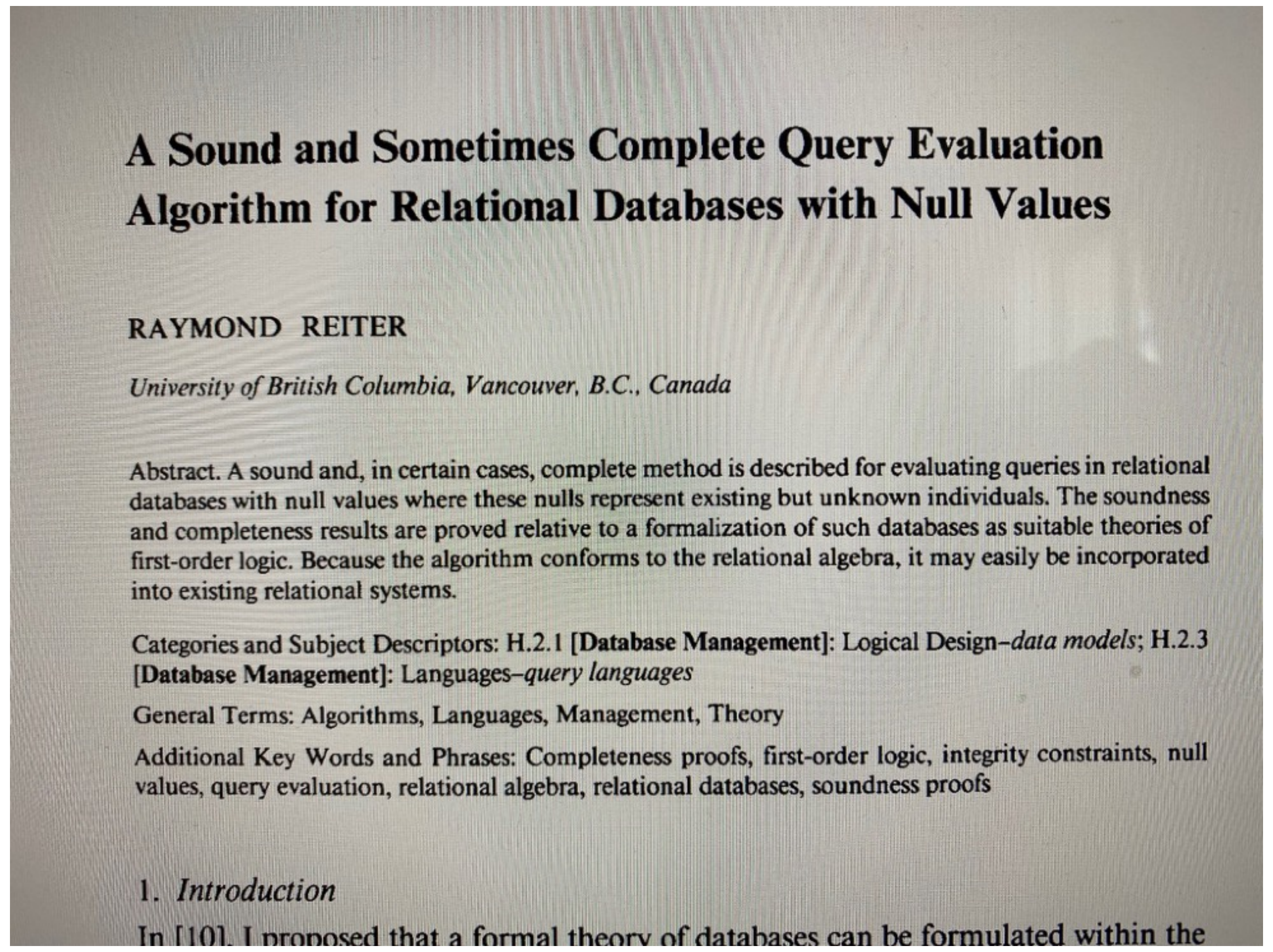
Approximations of Certain Answers in First-Order Logic

Leonid Libkin

Two classical papers from JACM and JCSS in 1986

Two classical papers from JACM and JCSS in 1986



A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values

RAYMOND REITER

University of British Columbia, Vancouver, B.C., Canada

Abstract. A sound and, in certain cases, complete method is described for evaluating queries in relational databases with null values where these nulls represent existing but unknown individuals. The soundness and completeness results are proved relative to a formalization of such databases as suitable theories of first-order logic. Because the algorithm conforms to the relational algebra, it may easily be incorporated into existing relational systems.

