CS 595 - Hot topics in database systems: **Data Provenance**

I. Database Provenance

I.1 Provenance Models and Systems

Boris Glavic

September 26, 2012

Outline

Causality and Responsibility

Causality

- Models which tuples were necessary to produce output tuple
- Necessary here is context dependent
  - Tuple is necessary assuming that other tuple do not exist

Responsibility

- Model how important a tuple was in deriving an output tuple
- Numeric value
  - 1: Absolutely necessary in deriving the tuple
  - $\rightarrow 1/\infty$: Very marginal necessity

Counterfactual Cause

- A tuple $t' \in I$ is counterfactual cause for a tuple t in result of query q
  - If removing it from the database causes t to disappear from the result of q
  - $\Rightarrow t'$ is strictly necessary to derive t

Definition (Counterfactual Cause)

$t' \in I$ is counterfactual cause for $t \in Q(I)$ iff

- $t \notin Q(I - \{t'\})$

ILLINOIS INSTITUTE OF TECHNOLOGY

Actual Cause

- Tuples influence result without being strictly necessary
  - E.g., three tuples are projected on one result tuple t
  - None of these tuples is a counterfactual cause
  - However, these tuples clearly caused t to be in result

Example

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Actual Cause

- Tuples influence result without being strictly necessary
  - E.g., three tuples are projected on one result tuple t
  - None of these tuples is a counterfactual cause
  - However, these tuples clearly caused t to be in result
- $\Rightarrow$Model that tuples are only necessary under certain conditions
  - E.g., removing two tuples from the example before $\Rightarrow$the remaining tuple to be a cause

Example

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Actual Cause

Definition (Actual Cause)

$t' \in I$ is actual cause for $t \in Q(I)$ iff

- exists $\Gamma \subset (I - \{t'\})$ (call contingency)
- $t \in Q(I - \Gamma)$ and $t \notin Q(I - \Gamma - \{t'\})$

ILLINOIS INSTITUTE OF TECHNOLOGY

Actual Cause Example

Example

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Actual Cause Example

Example

- Is r_1 a counterfactual cause?
  - $Q(I - \{r_1\}) = \{t_1\}$ $\Rightarrow$NO

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Actual Cause Example

Example

- Is r_1 an actual cause?
  - Chose contingency $\Gamma = \{r_2, r_3\}$
  - $Q(I - \Gamma) = \{t_1\}$
  - $Q(I - \Gamma - \{r_1\}) = \emptyset$
  - $\Rightarrow$YES

$q = \pi_a(R)$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Exogenous vs. Endogenous Tuple

Rationale

- Let user choose which tuples are considered as causes
  - Exclude trusted relation from reasoning
- User divides instance into
  - Potential causes I^n (endogenous)
  - Tuples which are not considered as causes I^x (exogenous)

Adapted Definitions

- Counterfactual Cause t' for t: $t' \in D^n \wedge t \notin Q(I - \{t'\})$
- Actual Cause t' for t:
  $t' \in D^n \wedge \exists \Gamma \subset D^n : t' \in Q(I - \Gamma) \wedge t' \notin Q(I - \Gamma - \{t'\})$

Responsibility

Rationale

- Not all causes are equal
- Some causes are more important than others
- $\Rightarrow$Create model that quantifies the importance of causes
- Tuples with large contingency are less important

Responsibility

Rationale

- Not all causes are equal
- Some causes are more important than others
- $\Rightarrow$Create model that quantifies the importance of causes
- Tuples with large contingency are less important

Definition (Responsibility)

The responsibility ρ_t of a cause t is computed as

- $\rho_t = \frac{1}{1 + min_\Gamma \|\Gamma\|}$
- Γ ranges over all contingencies for t

Responsibility Example

Example

- Compute responsibility ρ_{r_1} for t_1
- Find smalles contingency, test subsets of $I - \{r_1\}$

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Responsibility Example

Example

- Compute responsibility ρ_{r_1} for t_1
- Find smalles contingency, test subsets of $I - \{r_1\}$
  - $\{r_2\}$ NO

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Responsibility Example

Example

- Compute responsibility ρ_{r_1} for t_1
- Find smalles contingency, test subsets of $I - \{r_1\}$
  - $\{r_2\}$ NO
  - $\{r_3\}$ NO

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Responsibility Example

Example

- Compute responsibility ρ_{r_1} for t_1
- Find smalles contingency, test subsets of $I - \{r_1\}$
  - $\{r_2\}$ NO
  - $\{r_3\}$ NO
  - $\{r_2, r_3\}$ YES

$$q = \pi_a(R)$$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Responsibility Example

Example

- Compute responsibility ρ_{r_1} for t_1
- Find smalles contingency, test subsets of $I - \{r_1\}$
  - $\{r_2\}$ NO
  - $\{r_3\}$ NO
- $\rho_{r_1} = \frac{1}{1+\|\{r_2,r_3\}\|} = \frac{1}{3}$

$q = \pi_a(R)$

R

	a	b
r_1	1	7
r_2	1	6
r_3	1	5

Q

	a
t_1	1

Insensitivity to Query Rewrite

Causality is insensitive!

