Bisimulation from a graphical encoding (DPOs, cospans, relative POs and all that)

Fabio Gadducci
Dipartimento di Informatica, Pisa

some reminiscing...

- Back in 1995, my Ph.D. dealt with the search for algebraic presentations of rewriting systems.
- Differently from the Rewriting Logic formalism, the idea was not to equip an algebraic theory (i.e., a cartesian category) with additional operators...
- ...but to identify suitable categories for recovering the "terms as arrows, rewrites as cells" analogy!

rationale and method

- Equip set-theoretical formalisms (best suited for implementation purposes) with an algebraic presentation (best suited for inductive reasoning)
- The methodology
  - consider your favorite RS (states plus reductions)
  - find a free categorical presentation (consider e.g. cartesian categories for terms, monoidal categories for Petri nets), such that states are arrows
  - then rules are pairs of arrows, and computations are cells of the free 2-category

similarities and applications

- The methodology underlines e.g. the lambda calculus, and cartesian closed categories: objects are types, lambda-terms are arrows, beta-eta reductions are cells...

- The topic was tested for some RSs: infinite terms, (cyclic) term graphs, ...

  late 90's, mostly with Andrea

- The same mechanism was the basis for tile logic (since a double category is a 2-category in *Cats*), aimed at capturing process calculi specs

  late 90's, mostly with Ugo

dealing with graphs...

- Then, I moved to Berlin for a post-doctoral stay, and I tried the same ideas on DPO...
- but which is the category with "graphs as arrows"?
- Solution: graph cospans and free compact closed categories [GH, WADT97][GHL,CTCS99]

My current view of DPO

shortly, graph rewriting...

- Why graph rewriting (late Sixties, early Seventies)
  - ☑ generalizes Chomsky grammars (adding data sharing)
  - ☑ used in constraint solving and data structuring (70's)
  - ☑ applied as a (visual) specification technique (80's-90's)
- but...
  - ☐ no (obvious) algebraic structure (no induction)
  - ☐ neither (temporal) logic nor calculus

Many data structures (HLR, adhesive...)
for the same meta-approach

DPO approach

adhesive $\mathcal{C}$

arrows in $\mathcal{C}$ (possibly mono)

a rule

$$L \xleftarrow{\;l\;} K \xrightarrow{\;r\;} R$$

a derivation step

$$\begin{array}{ccccc} L & \xleftarrow{\;l\;} & K & \xrightarrow{\;r\;} & R \\ {\scriptstyle m_L}\downarrow & (1) & \downarrow{\scriptstyle m_K} & (2) & \downarrow{\scriptstyle m_R} \\ G & \xleftarrow[\;l^*\;]{} & D & \xrightarrow[\;r^*\;]{} & H \end{array}$$

pushout in $\mathcal{C}$

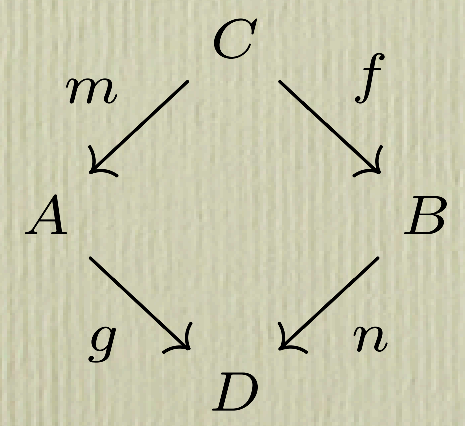
set of theoretical tools (concurrency, mostly)

shortly, adhesive categories

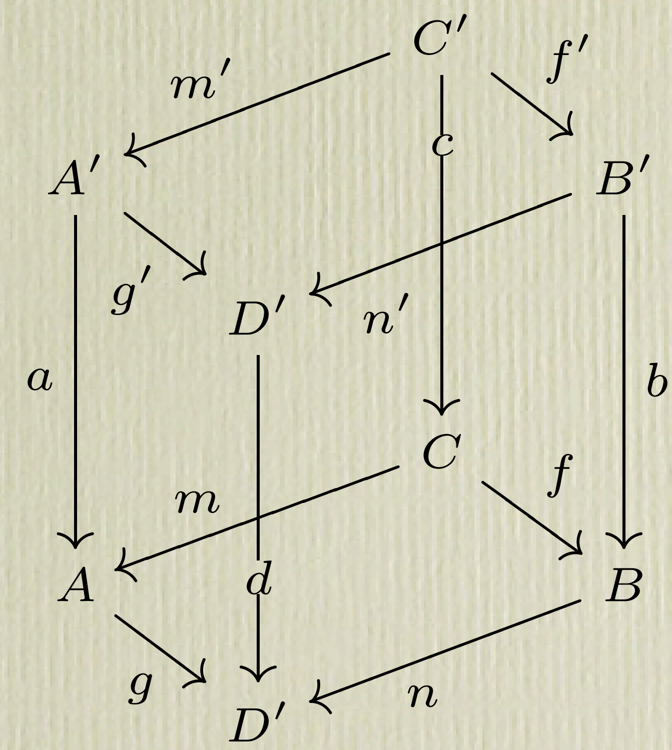
[LS04]

A category is adhesive if

1. it has pushouts along monos
2. it has pullbacks
3. pushout along monos are *Van Kampen squares*



a pushout



a Van Kampen square

cospan definition

cocomplete $\mathcal{C}$

arrows in $\mathcal{C}$

an arrow

$$A \xrightarrow{a} E \xleftarrow{b} B$$

composition

$$A \xrightarrow{a} E \xleftarrow{b} B \xrightarrow{c} D \xleftarrow{d} C$$

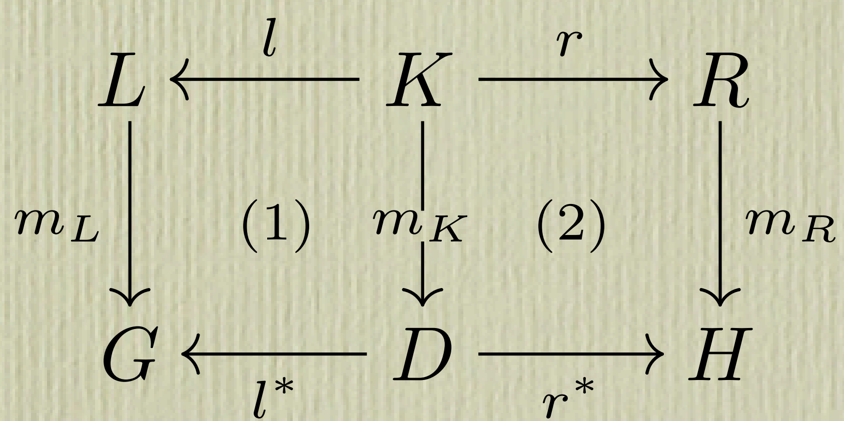
$$E \xrightarrow{e} F \xleftarrow{f} D$$

pushout in $\mathcal{C}$

powerful theoretical tool
(bi-categories of relations...)
[CW87]

