Incremental Materialisation in Datalog and its Relationship to Stream Reasoning

Boris Motik

University of Oxford

November 9, 2015

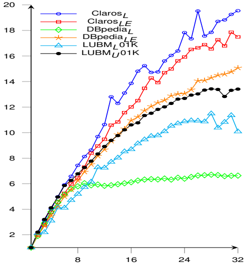
Table of Contents

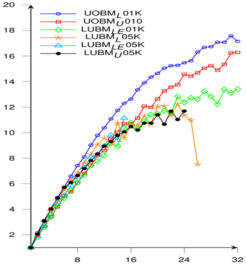
Table of Contents

RDFox: A Scalable RDF/Datalog Main-Memory Reasoner

- http://www.cs.ox.ac.uk/isg/tools/RDFox/
- RAM-based; currently centralised, but a distributed system is in the works
- Datalog reasoning via materialisation
  - Arbitrary (recursive) datalog rules, not just OWL 2 RL
  - Materialisation ⇒ precompute all facts in a preprocessing stage
  - Very effective parallelisation on multi-core architectures
- Efficient reasoning with *owl:sameAs* via rewriting
  - Known and widely-used technique, but correctness not trivial
- SPARQL query answering
  - Most of SPARQL 1.0 and some of SPARQL 1.1

Evaluation (I): Parallelisation Overhead and Speedup





- Speedup of up to 13x with 16 physical cores
- Increases to 19x with 32 virtual cores

Evaluation (II): Oracle's SPARC T5 (128/1024 cores, 4 TB)

	LUBM-50K		Claros		DBpedia	
Threads	sec	speedup	sec	speedup	sec	speedup
import	6.8k	—	168	—	952	—
1	27.0k	1.0x	10.0k	1.0x	31.2k	1.0x
16	1.7k	15.7x	906.0	11.0x	3.0k	10.4x
32	1.1k	24.0x	583.3	17.1x	1.8k	17.5x
48	920.7	29.3x	450.8	22.2x	2.0k	16.0x
64	721.2	37.4x	374.9	26.7x	1.2k	25.8x
80	523.6	51.5x	384.1	26.0x	1.2k	26.7x
96	442.4	60.9x	364.3	27.4x	825	37.8x
112	400.6	67.3x	331.4	30.2x	1.3k	24.3x
128	387.4	69.6x	225.7	44.3x	697.9	44.7x
256	—	—	226.1	44.2x	684.0	45.7x
384	—	—	189.1	52.9x	546.2	57.2x
512	—	—	153.5	65.1x	431.8	72.3x
640	—	—	140.5	71.2x	393.4	79.4x
768	—	—	130.4	76.7x	366.2	85.3x
896	—	—	127.0	78.8x	364.9	86.6x
1024	—	—	124.9	80.1x	358.8	87.0x
size	B/trp	Triples	B/trp	Triples	B/trp	Triples
aft imp	124.1	6.7G	80.5	18.8M	58.4	112.7M
aft mat	101.0	9.2G	36.9	539.2M	39.0	1.5G
import rate	1.0M		112k		120k	
mat. rate	6.1M		4.2M		4.0M	

Table of Contents

Why Incremental Reasoning?

- Common application scenario: continuous small changes in input data
  - Similar to stream reasoning!
- Materialisation can be expensive ⇒ starting from scratch is unacceptable!
- Incremental maintenance: minimise work needed to update materialisation
- State of the art (from the 90s):
  - the Counting algorithm
    - Basic variant applicable only to nonrecursive programs!
    - Extension to recursive programs rather complex
  - the Delete/Rederive (DRed) algorithm
    - Works for nonrecursive rules too

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

$A(a)$	1
$B(a)$	1

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x) \quad C_0(x) \leftarrow B(x) \quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1
$B(a)$	1
$C_0(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1
$B(a)$	1
$C_0(a)$	2

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation

$A(a)$	1
$B(a)$	1
$C_0(a)$	2
$C_1(a)$	1

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

$A(a)$	1
$B(a)$	1
$C_0(a)$	2
$C_1(a)$	1
...	

