

On the Expressiveness of Implicit Provenance in Query and Update Languages

Peter Buneman¹ James Cheney¹ Stijn Vansummeren²

¹University of Edinburgh, Scotland

²Hasselt University and Transnational University of Limburg, Belgium

Chris curates a DB

- create table DB(Beer, Type, Origin)
- insert into DB values(...)
- insert into DB
select * from S where origin='USA'
- insert into DB
select * from T where type='stout'

Beer	Type	Origin

Chris curates a DB

- create table DB(Beer, Type, Origin)
- insert into DB values(...)
- insert into DB
select * from S where origin='USA'
- insert into DB
select * from T where type='stout'

Beer	Type	Origin
Duvel	blond	Belgium

Chris curates a DB

- create table DB(Beer, Type, Origin)
- insert into DB values(...)
- insert into DB
select * from S where origin='USA'
- insert into DB
select * from T where type='stout'

Beer	Type	Origin
Duvel	blond	Belgium
Heineken	blond	USA
Bud	blond	USA

Chris curates a DB

- create table DB(Beer, Type, Origin)
- insert into DB values(...)
- insert into DB
`select * from S where origin='USA'`
- insert into DB
`select * from T where type='stout'`

Beer	Type	Origin
Duvel	blond	Belgium
Heineken	blond	USA
Bud	blond	USA
Guinness	stout	Ireland

Chris curates a DB

- create table DB(Beer, Type, Origin)
- insert into DB values(...)
- insert into DB
select * from S where origin='USA'
- insert into DB
select * from T where type='stout'

Beer	Type	Origin
Duvel	blond	Belgium
Heineken	blond	USA
Bud Guinness	blond stout	USA Ireland

Chris curates a DB

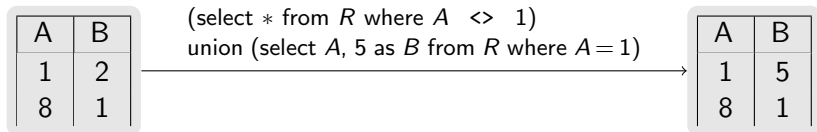
- create table DB(Beer, Type, Origin)
- insert into DB values(...)
- insert into DB
select * from S where origin='USA'
- insert into DB
select * from T where type='stout'

Beer	Type	Origin
Duvel	blond	Belgium
Heineken	blond	USA
Bud	blond	USA
Guinness	stout	Ireland

- **Provenance** is vital for assessing trustworthiness
- Manual provenance recording is tedious and error-prone
- Automatic provenance recording support?

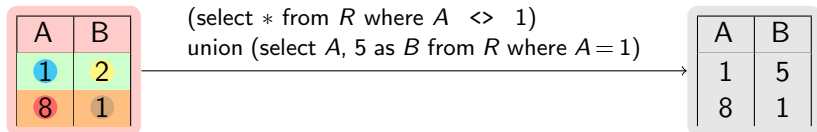
Implicit where-provenance

- Wang and Madnick (VLDB 1990) and Bhagwat et al. (VLDB 2004) for **atomic data values** in queries.
- This talk: provenance for tables, tuples, and atomic data values in queries and updates.



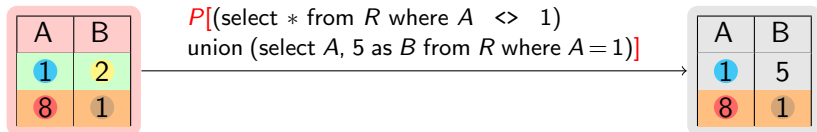
Implicit where-provenance

- Wang and Madnick (VLDB 1990) and Bhagwat et al. (VLDB 2004) for **atomic data values** in queries.
- This talk: provenance for tables, tuples, and atomic data values in queries and updates.



Implicit where-provenance

- Wang and Madnick (VLDB 1990) and Bhagwat et al. (VLDB 2004) for **atomic data values** in queries.
- This talk: provenance for tables, tuples, and atomic data values in queries and updates.