DPO connection

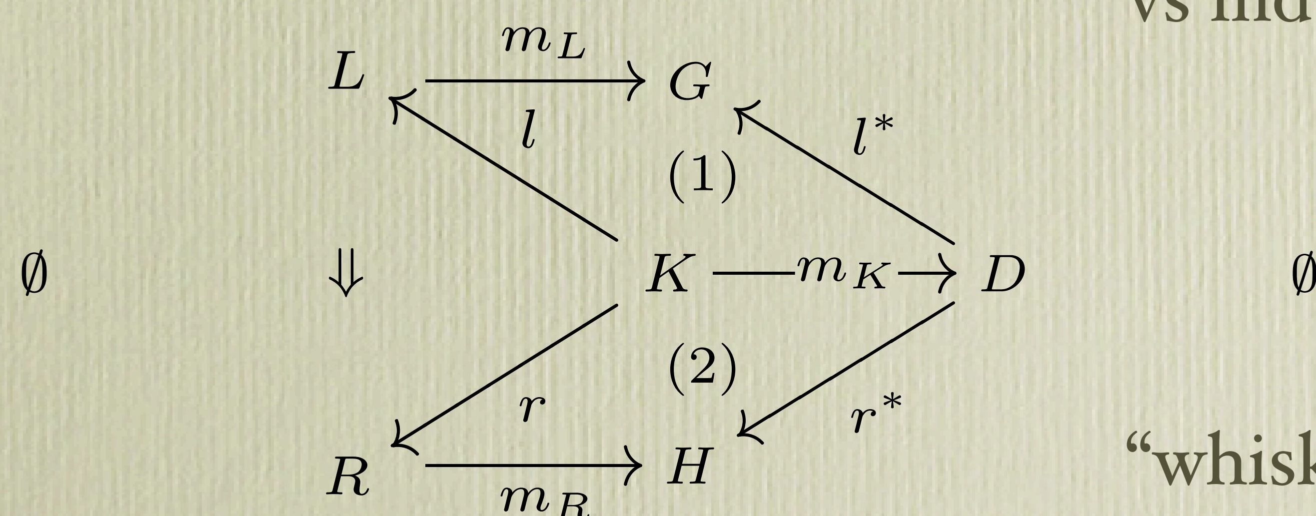
[GH97]



a rule

a derivation step

operational
vs induction



a "cell"

"whiskering"

DPOs vs. Cospans

- (A sub-category of) Cospans over graphs are the free compact closed (bi-)category built from the unary signature for graphs
- The DPO approach is operational: search for a match, build the PO complement...
- The free construction (using cospans) is algebraic: inductive closure of a set of basic rules...

recent facts on LTSs

some familiar remarks...

often, the operational semantics of a computational formalism is given by means of a reduction system...

$$(\lambda x.M)N \Rightarrow M[N/x]$$

functional paradigm

$$\alpha.P|\overline{\alpha} \Rightarrow P$$

process calculi

reductions are inductively built...

the states may have a complex structure...

$$\overline{\alpha} \mid \alpha.P \Rightarrow P$$

the semantics is often closed by contexts...

$$\frac{P \Longrightarrow Q}{C[P] \Longrightarrow C[Q]}$$

possibly forbidding some contexts from allowing the reduction to take place

our simple example...

$$\alpha.P|\overline{\alpha} \Rightarrow P$$

$$0 \xrightarrow{P} 1 \; \underset{[\_]}{\overset{\alpha.[\_]|\bar{\alpha}}{\Downarrow}} \; 1 \xrightarrow{C[\_]} 1$$

$$C[\_]$$

terms as arrows, types as objects

$$C[\_] = \beta.0 \mid [\_]$$

$$P = \overline{\beta}$$

$$\frac{\alpha.\overline{\beta} \mid \overline{\alpha} \Rightarrow \overline{\beta} \qquad \beta.0 \mid [\_]}{\beta.0 \mid \alpha.\overline{\beta} \mid \overline{\alpha} \Rightarrow \beta.0 \mid \overline{\beta}}$$

interaction vs computation

Albeit often self-intuitive, reduction may lack compositionality
(in capturing the behaviour of a process)

Or, in other words, it is important the
interaction of a process with an environment
as the basis for a semantical analysis!!

$$P \equiv Q \text{ if } \forall C[\_] : \mathcal{P}(C[P]) \iff \mathcal{P}(C[Q])$$

It would be nice to restrict to just a few contexts
(e.g., linear—not duplicating the process), still
preserving equivalence with respect to *all* contexts

a different style...

Since late 70s, Plotkin' SOS influenced the style of presenting the operational semantics

Labelled transition systems may enrich reductions with an observation of the actions offered to the environment

Thus, a process may be studied in isolation, by so called behavioural congruences

some facts

- After Milner's proposal for pi, use of reduction semantics has become increasingly popular for nominal calculi (consider e.g. mobile ambients)
- Still, it would be highly desirable to recover an observational semantics, possibly independently from the presentation of a calculus...
- In general terms, how to distill a suitable labelled transition system from a reduction system, at the same time ensuring congruence for the chosen behavioural equivalence ?

the context-as-label proposal

Aim: use enabling contexts as labels [Sewell98]

$$\frac{C[P] \Longrightarrow Q}{P \xrightarrow{C[\_]} Q}$$

problems...