Categories and Subject Descriptors: H.2.1 **[Database Management]**: Logical Design–*data models*; H.2.3 **[Database Management]**: Languages–*query languages*

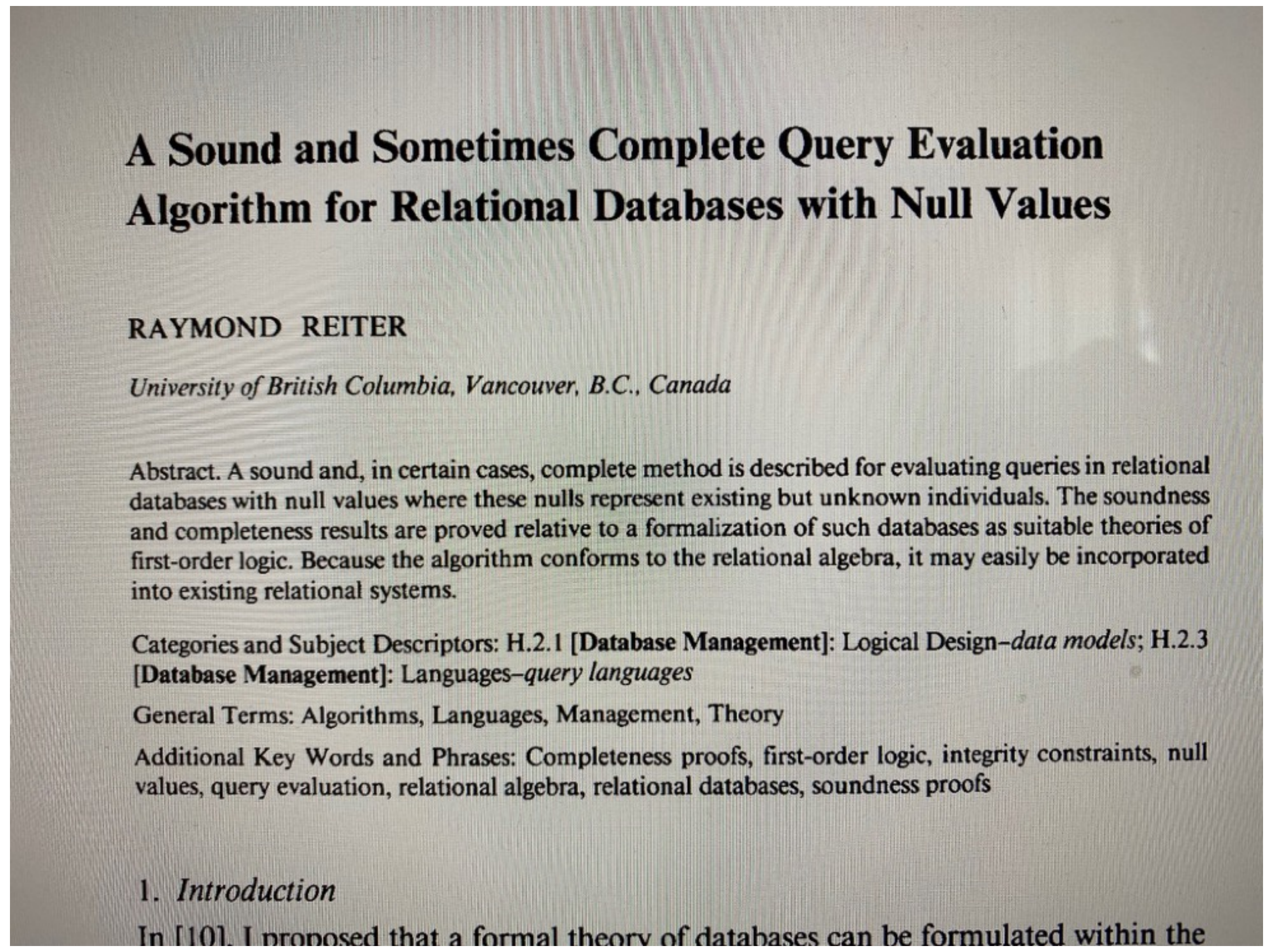
General Terms: Algorithms, Languages, Management, Theory

Additional Key Words and Phrases: Completeness proofs, first-order logic, integrity constraints, null values, query evaluation, relational algebra, relational databases, soundness proofs

1. *Introduction*

In [10], I proposed that a formal theory of databases can be formulated within the

Two classical papers from JACM and JCSS in 1986



A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values

RAYMOND REITER

University of British Columbia, Vancouver, B.C., Canada

Abstract. A sound and, in certain cases, complete method is described for evaluating queries in relational databases with null values where these nulls represent existing but unknown individuals. The soundness and completeness results are proved relative to a formalization of such databases as suitable theories of first-order logic. Because the algorithm conforms to the relational algebra, it may easily be incorporated into existing relational systems.