1 Provenance for Queries

2 Provenance for Updates

- **Objects:**

atoms – records – sets

- **Complex object queries:**

$$e ::= x \mid a \mid (A: e, \dots, B: e') \mid e.A \\ \mid \{e, \dots, e'\} \mid \cup e \mid \{e' \mid x \in e\} \mid \text{if } e_1 = e_2 \text{ then } e_t \text{ else } e_f$$

- **Objects:**

atoms – records – sets

- **Complex object queries:**

$$e ::= x \mid a \mid (A: e, \dots, B: e') \mid e.A \\ \mid \{e, \dots, e'\} \mid \bigcup e \mid \{e' \mid x \in e\} \mid \text{if } e_1 = e_2 \text{ then } e_t \text{ else } e_f$$

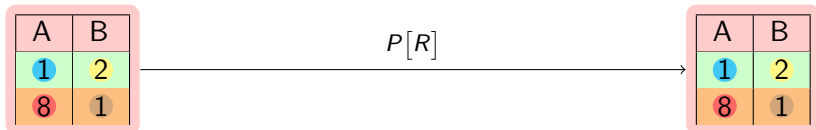
Example

$$\begin{aligned} & \text{select } * \text{ from } R \text{ where } A \lt; \! > 1 \\ & \quad \equiv \\ & \bigcup \{ \text{if } x.A = 1 \text{ then } \emptyset \text{ else } \{x\} \mid x \in R \} \end{aligned}$$

Provenance for queries

Queries **create** new objects

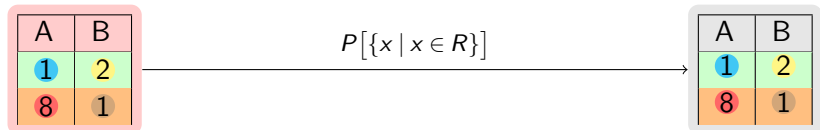
- Objects copied from the input retain their color.
- Objects resulting from constant, record, or set constructor are colored blank.



Provenance for queries

Queries **create** new objects

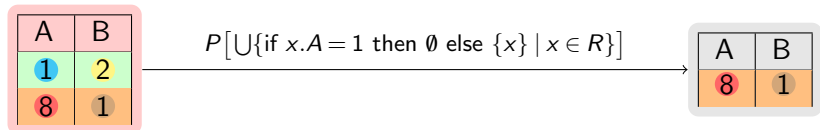
- Objects copied from the input retain their color.
- Objects resulting from constant, record, or set constructor are colored blank.



Provenance for queries

Queries **create** new objects

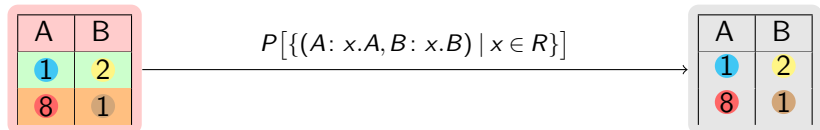
- Objects copied from the input retain their color.
- Objects resulting from constant, record, or set constructor are colored blank.



Provenance for queries

Queries **create** new objects

- Objects copied from the input retain their color.
- Objects resulting from constant, record, or set constructor are colored blank.



Question: Can every explicit f be expressed implicitly?

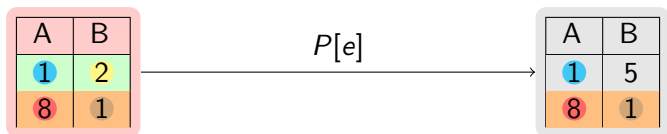
Answer: No, the provenance semantics $P[e]$ of a query e is:

- 1 Copying
- 2 Recording
- 3 Bounded Inventing

Theorem: Every explicit, copying, recording, and bounded inventing f can be expressed implicitly as the provenance semantics $P[e]$ of a query e .

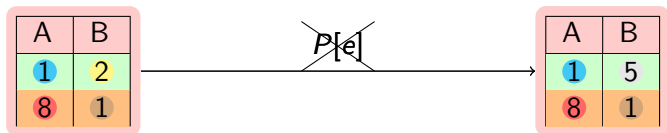
The implicit provenance semantics $P[e]$ of a query e is **copying**:

- Every non-blank object in the output is identical to the unique object in the input with the same color.