- The definition is purely declarative
- $\Rightarrow$ has to be insensitive
- E.g, $q \equiv q'$ and $t \in Q/Q'(I)$
- If t' is cause for t in q then $\exists \Gamma$ so that
  - $t \in Q(I - \Gamma) = Q'(I - \Gamma)$
  - $t \notin Q(I - \Gamma - \{t'\}) = Q'(I - \Gamma - \{t'\})$
- $\Rightarrow t'$ is cause for t in q'

ILLINOIS INSTITUTE OF TECHNOLOGY

Notation

Causality

- $Cau(q, t)$ is set of all actual causes for t
  - $Cau(q, t) = \{t' \mid t' \text{ is actual cause for } t\}$

Responsibility

- $\rho_{q,t}(t')$ is function mapping each cause t' its responsibility value

ILLINOIS INSTITUTE OF TECHNOLOGY

Computing Causes and Responsibility - Brute Force

Causes

- For each tuple t' in I
  - Enumerate all subsets Γ
  - For each such subset test
    - $Q(I - \Gamma)$ and $Q(I - \Gamma - \{t'\})$
  - If test is successful then t' is actual cause

ILLINOIS INSTITUTE OF TECHNOLOGY

Computing Causes and Responsibility - Brute Force

Causes

- For each tuple t' in I
  - Enumerate all subsets Γ
  - For each such subset test
    - $Q(I - \Gamma)$ and $Q(I - \Gamma - \{t'\})$
  - If test is successful then t' is actual cause

Complexity

- $\|I\|$ number of iterations
- In each iteration in worst case we have $2^{\|I\|-1}$ subsets to consider
- For each subset we have to execute two queries
- $O(\|I\| \times 2^{\|I\|} \times 2 \times cost(Q(I))) = O(2^{\|I\|})$

Computing Causes and Responsibility - Brute Force

Responsibility

- For each tuple t' in I init $minCont = \infty$
  - Enumerate all subsets Γ
  - For each such subset test
    - $Q(I - \Gamma)$ and $Q(I - \Gamma - \{t'\})$
  - If test is successful then
    - $minCont = min(\|\Gamma\|, minCont)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Computing Causes and Responsibility - Brute Force

Responsibility

- For each tuple t' in I init $minCont = \infty$
  - Enumerate all subsets Γ
  - For each such subset test
    - $Q(I - \Gamma)$ and $Q(I - \Gamma - \{t'\})$
  - If test is successful then
    - $minCont = min(\|\Gamma\|, minCont)$

Complexity

- $\|I\|$ number of iterations
- In each iteration in worst case we have $2^{\|I\|-1}$ subsets to consider
- For each subset we have to execute two queries
- $O(\|I\| \times 2^{\|I\|} \times 2 \times cost(Q(I))) = O(2^{\|I\|})$

Outline

Using Provenance for Cause Computation

Rationale

- Provenance contains all tuples that effect a tuple
- $\Rightarrow$Limit search for contingency to provenance
- Relationship with view update (delete tuple t from view)
  - View Update: Find set of tuples from the input that cause t to disappear
  - $\neq$ Find set of tuples so that after removal additional tuples cause t
  - Exogenous tuples!
  - Queries are assumed to be CQ's

ILLINOIS INSTITUTE OF TECHNOLOGY

Excursion: Datalog

Datalog

- Relational query language (set-semantics)
- Similar to Prolog: Queries are expressed as logical implications
- Declarative:
  - Query specifies what result is rather than how to compute it
- Expressive Power:
  - Supports recursion
  - Without recursion + with negation it is equivalent to relational algebra (no aggregation)
  - Without negation and recursion is equivalent to SPJ queries (using equality predicates only) - called Conjunctive Queries (CQ)