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation

$A(a)$	1
$B(a)$	1
$C_0(a)$	2
$C_1(a)$	1
...	
$C_n(a)$	1

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1
$B(a)$	1
$C_0(a)$	3
$C_1(a)$	1
...	
$C_n(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0
$B(a)$	1
$C_0(a)$	3
$C_1(a)$	1
...	
$C_n(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation
- Delete $A(a)$:
  - Decrease its counter

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0
$B(a)$	1
$C_0(a)$	2
$C_1(a)$	1
...	
$C_n(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation
- Delete $A(a)$:
  - Decrease its counter
  - The counter of $A(a)$ reaches zero, so propagate deletion

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0
~~$B(a)$~~	0
$C_0(a)$	2
$C_1(a)$	1
...	
$C_n(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation
- Delete $A(a)$:
  - Decrease its counter
  - The counter of $A(a)$ reaches zero, so propagate deletion
- Problem of this variant: delete $B(a)$
  - Decrease its counter

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0
~~$B(a)$~~	0
$C_0(a)$	1
$C_1(a)$	1
$\dots$	
$C_n(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation
- Delete $A(a)$:
  - Decrease its counter
  - The counter of $A(a)$ reaches zero, so propagate deletion
- Problem of this variant: delete $B(a)$
  - Decrease its counter
  - The counter of $B(a)$ reaches zero, so propagate deletion

Basic Counting (Nonrecursive Variant)

Example

$C_0(x) \leftarrow A(x)$ $C_0(x) \leftarrow B(x)$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0
~~$B(a)$~~	0
$C_0(a)$	1
$C_1(a)$	1
...	
$C_n(a)$	1

- Associate with each fact a counter initialised to zero
- Increment the counter after each derivation
- Delete $A(a)$:
  - Decrease its counter
  - The counter of $A(a)$ reaches zero, so propagate deletion
- Problem of this variant: delete $B(a)$
  - Decrease its counter
  - The counter of $B(a)$ reaches zero, so propagate deletion
  - However, $C_0(a)$ still has a cyclic derivation from $C_n(a)$
  - $\Rightarrow$ The algorithm does not delete $C_0(a), \ldots, C_n(a)$!
  - Reference counting is not a general garbage collection method

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1
$B(a)$	1

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1	
$B(a)$	1	
$C_0(a)$		1

- Associate with each fact an array of counters, one per iteration

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1	
$B(a)$	1	
$C_0(a)$		2

- Associate with each fact an array of counters, one per iteration

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1		
$B(a)$	1		
$C_0(a)$		2	
$C_1(a)$			1

- Associate with each fact an array of counters, one per iteration

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

$A(a)$	1		
$B(a)$	1		
$C_0(a)$		2	
$C_1(a)$			1
...			

- Associate with each fact an array of counters, one per iteration

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1				
$B(a)$	1				
$C_0(a)$		2			
$C_1(a)$			1		
...					
$C_n(a)$					1

- Associate with each fact an array of counters, one per iteration

Counting and Recursion

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

$A(a)$	1				
$B(a)$	1				
$C_0(a)$		2			1
$C_1(a)$			1		
...					
$C_n(a)$				1	

- Associate with each fact an array of counters, one per iteration

Counting and Recursion

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0				
~~$B(a)$~~	0				
$C_0(a)$		2			1
$C_1(a)$			1		
...					
$C_n(a)$				1	

- Associate with each fact an array of counters, one per iteration
- Delete $A(a)$ and $B(a)$ by undoing derivations

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0				
~~$B(a)$~~	0				
$C_0(a)$		0			1
$C_1(a)$			1		
...					
$C_n(a)$				1	

- Associate with each fact an array of counters, one per iteration
- Delete $A(a)$ and $B(a)$ by undoing derivations

Counting and Recursion

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0				
~~$B(a)$~~	0				
$C_0(a)$		0			1
$C_1(a)$			0		
...					
$C_n(a)$				1	

