Ontology-based Data Access
a.k.a. Queries and the Open World Assumption

David Toman

D. R. Cheriton School of Computer Science
University of Waterloo

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*?

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 1:

Definition (Implicit Definability)

A query Q is *implicitly definable in Ds* if $Q(M_1) = Q(M_2)$ for all pairs of databases $M_1 \models \mathcal{T}$ and $M_2 \models \mathcal{T}$ s. t. $A_i(M_1) = A_i(M_2)$ for all $A_i \in D$.

1. Chase/Craig Interpolation provides *rewriting* $\psi(D)$
2. In some cases φ is not implicitly definable

$\Rightarrow$ in particular when *OWA* plays a role (e.g., NULLs)

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 1:

Definition (Implicit Definability)

A query Q is *implicitly definable in Ds* if $Q(M_1) = Q(M_2)$ for all pairs of databases $M_1 \models \mathcal{T}$ and $M_2 \models \mathcal{T}$ s. t. $A_i(M_1) = A_i(M_2)$ for all $A_i \in D$.

1. Chase/Craig Interpolation provides *rewriting* $\psi(D)$
2. In some cases φ is not implicitly definable

$\Rightarrow$ in particular when *OWA* plays a role (e.g., NULLs)

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 2:

Definition (Certain Answers)

Answer to $\varphi(D)$ under $\mathcal{T}$:= $\mathrm{cert}_{\mathcal{T},D}(\varphi) = \bigcap_{M \models \mathcal{T} \cup D} \{\vec{a} \mid M, \vec{a} \models \varphi\}$

1. Essentially a variant of [Imielinski&Lipski] approach
2. Answer to φ is *always* defined (unlike in OPTION 1)

. . . any drawbacks?

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 2:

Definition (Certain Answers)

$$\text{Answer to } \varphi(D) \text{ under } \mathcal{T} := \mathrm{cert}_{\mathcal{T},D}(\varphi) = \bigcap_{M \models \mathcal{T} \cup D} \{\vec{a} \mid M, \vec{a} \models \varphi\}$$

1. Essentially a variant of [Imielinski&Lipski] approach
2. Answer to φ is *always* defined (unlike in OPTION 1)

... any drawbacks?

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 2:

Definition (Certain Answers)

$$\text{Answer to } \varphi(D) \text{ under } \mathcal{T} := \text{cert}_{\mathcal{T},D}(\varphi) = \bigcap_{M \models \mathcal{T} \cup D} \{\vec{a} \mid M, \vec{a} \models \varphi\}$$

1. Essentially a variant of [Imielinski&Lipski] approach
2. Answer to φ is *always* defined (unlike in OPTION 1)

. . . any drawbacks?

ODBA: Queries and Ontologies

IDEA:

Queries answers are *logical consequences* of *explicit data*
combined with *background knowledge*
$\Rightarrow$ Ontology-based Data Access (OBDA)

Example

- Bob is a BOSS (explicit data)
- Every BOSS is an EMPloyee (ontology)

List all EMPloyees $\Rightarrow$ {Bob} (query)

ODBA: Queries and Ontologies

IDEA:

Queries answers are *logical consequences* of *explicit data*
combined with *background knowledge*

$\Rightarrow$ Ontology-based Data Access (OBDA)

Example

- Bob is a BOSS (explicit data)
- Every BOSS is an EMPloyee (ontology)

List all EMPloyees $\Rightarrow \{\text{Bob}\}$ (query)

Difficulties: User Expectations

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$

Difficulties: User Expectations

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$

User: *Does Sue have a phone number?*

Information System: *YES*

Difficulties: User Expectations

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$

User: *Does Sue have a phone number?*

Information System: *YES*

User: *OK, tell me Sue's phone number!*

Information System: *(no answer)*

Difficulties: User Expectations

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$

User: *Does Sue have a phone number?*

Information System: *YES*

User: *OK, tell me Sue's phone number!*

Information System: *(no answer)*

User:

Why? Certain Answers

Example (Unintuitive Behaviour of Queries:)

1. $\exists x.Phone(\texttt{"Sue"}, x)$?
2. $Phone(\texttt{"Sue"}, x)$?

under $\mathcal{T} = \{\forall x.Person(x) \to \exists y.Phone(x, y)\}$
and $D = \{Person(\texttt{"Sue"})\}$.