Excursion: Datalog rules

Queries

- Set of datalog rules

Datalog rule

- $q(\vec{X}) :- R_1(\vec{X_1}), \ldots, R_n(\vec{X_n})$
- R_i's are relations
- $\vec{X}$ and $\vec{X_i}$ are lists of variables and/or constants
- The variables in $\vec{X}$ have to appear in at least one $\vec{X_i}$
- Head: $q(\vec{X})$ is called the head
- Body: $R_1(\vec{X_1}), \ldots, R_n(\vec{X_n})$ is called the body of the rule
- Single rule (conjunctive query) = SPJ query

Excursion: Evaluating CQ's

Valuation θ

- a replacement of variables in body ($\Rightarrow$also in head) with constants
- such that every atom $R_i(\theta(\vec{X_i}))$ is a tuple in the instance I

Result of Conjunctive Query

- For each valuation θ
- add $\theta(\vec{X})$ to the result of query

Excursion: Example CQ

Example

$$q(b) : -E(a,b), A(c,a), P(c, "CS")$$

Q

	Name
t_1	Peter
t_2	Gertrud

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Excursion: Example CQ

Example

$$q(b) :- E(a,b), A(c,a), P(c, "CS")$$
$$\theta_{t_1} = \{a = 1, b = "Peter", c = "Server"\}$$

Q

	Name
t_1	Peter
t_2	Gertrud

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Excursion: Example CQ

Example

$$q(b) :- E(a, b), A(c, a), P(c, "CS")$$
$$\theta_{t_2} = \{a = 2, b = "Gertrud", c = "Server"\}$$

Q

	Name
t_1	Peter
t_2	Gertrud

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Excursion: Example CQ

Example

$$q(b) :- E(a,b), A(c,a), P(c, "CS")$$
$$\theta'_{t_2} = \{a = 2, b = "Gertrud", c = "Webpage"\}$$

Q

	Name
t_1	Peter
t_2	Gertrud

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Excursion: Boolean queries

Boolean query

- Conjunctive query with empty head
- Evaluates to $\{true, false\}$

Example

- `Department(Name,Headcount,Budget)`
- $q() : -Dep("CS", b, c)$
- Evaluates to `true` if there is an "CS" department

ILLINOIS INSTITUTE OF TECHNOLOGY

Excursion: Union of Conjunctive Queries

Union of Conjunctive Queries

- Set of datalog rules with same name and arity in head
- Evaluation: union the evaluation results for all rules

Example

$$q(a) :- R(a)$$
$$q(b) :- R(b), S(b)$$

is the same as $R \cup (R \bowtie S)$ (natural join)

Excursion: Union of Conjunctive Queries

Union of Conjunctive Queries

- Set of datalog rules with same name and arity in head
- Evaluation: union the evaluation results for all rules

Example

$$q(a) : -R(a)$$
$$q(b) : -R(b), S(b)$$

is the same as $R \cup (R \bowtie S)$ (natural join)

Excursion: Recursive Datalog and Negation

Recursive Datalog

- Set of datalog rules with same name and arity in head
- + rules can reference themselves or other rules in the body
- Evaluation:
  - $I' = I$
  - Evaluate one rule and add result to I'
  - Repeat until no more new data can be added

Example

$$ancestor(a, b) :- parent(a, c), ancestor(c, b)$$
$$ancestor(a, b) :- parent(a, b)$$

Excursion: Recursive Datalog and Negation

Datalog with Negation

- Allow negated atoms in the body
- Evaluation:
  - Find valuations so that for every negated atom $\neg R_i(\vec{X_i})$
  - $R_i(\theta(\vec{X_i}))$ is not in the instance

Example

$$Civil(a) :- Person(a), \neg WorksInArmy(a)$$

ILLINOIS INSTITUTE OF TECHNOLOGY

The $\mathbb{B}[I]$ Semiring for conjunctive queries

Recall

- $\mathbb{B}[I]$ is boolean expressions over variables presenting the tuples in I
- E.g., $(t_1 \wedge t_2) \vee t_3$
- Here always formulas in *DNF* (disjunctive normal form)

Provenance for conjunctive queries

- CQ: $q : -a_1, \ldots, a_m$
- Valuation θ with $\theta(a_i) = t_i$
- X_t is boolean variable for tuple t
- Formula $c^\theta = X_1 \wedge \ldots \wedge X_m$
- Provenance $\Phi(q, t) = \bigvee_{\theta : q \rightarrow I} c^\theta$

