Composition and Substitution in Provenance and Workflows

~ or ~

Basic Boxology for Provenance

Peter Buneman
Adrià Gascón
Dave Murray-Rust
University of Edinburgh

Funded by: NSF Data Citation Project and EPSRC SOCIAM Project

The Old Chestnut: fine-grain vs coarse-grain provenance

Claimed for the PROV model, but no details given

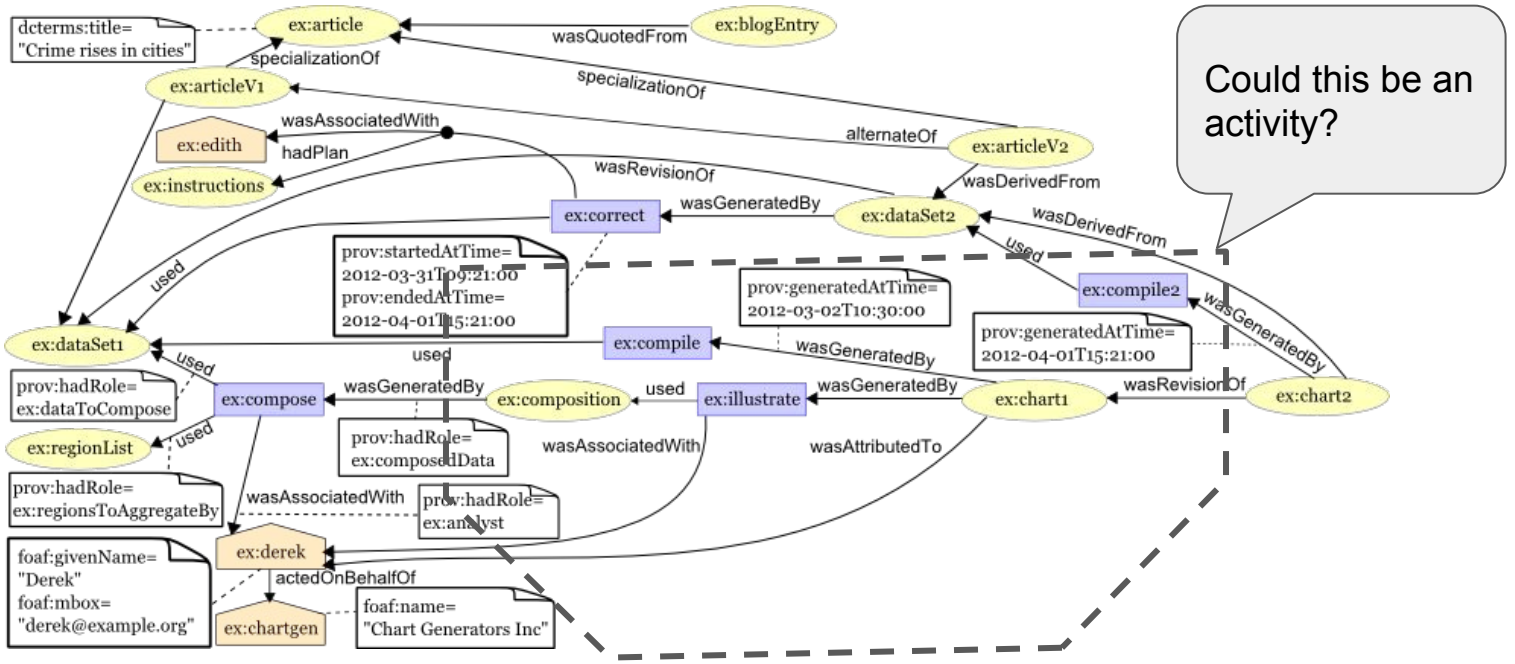
[Amsterdamer, Davidson, Deutch, Milo, Stoyanovich and Tannen. VLDB 2011] ties together data and workflow model and allows one to *view* provenance at various levels of granularity.

[Buneman, Cheney and Kostylev TAPP 2011] Form of graph homomorphism to explain granularity.

But ...