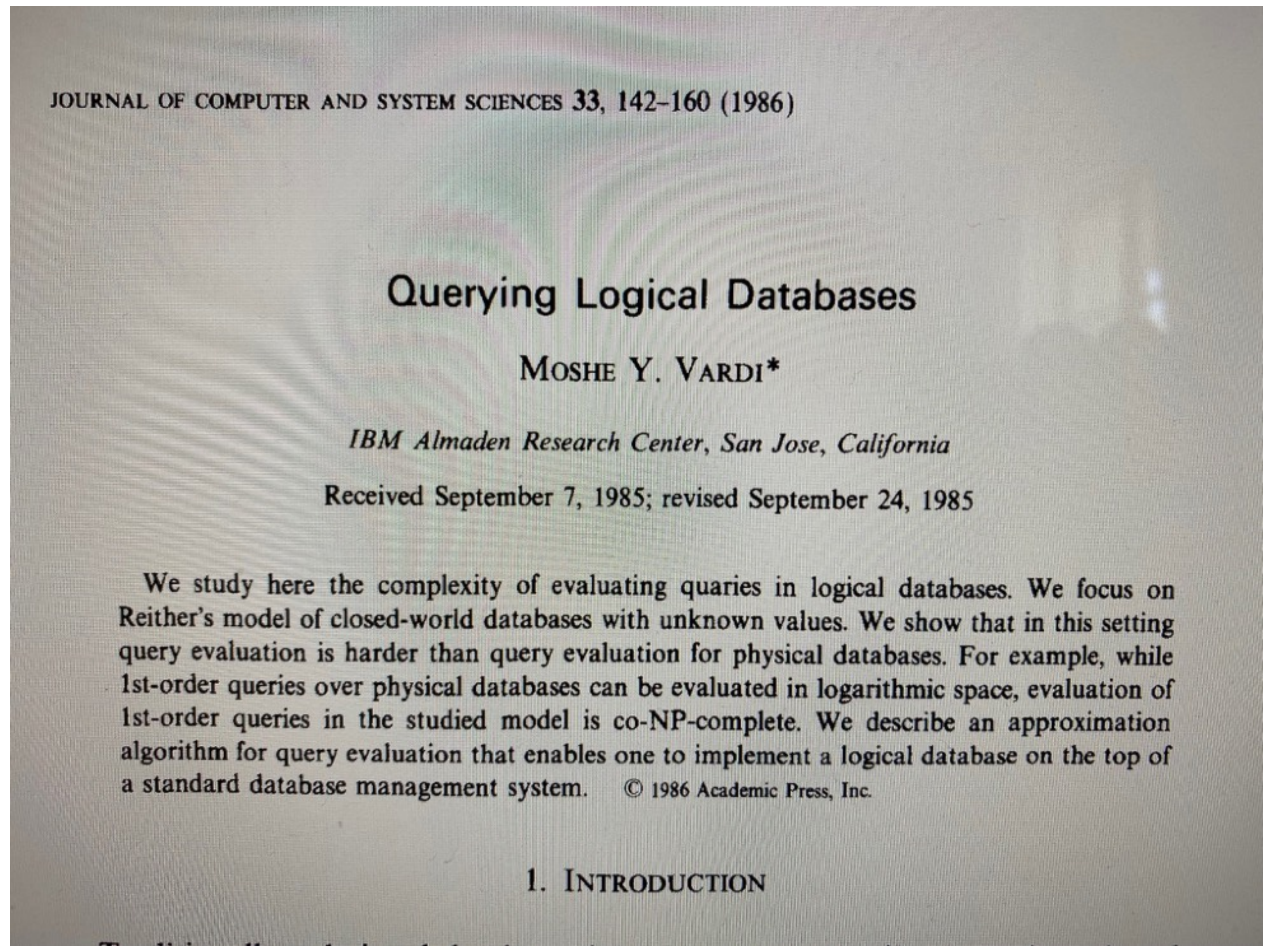
Categories and Subject Descriptors: H.2.1 **[Database Management]**: Logical Design–*data models*; H.2.3 **[Database Management]**: Languages–*query languages*

General Terms: Algorithms, Languages, Management, Theory

Additional Key Words and Phrases: Completeness proofs, first-order logic, integrity constraints, null values, query evaluation, relational algebra, relational databases, soundness proofs

1. *Introduction*

In [10], I proposed that a formal theory of databases can be formulated within the



JOURNAL OF COMPUTER AND SYSTEM SCIENCES **33**, 142–160 (1986)

