Provenance Analyzer: Exploring Provenance Semantics with Logic Rules

by Saumen Dey, **Sean Riddle**, and Bertram Ludäscher

Overview

- The Problem
  - The Open Provenance Model
  - Toward a Temporal Semantics
  - Provenance Analyzer prototype
- Approach
  - Deductive
  - Abduce Time
  - Abduce Partial Order

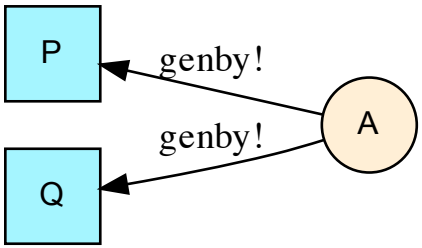
Open Provenance Model

- Graph model, N = P (*processes*) U A (*artifacts*)
- Edge types
  - A genby P (*artifact* A *was generated by process* P)
  - P1 informedBy P2 (*process* P1 *influenced the execution of process* P2)
  - A1 derivedFrom A2 (*artifact* A1 *used* A2 *in its creation*)
  - P used A (*process* P *used artifact* A)
- Time is optional annotation
- Legality
  - Each datum must have one immediate creating process
  - Broken use-generate-derive triangles [1] are not allowed
- Few completeness restrictions (a trace does not have to say very much – it is difficult to tell at a glance how detailed/ restrictive a trace is)

Legal OPM Trace

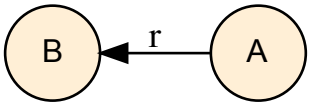
- Each artifact can only be precisely generated by one process

Invalid!

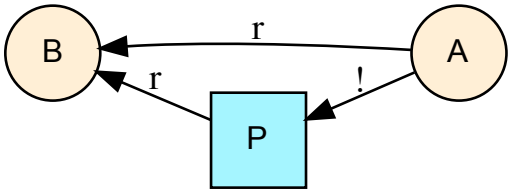


- Each precise derived from edge is part of a use-generate-derive triangle

If…

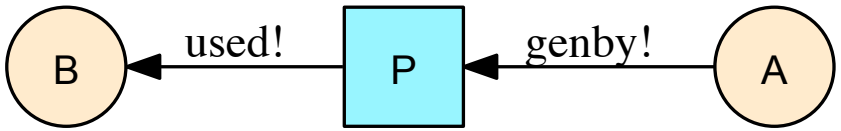


then,

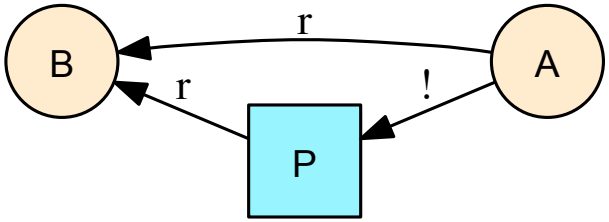


Example

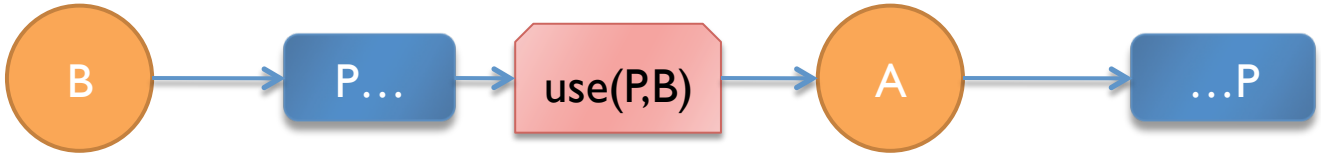
- ! Indicates edge has a role, used in algorithm in [1], use-generate-derive triangles