Why? Certain Answers

Example (Unintuitive Behaviour of Queries:)

1. $\exists x.Phone(\texttt{"Sue"}, x)? \Rightarrow$ YES
2. $Phone(\texttt{"Sue"}, x)? \Rightarrow \{\,\}$

under $\mathcal{T} = \{\forall x.Person(x) \rightarrow \exists y.Phone(x, y)\}$
and $D = \{Person(\texttt{"Sue"})\}$.

The problem: Users (essentially) EXPECT CWA

What does $\mathcal{A} = \{EMP(Bob), EMP(Sue)\}$ mean?

OWA: $Bob^{\mathcal{I}} \in EMP^{\mathcal{I}}, Sue^{\mathcal{I}} \in EMP^{\mathcal{I}}$ (KR folks)

CWA: $\{Bob^{\mathcal{I}}, Sue^{\mathcal{I}}\} = EMP^{\mathcal{I}}$ (DB folks and users)

...at least for *their* relations (i.e., in the conceptual schema).

The problem: Users (essentially) EXPECT CWA

What does $\mathcal{A} = \{EMP(Bob), EMP(Sue)\}$ mean?

OWA: $Bob^{\mathcal{I}} \in EMP^{\mathcal{I}}, Sue^{\mathcal{I}} \in EMP^{\mathcal{I}}$ (KR folks)

CWA: $\{Bob^{\mathcal{I}}, Sue^{\mathcal{I}}\} = EMP^{\mathcal{I}}$ (DB folks and users)

…at least for *their* relations (i.e., in the conceptual schema).

Simulations:

CWA in OWA: *closure axioms*: $\forall x.EMP(x) \rightarrow (x = Bob) \vee (x = Sue)$;

OWA in CWA: *auxiliary symbols*: $ExpEMP(Bob), ExpEMP(Sue)$
and *constraints*: $\forall x.ExpEMP(x) \rightarrow EMP(x)$

Certain Answers: What is the Price?

Example

- Schema&Data:

$$\begin{array}{rcl} \mathcal{T} & = & \{ \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \\ & & \quad \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \qquad \} \\ D & = & \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \\ & & \qquad Colour = \{r, g, b\} \ \} \end{array}$$

Certain Answers: What is the Price?

Example

- Schema&Data:

$$\begin{aligned} \mathcal{T} &= \{\ \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \\ &\qquad \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \qquad\qquad \} \\ D &= \{\ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad Colour = \{r, g, b\}\ \} \end{aligned}$$

- Query:

$$\exists x, y, c. Edge(x, y) \wedge ColNode(x, c) \wedge ColNode(y, c)$$

Certain Answers: What is the Price?

Example

- Schema&Data:

$$\begin{aligned} \mathcal{T} &= \{ \forall x, y.ColNode(x, y) \leftrightarrow Node(x), \\ &\quad\ \forall x, y.ColNode(x, y) \leftrightarrow Colour(y) \quad \} \\ D &= \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \\ &\quad\ Colour = \{r, g, b\} \ \} \end{aligned}$$

- Query:

$$\exists x, y, c.Edge(x, y) \land ColNode(x, c) \land ColNode(y, c)$$

$\Rightarrow$ the graph ($Node$, $Edge$) is NOT 3-colourable.

Certain Answers: What is the Price?

Example

- Schema&Data:

$$\mathcal{T} = \{ \forall x, y.ColNode(x,y) \leftrightarrow Node(x), \quad \forall x, y.ColNode(x,y) \leftrightarrow Colour(y) \}$$

$$D = \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \quad Colour = \{r, g, b\} \}$$

- Query:

$$\exists x, y, c.Edge(x,y) \wedge ColNode(x,c) \wedge ColNode(y,c)$$

$\Rightarrow$ the graph (*Node*, *Edge*) is NOT 3-colourable.

coNP-complete for all DLs between $\mathcal{AL}$ and $\mathcal{SHIQ}$

Certain Answers: What is the Price?