- how to minimize the labels (otherwise, of little use)
- how to recover congruence?
- how to establish meaning (e.g., correspondence results)?

the Relative PO choice

$$L : \sigma \Longrightarrow R : \sigma$$

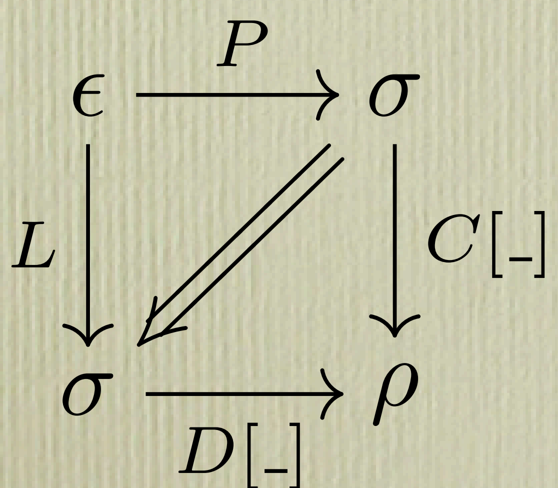
a rule

proposed by Leifer and Milner {00}

$$D[\_]$$

an enabling context

generalised by Sassone and Sobocinski {03}

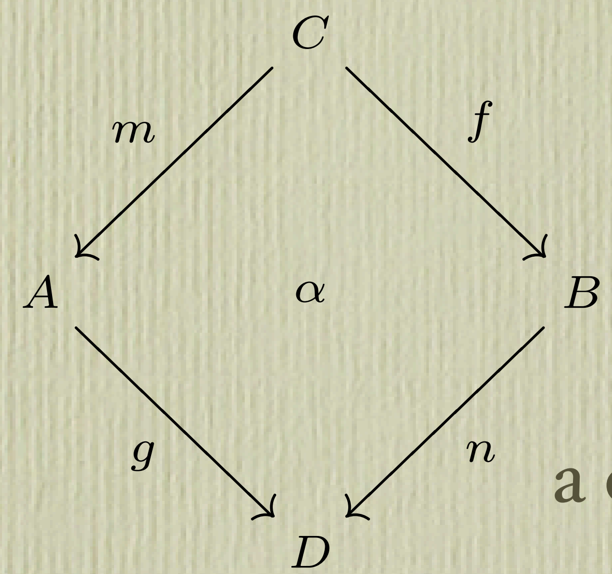


a commutative, "minimal" diagram

$$P \stackrel{C[\_]}{\Longrightarrow} D[R]$$

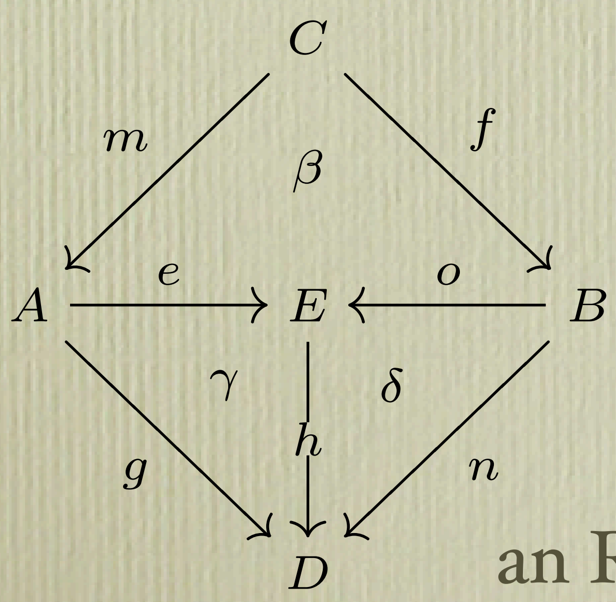
the labelled transition

the RPO definition



a commutative diagram

a candidate is an RPO if it uniquely factorizes all possible candidates



an RPO candidate

an IPO is a commutative diagram which is its own RPO (i.e., E = D, etc.)

alternative RPO definition

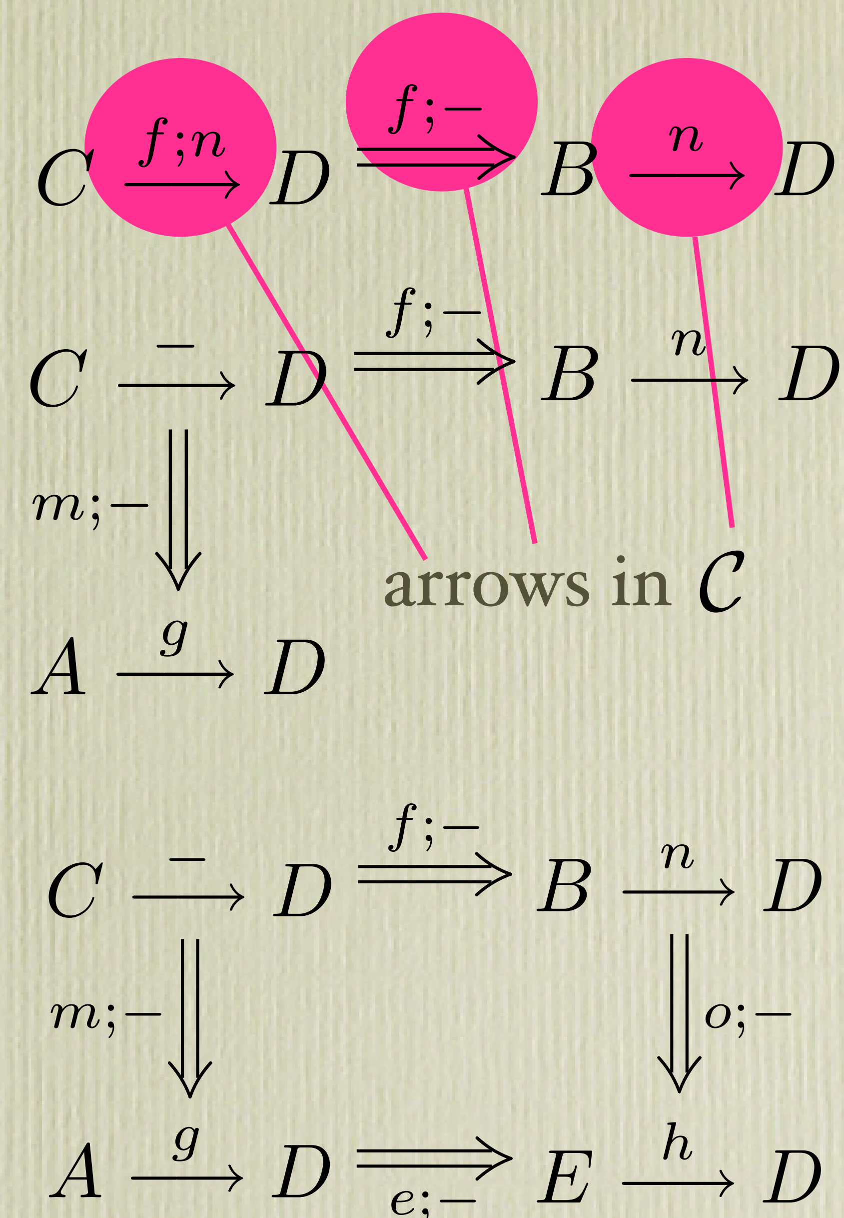
co-slice category

$(\mathcal{C} \Downarrow id)$

span of arrows (commutative diagram)

pushout (RPO)

(IPO iff E = D)



a derivation

$$
\begin{array}{ccc}
0 & \xrightarrow{\beta.0|\alpha.\overline{\beta}} & 1 \\
{\scriptstyle \alpha.\overline{\beta}|\overline{\alpha}}\downarrow & \swarrow & \downarrow{\scriptstyle [\_]|\overline{\alpha}} \\
1 & \xrightarrow[\beta.0|[\_]]{} & 1
\end{array}
$$

not necessarily a pushout,
but an IPO (no further
factorizing of the diagram)

$$\beta.0 \mid \alpha.\overline{\beta} \xRightarrow{[\_]|\overline{\alpha}} \beta.0 \mid \overline{\beta}$$

problems...