Example Provenance for CQ

Example

$$q(b) : -E(a,b), A(c,a), P(c,"CS")$$
$$\theta_{t_2} = \{a = 2, b = "Gertrud", c = "Server"\}$$
$$\theta'_{t_2} = \{a = 2, b = "Gertrud", c = "Webpage"\}$$
$$c^{\theta_{t_2}} = (e_2 \wedge a_2 \wedge p_1) \; c^{\theta'_{t_2}} = (e_2 \wedge a_3 \wedge p_2)$$
$$\Phi(q, t_2) = (e_2 \wedge a_2 \wedge p_1) \vee (e_2 \wedge a_3 \wedge p_2)$$

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Q

	Name
t_1	Peter
t_2	Gertrud

OF TECHNOLOGY

Determine Causes based on Provenance

- Testing whether t is in the result of removing some tuples

Determine Causes based on Provenance

- Testing whether t is in the result of removing some tuples
- $\Rightarrow$Deletion propagation

Determine Causes based on Provenance

- Testing whether t is in the result of removing some tuples
- $\Rightarrow$Deletion propagation
- $\Rightarrow$Set variables for deleted tuples S to *false*

Determine Causes based on Provenance

- Testing whether t is in the result of removing some tuples
- $\Rightarrow$Deletion propagation
- $\Rightarrow$Set variables for deleted tuples S to *false*
- For set S of tuples: $\Phi[S = \mathit{false}]$ sets all variables corresponding to tuples in S to false
  - If $\Phi(q, t)[S = \mathit{false}] = \mathit{true}$: tuple t in result of $Q(I - S)$
  - If $\Phi(q, t)[S = \mathit{false}] = \mathit{false}$: tuple t not in result of $Q(I - S)$

Determine Causes based on Provenance

- Testing whether t is in the result of removing some tuples
- $\Rightarrow$Deletion propagation
- $\Rightarrow$Set variables for deleted tuples S to *false*
- For set S of tuples: $\Phi[S = \mathit{false}]$ sets all variables corresponding to tuples in S to false
  - If $\Phi(q, t)[S = \mathit{false}] = \mathit{true}$: tuple t in result of $Q(I - S)$
  - If $\Phi(q, t)[S = \mathit{false}] = \mathit{false}$: tuple t not in result of $Q(I - S)$
- $\Rightarrow$We can test whether t' is cause by
  - Testing for every subset Γ of the provenance whether $\Phi(q, t)[\Gamma = \mathit{false}] = \mathit{true}$
  - ... and $\Phi(q, t)[(\Gamma \cup \{t'\}) = \mathit{false}]$

Determine Causes based on Provenance

- Testing whether t is in the result of removing some tuples
- $\Rightarrow$Deletion propagation
- $\Rightarrow$Set variables for deleted tuples S to *false*
- For set S of tuples: $\Phi[S = \mathit{false}]$ sets all variables corresponding to tuples in S to false
  - If $\Phi(q,t)[S = \mathit{false}] = \mathit{true}$: tuple t in result of $Q(I - S)$
  - If $\Phi(q,t)[S = \mathit{false}] = \mathit{false}$: tuple t not in result of $Q(I - S)$
- $\Rightarrow$We can test whether t' is cause by
  - Testing for every subset Γ of the provenance whether $\Phi(q,t)[\Gamma = \mathit{false}] = \mathit{true}$
  - ...and $\Phi(q,t)[(\Gamma \cup \{t'\}) = \mathit{false}]$
- $\Rightarrow$still exponential, but in size of provenance

Determine Causes based on Provenance cont.

- Exploit structure of the formula for more efficient computation
- $\Phi(Q, t)$ evaluates to false, if every conjunct evaluates to false
- A conjunct evaluates to false if any of its variables it set to false
- $\Rightarrow$To make t' a cause
  - Let $\mathcal{C}(t')$ be the set of conjuncts that contain t'
  - Set one variable form every conjunct not in $\mathcal{C}(t')$ to false (contingency)
    - Use variable that is not in a conjunct in $\mathcal{C}(t')$!
  - $\Rightarrow$the resulting formula is true
  - Setting $X_{t'}$ to false will make all conjuncts in $\mathcal{C}(t')$ false
  - $\Rightarrow t'$ is cause

Determine Causes based on Provenance cont.

Caveat

- There may not be variables that are only in conjuncts not in $\mathcal{C}(t')$!
  - Example: $X_{t_2} \vee X_{t_2} X_{t_1}$.
- This is the case for redundant conjuncts
- $\Rightarrow$apply absorption $a \wedge b \vee a = a$
- $\Rightarrow$All tuples corresponding to variables in this formula are actual causes
- $\Rightarrow$Polynomial complexity!

Determine Causes based on Provenance cont.