- Associate with each fact an array of counters, one per iteration
- Delete $A(a)$ and $B(a)$ by undoing derivations

Counting and Recursion

Example

$C_0(x) \leftarrow A(x)$ $\quad$ $C_0(x) \leftarrow B(x)$ $\quad$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad$ $C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0				
~~$B(a)$~~	0				
$C_0(a)$		0			1
$C_1(a)$			0		
. . .					
$C_n(a)$				0	

- Associate with each fact an array of counters, one per iteration
- Delete $A(a)$ and $B(a)$ by undoing derivations

Counting and Recursion

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~	0				
~~$B(a)$~~	0				
$C_0(a)$		0			0
$C_1(a)$			0		
...					
$C_n(a)$				0	

- Associate with each fact an array of counters, one per iteration
- Delete $A(a)$ and $B(a)$ by undoing derivations

Inefficiency of Recursive Counting

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

$A(a)$	1				
$B(a)$	1				
$C_0(a)$		2			1
$C_1(a)$			1		
...					
$C_n(a)$				1	

Inefficiency of Recursive Counting

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

$A(a)$	1				
$B(a)$	1				
$C_0(a)$	1	2			1
$C_1(a)$			1		
...					
$C_n(a)$				1	

- Add $C_0(a)$ explicitly

Inefficiency of Recursive Counting

Example

$C_0(x) \leftarrow A(x)$ $C_0(x) \leftarrow B(x)$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $C_0(x) \leftarrow C_n(x)$

$A(a)$	1				
$B(a)$	1				
$C_0(a)$	1	2			1
$C_1(a)$		1			
...					
$C_n(a)$				1	

- Add $C_0(a)$ explicitly
- We must update all counts to reflect the new state although there is no change in the available facts!

The DRed Algorithm at a Glance

Delete/Rederive (DRed): state of the art incremental maintenance algorithm

Example

$C_0(x) \leftarrow A(x) \qquad C_0(x) \leftarrow B(x) \qquad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n \qquad C_0(x) \leftarrow C_n(x)$

$A(a)$ $B(a)$

The DRed Algorithm at a Glance

Delete/Rederive (DRed): state of the art incremental maintenance algorithm

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

- Materialise initial facts

$A(a)$
$B(a)$
$C_0(a)$
$C_1(a)$
$\ldots$
$C_n(a)$

The DRed Algorithm at a Glance

Delete/Rederive (DRed): state of the art incremental maintenance algorithm

Example

$C_0(x) \leftarrow A(x)$ $C_0(x) \leftarrow B(x)$ $C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $C_0(x) \leftarrow C_n(x)$

- Materialise initial facts
- Delete $A(a)$ using DRed:

~~$A(a)$~~
$B(a)$
$C_0(a)$
$C_1(a)$
$\ldots$
$C_n(a)$

The DRed Algorithm at a Glance

Delete/Rederive (DRed): state of the art incremental maintenance algorithm

Example

$C_0(x) \leftarrow A(x) \quad C_0(x) \leftarrow B(x) \quad C_i(x) \leftarrow C_{i-1}(x) \text{ for } 1 \leq i \leq n \quad C_0(x) \leftarrow C_n(x)$

- Materialise initial facts
- Delete $A(a)$ using DRed:
  1. Delete all facts with a derivation from $A(a)$

$$
\begin{aligned}
C_0(x)^D &\leftarrow A(x)^D \\
C_0(x)^D &\leftarrow B(x)^D \\
C_i(x)^D &\leftarrow C_{i-1}(x)^D \text{ for } 1 \leq i \leq n \\
C_0(x)^D &\leftarrow C_n(x)^D
\end{aligned}
$$

~~$A(a)$~~
$B(a)$
~~$C_0(a)$~~
~~$C_1(a)$~~
~~...~~
~~$C_n(a)$~~

The DRed Algorithm at a Glance

Delete/Rederive (DRed): state of the art incremental maintenance algorithm

Example

$C_0(x) \leftarrow A(x)$ $\quad C_0(x) \leftarrow B(x)$ $\quad C_i(x) \leftarrow C_{i-1}(x)$ for $1 \leq i \leq n$ $\quad C_0(x) \leftarrow C_n(x)$