Example

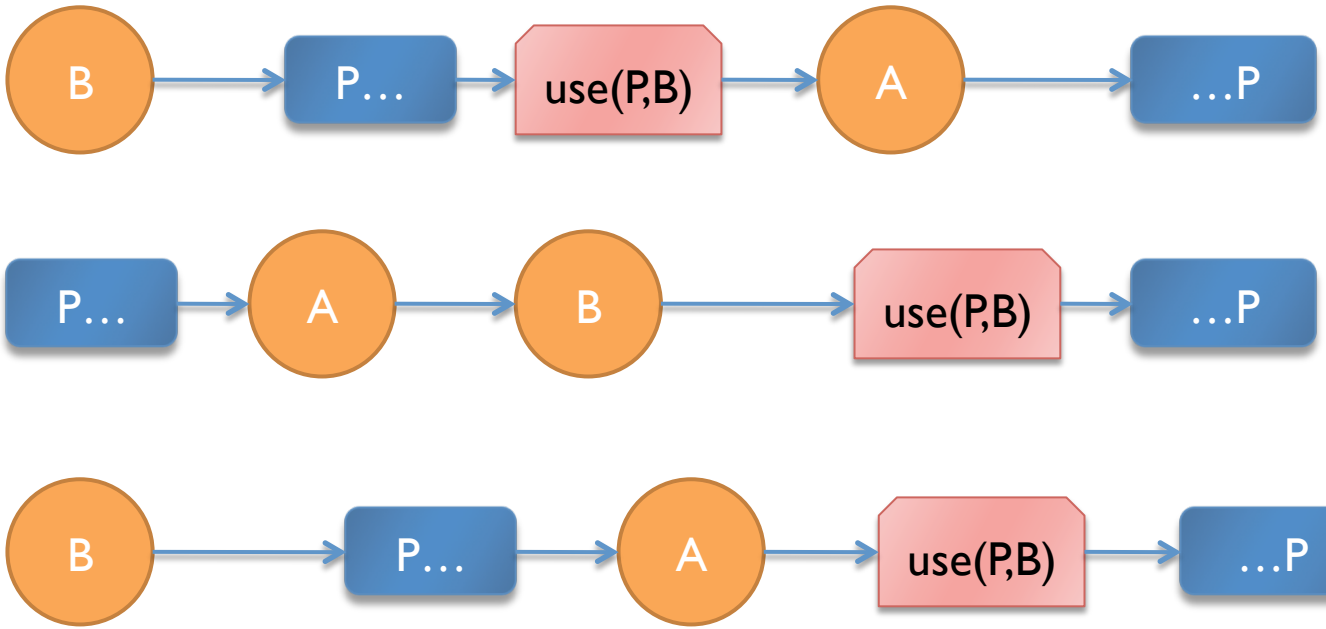


- ! Indicates edge has a role
- Forms a use-generate-derive triangle, which encodes additional semantics according to [1]

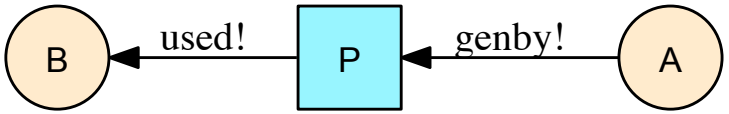


• P used B to create A

Many Different Possible Worlds



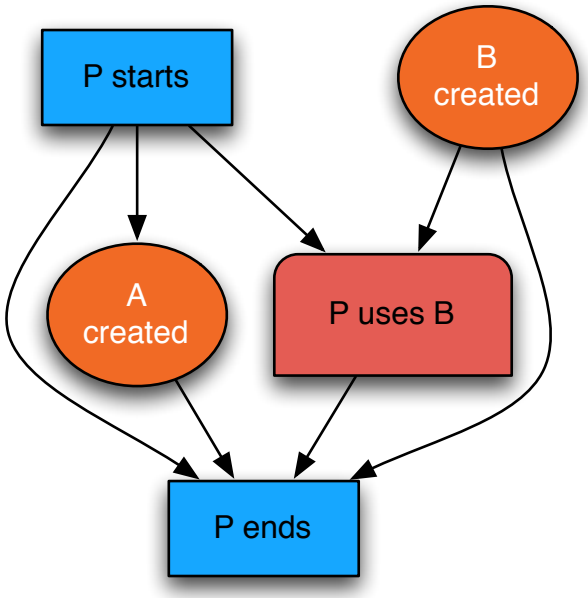
(5 models elided)



Deductive Algorithm

- From Kwasnikowska et al.[1]
- Graph patterns → Temporal constraints
- .

$$(\text{AX } 2)\ \frac{\begin{array}{c}P\\ \uparrow !\\ A\end{array}}{\text{begin}(P) \preceq \text{create}(A) \preceq \text{end}(P)}$$



Implementation

- Implemented in ASP (answer set programming) system DLV: dlvsystem.com
- Instrumented/interconnected with Python
- Sets of rules run in different combinations:
  - Definitions of axioms in different terms (deductive, abductive over different structures)
  - Definitions of OPM legality
  - Rules to generate linear extensions of a partial order (ie., total orders that obey the partial order)