- what is the associated equivalence? Hard to assess...
- from rules to labelled rules, instead of labelled transitions

mixing graphs and processes

sad fact...

- the category with sets of process variables as objects and structurally congruent processes as arrows does not have all POs...
- (grupoidal) relative POs exist (this is the reason for their introduction), but then analysing the bisimilarity induced by the transition system is very hard...

cospans of adhesive cats

for adhesive $\mathcal{C}$ its cospan category has (G)RPOs

[SS05]

hence, an observational semantics
for graph rewriting systems

(subsumes [HK04])

sanity check: maps processes into graphs,
reduction rules into DPO rules, and
check out the resulting bisimilarity

[GM06]

we need a graph cospan...

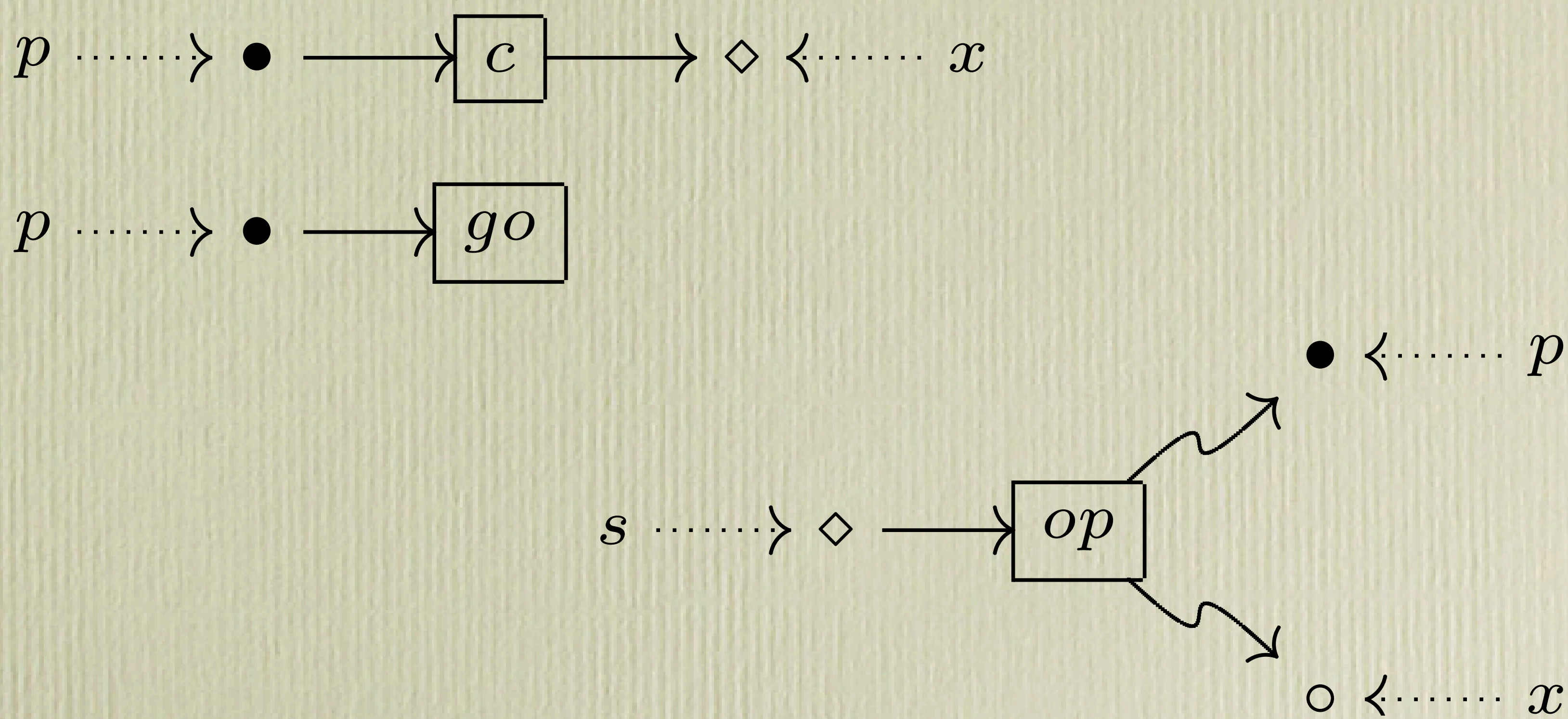
a discrete graph cospan *(r, G,v)* is a graph *G* with

- a function *v* from a set of variables *V* to *N*; and
- a function *r* from a from a set of roots *R* to *N*.

A process *P* is mapped to a discrete graph cospan *G(P)* with set of variables *fn(P)* and set of roots *{p}*

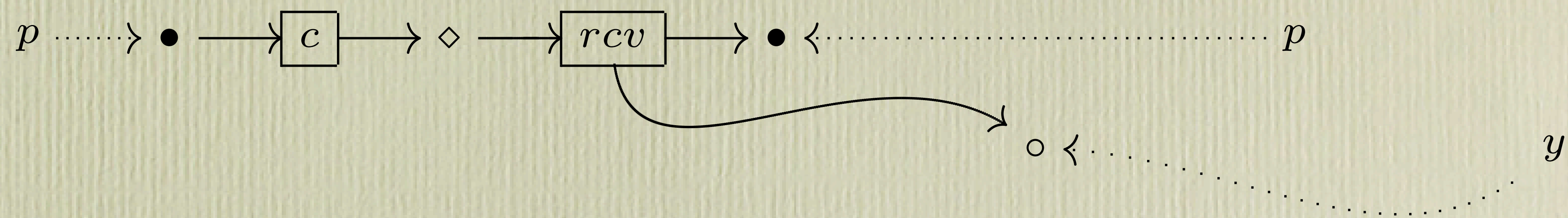
(in other terms, the functions trace the free names of a process, as well as its top operators)

...with four types of edges...