~~$A(a)$~~
$B(a)$
$C_0(a)$
$C_1(a)$
$\ldots$
$C_n(a)$

- Materialise initial facts
- Delete $A(a)$ using DRed:
  1. Delete all facts with a derivation from $A(a)$

$$
\begin{aligned}
C_0(x)^D &\leftarrow A(x)^D \\
C_0(x)^D &\leftarrow B(x)^D \\
C_i(x)^D &\leftarrow C_{i-1}(x)^D \text{ for } 1 \leq i \leq n \\
C_0(x)^D &\leftarrow C_n(x)^D
\end{aligned}
$$

  2. Rederive facts that have an alternative derivation

$$
\begin{aligned}
C_0(x) &\leftarrow C_0(x)^D \wedge A(x) \\
C_0(x) &\leftarrow C_0(x)^D \wedge B(x) \\
C_i(x) &\leftarrow C_i(x)^D \wedge C_{i-1}(x) \text{ for } 1 \leq i \leq n \\
C_0(x) &\leftarrow C_0(x)^D \wedge C_n(x)
\end{aligned}
$$

Improvement: the Forward/Backward/Forward Algorithm

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

$A(a)$
$B(a)$
$C_0(a)$
$C_1(a)$
$\ldots$
$C_n(a)$

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

Improvement: the Forward/Backward/Forward Algorithm

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

$A(a)$
$B(a)$
$C_0(a)$
$C_1(a)$
...
$C_n(a)$

- Delete $A(a)$ using FBF:

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

IMPROVEMENT: THE FORWARD/BACKWARD/FORWARD ALGORITHM

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

$A(a)$
$B(a)$
$C_0(a)$
$C_1(a)$
$\dots$
$C_n(a)$

?

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

Improvement: the Forward/Backward/Forward Algorithm

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

~~$A(a)$~~ ×
$B(a)$
$C_0(a)$
$C_1(a)$
...
$C_n(a)$

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?
  2. No ⇒ delete

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

Improvement: the Forward/Backward/Forward Algorithm

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

~~$A(a)$~~ ×
$B(a)$
$C_0(a)$?
$C_1(a)$
...
$C_n(a)$

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?
  2. No $\Rightarrow$ delete
  3. As in DRed, identify $C_0(a)$ as derivable from $A(a)$

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

IMPROVEMENT: THE FORWARD/BACKWARD/FORWARD ALGORITHM

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

~~$A(a)$~~	$\times$
$B(a)$	?
$C_0(a)$	?
$C_1(a)$	
...	
$C_n(a)$	

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?
  2. No $\Rightarrow$ delete
  3. As in DRed, identify $C_0(a)$ as derivable from $A(a)$
  4. Apply the rules to $C_0(a)$ 'backwards' $\Rightarrow$ by $C_0(x) \leftarrow B(x)$, we get $B(a)$

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

Improvement: the Forward/Backward/Forward Algorithm

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

~~$A(a)$~~	$\times$
$B(a)$	$\checkmark$
$C_0(a)$	?
$C_1(a)$	
$\cdots$	
$C_n(a)$	

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?
  2. No $\Rightarrow$ delete
  3. As in DRed, identify $C_0(a)$ as derivable from $A(a)$
  4. Apply the rules to $C_0(a)$ 'backwards' $\Rightarrow$ by $C_0(x) \leftarrow B(x)$, we get $B(a)$
  5. $B(a)$ is explicit so it is derivable

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

Improvement: the Forward/Backward/Forward Algorithm

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

~~$A(a)$~~	$\times$
$B(a)$	$\checkmark$
$C_0(a)$	$\checkmark$
$C_1(a)$	
$\cdots$	
$C_n(a)$	