Querying Logical Databases

MOSHE Y. VARDI*

IBM Almaden Research Center, San Jose, California

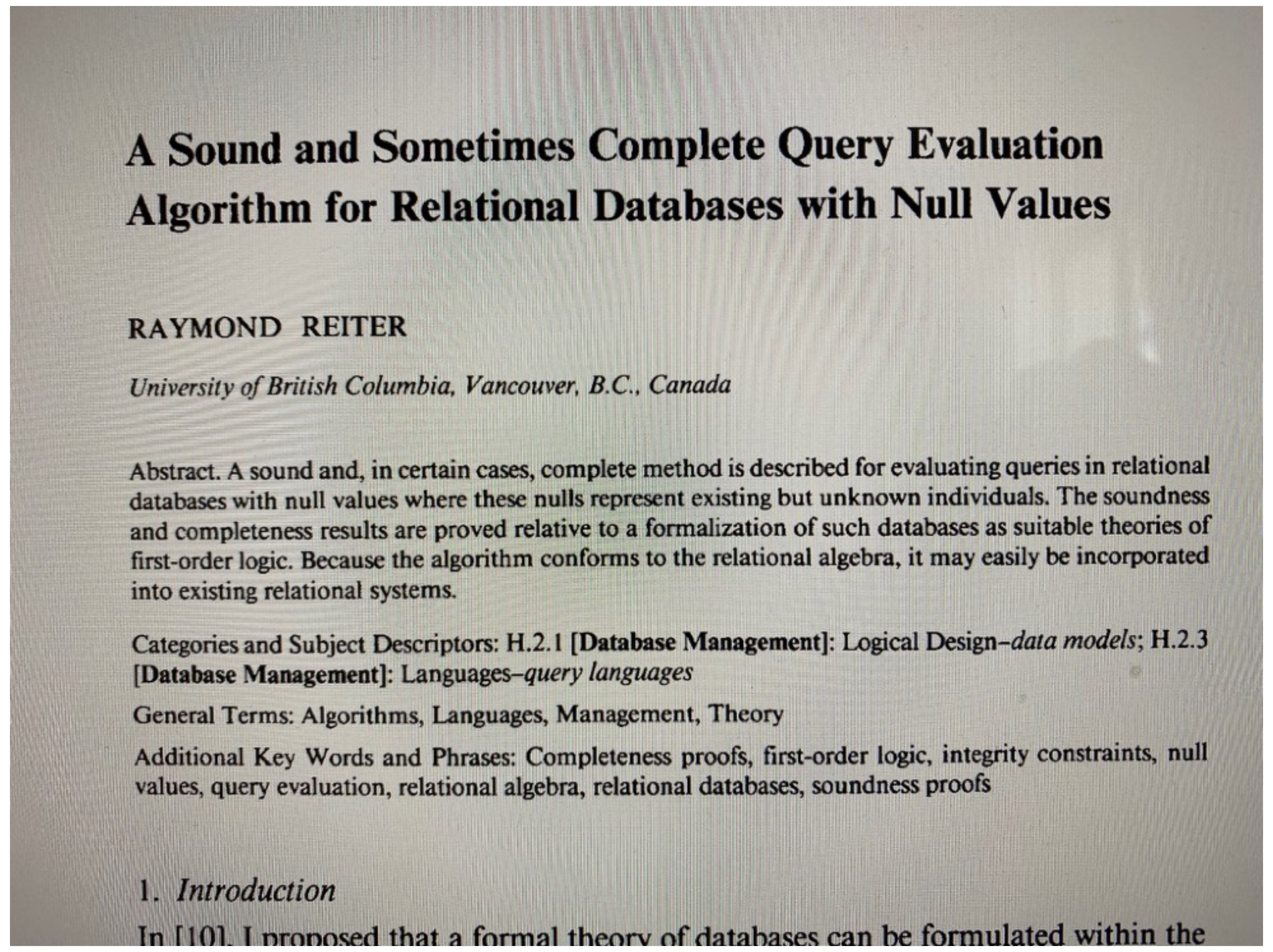
Received September 7, 1985; revised September 24, 1985

We study here the complexity of evaluating quaries in logical databases. We focus on Reither's model of closed-world databases with unknown values. We show that in this setting query evaluation is harder than query evaluation for physical databases. For example, while 1st-order queries over physical databases can be evaluated in logarithmic space, evaluation of 1st-order queries in the studied model is co-NP-complete. We describe an approximation algorithm for query evaluation that enables one to implement a logical database on the top of a standard database management system.

© 1986 Academic Press, Inc.

1. INTRODUCTION

Two classical papers from JACM and JCSS in 1986



A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values

RAYMOND REITER

University of British Columbia, Vancouver, B.C., Canada

Abstract. A sound and, in certain cases, complete method is described for evaluating queries in relational databases with null values where these nulls represent existing but unknown individuals. The soundness and completeness results are proved relative to a formalization of such databases as suitable theories of first-order logic. Because the algorithm conforms to the relational algebra, it may easily be incorporated into existing relational systems.