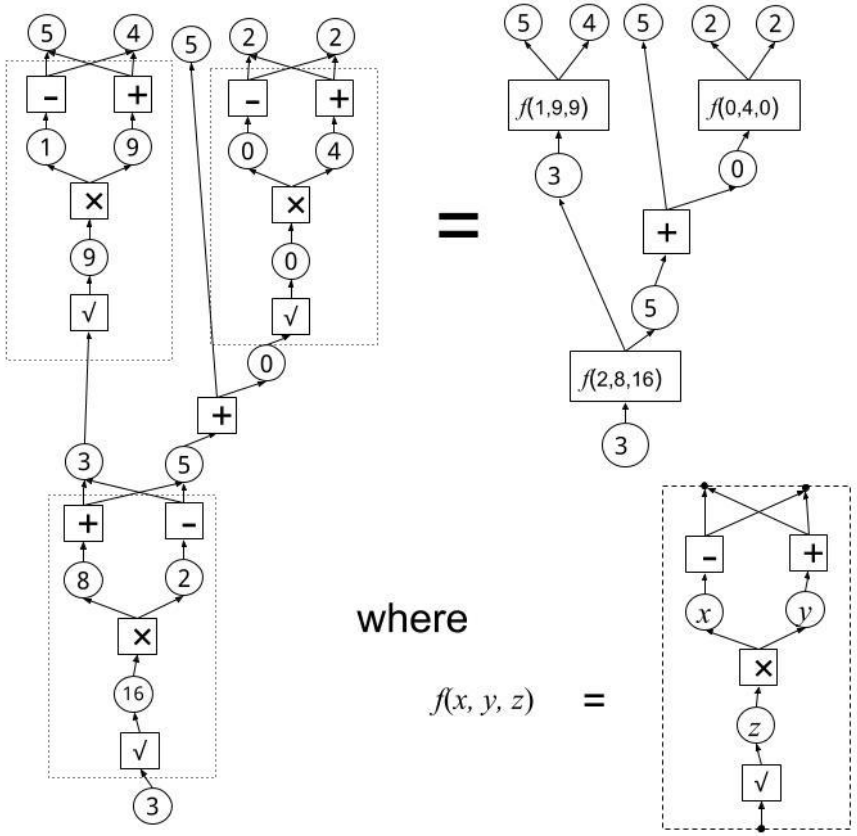
- How does one compose/combine provenance graphs?
- How does one create a subroutine in a workflow?
- Can we define a notion of substitution for provenance graphs?
- Can we build a “semantic” compressor for provenance graphs with repetitive substructures?

PROV. Any restriction on how activities can be grouped into larger activities?



Roughly speaking: none (thanks to Paolo Missier)

PROV substitution – perhaps for compression



We want to define some kind of graph substitution.

But there's a rather obvious "plumbing" problem if we stick to conventional (binary relation) graphs.

Graph substitution?

Computer scientists must have thought about this!

- [A] Alur, R., Benedikt, M., Etessami, K., Godefroid, P., Reps, T.W., Yannakakis, M.: Analysis of recursive state machines. ACM Trans. Program. Lang. Syst. 27(4) (2005)
- Drewes, F., Kreowski, H.J., Habel, A.: Hyperedge replacement, graph grammars. In: Handbook of Graph Grammars. (1997) 95–162

Variations on a model in which the nodes are *boxes* with labeled input and output ports

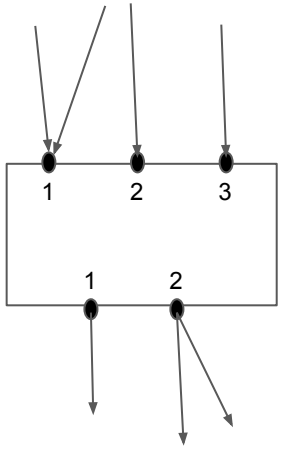
Somewhat unsurprisingly this model is used for provenance in

- [D] Davidson, S.B., Khanna, S., Milo, T., Panigrahi, D., Roy, S.: Provenance views for module privacy. PODS 2011

Both [A] and [D] prove results about dependency or connectivity under substitution