ASP Example

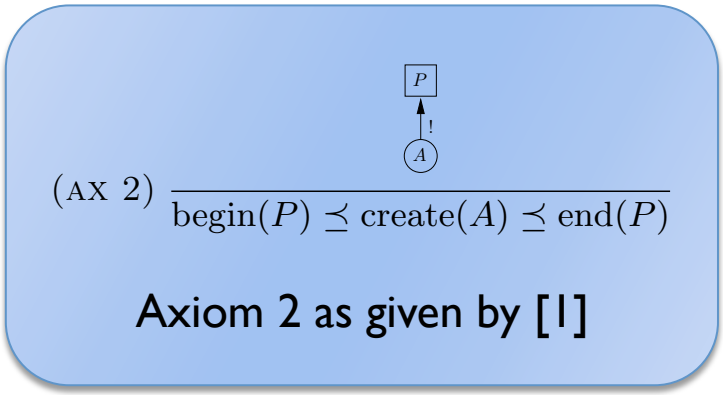
- a :- not b.
  b :- not a.
- Evaluate bodies of rules wrt model; does it produce same model?
- Is {a} a model? Yes.
  - a :- true
  - b :- false
- Is {b} a model? Yes.
  - a :- false
  - b :- true
- Is {a, b} a model? No.
  - a :- false
  - b :- false

Implementation - Deductive

- Implemented axioms and temporal inference rules from [1]
- Deductive generation of constraints from graph structure

```
leq(beginP, createA) :-
  pGenBy(a, p).
leq(createA, endP) :-
  pGenBy(a, p).
```

Axiom 2 prototype implementation



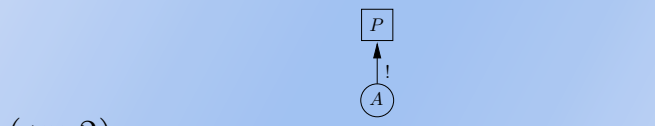
Axiom 2 as given by [1]

Implementation – Abduce Time

- Abductive approach guesses mapping of events to time points and throws out violators.

```
false :-
   pGenBy(a, p),
   at(createA, TA),
   at(beginP, TP),
   TP > TA.
false :-
   pGenBy(a, p),
   at(createA, TA),
   at(endP, TP),
   TA > TP.
```

Axiom 2 prototype implementation



$$(\text{AX } 2)\ \frac{}{\text{begin}(P) \preceq \text{create}(A) \preceq \text{end}(P)}$$

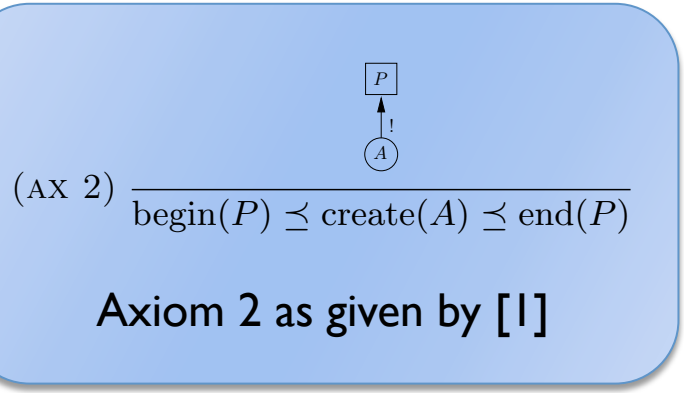
Axiom 2 as given by [1]

Implementation – Abduce Partial Order

- Abductive approach guesses mapping of events to events and then throws out all that do not describe a partial order (or total order if desired).

```
false :-
  pGenBy(a, p),
   after(beginP, createA).
false :-
  pGenBy(a, p),
  after(createA, endP).
```

