A Process Algebra Approach to Provenance

James Cheney

Lab Lunch

February 7, 2006

What is process algebra?

- A symbolic system for describing and understanding the behavior of concurrent processes
- Examples: CSP, CCS, π-calculus, variants
- Basic ideas:

$P\|Q$	parallel composition
$P + Q$	choice
$c(x).P$	input x from channel c, then do P
$\bar{c}e.P$	output e to channel c, then do P

- Half of the audience has seen this before

What is provenance?

- Information about the creation, modification, derivation, or other history of something
- Real-world examples: birth certificate, passport stamps, travel stickers on luggage
- Digital examples: links, version control changelogs, email headers
- Problem: Requires extra effort, discipline to maintain provenance manually (especially for automatic processes); many users aren't that patient
- Self-reported provenance may be unreliable (dishonesty/laziness/human error)
- Half of the audience has seen this before

Provenance Challenges

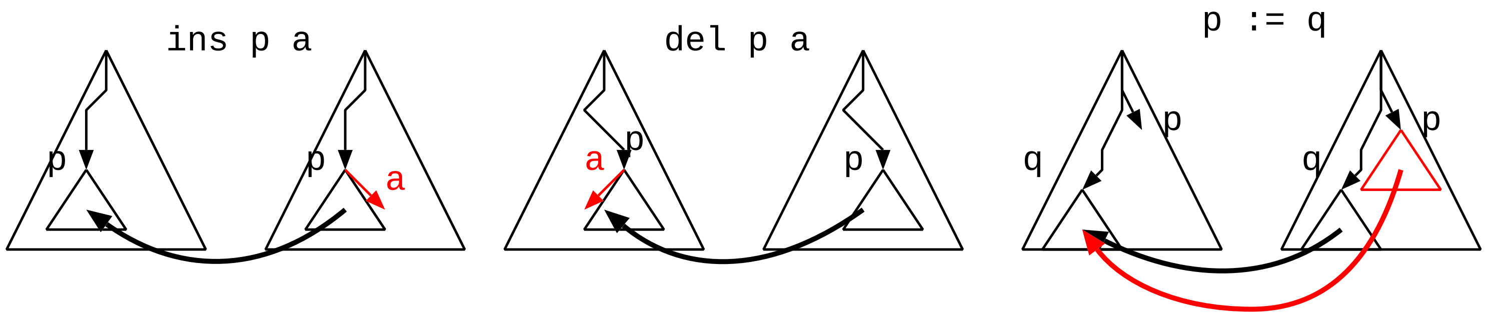
- There are (at least) two significant challenges in tracking and managing provenance
  - *Policy*: that is, what should we be doing, and how can we argue that what we do is sufficient/correct?
  - *Mechanism*: that is, how to build systems that effectively and efficiently capture (and exploit) provenance information?
- Most existing work focuses on (2), but without a good answer for (1), it's not clear to me how to evaluate an answer to (2).

Formal Foundation

- In a previous lab lunch Peter discussed work on provenance semantics for a simple tree update language