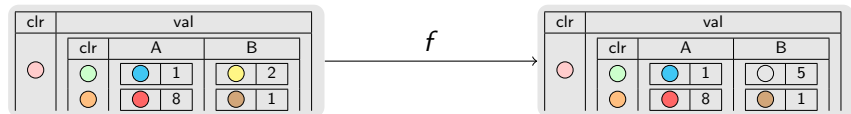
The implicit provenance semantics $P[e]$ of a query e is **copying**:

- Every non-blank object in the output is identical to the unique object in the input with the same color.



The implicit provenance semantics $P[e]$ of a query e is **copying**:

- Every non-blank object in the output is identical to the unique object in the input with the same color.



The implicit provenance semantics $P[e]$ of a query e is **copying**:

- Every non-blank object in the output is identical to the unique object in the input with the same color.

Corollary:

- Non-copying explicit f cannot be expressed implicitly.

The implicit provenance semantics $P[e]$ of a query e is **recording**:

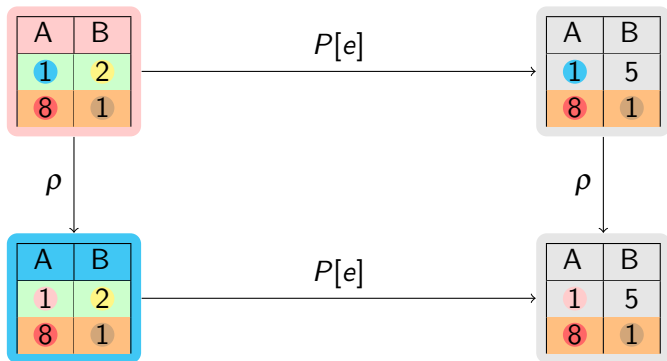
- The explicit expression for $P[e]$ never compares colors:

if $e_1 = e_2$ then e_t else e_f

can occur only if e_1 and e_2 do not output colors.

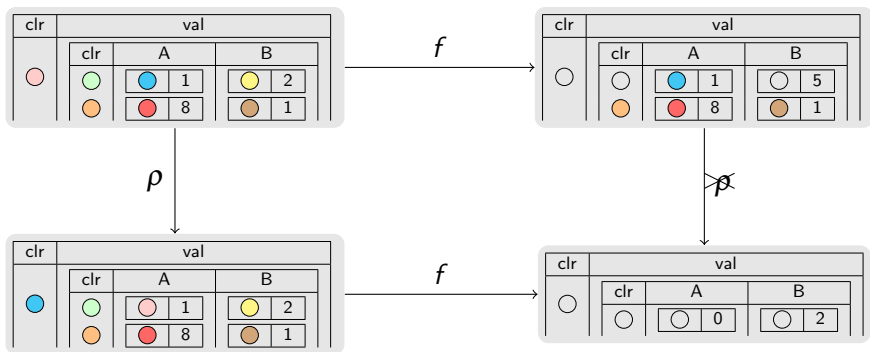
The implicit provenance semantics $P[e]$ of a query e is **recording**:

- The explicit expression for $P[e]$ never compares colors
- $P[e]$ is therefore also **propagating**: it commutes with recolorings



The implicit provenance semantics $P[e]$ of a query e is **recording**:

- The explicit expression for $P[e]$ never compares colors
- $P[e]$ is therefore also **propagating**: it commutes with recolorings



The implicit provenance semantics $P[e]$ of a query e is **recording**:

- The explicit expression for $P[e]$ never compares colors
- $P[e]$ is therefore also **propagating**: it commutes with recolorings

Corollary:

- Non-propagating explicit f cannot be expressed implicitly
- But such f *query* provenance rather than *define* it

The implicit provenance semantics $P[e]$ of a query e is **recording**:

- The explicit expression for $P[e]$ never compares colors
- $P[e]$ is therefore also **propagating**: it commutes with recolorings

Corollary:

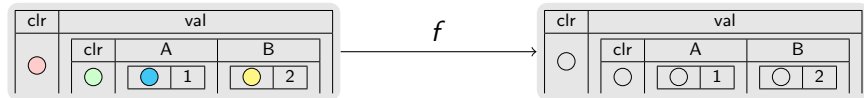
- Non-propagating explicit f cannot be expressed implicitly
- But such f *query* provenance rather than *define* it

Related work:

- [Geerts et al, ICDE 2006] and [Geerts and Van den Bussche, DBPL 2007] consider the queries f that *explicitly query* provenance, and show them expressively complete to the *color algebra*, an extension of the relational algebra that *implicitly queries* provenance.