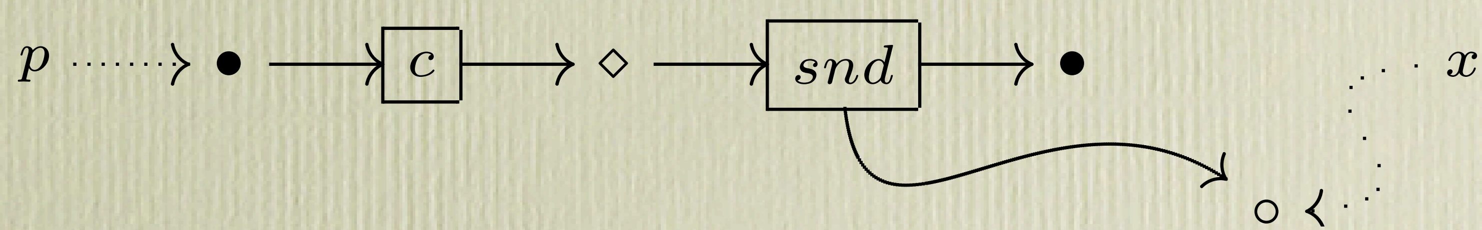


(for label *op* either *snd* or *rcv*)

..and the coalescing of nodes

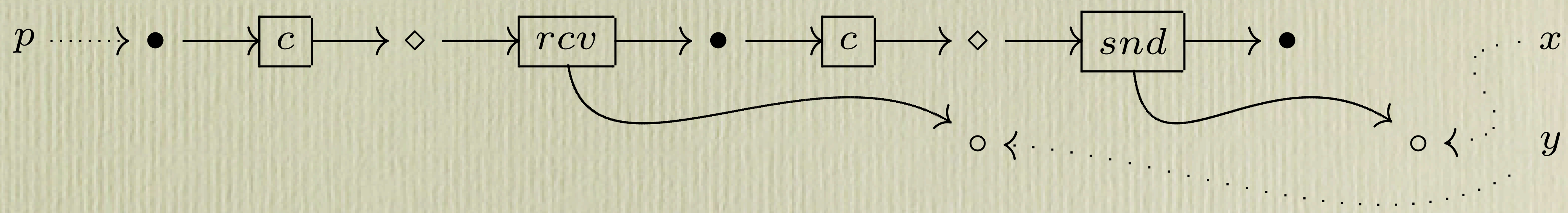


a graph cospan...



...another
graph cospan...

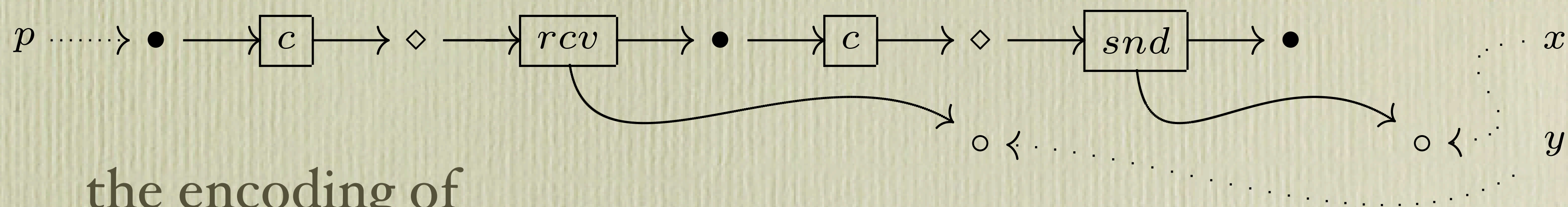
...and the coalescing of nodes.



...and their sequential composition!!

(the actual encoding for $y.\overline{x}.0$)

encoding restriction



the encoding of

$$y.\overline{x}.0$$

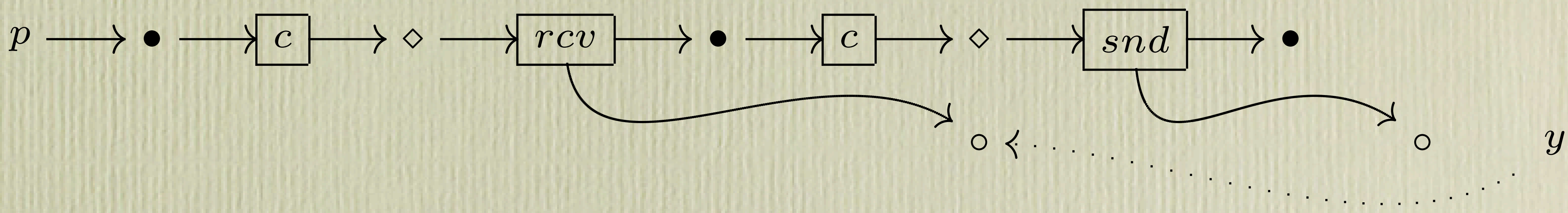
is post-composed with the cospan

$$x \dashrightarrow \circ$$

$$y \dashrightarrow \circ \dashleftarrow y$$

restriction takes a node out of the interface (making it convertible)

dealing with conversion



the graph is the encoding for $(\nu x)(y.\overline{x}.0)$

(and also for any renaming of the *x* variable !!)