Example

- Schema&Data:

$$\mathcal{T} = \{ \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \quad \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \}$$

$$D = \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \quad Colour = \{r, g, b\} \}$$

- Query:

$$\exists x, y, c. Edge(x, y) \wedge ColNode(x, c) \wedge ColNode(y, c)$$

$\Rightarrow$ the graph (*Node*, *Edge*) is NOT 3-colourable.

coNP-complete for all DLs between $\mathcal{AL}$ and $\mathcal{SHIQ}$ (DATA complexity!)

Can this be Done Efficiently at all?

Question

Can there be a *non-trivial* schema language for which *query answering* (under certain answer semantics) is *tractable*?

Can this be Done Efficiently at all?

Question

Can there be a *non-trivial* schema language for which *query answering* (under certain answer semantics) is *tractable*?

YES: *Conjunctive queries* (or positive) and
certain *(dialects of) Description Logics* (or OWL profiles):

1. The DL-Lite family
   $\Rightarrow$ conjunction, $\perp$, domain/range, unqualified $\exists$, role inverse, UNA
   $\Rightarrow$ certain answers in AC_0 for data complexity (i.e., maps to SQL)
2. The $\mathcal{EL}$ family
   $\Rightarrow$ conjunction, qualified $\exists$
   $\Rightarrow$ certain answers *PTIME-complete* for data complexity

Can this be Done Efficiently at all?

Question

Can there be a *non-trivial* schema language for which *query answering* (under certain answer semantics) is *tractable*?

YES: *Conjunctive queries* (or positive) and
certain *(dialects of) Description Logics* (or OWL profiles):

1. The DL-Lite family

   $\Rightarrow$ conjunction, $\perp$, domain/range, unqualified $\exists$, role inverse, UNA

   $\Rightarrow$ certain answers in AC_0 for data complexity (i.e., maps to SQL)

2. The $\mathcal{EL}$ family

   $\Rightarrow$ conjunction, qualified $\exists$

   $\Rightarrow$ certain answers *PTIME-complete* for data complexity

...schemas are *weak on purpose*: queries *must not* be definable.

DL-Lite Family of DLs

Definition (DL-Lite family: Schemata and TBoxes)

1. *Roles* R and *concepts* C as follows:

$$R ::= P \mid P^{-} \qquad C ::= \bot \mid A \mid \exists R$$

2. Schemas are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*

$$C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C \qquad R_1 \sqsubseteq R_2$$

Definition (DL-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA here: ABox does NOT say “these are all the tuples”!

DL-Lite Family of DLs

Definition (DL-Lite family: Schemata and TBoxes)

1. *Roles* R and *concepts* C as follows:

$$R ::= P \mid P^- \qquad C ::= \bot \mid A \mid \exists R$$

2. Schemas are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*

$$C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C \qquad R_1 \sqsubseteq R_2$$

Definition (DL-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA here: ABox does NOT say "these are all the tuples"!

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the query.

Example

TBox (Schema): $\mathit{Employee} \sqsubseteq \exists \mathit{Works}$

$\exists \mathit{Works}^{-} \sqsubseteq \mathit{Project}$

Conjunctive Query: $\exists y.\mathit{Works}(x, y) \wedge \mathit{Project}(y)$

Example

TBox (Schema): $Employee \sqsubseteq \exists Works$
$\exists Works^{-} \sqsubseteq Project$

Conjunctive Query: $\exists y.Works(x, y) \wedge Project(y)$

Rewriting:

$$
\begin{aligned}
Q^{\dagger} = \; & (\exists y.Works(x, y) \wedge Project(y)) \vee \\
& (\exists y, z.Works(x, y) \wedge Works(z, y)) \vee \\
& (\exists y.Works(x, y)) \vee \\
& (Employee(x))
\end{aligned}
$$

Example

TBox (Schema):

$$Employee \sqsubseteq \exists Works$$
$$\exists Works^- \sqsubseteq Project$$

Conjunctive Query: $\exists y.Works(x,y) \wedge Project(y)$

Rewriting:

$$\begin{aligned} Q^\dagger = \; & (\exists y.Works(x,y) \wedge Project(y)) \vee \\ & (\exists y,z.Works(x,y) \wedge Works(z,y)) \vee \\ & (\exists y.Works(x,y)) \vee \\ & (Employee(x)) \end{aligned}$$