The implicit provenance semantics $P[e]$ of a query e is **bounded inventing**:

- Only constants appearing in e will be colored blank.



The implicit provenance semantics $P[e]$ of a query e is **bounded inventing**:

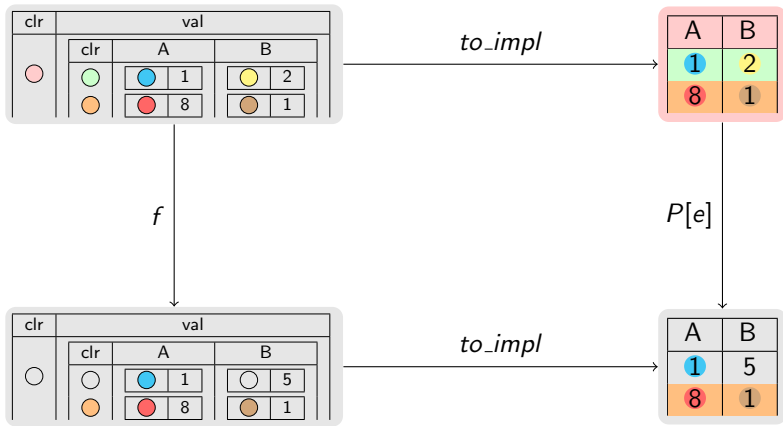
- Only constants appearing in e will be colored blank.

Corollary:

- Unbounded inventing explicit f cannot be expressed implicitly.
- But such f are not “domain preserving” w.r.t. provenance.

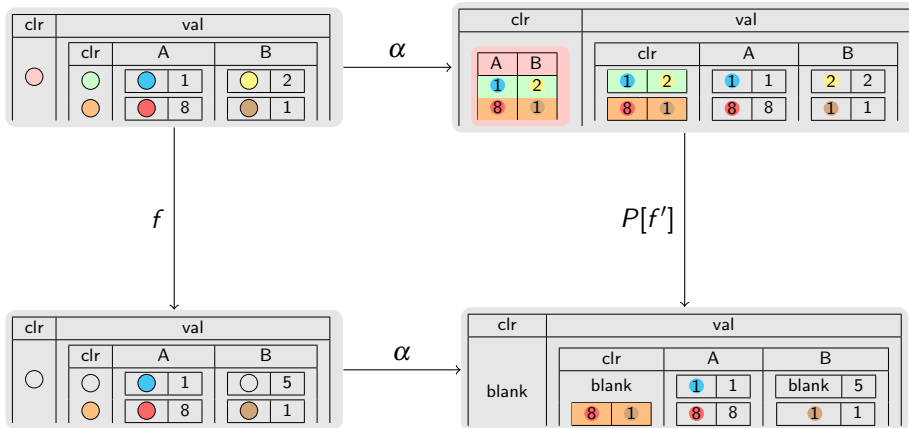
Main Result

Theorem: Every explicit, copying, recording, and bounded inventing f can be expressed implicitly as the provenance semantics $P[e]$ of a query e .



Proof Sketch

Lemma: Every explicit, recording f is **polymorphic** w.r.t. colors.



Question: Is the ability to construct deeply nested objects vital here?

Answer: No, it is possible to construct $P[e]$ equivalent to f such that e never constructs deeper intermediate objects than f .

Question: Is the ability to construct deeply nested objects vital here?

Answer: No, it is possible to construct $P[e]$ equivalent to f such that e never constructs deeper intermediate objects than f .

Question: What happens when f is propagating instead of recording?

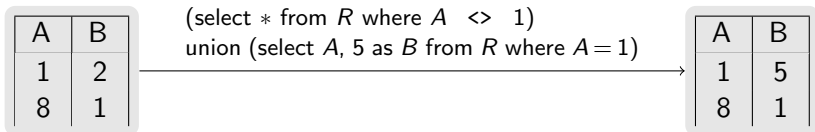
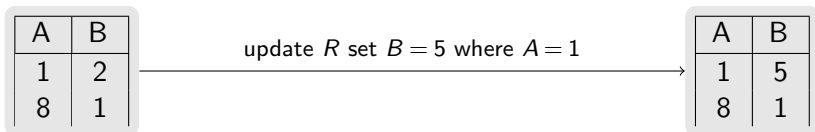
Conjecture: Every propagating f can be written as a recording one.