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?
  2. No $\Rightarrow$ delete
  3. As in DRed, identify $C_0(a)$ as derivable from $A(a)$
  4. Apply the rules to $C_0(a)$ 'backwards' $\Rightarrow$ by $C_0(x) \leftarrow B(x)$, we get $B(a)$
  5. $B(a)$ is explicit so it is derivable
  6. So $C_0(a)$ is derivable too

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

IMPROVEMENT: THE FORWARD/BACKWARD/FORWARD ALGORITHM

- Facts often have many derivations, so many facts get deleted in the first step
- The Forward/Backward/Forward algorithm looks for alternatives immediately

~~$A(a)$~~	$\times$
$B(a)$	✓
$C_0(a)$	✓
$C_1(a)$	
$\dots$	
$C_n(a)$	

- Delete $A(a)$ using FBF:
  1. Is $A(a)$ derivable in any other way?
  2. No $\Rightarrow$ delete
  3. As in DRed, identify $C_0(a)$ as derivable from $A(a)$
  4. Apply the rules to $C_0(a)$ 'backwards' $\Rightarrow$ by $C_0(x) \leftarrow B(x)$, we get $B(a)$
  5. $B(a)$ is explicit so it is derivable
  6. So $C_0(a)$ is derivable too
  7. Stop propagation and terminate

B. Motik, Y. Nenov, R. Piro, and I. Horrocks.:

- Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm. AAAI 2015
- Combining Rewriting and Incremental Materialisation Maintenance for Datalog Programs with Equality. IJCAI 2015

EVALUATION (III): INCREMENTAL REASONING

Dataset		$\|E^-\|$	$\|I \setminus I'\|$	Rematerialise		DRed					B/F				
				Time	Derivations	Time		Derivations			Time		Derivations		
				(s)	Fwd	(s)	$\|D\|$	DR2	DR4	DR5	(s)	$\|C\|$	Bwd	Sat	Del Prop
	LUBM-1k-L	**100**	113	139.4	212.5M	0.0	1.0k	1.1k	0.8k	1.0k	0.0	0.5k	0.2k	0.3k	0.2k
$\|E\| = 133.6$M		**5.0k**	5.5k	101.8	212.5M	0.2	55.5k	67.2k	46.9k	59.8k	0.2	23.0k	9.3k	13.7k	7.4k
$\|I\| = 182.4$M		**2.5M**	2.7M	138.5	208.8M	39.4	10.3M	15.2M	6.6M	11.5M	32.8	10.0M	4.1M	5.6M	3.7M
$M_t = 121.5$s		**5.0M**	5.5M	91.8	205.0M	54.8	17.8M	26.3M	10.5M	18.9M	62.3	18.8M	7.8M	10.1M	7.5M
$M_d = 212.5$M		**7.5M**	8.3M	89.2	201.3M	71.5	24.3M	35.5M	13.6M	24.3M	85.4	26.7M	11.0M	14.0M	11.2M
		10.0M	11.0M	99.5	197.5M	127.9	30.0M	43.1M	15.9M	28.1M	102.2	34.1M	14.0M	17.4M	15.0M
	UOBM-1k-Uo	**100**	160	3482.0	3.6G	8797.6	1.8G	2.6G	53.2M	2.6G	5.4	0.8k	0.5k	1.3k	0.5k
$\|E\| = 254.8$M		**5.0k**	85.2k	3417.8	3.6G	9539.3	1.8G	2.6G	53.2M	2.6G	28.2	105.9k	17.9k	42.1k	104.1k
$\|I\| = 2.2$G		**17.0M**	130.9M	3903.1	3.4G	8934.3	1.8G	2.7G	63.7M	2.5G	988.8	175.8M	47.6M	104.0M	196.7M
$M_t = 5034.0$s		**34.0M**	269.0M	4084.1	3.2G	9492.5	1.9G	2.8G	68.4M	2.4G	1877.2	340.7M	87.5M	182.3M	401.1M
$M_d = 3.6$G		**51.0M**	422.8M	4010.0	3.0G	10659.3	1.9G	2.9G	71.5M	2.2G	2772.7	513.7M	125.2M	246.8M	622.0M
		68.0M	581.4M	3981.9	2.8G	11351.6	1.9G	2.9G	73.3M	2.1G	3737.3	687.0M	162.5M	289.5M	848.6M
	Claros-L	**100**	212	62.9	128.6M	0.0	0.8k	1.0k	0.2k	0.5k	0.0	0.6k	0.3k	0.7k	0.5k
$\|E\| = 18.8$M		**5.0k**	11.3k	62.8	128.6M	0.4	37.8k	50.7k	10.9k	23.9k	0.4	29.1k	18.8k	35.3k	26.8k
$\|I\| = 74.2$M		**0.6M**	1.3M	62.3	125.6M	32.3	4.1M	5.5M	1.1M	2.5M	14.9	3.1M	2.0M	3.6M	3.0M
$M_t = 78.9$s		**1.2M**	2.6M	61.2	122.6M	53.2	7.8M	10.8M	2.0M	4.8M	33.6	6.1M	3.8M	6.7M	6.0M
$M_d = 128.6$M		**1.7M**	4.0M	60.5	119.5M	73.6	11.4M	15.9M	2.8M	6.8M	47.8	8.9M	5.6M	9.5M	9.1M
		2.3M	5.5M	60.0	116.3M	91.0	14.8M	20.9M	3.6M	8.6M	60.6	11.7M	7.3M	12.0M	12.3M
	Claros-LE	**100**	0.5k	3992.8	12.6G	0.0	1.3k	2.0k	0.3k	0.9k	0.0	1.0k	0.7k	1.0k	1.1k
$\|E\| = 18.8$M		**2.5k**	178.9k	5235.1	12.6G	8077.4	5.5M	11.7G	176.6k	11.7G	10.3	216.4k	161.2k	8.8M	320.0k
$\|I\| = 533.7$M		**5.0k**	427.5k	4985.1	12.6G	7628.2	6.0M	11.7G	186.0k	11.7G	16.5	485.6k	369.0k	8.9M	769.3k
$M_t = 4024.5$s		**7.5k**	609.6k	4855.0	12.6G	7419.1	6.5M	11.7G	193.9k	11.7G	19.5	683.4k	516.8k	9.0M	1.1M
$M_d = 12.9$G		**10.0k**	780.8k	5621.3	12.6G	7557.9	6.8M	11.7G	207.6k	11.7G	3907.2	6.0M	723.0M	11.7G	16.9M