Query Execution:

$$Q^\dagger\left(\begin{array}{l}\{\,Employee(bob),\\ \;\;Works(sue, slides)\,\}\end{array}\right)$$

Example

TBox (Schema): $Employee \sqsubseteq \exists Works$
$\exists Works^- \sqsubseteq Project$

Conjunctive Query: $\exists y.Works(x, y) \wedge Project(y)$

Rewriting:

$$
\begin{aligned}
Q^\dagger = \; & (\exists y.Works(x, y) \wedge Project(y)) \vee \\
& (\exists y, z.Works(x, y) \wedge Works(z, y)) \vee \\
& (\exists y.Works(x, y)) \vee \\
& (Employee(x))
\end{aligned}
$$

Query Execution:

$$
Q^\dagger \left(\begin{array}{l} \{ Employee(bob), \\ \;\; Works(sue, slides) \} \end{array} \right) = \{bob, sue\}
$$

QuOnto: Rewriting Approach [Calvanese et al.]

Input: Conjunctive query Q, DL-Lite TBox $\mathcal{T}$
$R = \{Q\}$;
repeat
  foreach *query* $Q' \in R$ **do**
    foreach *axiom* $\alpha \in \mathcal{T}$ **do**
      if α *is applicable to* Q' **then**
        $R = R \cup \{Q'[\text{lhs}(\alpha)/\text{rhs}(\alpha)]\}$
    foreach *two atoms* D_1, D_2 *in* Q' **do**
      if D_1 *and* D_2 *unify* **then**
        $\sigma = MGU(D_1, D_2)$; $R = R \cup \{\lambda(Q', \sigma)\}$;
until *no query unique up to variable renaming can be added to* R;
return $Q^\dagger := (\bigvee R)$

QuOnto: Rewriting Approach [Calvanese et al.]

Input: Conjunctive query Q, DL-Lite TBox $\mathcal{T}$
$R = \{Q\}$;
repeat
  foreach *query* $Q' \in R$ **do**
    foreach *axiom* $\alpha \in \mathcal{T}$ **do**
      if α *is applicable to* Q' **then**
        $R = R \cup \{Q'[\mathsf{lhs}(\alpha)/\mathsf{rhs}(\alpha)]\}$
    foreach *two atoms* D_1, D_2 *in* Q' **do**
      if D_1 *and* D_2 *unify* **then**
        $\sigma = MGU(D_1, D_2)$; $R = R \cup \{\lambda(Q', \sigma)\}$;
until *no query unique up to variable renaming can be added to* R;
return $Q^\dagger := (\bigvee R)$

Theorem

$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q$ *if and only if* $\mathcal{A}, \vec{a} \models Q^\dagger$

QuOnto: Rewriting Approach [Calvanese et al.]

Input: Conjunctive query Q, DL-Lite TBox $\mathcal{T}$
$R = \{Q\}$;
repeat
  foreach *query* $Q' \in R$ **do**
    foreach *axiom* $\alpha \in \mathcal{T}$ **do**
      if α *is applicable to* Q' **then**
        $R = R \cup \{Q'[\text{lhs}(\alpha)/\text{rhs}(\alpha)]\}$
    foreach *two atoms* D_1, D_2 *in* Q' **do**
      if D_1 *and* D_2 *unify* **then**
        $\sigma = MGU(D_1, D_2)$; $R = R \cup \{\lambda(Q', \sigma)\}$;
until *no query unique up to variable renaming can be added to* R;
return $Q^\dagger := (\bigvee R)$

Theorem

$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q$ *if and only if* $\mathcal{A}, \vec{a} \models Q^\dagger$ $\Leftarrow$ *can be VERY large*

$\mathcal{EL}$ Family of DLs

Definition ($\mathcal{EL}$-Lite family: Schemata and TBoxes)

1. *Concepts* C as follows:
$$C ::= A \mid \top \mid \bot \mid C \sqcap C \mid \exists R.C$$
2. Schemas are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*
$$C_1 \sqsubseteq C_2 \qquad\qquad R_1 \sqsubseteq R_2$$

Definition ($\mathcal{EL}$-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA again: ABox does NOT say "these are all the tuples"!