Model: A *network* consists of

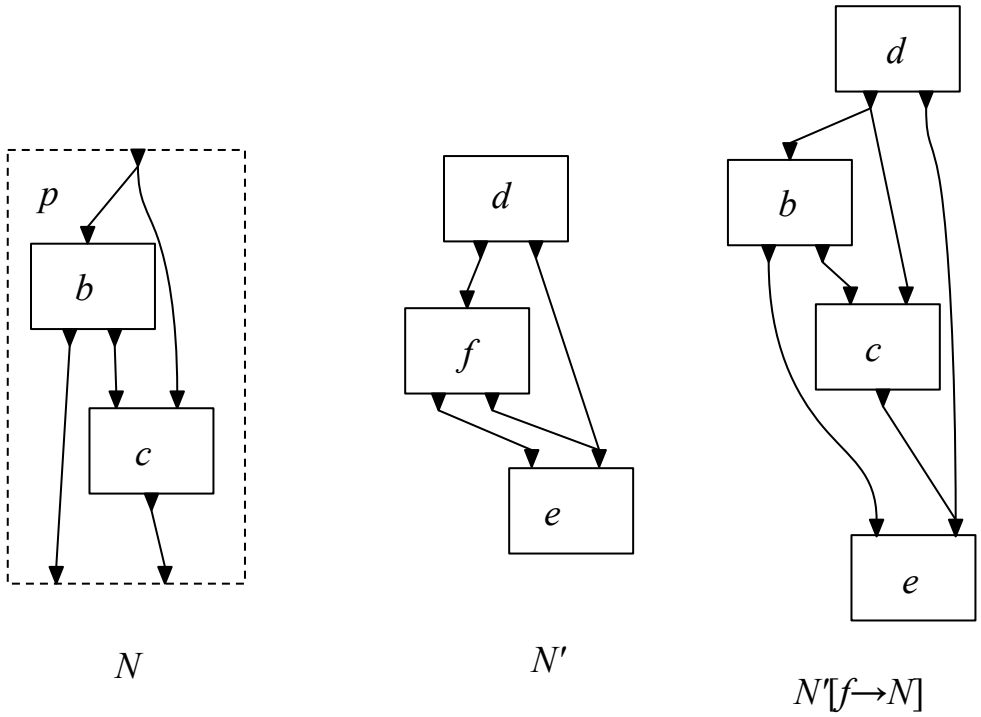
- *boxes*, with numbered input and output ports
- *edges*, that connect input to output ports



A 3-2 box

This is a big abstraction. We are assuming just one box shape and no box or edge labels.

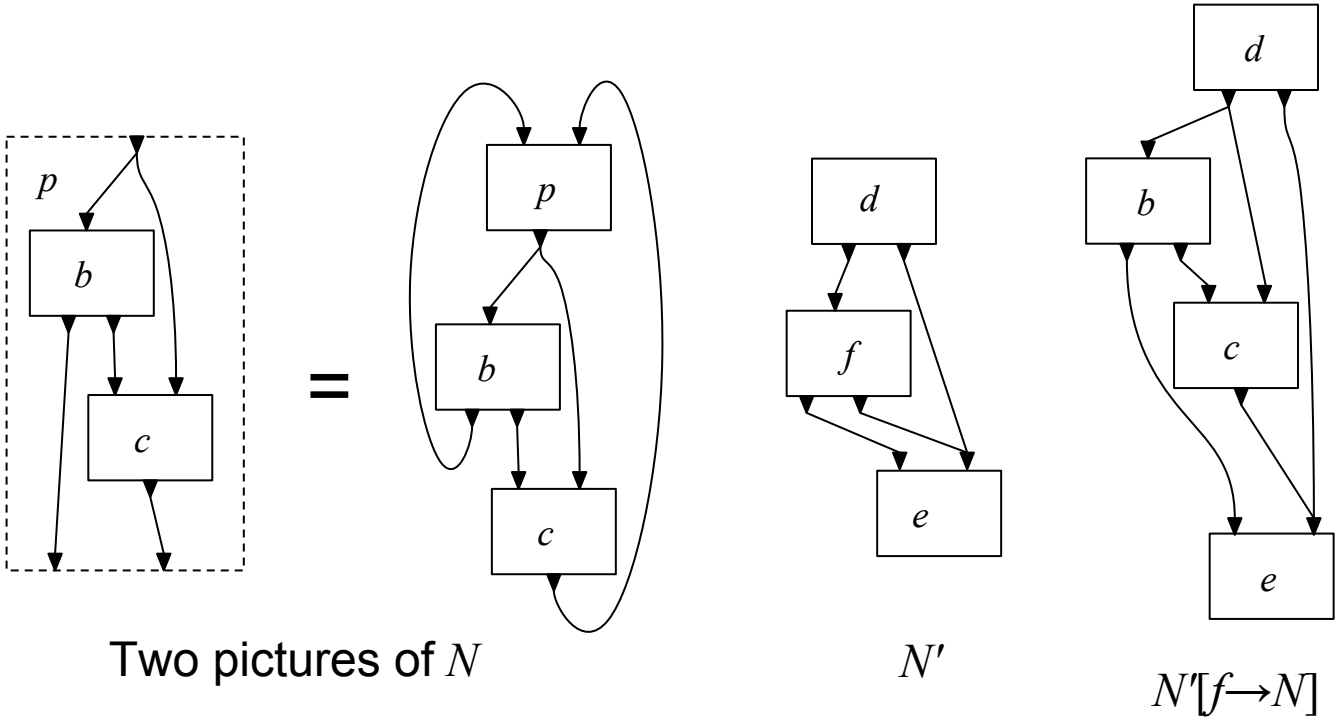
Substitution:



N N' $N'[f \rightarrow N]$

But we need two kinds of box!

Eversion



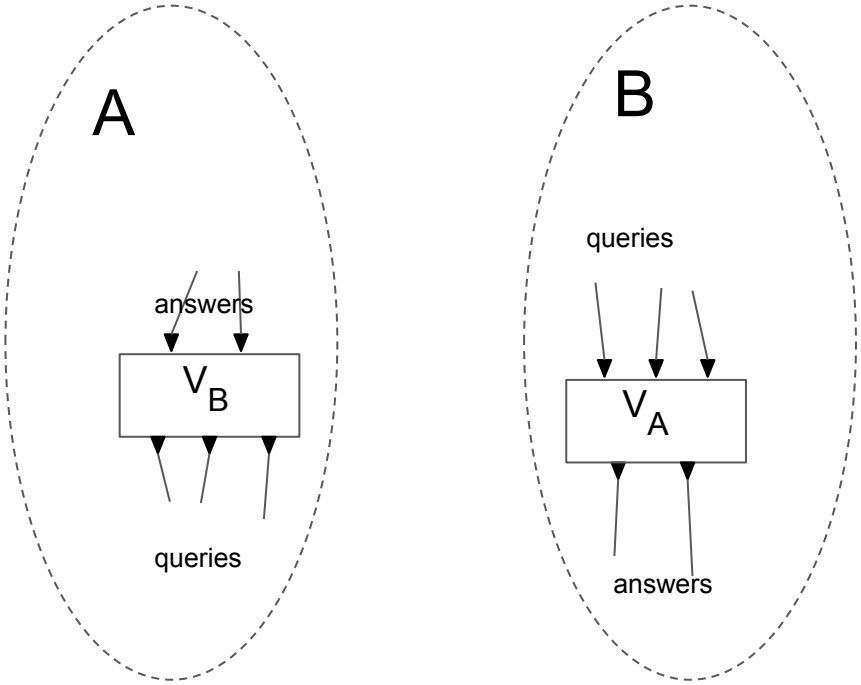
Two pictures of N N' $N'[f \rightarrow N]$

Now substitution looks different:

- Join the inputs of p to the corresponding outputs of f
- Join the inputs of f to the corresponding outputs of p
- Cause f and p to disappear

It's a symmetrical operation: $N[p, f]N'$ – a *join* of an (m,n)-box to an (n,m) box
(In fact the definition is a bit more subtle – need to deal with "self" edges.)

Do we need a join?



The combined graph is $A[V_B,V_A]B$

A represents a process that anonymizes and exports clinical data in response to queries from…

B, which represents the activity of a scientific research program

Neither A nor B can "look inside" each other.

A represents B as an (external) activity V_B and B represents A in as V_A

At some point B's research indicates that there is a patient at risk in A's database. Someone needs to combine the two provenance graphs!