Table of Contents

APPLICATION: CONTEXT-AWARE MOBILE SERVICES (SAMSUNG)

- Use sensors (WiFi, GPS, . . .) to identify the context
  - E.g., 'at home', 'in a shop', 'with a friend' . . .
- Adapt behaviour depending on the context
  - 'If with a friend who has birthday, remind to congratulate'
- Declaratively describe contexts and adaptations
  - Use a bunch of rules
  - E.g., 'If can see home WiFi, then context is "at home"'
- Interpret rules in real-time via incremental reasoning
  - We used DRed
- User detect events by registering continuous queries
- The streaming aspect was lightweight:
  - The database always reflects the 'current' state of the world
  - Continuous queries just monitor this 'current' state
  - Queries cannot refer to states at different time points

Querying/Reasoning Across Time-Points

Querying a Stream of Events

- The database is an ever-filling set of events with time-points
- E.g., 'Quote for AAPL at 9am is $121'
- Queries must explicitly refer to events
- E.g., 'The price of AAPL at 9.05am?' makes no sense $\Rightarrow$ no global world-view
- $\Rightarrow$ 'The quote for AAPL at time t with $t <$ 9.05am and no quote from t to 9.05am'

Querying an Evolving World View

- There is a complete database state ('world view') for each time-point t
- We can have inertia rules
- E.g., 'The price a stock at any point t is the price of the most recent quote'
- Now 'The price of AAPL at 9.05am?' is correct as we have a notion of 'Price at time t'

- Windowing could be viewed as an implementation detail
  - Prevents memory from filling, but does not play part in the definition of a model
- Can we use incremental materialisation algorithms for stream reasoning?