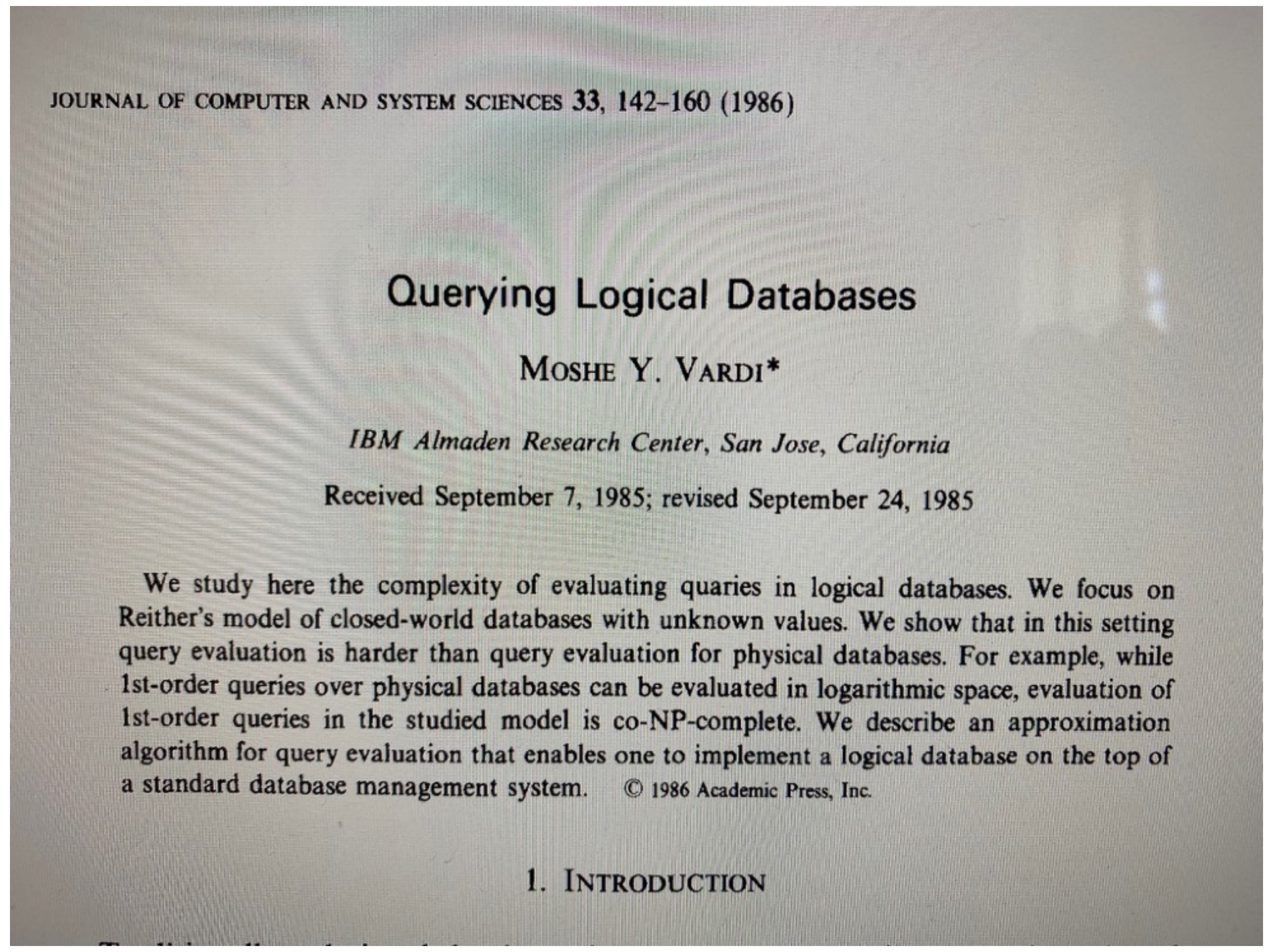
Categories and Subject Descriptors: H.2.1 **[Database Management]**: Logical Design–*data models*; H.2.3 **[Database Management]**: Languages–*query languages*

General Terms: Algorithms, Languages, Management, Theory

Additional Key Words and Phrases: Completeness proofs, first-order logic, integrity constraints, null values, query evaluation, relational algebra, relational databases, soundness proofs

1. *Introduction*

In [10], I proposed that a formal theory of databases can be formulated within the



JOURNAL OF COMPUTER AND SYSTEM SCIENCES 33, 142–160 (1986)

Querying Logical Databases

MOSHE Y. VARDI*

IBM Almaden Research Center, San Jose, California

Received September 7, 1985; revised September 24, 1985

We study here the complexity of evaluating quaries in logical databases. We focus on Reither's model of closed-world databases with unknown values. We show that in this setting query evaluation is harder than query evaluation for physical databases. For example, while 1st-order queries over physical databases can be evaluated in logarithmic space, evaluation of 1st-order queries in the studied model is co-NP-complete. We describe an approximation algorithm for query evaluation that enables one to implement a logical database on the top of a standard database management system. © 1986 Academic Press, Inc.

