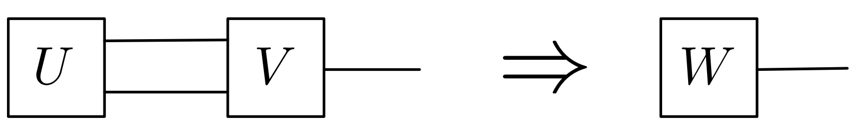
PROP Rewriting

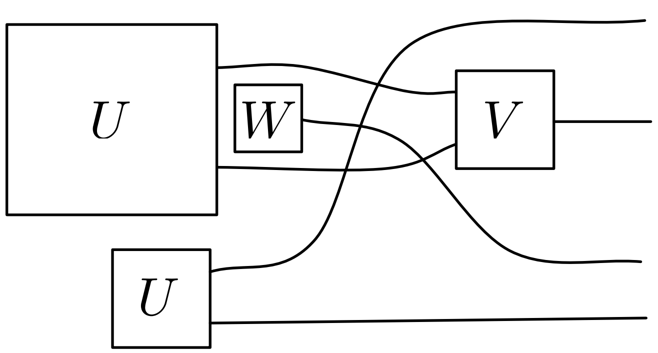
By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

More generally, rewriting is important for completeness proofs
that often rely on normal forms

But rewriting modulo the axioms of PROPs is tough...





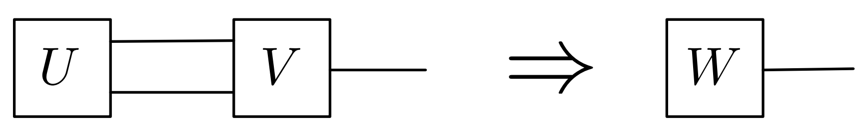
PROP Rewriting

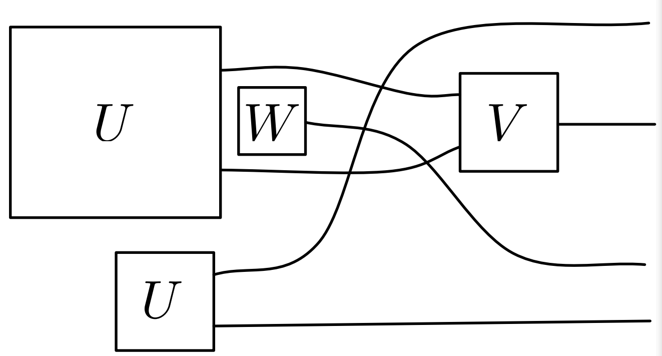
By orienting the equations of an SMTs,
one obtains a rewriting system

If the system is terminating and confluent,
one can check equivalence via rewriting

More generally, rewriting is important for completeness proofs
that often rely on normal forms

But rewriting modulo the axioms of PROPs is tough...





French School of Rewriting
(Yves la Font,
Samuel Mimram,
Philippe Malboss, ...)

Confluence for PROP Rewriting

Lafont 2003 - Mimram 2014

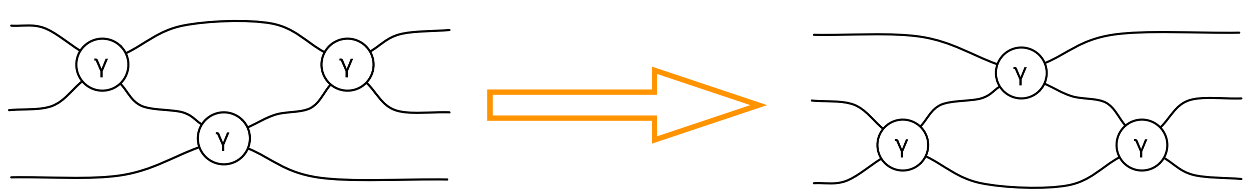
A finite rewriting system,
can generate infinitely many critical pairs

Confluence for PROP Rewriting

Lafont 2003 - Mimram 2014

A finite rewriting system,
can generate infinitely many critical pairs

One rule (directed Yang-Baxter)