$\mathcal{EL}$ Family of DLs

Definition ($\mathcal{EL}$-Lite family: Schemata and TBoxes)

1. *Concepts C* as follows:

$$C ::= A \mid \top \mid \bot \mid C \sqcap C \mid \exists R.C$$

2. Schemas are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*

$$C_1 \sqsubseteq C_2 \qquad\qquad R_1 \sqsubseteq R_2$$

Definition ($\mathcal{EL}$-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA again: ABox does NOT say "these are all the tuples"!

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the data.

Combined Approach

Can an approach based on *rewriting* be used for $\mathcal{EL}$?

Combined Approach

Can an approach based on *rewriting* be used for $\mathcal{EL}$?

NO: $\mathcal{EL}$ is PTIME-complete for data complexity.

Combined Approach

Can an approach based on *rewriting* be used for $\mathcal{EL}$?

NO: $\mathcal{EL}$ is PTIME-complete for data complexity.

Combined Approach

We effectively transform

1. the ABox $\mathcal{A}$ to a relational database $D_{\mathcal{A}}$ using constraints in $\mathcal{T}$,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

...both *polynomial* in the input(s)

Combined Approach

Can an approach based on *rewriting* be used for $\mathcal{EL}$?

NO: $\mathcal{EL}$ is PTIME-complete for data complexity.

Combined Approach

We effectively transform

1. the ABox $\mathcal{A}$ to a relational database $D_{\mathcal{A}}$ using constraints in $\mathcal{T}$,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

…both *polynomial* in the input(s)

Theorem (Lutz, T., Wolter: IJCAI'09)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \textit{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

Example (with DL-Lite schema)

TBox (Schema): $\mathit{Employee} \sqsubseteq \exists \mathit{Works}$

$\exists \mathit{Works}.\top \sqsubseteq \exists \mathit{Works}.\mathit{Project}$

Conjunctive Query: $\exists y.\mathit{Works}(x, y) \wedge \mathit{Project}(y)$

Data: $\{\mathit{Employee}(\mathit{bob}), \mathit{Works}(\mathit{sue}, \mathit{slides})\}$

Example (with DL-Lite schema)

TBox (Schema): $Employee \sqsubseteq \exists Works$
$\exists Works.\top \sqsubseteq \exists Works.Project$

Conjunctive Query: $\exists y.Works(x, y) \wedge Project(y)$

Data: $\{Employee(bob), Works(sue, slides)\}$

Rewriting:

1. $D_{\mathcal{A}} = \{\ Employee(bob), Works(bob, c_{Works}),$
   $Works(sue, slides), Project(c_{Works}), Project(slides)\ \}$
2. $Q^{\ddagger} = Q \wedge (x \neq c_w)$

Example (with DL-Lite schema)

TBox (Schema): $Employee \sqsubseteq \exists Works$
$\exists Works.\top \sqsubseteq \exists Works.Project$

Conjunctive Query: $\exists y.Works(x, y) \wedge Project(y)$

Data: $\{Employee(bob), Works(sue, slides)\}$

Rewriting:

1. $D_{\mathcal{A}} = \{ \; Employee(bob), Works(bob, c_{Works}), Works(sue, slides), Project(c_{Works}), Project(slides) \; \}$
2. $Q^{\ddagger} = Q \wedge (x \neq c_w)$

Query Execution:

$$Q^{\ddagger}(D_{\mathcal{A}}) = \{bob, sue\}$$

Summary

1. Answering queries over databases *with respect to schema constraints/ontologies* is hard.

2. Choice between:

   Query Definability:

   $\Rightarrow$ expressive schema languages and queries
   $\Rightarrow$ rewritten queries in AC_0 ($\sim$ efficient)
   $\Rightarrow$ but rewriting *is hard to find* and may *not exist*

   Certain Answers:

   $\Rightarrow$ weak schema languages and positive queries only
   $\Rightarrow$ rewritten queries still complex (data complexity)
   $\Rightarrow$ but certain answers are *always defined*