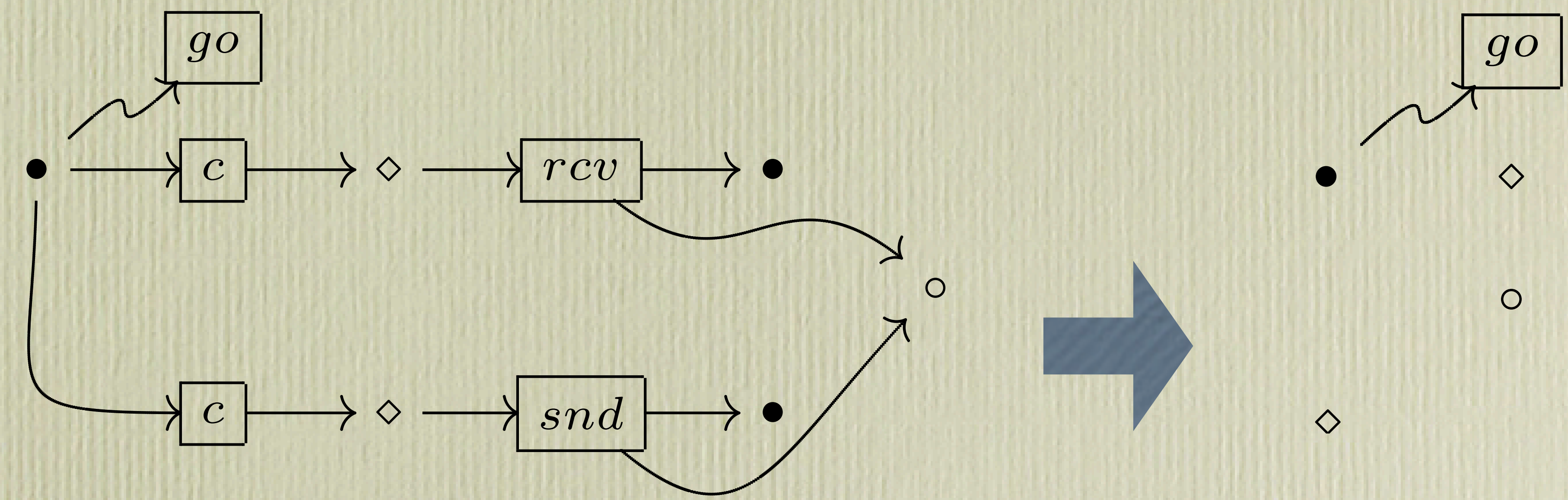
sound and complete

Let *P* be a process.

1. A graph cospan *G(P)* –i.e., the encoding for *P*– can be defined by induction on the operators of the calculus occurring in *P*

2. Moreover, let *R* be any other process. Then, *P* is structurally congruent to *R* if and only if *G(P)* is isomorphic as a graph cospan to *G(R)*.

the rewriting rule

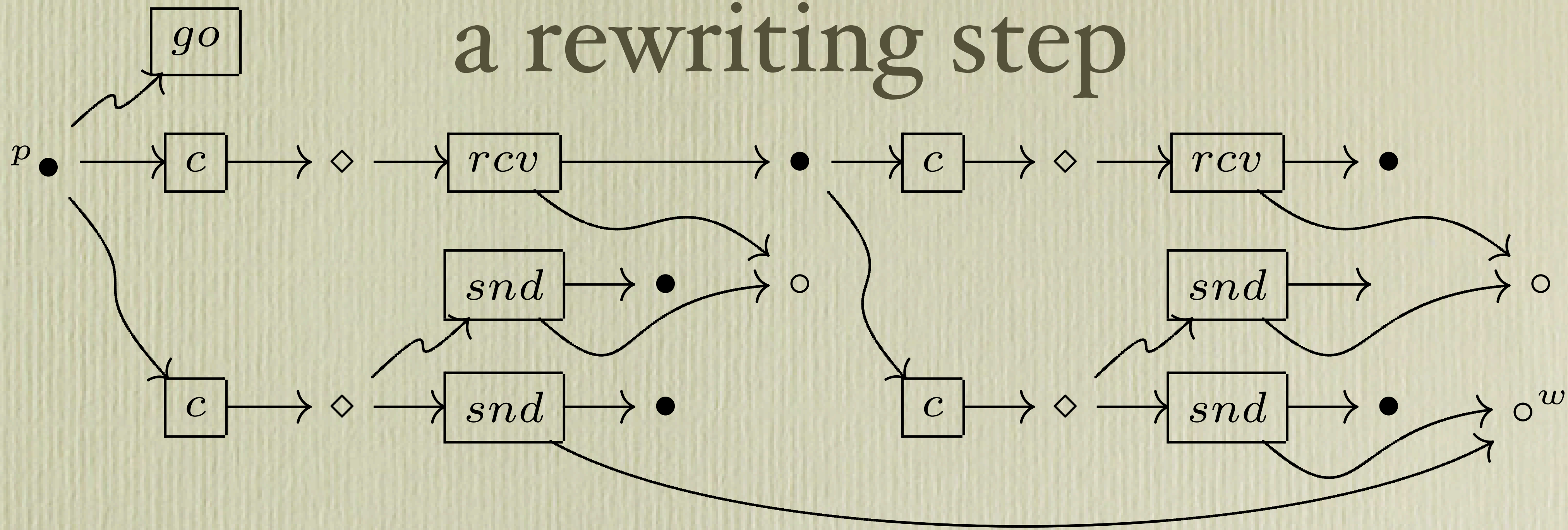


a redex is found

nodes are coalesced
edges are removed

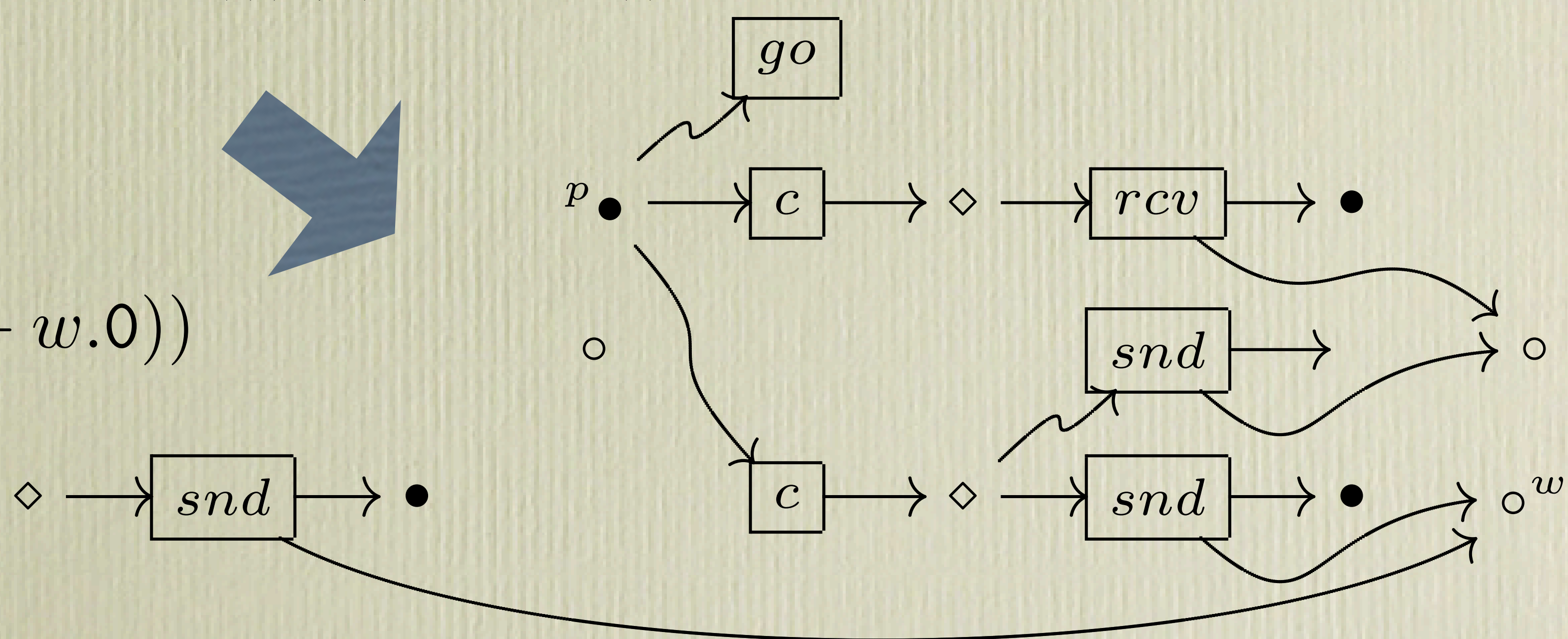
no other rule is needed
(graph morphisms take care of the context inside which the reduction might be occurring)