Representing this in PROV is subject of ongoing work with Luc Moreau

A simple result

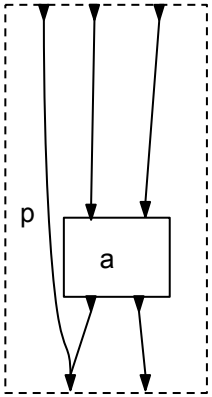
Let $M=(B_M,E_M)$ be a sub-network of $N=(B_N,E_N)$. There are networks $R=(B_M \cup \{a\}, E_M)$ and $S = (B_N - B_M \cup \{b\}, E_S)$ such that $N = R[a,b]S$

In other words: any subnetwork of a network can be treated as single box (b) and recovered with a join.

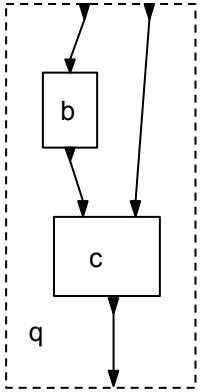
Moreover the recovery is "on the nose": it preserves the identity of the boxes

Is join fundamental or are there other more basic operations on networks?

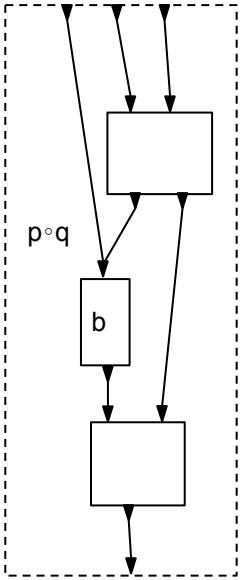
Parallel and serial composition



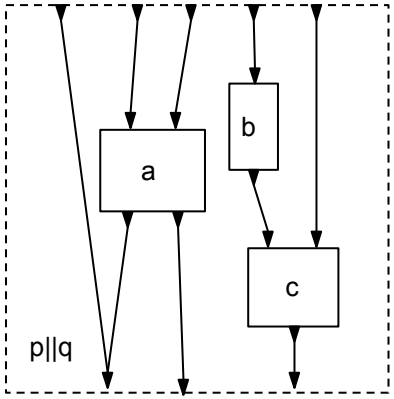
2-3 box



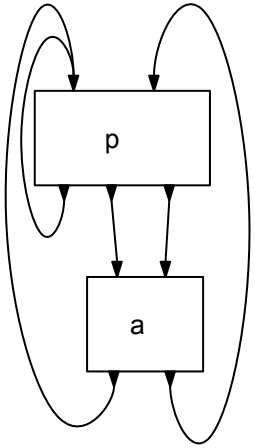
1-2 box

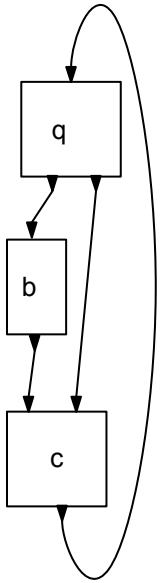


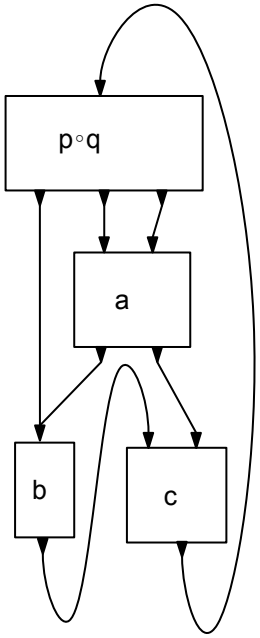
1-3 box



3-5 box







But there's a subtlety:

For substitution, $N'[f \rightarrow N]$, we may expect the new nodes from N to be disjoint from those in N' (we may explicitly copy them to ensure this)

In join $N[a, b]N'$ we do not want to assume that $N' \cap N = \varnothing$ for in any realistic situation (back-channels etc.)