1 Provenance for Queries

2 Provenance for Updates

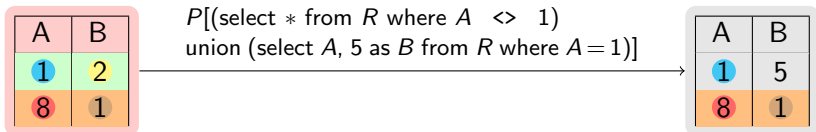
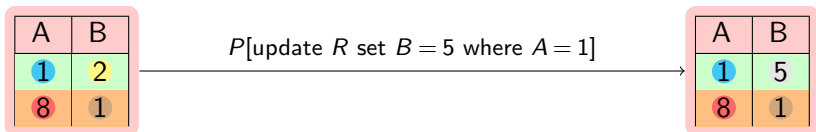
What about updates?

Updates are usually no more expressive than queries.



What about updates?

Updates are usually no more expressive than queries.



- **Objects:**

atoms – records – sets

- **Complex object updates (Liefke and Davidson, 1999):**

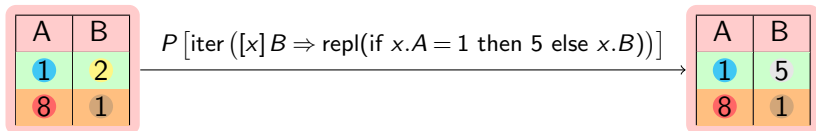
$$u ::= \text{skip} \mid u_1; u_2 \mid \text{repl } e \mid [x] u \mid A \Rightarrow u \mid \text{add } e \mid \text{drop } A \\ \mid \text{insert } e \mid \text{remove } e \mid \text{iter } u$$

Theorem: Updates and queries are equally expressive under the “normal” semantics

Provenance for updates

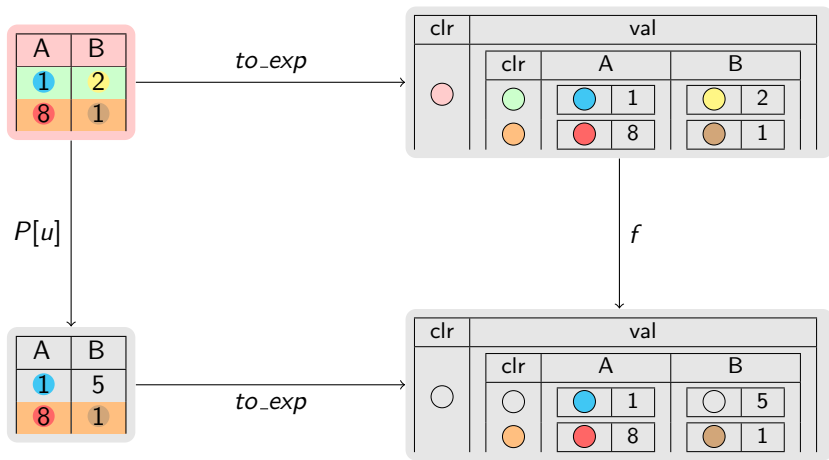
Updates **modify** objects

- Objects retain their color, unless they are replaced by another object.



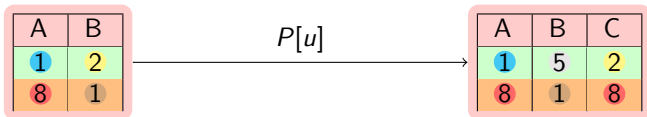
Implicit vs explicit provenance

Theorem: For every update u there exists a query f that **explicitly** implements the **implicit** provenance semantics $P[u]$.



The implicit provenance semantics $P[u]$ of an update u is:

- kind preserving



- recording
- bounded inventing

Theorem: Every explicit, kind preserving, recording, and bounded inventing f can be expressed implicitly as the provenance semantics $P[u]$ of an update.

Queries and updates **implicitly** define provenance

- Queries create new objects, updates modify them
- Readily implemented

Formal justification for the proposed semantics

- It is **expressively complete** w.r.t. natural classes of queries that **explicitly** manipulate provenance.