1. INTRODUCTION

Idea:

A database with incomplete information is a logical theory

Querying such a database is logical entailment:
the database entails the query

This is computationally hard

Hence we need to approximate

How it works

How it works

Relation R

A	B
1	x
x	2
3	y

Defines complete databases

$$\varphi_{OWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \right)$$

More common assumption

$$\varphi_{CWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \wedge \forall u \forall v \left(R(u,v) \rightarrow (u,v) = (1,x) \vee (u,v) = (x,2) \vee (u,v) = (3,y) \right) \right)$$

How it works

Relation R

A	B
1	x
x	2
3	y

Defines complete databases

$$\varphi_{OWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \right)$$

More common assumption

$$\varphi_{CWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \wedge \forall u \forall v \left(R(u,v) \rightarrow (u,v) = (1,x) \vee (u,v) = (x,2) \vee (u,v) = (3,y) \right) \right)$$

Given a query ψ, to answer check whether

$$\models \varphi_{CWA} \rightarrow \psi$$

How it works

Relation R

A	B
1	x
x	2
3	y

Defines complete databases

$$\varphi_{OWA} = \exists x \exists y \left(R(1,x) \land R(x,2) \land R(3,y) \right)$$

More common assumption

$$\varphi_{CWA} = \exists x \exists y \left(R(1,x) \land R(x,2) \land R(3,y) \land \forall u \forall v \left(R(u,v) \rightarrow (u,v) = (1,x) \lor (u,v) = (x,2) \lor (u,v) = (3,y) \right) \right)$$

Given a query ψ, to answer check whether

$$\models \varphi_{CWA} \rightarrow \psi$$

ψ is certainly true

How it works

Relation R

A	B
1	x
x	2
3	y

Defines complete databases

$$\varphi_{OWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \right)$$

More common assumption

$$\varphi_{CWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \wedge \forall u \forall v \left(R(u,v) \rightarrow (u,v) = (1,x) \vee (u,v) = (x,2) \vee (u,v) = (3,y) \right) \right)$$

Given a query ψ, to answer check whether

$$\models \varphi_{CWA} \rightarrow \psi$$

ψ is certainly true

To approximate find a translation $\psi \mapsto \alpha$ so that

$$R \models \alpha \quad \text{implies} \quad \models \varphi_{CWA} \rightarrow \psi$$

How it works

Relation R

A	B
1	x
x	2
3	y

Defines complete databases

$$\varphi_{OWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \right)$$

More common assumption

$$\varphi_{CWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \wedge \forall u \forall v \left(R(u,v) \rightarrow (u,v) = (1,x) \vee (u,v) = (x,2) \vee (u,v) = (3,y) \right) \right)$$

Given a query ψ, to answer check whether

$$\models \varphi_{CWA} \rightarrow \psi$$

ψ is certainly true

To approximate find a translation $\psi \mapsto \alpha$ so that

$$R \models \alpha \quad \text{implies} \quad \models \varphi_{CWA} \rightarrow \psi$$

α approximates certain answer

How it works

Relation R

A	B
1	x
x	2
3	y

Defines complete databases

$$\varphi_{OWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \right)$$

More common assumption

$$\varphi_{CWA} = \exists x \exists y \left(R(1,x) \wedge R(x,2) \wedge R(3,y) \wedge \ \forall u \forall v \ \left(R(u,v) \rightarrow (u,v) = (1,x) \vee (u,v) = (x,2) \vee (u,v) = (3,y) \right) \right)$$

Given a query ψ, to answer check whether

$$\models \varphi_{CWA} \rightarrow \psi$$

ψ is certainly true

To approximate find a translation $\psi \mapsto \alpha$ so that

$$R \models \alpha \quad \text{implies} \quad \models \varphi_{CWA} \rightarrow \psi$$

α approximates certain answer

Example: $\exists u \exists v \left(R(u,v) \wedge (u \neq v) \right)$ is certainly true

What we learned back then

- Answering queries is computationally hard (coNP-hard)
- Everything works well for unions of conjunctive queries
  - $\wedge$, $\vee$, $\exists$ fragment of first-order logic
- Approximation schemes are rather complex (more so in Reiter's paper)
  - Neither of them was implemented (implementable?)