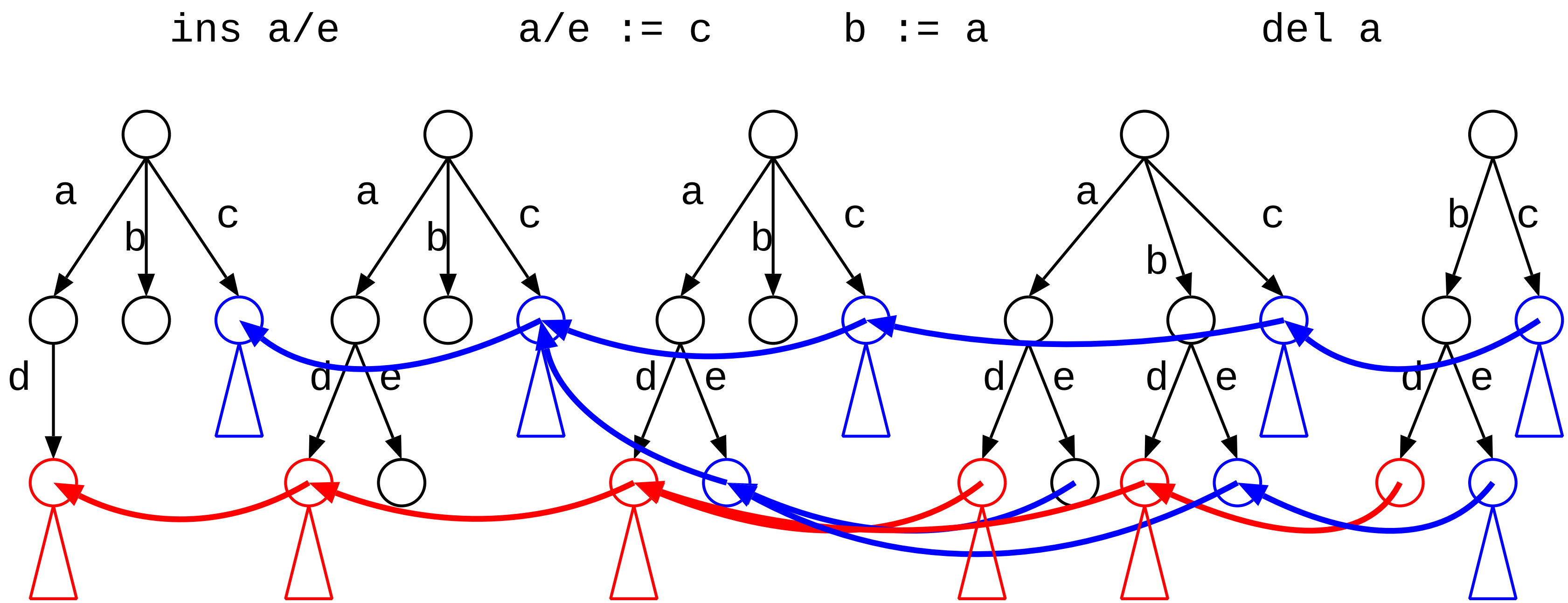
$$u ::= u; u' \mid \textsf{ins}\ p \mid \textsf{del}\ p \mid p := q$$

- *History* is the sequence of versions, with "links" between each version reflecting changes



Example

- Example



- (Many links omitted for readability.)
- So, we can tell that b/d was originally from under a, and that b/e is a copy of c.

How does this help?

- The history expresses the most detailed form of provenance information we are interested in.
- This can be used to evaluate other approaches w.r.t. accuracy and correctness.
- Key question:

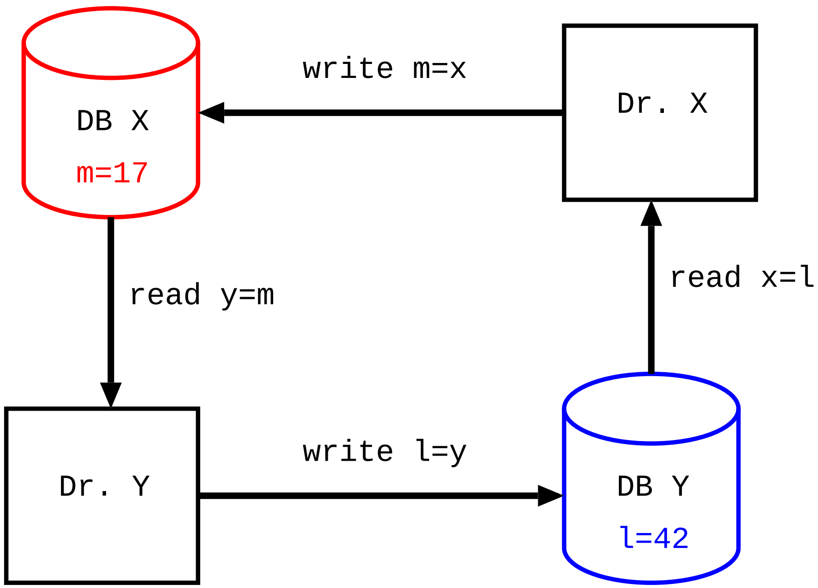
  Given a provenance tracking system, what questions about the history can be answered with certainty (given only the final state and provenance information)?