a rewriting step



$$(\nu x)(x.((\nu y)(y.0 \mid (\overline{y}.0 + w.0))) \mid (\overline{x}.0 + w.0))$$

$$(\nu y)(y.0 \mid (\overline{y}.0 + w.0))$$



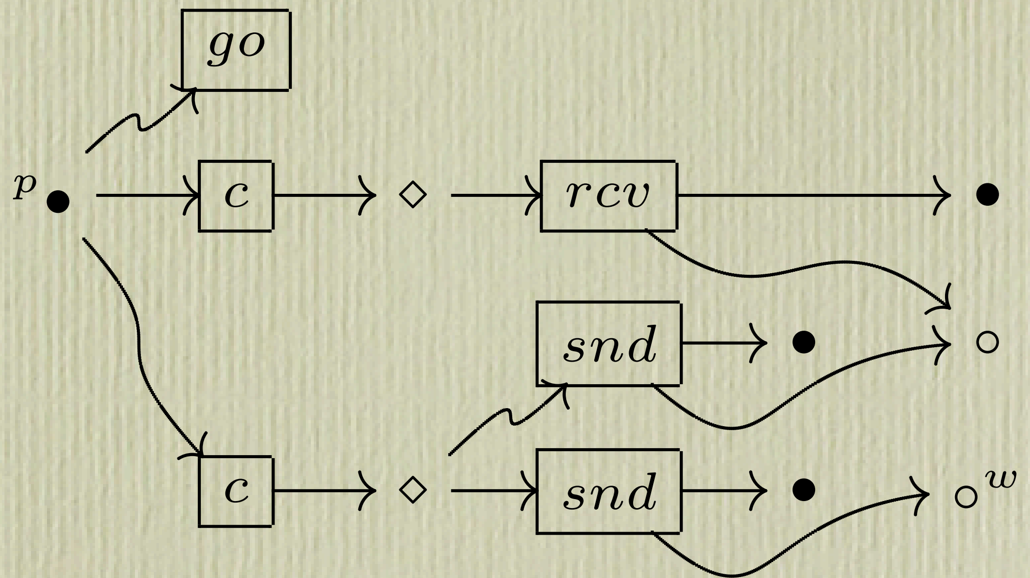
sound and complete

Let *P* be a process.

1. If *P* reduces to a process *R*, then there exists a graph cospan *H* and a rewrite from *G(P)* to *H*, such that *G(R)* is isomorphic as a graph to *H*, up-to some garbage collection.

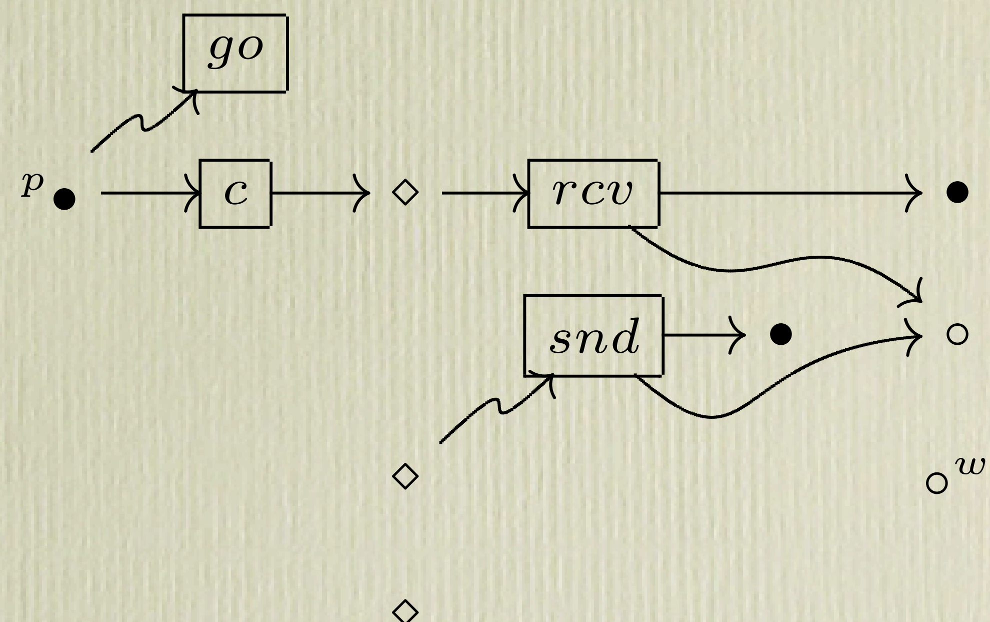
2. If *G(P)* rewrites to a graph cospan *H*, then there exists a process *R* and a reduction from *P* to *R*, such that *H* is isomorphic as a graph to *G(R)*, up-to some garbage collection.