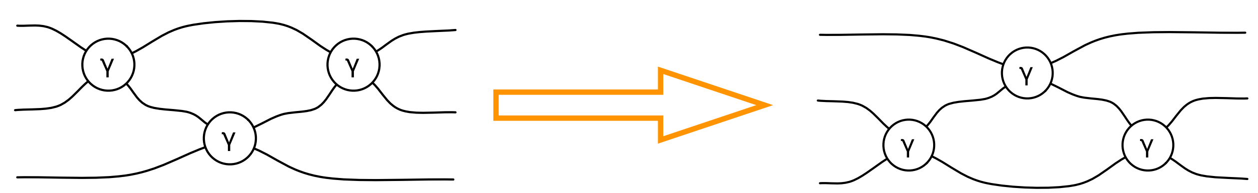


Confluence for PROP Rewriting

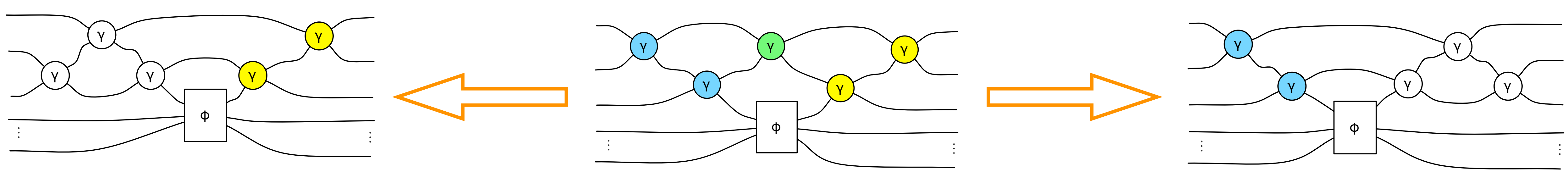
Lafont 2003 - Mimram 2014

A finite rewriting system,
can generate infinitely many critical pairs

One rule (directed Yang-Baxter)



Infinitely many critical pairs: one for each diagram **ϕ**



Rewriting Modulo Symmetric Monoidal Structure (LICS 2016)

One solution to both problems: DPO rewriting with interfaces!

Rewriting Modulo Symmetric Monoidal Structure (LICS 2016)

One solution to both problems: DPO rewriting with interfaces!

If the theory contains a special Frobenius structure,	If the theory does not contain a special Frobenius structure,

Rewriting Modulo Symmetric Monoidal Structure (LICS 2016)

One solution to both problems: DPO rewriting with interfaces!

If the theory contains a special Frobenius structure,	If the theory does not contain a special Frobenius structure,
then PROP rewriting = DPO rewriting with interfaces	

Rewriting Modulo Symmetric Monoidal Structure (LICS 2016)

One solution to both problems: DPO rewriting with interfaces!

If the theory contains a special Frobenius structure,

then
PROP rewriting
=
DPO rewriting
with interfaces

If the theory does not contain a special Frobenius structure,

then
PROP rewriting
=
convex DPO rewriting
with interfaces

Rewriting Modulo Symmetric Monoidal Structure (LICS 2016)

One solution to both problems: DPO rewriting with interfaces!

If the theory contains a special Frobenius structure,

then
PROP rewriting
=
DPO rewriting
with interfaces

If the theory does not contain a special Frobenius structure,

then
PROP rewriting
=
convex DPO rewriting
with interfaces

Paves the way to challenging and promising research paths...

Rewriting Modulo Symmetric Monoidal Structure (LICS 2016)

One solution to both problems: DPO rewriting with interfaces!

If the theory contains
a special Frobenius structure,

then
PROP rewriting
=
DPO rewriting
with interfaces

We know how to prove confluence

If the theory does not contain
a special Frobenius structure,

then
PROP rewriting
=
convex DPO rewriting
with interfaces

Paves the way to challenging and promising research paths...

Future Works

Future Works

SMTs with Special Frobenius Structures are closely related to Geometric Logic

Future Works

SMTs with Special Frobenius Structures are closely related to Geometric Logic

Like term rewriting plays a crucial role for equational logic, hopefully, DPO rewriting (with interfaces!) may play analogous role for Geometric Logic....

Future Works

SMTs with Special Frobenius Structures are closely related to
Geometric Logic
Like term rewriting plays a crucial role for equational logic, hopefully, DPO rewriting (with interfaces!) may play analogous role for Geometric Logic....

Future Works

SMTs with Special Frobenius Structures are closely related to Geometric Logic

Like term rewriting plays a crucial role for equational logic, hopefully, DPO rewriting (with interfaces!) may play analogous role for Geometric Logic....

SMTs with Special Frobenius Structures are closely related to Relational Structures

(more precisely, to Cartesian Bicategories of Relations by Carboni and Walters)

Future Works

SMTs with Special Frobenius Structures are closely related to
Geometric Logic
Like term rewriting plays a crucial role for equational logic, hopefully, DPO rewriting (with interfaces!) may play analogous role for Geometric Logic....

SMTs with Special Frobenius Structures are closely related to
Relational Structures
(more precisely, to Cartesian Bicategories of Relations by Carboni and Walters)

A functorial semantics for them is still not understood

Future Works

Future Works

Most of the theory of convex DPO rewriting
has to be developed

Future Works

Most of the theory of convex DPO rewriting has to be developed

Future Works

Most of the theory of convex DPO rewriting has to be developed

We need tools for supporting combinatorial reasoning

1) Implementing rewriting with Interfaces (for arbitrary matches and rules)
2) Automatically proving confluence
3) (Semi-)Automatically check equivalence