Problems

- Considered sequential language involving one database only
- Real world examples involve multiple databases, users acting in parallel
- This introduces many problems that don't come up in the single-threaded, single-database case.
  - multiple agents/authors
  - synchronization
  - concurrent queries/updates
  - read/write conflicts

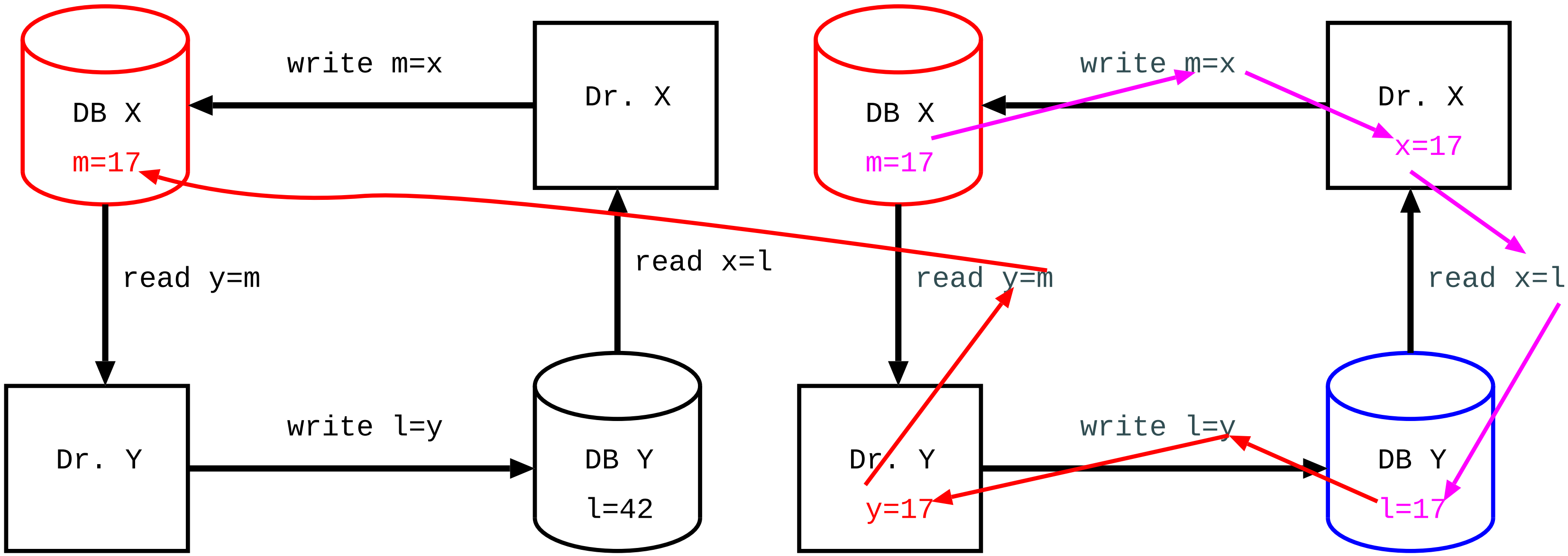
Example

- Consider following (realistic) situation:
  1. Dr. X copies some data from DB Y and incorporates it into DB X
  2. Meanwhile, Dr. Y updates DB Y with data from DB X



What happened?