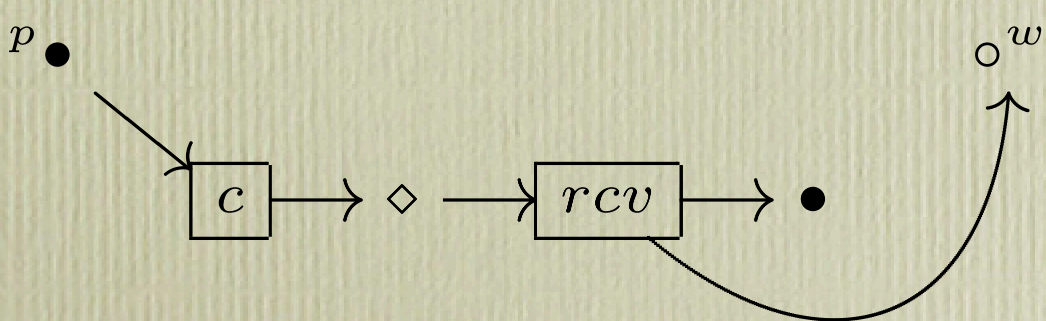
playing the RPO game



the source

$(\nu x)(\overline{x}.0 \mid (x.0 + w.0))$





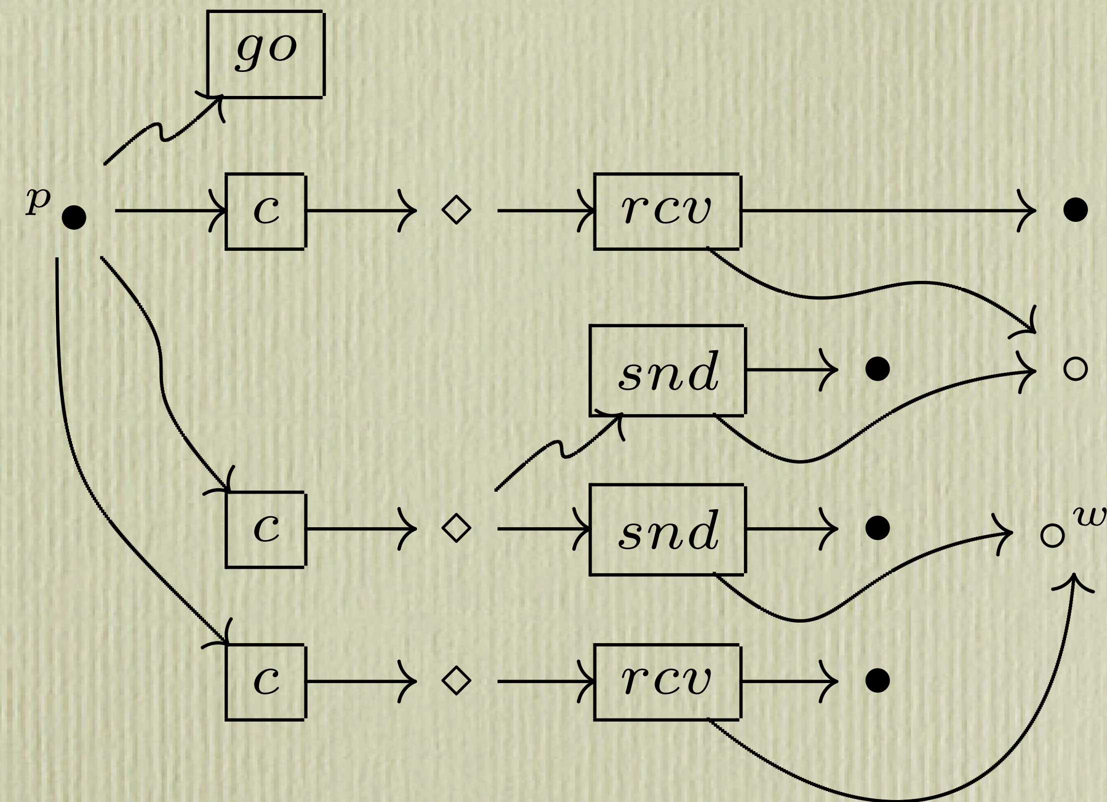
the label

$\overline{w}.0$

the target

$(\nu x)(\overline{x}.0)$

labels from contexts

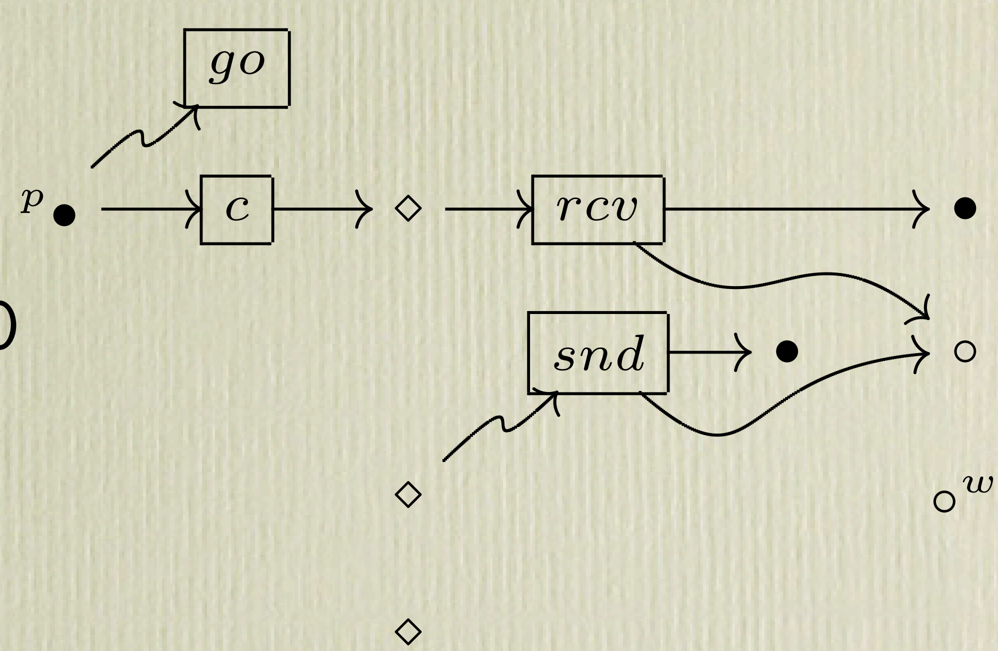


(the label is the minimal context allowing the rewriting step!!)

$$(\nu x)(\overline{x}.0)$$

$$(\nu x)(\overline{x}.0 \mid (x.0 + w.0)) \mid \overline{w}.0$$

rewrites to



most astonishingly...

- Let P, Q be (possibly recursive) CCS processes. Then, they are strongly bisimilar if and only if their graphical encodings are so. [BGK06]

- Bad thing: difficult!

- Good thing: first correspondence result of its kind for ANY recursive calculus!

current work

- Bisimilarity for other calculi (fusion, ...)
- Analysis for other graph-like adhesive categories
- Does these categories have a “terms as arrows, types as objects” characterization?