Axiom 2 prototype implementation

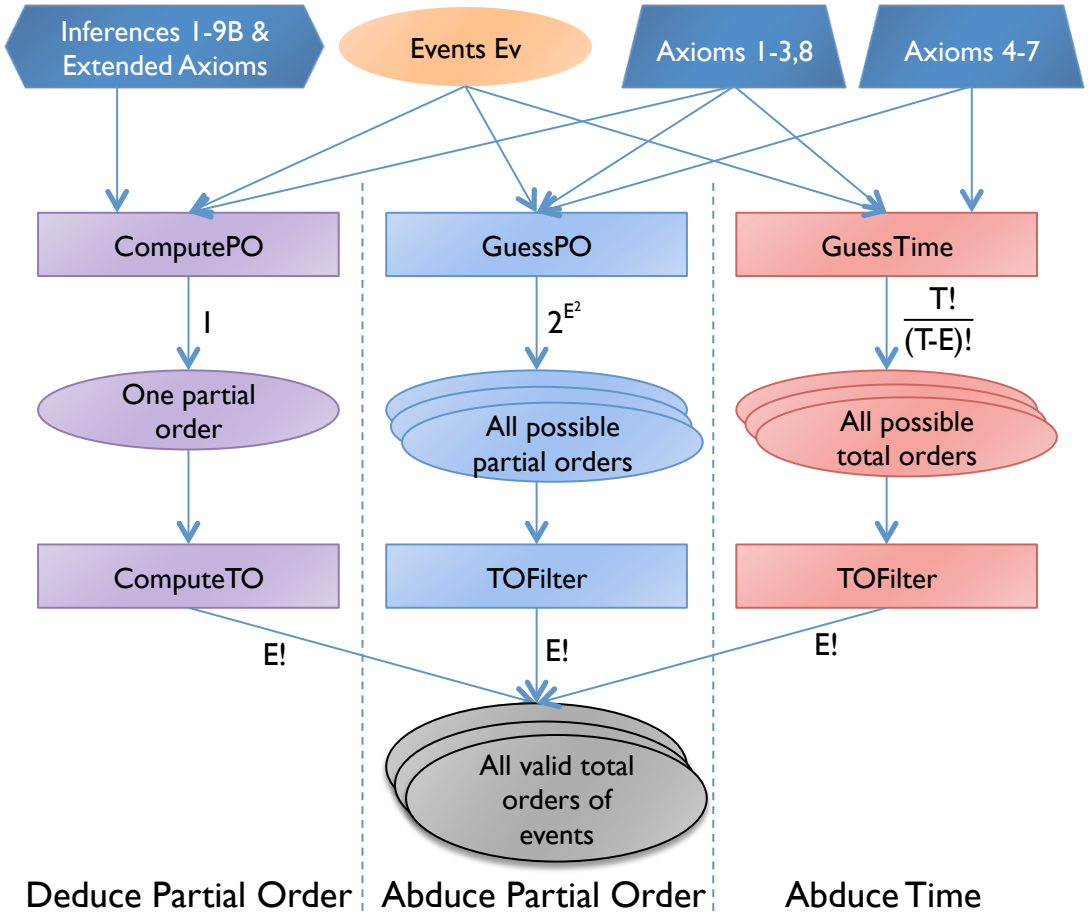


Axiom 2 as given by [1]

Abduce Time vs. Abduce Partial Order

- Abduce Time faster on small examples, sensitive to definition of time points (too many can cause drastic slowdown)
- Abduce Time allows for constraints that depend on time, eg., creation events can only occur with a certain maximum frequency

Design


Inferences 1-9B & Extended Axioms
Events Ev
Axioms 1-3,8
Axioms 4-7
ComputePO
GuessPO
GuessTime
1
2^{E^2}
$\frac{T!}{(T-E)!}$
One partial order
All possible partial orders
All possible total orders
ComputeTO
TOFilter
TOFilter
E!
E!
E!
All valid total orders of events
Deduce Partial Order
Abduce Partial Order
Abduce Time

Conclusions and Future Work

- Provides measure of constrained-ness (how many total orders of events correspond to this trace?)
- Can query what is true in some/all models (analogous to brave/cautious reasoning in LP)
- Allows experimentation with axioms
- Future work
  - Combinations of axioms that describe easily characterizable subsets of temporal constraints (eg., no-use inequalities)
  - Increase performance with subsumption

Provenance Analyzer

- Available at: http://code.google.com/p/provenance-analyzer
- Works Cited

1) Kwasnikowska, Natalia, Luc Moreau, and Jan Van den Bussche. "A formal account of the open provenance model." University of Southampton, ECS Technical Report 21819 *(2010)*.
2) Moreau, Luc, et al. "The open provenance model core specification (v1. 1)." *Future Generation Computer Systems 27.6 (2011): 743-756.*

- Acknowledgements

  - Work supported in part by NSF awards DGE-0841297, OCI-0830944, and IIS-1118088.