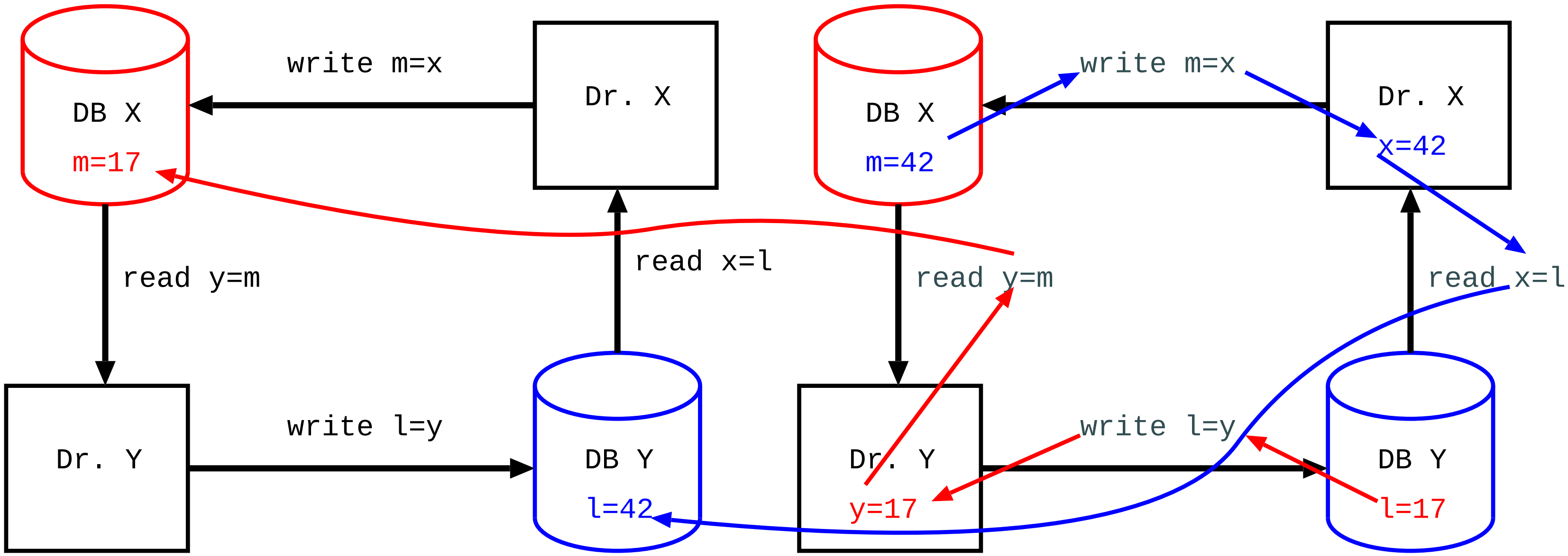
- Depending on the order of operations, there are several outcomes.



- Y reads and writes, then X reads and writes.

What happened?

- Depending on the order of operations, there are several outcomes.



- X and Y read, then X and Y write.

Process algebra to the rescue!

- Consider simple algebra for processes that communicate with databases (and each other)

$$
\begin{array}{rcl}
P & ::= & P|Q \mid P+Q \mid \alpha.P \mid 0 \\
\alpha & ::= & \mathsf{query}\ db\ e(x) \mid \mathsf{upd}\ db\ u \mid \bar{c}e \mid c(x) \\
e & ::= & l \mid n \mid x \\
u & ::= & l := e \mid \mathsf{ins}\ l = e \mid \mathsf{del}\ l \mid u; u'
\end{array}
$$

- db: database name, c: channel between processes
- To keep things simple, “databases” are just flat maps from labels l to integer values n.

$$\delta : Lab \rightharpoonup \mathbb{N}$$

Standard semantics

- Queries and updates on a database δ:

$$
\begin{aligned}
[\![n]\!](\delta) &= n \\
[\![l]\!](\delta) &= \delta(l) \\
[\![u; u']\!](\delta) &= [\![u']\!]([\![u]\!](\delta)) \\
[\![\mathsf{ins}\ l\ v]\!](\delta) &= \delta \uplus \{l \mapsto v\} \\
[\![\mathsf{del}\ l]\!](\delta) &= \delta - l \\
[\![l := v]\!](\delta) &= \delta[l := v]
\end{aligned}
$$

Note that expressions and updates must be ground (no free variables) when evaluated.