For example:

If N and N' are strongly connected then so is $N[a,b]N'$ provided $N' \cap N = \varnothing$

Similar observations apply to parallel and serial composition

Further observations

A network is

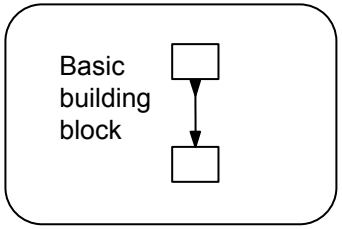
- *terminal free* if every box has at least one input and at least one output port
- *full* if every port has an incident edge
- *strongly connected* if there's a directed path from any box to any other

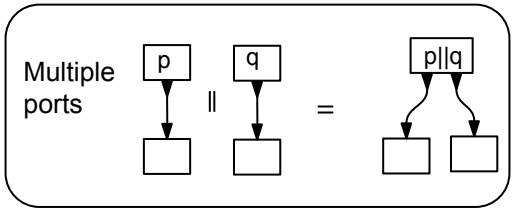
The first two are preserved under join, parallel composition and serial composition. The third is preserved if the networks are disjoint

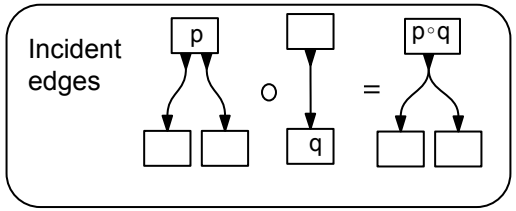
We will normally assume that our networks are full

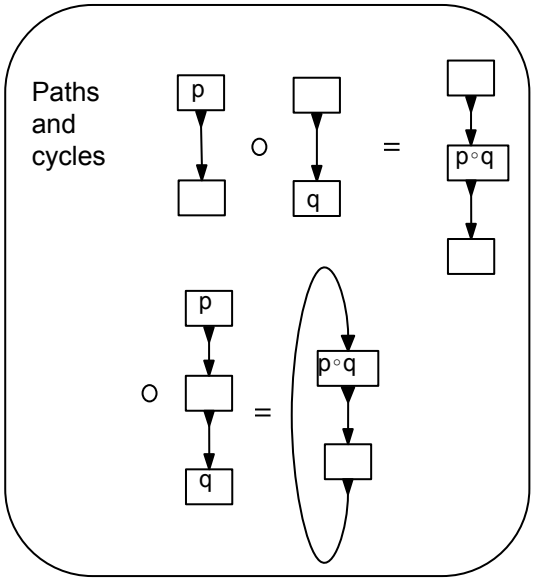
Interesting fact (cf series parallel graphs): any network can be constructed with series and parallel composition