A fresh look 35 years later

Rather than one translation $\psi \mapsto \alpha$ we have two: $\psi \mapsto \psi^t$ and $\psi \mapsto \psi^f$

A fresh look 35 years later

Rather than one translation $\psi \mapsto \alpha$ we have two: $\psi \mapsto \psi^t$ and $\psi \mapsto \psi^f$

$$
\begin{aligned}
R(\bar{x})^t &:= R(\bar{x}) \\
(x = y)^t &:= (x = y) \\
(\psi_1 \wedge \psi_2)^t &:= \psi_1^t \wedge \psi_2^t \\
(\exists x\ \psi)^t &:= \exists x\ \psi^t \\
(\neg \psi)^t &:= \psi^f
\end{aligned}
$$

A fresh look 35 years later

Rather than one translation $\psi \mapsto \alpha$ we have two: $\psi \mapsto \psi^t$ and $\psi \mapsto \psi^f$

$$
\begin{aligned}
R(\bar{x})^t &:= R(\bar{x}) \\
(x = y)^t &:= (x = y) \\
(\psi_1 \wedge \psi_2)^t &:= \psi_1^t \wedge \psi_2^t \\
(\exists x\ \psi)^t &:= \exists x\ \psi^t \\
(\neg \psi)^t &:= \psi^f
\end{aligned}
$$

$$
\begin{aligned}
R(\bar{x})^f &:= \neg \exists y \left(R(\bar{y}) \wedge \bar{x} \Uparrow \bar{y} \right) \\
(x = y)^f &:= \neg(x = y) \wedge \neg \mathsf{null}(x) \wedge \neg \mathsf{null}(y) \\
(\psi_1 \wedge \psi_2)^f &:= \psi_1^f \vee \psi_2^f \\
(\exists x\ \psi)^f &:= \forall x\ \psi^f \\
(\neg \psi)^f &:= \psi^t
\end{aligned}
$$

A fresh look 35 years later

$\bar{x}$ and $\bar{y}$ unify by mapping variables to constants

Rather than one translation $\psi \mapsto \alpha$ we have two: $\psi \mapsto \psi^t$ and $\psi \mapsto \psi^f$

$$
\begin{aligned}
R(\bar{x})^t &:= R(\bar{x}) \\
(x = y)^t &:= (x = y) \\
(\psi_1 \wedge \psi_2)^t &:= \psi_1^t \wedge \psi_2^t \\
(\exists x\ \psi)^t &:= \exists x\ \psi^t \\
(\neg\psi)^t &:= \psi^f
\end{aligned}
$$

$$
\begin{aligned}
R(\bar{x})^f &:= \neg\exists y\ \big(R(\bar{y}) \wedge \bar{x} \Uparrow \bar{y}\big) \\
(x = y)^f &:= \neg(x = y) \wedge \neg\mathsf{null}(x) \wedge \neg\mathsf{null}(y) \\
(\psi_1 \wedge \psi_2)^f &:= \psi_1^f \vee \psi_2^f \\
(\exists x\ \psi)^f &:= \forall x\ \psi^f \\
(\neg\psi)^f &:= \psi^t
\end{aligned}
$$

A fresh look 35 years later

$\bar{x}$ and $\bar{y}$ unify by mapping variables to constants

Rather than one translation $\psi \mapsto \alpha$ we have two: $\psi \mapsto \psi^t$ and $\psi \mapsto \psi^f$

$$\begin{aligned} R(\bar{x})^t &:= R(\bar{x}) \\ (x = y)^t &:= (x = y) \\ (\psi_1 \wedge \psi_2)^t &:= \psi_1^t \wedge \psi_2^t \\ (\exists x\ \psi)^t &:= \exists x\ \psi^t \\ (\neg\psi)^t &:= \psi^f \end{aligned}$$

$$\begin{aligned} R(\bar{x})^f &:= \neg\exists y\ \big(R(\bar{y}) \wedge \bar{x} \Uparrow \bar{y}\big) \\ (x = y)^f &:= \neg(x = y) \wedge \neg\mathrm{null}(x) \wedge \neg\mathrm{null}(y) \\ (\psi_1 \wedge \psi_2)^f &:= \psi_1^f \vee \psi_2^f \\ (\exists x\ \psi)^f &:= \forall x\ \psi^f \\ (\neg\psi)^f &:= \psi^t \end{aligned}$$

What we can prove:

ψ^t produces a subset of certain answers to ψ (thus $\models \psi^t \to \psi$)

ψ^f produces a subset of certain answers to $\neg\psi$ (thus $\models \psi^f \to \neg\psi$)

On a database without nulls ψ and ψ^t coincide

For unions of conjunctive queries ψ and ψ^t coincide

Does it work?

In theory, yes, in practice not quite. But we can be a bit smarter

Does it work?

In theory, yes, in practice not quite. But we can be a bit smarter

Issue: unrestricted negation and disjunction— $R(\bar{x})^f := \neg \exists y \left(R(\bar{y}) \wedge \bar{x} \Uparrow \bar{y} \right)$

$(\psi_1 \wedge \psi_2)^f := \psi_1^f \vee \psi_2^f$ produce **HUGE** sets

Does it work?