Standard semantics

- Configurations $\Delta; P$ consist of a collection of databases $\Delta = \{db_1 \mapsto \delta_1, \ldots\}$ and a process P
- All the standard process reduction steps lift:

$$\frac{P \to Q}{\langle \Delta; P \rangle \to \langle \Delta; Q \rangle}$$

- In addition, we have steps

$$\langle \Delta; \textsf{query}\ db\ e(x).P|Q \rangle \to \langle \Delta; P[[\![e]\!](\Delta(db))/x]|Q \rangle$$

$$\langle \Delta; \textsf{upd}\ db\ u.P|Q \rangle \to \langle \Delta[db := [\![u]\!](\Delta(db))]; P|Q \rangle$$

In both cases, transition only if $[\![e]\!](\Delta(db))$ or $[\![u]\!](\Delta(db))$ is defined.

Provenance semantics

- Idea: Annotate values with source information $db.t.l$, meaning "came from db at time t in location l". Annotations can also be empty ($\bot$).

$$
\begin{array}{rcl}
[\![n]\!]^\alpha(\delta) & = & n^\bot \\
[\![l]\!]^\alpha(\delta) & = & \delta(l)^l \\
[\![u;u']\!](\delta) & = & [\![u']\!]([\![u]\!](\delta)) \\
[\![\mathsf{ins}\ l\ v^\alpha]\!](\delta) & = & \delta \uplus \{l \mapsto v^\alpha\} \\
[\![\mathsf{del}\ l]\!](\delta) & = & \delta - l \\
[\![l := v^\alpha]\!](\delta) & = & \delta[l := v^\alpha]
\end{array}
$$

Provenance semantics

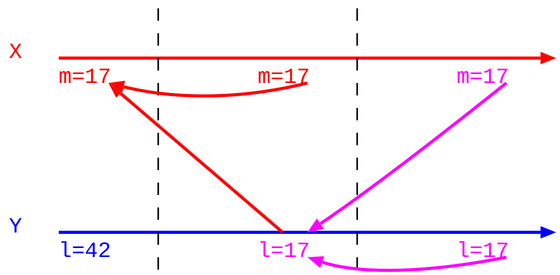
- Assume a global integer clock t (for simplicity).
- Use db and clock time to label data obtained via queries
- Clock steps only occur when a database is updated.

$$\langle t; \Delta; \mathsf{query}\ db\ e(x).P|Q\rangle \to \langle t; \Delta; P[[\![e]\!]^{db.t}(\Delta(db))/x]|Q\rangle$$

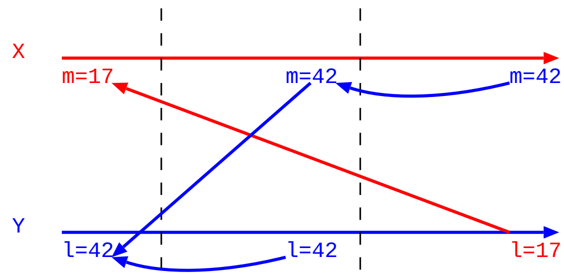
$$\langle t; \Delta; \mathsf{upd}\ db\ u.P|Q\rangle \to \langle t+1; \Delta[db := [\![u]\!](\Delta(db))]; P|Q\rangle$$

History

- We can now define a history of a configuration C as a sequence of configurations ending in C.
- Data can flow into processes, stay there for several time steps, then flow into a DB.
- This semantics can be used as a starting point for evaluating techniques for tracking provenance in a distributed setting.



vs



Conclusions

- Next steps: identifying interesting provenance systems and assertion languages, proofs of correctness.
- Extensions to process language to support locking may be needed
- Also, the synchronous time model is unrealistically simplistic (and distinguishes too much)
- Cryptographic protocols may be needed in non-cooperative settings.