Open-World Finite Query Answering
Under Number Restrictions

Antoine Amarilli\textsuperscript{1,2}

\textsuperscript{1}Télécom ParisTech, Paris, France

\textsuperscript{2}University of Oxford, Oxford, United Kingdom

August 12, 2014
Open-world query answering

- Evaluate query $q$ over instance $I$, open-world assumption:
  - The instance $I$ is correct but incomplete
  - Consider all possible completions $J$ satisfying constraints $\Sigma$
  - Certain answers to query $q$ among those completions

$\Rightarrow$ Formally: $I, \Sigma \models q$ if $J \models q$ for all $J \supseteq I$ s.t. $J \models \Sigma$
Open-world query answering

- Evaluate query $q$ over instance $I$, open-world assumption:
  - The instance $I$ is correct but incomplete
  - Consider all possible completions $J$ satisfying constraints $\Sigma$
  - Certain answers to query $q$ among those completions
  $\Rightarrow$ Formally: $I, \Sigma \models q$ if $J \models q$ for all $J \supseteq I$ s.t. $J \models \Sigma$

- Constraints:
  - TGDs, especially inclusion dependencies (ID)
    $\Rightarrow$ Unary inclusion dependencies (UID): $R[A] \subseteq S[B]$
  - Number restrictions, especially functional dependencies (FD)
Open-world query answering

- Evaluate query q over instance I, open-world assumption:
  - The instance I is correct but incomplete
  - Consider all possible completions J satisfying constraints \( \Sigma \)
  - Certain answers to query q among those completions
  \[ \Rightarrow \text{Formally: } I, \Sigma \models q \text{ if } J \models q \text{ for all } J \supseteq I \text{ s.t. } J \models \Sigma \]

- Constraints:
  - TGDs, especially inclusion dependencies (ID)
    \[ \Rightarrow \text{Unary inclusion dependencies (UID): } R[A] \subseteq S[B] \]
  - Number restrictions, especially functional dependencies (FD)

- Finite vs unrestricted QA

  **Instance:** List of employees
  **Constraint 1:** Each employee reviews some employee (UID)
  **Constraint 2:** At most one reviewer per employee (FD)
  **Query:** Are all employees reviewed?
Table of Contents

1. Introduction
2. Existing approaches
3. Result
4. Proof ideas
5. Conclusion
Undecidability barrier

- Entailment of IDs and FDs is undecidable [Mitchell, 1983]
- Already for binary IDs and unary FDs:
  \[ R[A, B] \subseteq S[C, D], \quad R[A] \rightarrow R[B] \]
  \[ \Rightarrow \] QA (finite or not) is also undecidable [Calì et al., 2003]
  (Remark: this proof requires constants in the query)
Undecidability barrier

- Entailment of IDs and FDs is undecidable [Mitchell, 1983]
- Already for binary IDs and unary FDs:
  \[ R[A, B] \subseteq S[C, D], \ R[A] \rightarrow R[B] \]
  \[ \implies \ QA \ (finite \ or \ not) \ is \ also \ undecidable \ [\text{Calì et al., 2003}] \]
  (Remark: this proof requires constants in the query)

  \[ \implies \ We \ can’t \ have \ everything \]
Idea 1: Separability

- The **chase** for IDs: universal model
- **Intuition**: apply all IDs with fresh elements
- FDs are **separable** from IDs if they do not impact the chase
- Sufficient conditions for separability, e.g., **non-conflicting**:
  - exported positions must not be a **strict superset** of a key
- When separable, we can **ignore** FDs (just check them on I)
Idea 1: Separability

- The **chase** for IDs: universal model
- **Intuition**: apply all IDs with fresh elements
- FDs are **separable** from IDs if they do not impact the chase
- Sufficient conditions for separability, e.g., **non-conflicting**:
  - exported positions must not be a **strict superset** of a key
- When separable, we can **ignore** FDs (just check them on \( I \))
  - The chase is **infinite** in general so it doesn’t work in the finite
  - Finite QA **undecidable** for separable IDs/FDs [Rosati, 2006]
    (intuition: their **finite consequences** may not be separable)
Idea 2: Finite controllability

- **Finite controllability** means that finite and infinite QA coincide
- IDs are **finitely controllable** [Rosati, 2006]
  - Construction: finite chase (chase with distant reuses)
- Generalizes to the **guarded fragment** [Barany et al., 2010]
  - (Guarded means that $\forall/\exists$ must be covered by an atom)
  - Intuition: query acyclification and cycle blowup
- Generalises to IDs/FDs with **foreign keys** condition [Rosati, 2006]
Idea 2: Finite controllability

- **Finite controllability** means that finite and infinite QA coincide
- IDs are **finitely controllable** [Rosati, 2006]
  - Construction: finite chase (chase with distant reuses)
- Generalizes to the **guarded fragment** [Barany et al., 2010]
  - (Guarded means that $\forall/\exists$ must be covered by an atom)
  - Intuition: query acyclification and cycle blowup
- Generalises to IDs/FDs with **foreign keys** condition [Rosati, 2006]
  - FDs are **not expressible** in the guarded fragment.
  - IDs/FDs are **not** finitely controllable!
Idea 3: Arity-two