Example

- $q(b) : -E(a, b), A(c, a), P(c, "CS")$
- $\Phi(q, t_2) = (e_2 \wedge a_2 \wedge p_1) \vee (e_2 \wedge a_3 \wedge p_2)$
- No redundant conjuncts
- $\Rightarrow Cau(q, t_2) = \{e_2, a_2, a_3, p_1, p_2\}$

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Q

	Name
t_1	Peter
t_2	Gertrud

OF TECHNOLOGY

Endogenous Provenance

- Exogenous tuples cannot be used
- Set all exogenous tuple variables to true $\Rightarrow \Phi^n(q, t)$
- $\Rightarrow$Compute non-redundant conjuncts based on $\Phi^n(q, t)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Endogenous Provenance

Example

- $q(b) :- E(a, b), A(c, a), P(c, "CS")$
- $\Phi(q, t_2) = (e_2 \wedge a_2 \wedge p_1) \vee (e_2 \wedge a_3 \wedge p_2)$
- $I^x = \{p_1\} \cup A$
- $\Rightarrow \Phi^n(q, t_2) = (e_2 \wedge true \wedge true) \vee (e_2 \wedge true \wedge p_2) = e_2 \vee (e_2 \wedge p_2) = e_2$

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Assigned

	PName	Id
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Project

	PName	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Q

	Name
t_1	Peter
t_2	Gertrud

Outline

Recap

Causality based Provenance

- **Rationale**: Models necessity negatively
  - Cause not there $\Rightarrow$ Tuple not there
- **Representation**: Set of tuples (Cause) / numeric value (Responsibility)
  - Contingency Γ
- **Declarative Definition**:
  - For any queries
- **Provenance Based Definition**:
  - SPJ queries

Recap

Responsibility

- Quantifies responsibility of cause
- Defined over size of smallest contingency

Provenance Model Comparison

Property	Why	Lin	PI-CS	Where	How	Causality
Representation	Set of Set of Tuples	List of Set of Tuples	Set/Bag of List of Tuples	Sets of Attribute Value Positions	Values of provenance semiring	Set of causes + numeric responsibility value
Granularity	Tuple	Tuple	Tuple	Attribute Value	Tuple	Tuple
Language Support	USPJ	ASPJ-Set	ASPJ-Set + Nested subqueries	U-SPJ	A^{*}SPJ-UD^{*}	SPJ
Semantics	Set	Set + Bag*	Bag	Set	Set + Bag	Set
Variants	Wit, Why, IWhy	Set/Bag	Influence + Copy	SPJ + Insensitive + Insensitive Union	semirings	For multiple tuples
Definition	Decl. - Synt. - Decl./Synt.	Decl. + Synt.	Decl. + Synt.	Synt.	Synt.	Decl.
Design Principles	Sufficiency - No false positives	Sufficiency + No false negatives + no false positives	Sufficiency + No false negatives + No false positives	Copying	Equivalent to query evaluation	Contextual Necessity
Systems	-	WHIPS	Perm	DBNotes	ORCHESTRA	Thesias*
Insensitivity	Yes - No - Yes	No	No	No - Yes - Yes	Yes	Yes

Literature I

A. Meliou and D. Suciu.
Tiresias: the database oracle for how-to queries.
In Proceedings of the 2012 international conference on Management of Data, 337–348, ACM, 2012.

Alexandra Meliou, Wolfgang Gatterbauer, Suman Nath, and Dan Suciu.
Tracing data errors with view-conditioned causality.
In SIGMOD Conference, 505-516, 2011.

James Cheney.
Causality and the semantics of provenance.
In DCM, 63-74, 2010.

A. Meliou, W. Gatterbauer, J.Y. Halpern, C. Koch, K.F. Moore, and D. Suciu.
Causality in databases.
IEEE Data Engineering Bulletin, 2010.

A. Meliou, W. Gatterbauer, K.F. Moore, and D. Suciu.
The Complexity of Causality and Responsibility for Query Answers and non-Answers.
Proceedings of the VLDB Endowment, 4(1):34–45, 2010.

I. Beer, S. Ben-David, H. Chockler, A. Orni, and R. Trefler.
Explaining counterexamples using causality.
In Computer Aided Verification, 94–108, Springer, , 2009.

Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu.
Why so? or why no? functional causality for explaining query answers.
Technical report, University of Washington, 2009.

ILLINOIS INSTITUTE OF TECHNOLOGY

Literature II

T. Eiter and T. Lukasiewicz.
Complexity results for structure-based causality* 1.
Artificial Intelligence, 142(1):53–89, 2002.

J. Pearl.
Causality: models, reasoning, and inference.
Cambridge Univ Pr, 2000.