In theory, yes, in practice not quite. But we can be a bit smarter

Issue: unrestricted negation and disjunction— $R(\bar{x})^f := \neg \exists y \left(R(\bar{y}) \wedge \bar{x} \Uparrow \bar{y} \right)$

$(\psi_1 \wedge \psi_2)^f := \psi_1^f \vee \psi_2^f$ produce **HUGE** sets

$$
\begin{aligned}
R(\bar{x})^+ &:= R(\bar{x}) \\
(x = y)^+ &:= (x = y) \\
(x \neq y)^+ &:= (x \neq y) \wedge \neg \mathsf{null}(x) \wedge \neg \mathsf{null}(y) \\
(\psi_1 \wedge \psi_2)^+ &:= \psi_1^+ \wedge \psi_2^+ \\
(\exists x \ \psi)^+ &:= \exists x \ \psi^+ \\
(\neg \psi)^+ &:= \neg \exists \bar{y} \left(\psi^?(\bar{y}) \wedge \bar{x} \Uparrow \bar{y} \right)
\end{aligned}
$$

Does it work?

In theory, yes, in practice not quite. But we can be a bit smarter

Issue: unrestricted negation and disjunction — $R(\bar{x})^f := \neg \exists y \left(R(\bar{y}) \land \bar{x} \Uparrow \bar{y} \right)$

$(\psi_1 \land \psi_2)^f := \psi_1^f \lor \psi_2^f$ produce **HUGE** sets

$$
\begin{aligned}
R(\bar{x})^+ &:= R(\bar{x}) \\
(x = y)^+ &:= (x = y) \\
(x \neq y)^+ &:= (x \neq y) \land \neg \mathsf{null}(x) \land \neg \mathsf{null}(y) \\
(\psi_1 \land \psi_2)^+ &:= \psi_1^+ \land \psi_2^+ \\
(\exists x\ \psi)^+ &:= \exists x\ \psi^+ \\
(\neg \psi)^+ &:= \neg \exists \bar{y} \left(\psi^?(\bar{y}) \land \bar{x} \Uparrow \bar{y} \right)
\end{aligned}
$$

$$
\begin{aligned}
R(\bar{x})^? &:= R(\bar{x}) \\
(x = y)^? &:= (x = y) \lor \mathsf{null}(x) \lor \mathsf{null}(y) \\
(x \neq y)^? &:= (x \neq y) \\
(\psi_1 \land \psi_2)^? &:= \psi_1^? \land \exists \bar{y} \left(\psi_2^?[\bar{y}/\bar{x}] \land \bar{x} \Uparrow \bar{y} \right) \\
(\exists x\ \psi)^? &:= \exists x\ \psi^? \\
(\neg \psi)^? &:= \neg \psi^+
\end{aligned}
$$

Does it work?

In theory, yes, in practice not quite. But we can be a bit smarter

Issue: unrestricted negation and disjunction— $R(\bar{x})^f := \neg \exists y \left(R(\bar{y}) \wedge \bar{x} \Uparrow \bar{y}\right)$

$(\psi_1 \wedge \psi_2)^f := \psi_1^f \vee \psi_2^f$ produce **HUGE** sets

$$
\begin{aligned}
R(\bar{x})^+ &:= R(\bar{x}) \\
(x = y)^+ &:= (x = y) \\
(x \neq y)^+ &:= (x \neq y) \wedge \neg\mathsf{null}(x) \wedge \neg\mathsf{null}(y) \\
(\psi_1 \wedge \psi_2)^+ &:= \psi_1^+ \wedge \psi_2^+ \\
(\exists x\ \psi)^+ &:= \exists x\ \psi^+ \\
(\neg\psi)^+ &:= \neg \exists \bar{y} \left(\psi^?(\bar{y}) \wedge \bar{x} \Uparrow \bar{y}\right)
\end{aligned}
$$

$$
\begin{aligned}
R(\bar{x})^? &:= R(\bar{x}) \\
(x = y)^? &:= (x = y) \vee \mathsf{null}(x) \vee \mathsf{null}(y) \\
(x \neq y)^? &:= (x \neq y) \\
(\psi_1 \wedge \psi_2)^? &:= \psi_1^? \wedge \exists \bar{y}\left(\psi_2^?[\bar{y}/\bar{x}] \wedge \bar{x} \Uparrow \bar{y}\right) \\
(\exists x\ \psi)^? &:= \exists x\ \psi^? \\
(\neg\psi)^? &:= \neg\psi^+
\end{aligned}
$$

Tried in TPC-H queries with negation, on databases of sizes up to 10GB.
Scales surprisingly well in about 75% of cases

Moral

- Don’t forget old papers
  - Especially written by giants
- But don’t take them as-is many years later
  - Be inspired and rethink

Why now?

New database theory book

Freely available on GitHub

Over half of the material (≈600pp) already released

We needed a clean chapter on incomplete data

Marcelo Arenas, Pablo Barceló, Leonid Libkin,
Wim Martens, Andreas Pieris

Database Theory

Querying Data

(Preliminary Version)

July 14, 2022

Santiago Paris
Bayreuth Edinburgh