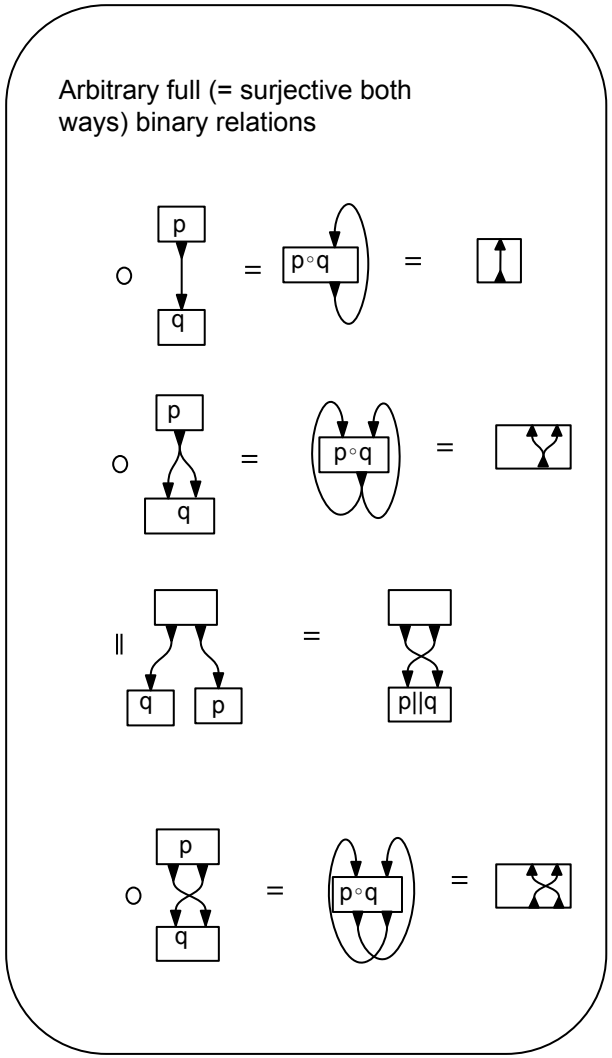
Legoland





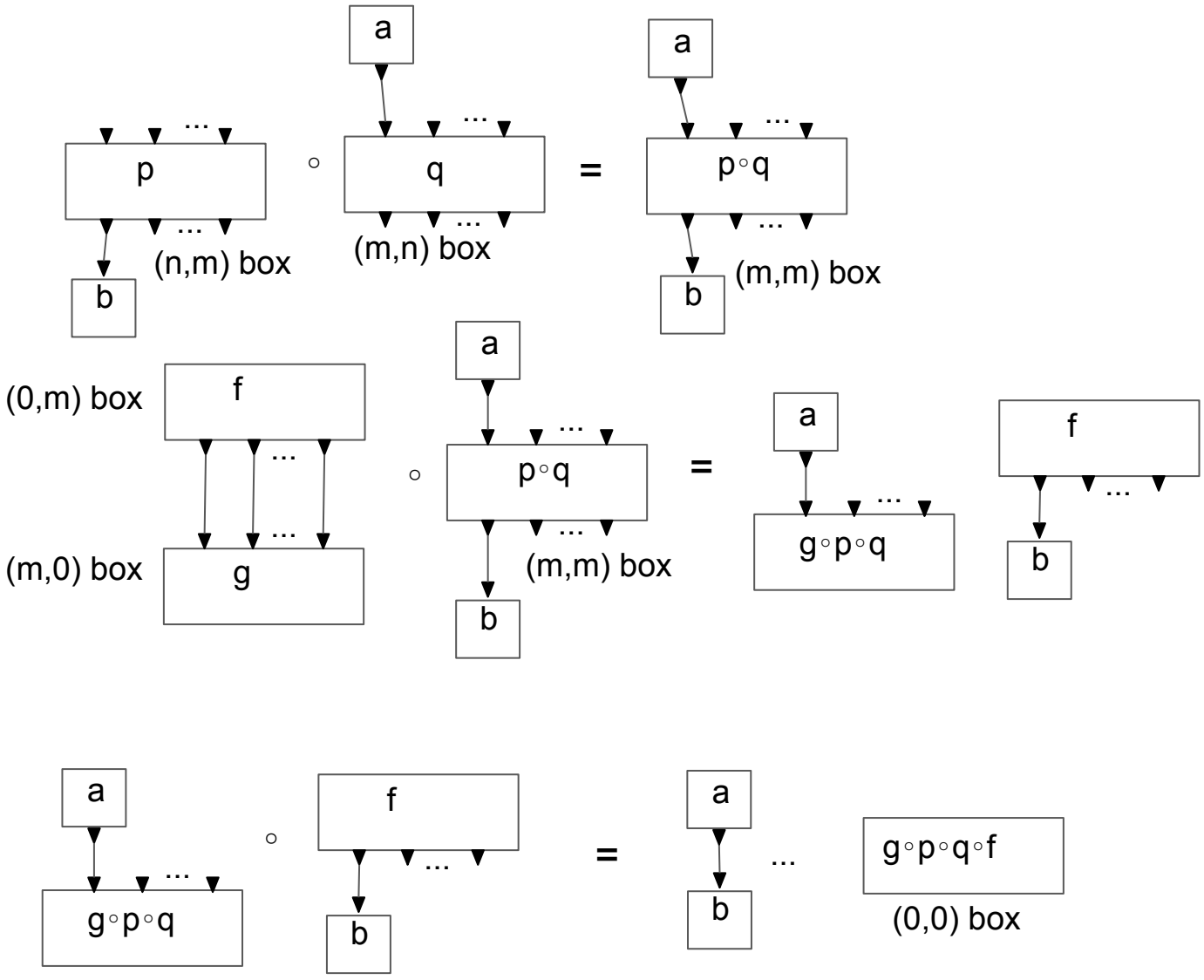




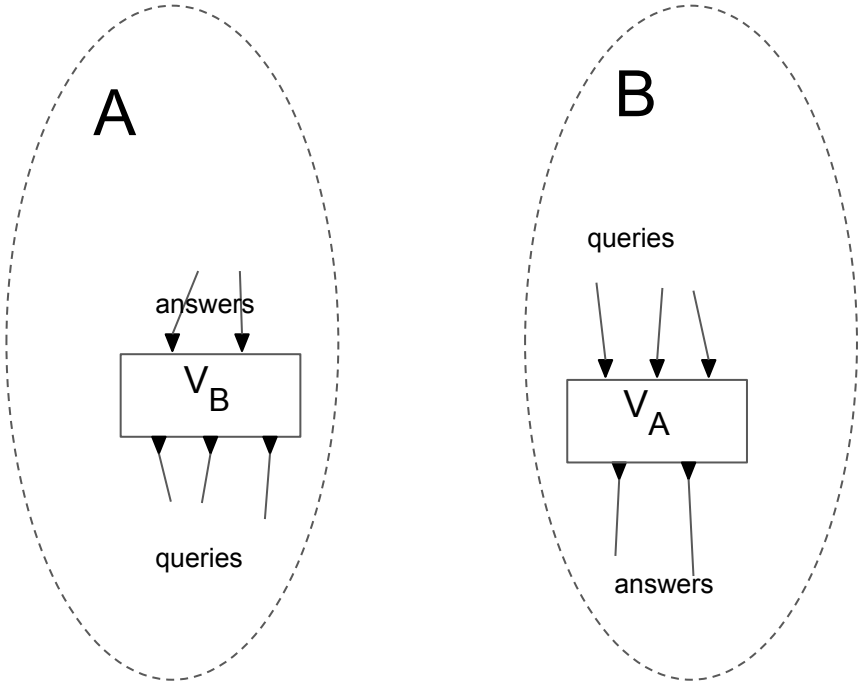


Any network can be constructed from (copies of) the basic building block with parallel and serial composition.

Join can be implemented “on the nose” with serial composition



Back to PROV and the problem of combining the provenance of communicating processes



We require

1. something like port numbers
2. account for node shapes, node labels and edge labels and rules for combining edge labels some algebra on edge labels

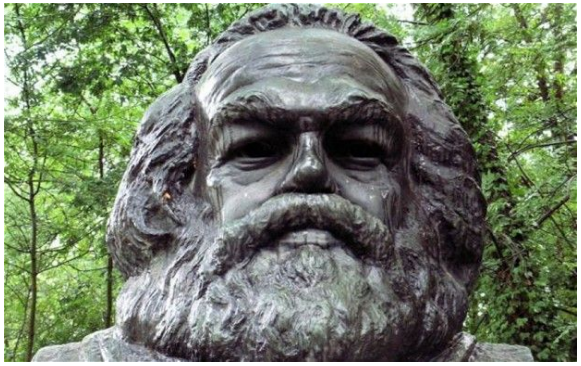
Instead of (1) we can identify messages by URIs

But we still need a way for each party to identify the other (where did the message go to / come from?)

Taken together, it's not so easy to represent a satisfactory join in PROV

Conclusions (minimal)

- Non-disjoint composition (parallel and serial) and join are needed at least for combining provenance graphs
- Relationship with series-parallel graphs? (Sarah-Cohen Boulakia's work on workflows)
  - Bao, Z., Cohen-Boulakia, S., Davidson, S. B., Eyal, A., & Khanna, S. (2009, March). Differencing provenance in scientific workflows. *ICDE'09.*
  - A (non-trivial) network is *rooted* at r if there is a path from every box to r
  - Rooted (parallel and serial) composition produce rooted graphs
  - Is there a legoland for rooted networks?
- A (linear) syntax for building networks (something like λ-terms)
- Language for graph substitution
- Reasoning about dependency and causality



“The computer scientists have only *interpreted* provenance, in various ways. The point, however, is to *construct* it.”

Thank you