- Finite and unrestricted QA **decidable** in arity-two for the two-variable guarded fragment and **counting constraints** [Pratt-Hartmann, 2009]
  - Intuition: again, encode the acyclic part of the query
  - Satisfiability **decidable** by reduction to an inequation system

- Explicit construction for DLs [Ibáñez-García et al., 2014]
Idea 3: Arity-two

- Finite and unrestricted QA **decidable** in arity-two for the two-variable guarded fragment and **counting constraints** [Pratt-Hartmann, 2009]
  - **Intuition:** again, encode the acyclic part of the query
  - Satisfiability **decidable** by reduction to an inequation system

- **Explicit construction** for DLs [Ibáñez-García et al., 2014]
  - Only for **arity-two** signatures
  - No clear way to **generalize** to higher arity
Table of Contents

1 Introduction
2 Existing approaches
3 Result
4 Proof ideas
5 Conclusion
Our setting

- So:
  - Finite QA
  - TGDs and EGDs with interaction (not FC)
  - High-arity signatures
  ⇒ Can we have all three?
Our setting

- So:
  - Finite QA
  - TGDs and EGDs with interaction (not FC)
  - High-arity signatures
  \[\Rightarrow\] Can we have all three?
  \[\Rightarrow\] What if we restrict the language to UIDs and FDs?
Our setting

- So:
  - **Finite QA**
  - TGDs and EGDs with interaction (not FC)
  - **High-arity** signatures

⇒ Can we have **all three**?

⇒ What if we restrict the language to **UIDs** and **FDs**?
  - No direct encoding to arity-two (unlike UIDs/UKDs...)
  - **UIDs** are important IDs in practice
  - **UIDs** match the DL intuition
  - **UIDs** are less expressive than BIDs
  - and...
Finite closure for UIDs and FDs

- Implication of UIDs/FDs is **decidable** and PTIME [Cosmadakis et al., 1990]
- Unrestricted and finite **do not coincide**
Finite closure for UIDs and FDs

- Implication of UIDs/FDs is **decidable** and PTIME [Cosmadakis et al., 1990]
- Unrestricted and finite do not coincide
- For unrestricted: implication of FDs and UIDs in isolation
Finite closure for UIDs and FDs

- Implication of UIDs/FDs is **decidable** and PTIME [Cosmadakis et al., 1990]
- Unrestricted and finite do not coincide
- For unrestricted: implication of FDs and UIDs in isolation
- For finite: add cycle reversal:
  - Consider only unary FDs: $R[i] 	o S[j]$
  - When $R[i] \subseteq S[j]$ we have $|R[i]| \leq |S[j]|$
  - When $R[i] \to S[j]$ we have $|R[i]| \geq |S[j]|$
  - Inequality cycles with this encoding
Finite closure for UIDs and FDs

- Implication of UIDs/FDs is **decidable** and PTIME [Cosmadakis et al., 1990]
- Unrestricted and finite do not coincide
- For **unrestricted**: implication of FDs and UIDs in isolation
- For **finite**: add cycle reversal:
  - Consider only **unary FDs**: $R[i] \rightarrow S[j]$
  - When $R[i] \subseteq S[j]$ we have $|R[i]| \leq |S[j]|$
  - When $R[i] \rightarrow S[j]$ we have $|R[i]| \geq |S[j]|$
  - Inequality **cycles** with this encoding
  \[\Rightarrow\] In the finite, such cycles must be reversed
Finite closure example

- $R[2] \subseteq R[1]$
Finite closure example

- $R[2] \subseteq R[1]$
Finite closure example

- $R[2] \subseteq R[1]$

\[ I \]
\[ R(\bullet \bullet \bullet) \]
\[ R(\bullet \bullet ) \]
\[ R(\bullet ) \]
\[ R(\bullet \bullet \bullet \bullet ) \]
Finite closure example

- $R[2] \subseteq R[1]$
Finite closure example

\[ R(\bullet) \subseteq R(\circ) \]

\[ R(\circ) \rightarrow R(\bullet) \]

\[ |R[2]| \leq |R[1]| \]

\[ |R[1]| \leq |R[2]| \]
Finite closure example

- $R[2] \subseteq R[1]$

$\implies |R[2]| \leq |R[1]|$
$\implies |R[1]| \leq |R[2]|$
Finite closure example

- $R[2] \subseteq R[1]$
- $|R[2]| \leq |R[1]|$
- $|R[1]| \leq |R[2]|$
- Add $R[1] \subseteq R[2]$
Finite closure example

- $R[2] \subseteq R[1]$
  - $|R[2]| \leq |R[1]|$
  - $|R[1]| \leq |R[2]|$
- Add $R[1] \subseteq R[2]$
  - No finite model!
In arity-two, UIDs/UFDs finitely controllable up to finite closure [Rosati, 2008, Ibáñez-García et al., 2014]

⇒ To perform finite QA on instance I, UIDs/UFDs Σ:

- Compute Σ* the finite closure of Σ
- Check if I satisfies the UFDs of Σ*
- Perform unrestricted QA with I and Σ*
- Easy because UIDs/UFDs are non-conflicting so separable
In arity-two, UIDs/UFDs **finitely controllable up to finite closure** [Rosati, 2008, Ibáñez-García et al., 2014]

⇒ To perform finite QA on instance $I$, UIDs/UFDs $\Sigma$:

- Compute $\Sigma^*$ the finite closure of $\Sigma$
- Check if $I$ satisfies the UFDs of $\Sigma^*$
- Perform unrestricted QA with $I$ and $\Sigma^*$
- Easy because UIDs/UFDs are non-conflicting so separable

⇒ Does this also hold with higher-arity relations and FDs?
The result

**Theorem**

*UIDs and FDs, though not finitely controllable, are finitely controllable up to finite closure, on arbitrary arity signatures.*
The result

Theorem

UIDs and FDs, though not finitely controllable, are finitely controllable up to finite closure, on arbitrary arity signatures.

⇒ It suffices to show that for any $k$, $l$, and $\Sigma^*$, there is a finite completion of $l$ by $\Sigma^*$ which is universal for queries of size $\leq k$.
Table of Contents

1 Introduction
2 Existing approaches
3 Result
4 Proof ideas
5 Conclusion
Quotienting the chase

\[ R[2] \subseteq R[1] \]

- Consider \( k \)-neighborhood equivalence
Quotienting the chase

Consider $k$-neighborhood equivalence

Quotient the chase by this relation

$R[2] \subseteq R[1]$
Quotienting the chase

\[ R[2] \subseteq R[1] \]

- Consider \( k \)-neighborhood equivalence
- Quotient the chase by this relation
Quotienting the chase

\( R[2] \subseteq R[1] \)

- Consider \( k \)-neighborhood equivalence
- Quotient the chase by this relation
- May violate FDs
### Quotienting the chase

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>chase</td>
<td>chase/≡₂</td>
<td>R(●○○)</td>
</tr>
</tbody>
</table>

- Consider $k$-neighborhood equivalence
- **Quotient** the chase by this relation
- May violate FDs
Quotienting the chase

\[ R[2] \subseteq R[1] \]

- Consider \( k \)-neighborhood equivalence
- Quotient the chase by this relation
- May violate FDs
- Not universal (cycles) even for \( \leq k \)
Quotienting the chase

\[ R[2] \subseteq R[1] \]

- Consider \( k \)-neighborhood equivalence
- Quotient the chase by this relation
- May violate FDs
- Not universal (cycles) even for \( \leq k \)
Quotienting the chase

\[ R[2] \subseteq R[1] \]

- Consider \( k \)-neighborhood equivalence
- Quotient the chase by this relation
- May violate FDs
- Not universal (cycles) even for \( \leq k \)
- Yet universal for \( \leq k \) acyclic queries
Quotienting the chase

\[ R[2] \subseteq R[1] \]

- Consider \( k \)-neighborhood equivalence
- Quotient the chase by this relation
- May violate FDs
- Not universal (cycles) even for \( \leq k \)
- Yet universal for \( \leq k \) acyclic queries
- Keep a homomorphism to this quotient
Frugal chase steps

- Follow the chase

\[ R(\text{circle}, \text{rectangle}, \text{triangle}) \]
Frugal chase steps

Follow the chase

\[ R(\bullet \square \triangle) \]

\[ R[3] \subseteq S[1] \]
Frugal chase steps

Follow the chase

R(● □ △)
S(△ * * * *)
### Frugal chase steps

- Follow the chase
- Partition positions:
  - Exported position
  - Dangerous positions (determiners for an UFD)
  - Non-dangerous positions (the rest)
Frugal chase steps

- Follow the chase
- Partition positions:
  - Exported position
  - Dangerous positions (determiners for an UFD)
  - Non-dangerous positions (the rest)
Frugal chase steps

- Follow the chase
- Partition positions:
  - Exported position
  - Dangerous positions (determiners for an UFD)
  - Non-dangerous positions (the rest)
- Create fresh elements for dangerous
Frugal chase steps

- Follow the chase
- Partition positions:
  - Exported position
  - Dangerous positions (determiners for an UFD)
  - Non-dangerous positions (the rest)
- Create fresh elements for dangerous positions
Frugal chase steps

- Follow the chase
- Partition positions:
  - Exported position
  - Dangerous positions (determiners for an UFD)
  - Non-dangerous positions (the rest)
- Create fresh elements for dangerous
- Reuse clusters for non-dangerous created by initial chasing
Frugal chase steps

- Follow the chase
- Partition positions:
  - Exported position
  - Dangerous positions (determiners for an UFD)
  - Non-dangerous positions (the rest)
- Create fresh elements for dangerous
- Reuse clusters for non-dangerous created by initial chasing
Infinite functional paths

... but only within a cycle

Connect it back (match elements)

More complex if many positions of many relations are involved...

Uses cardinality along cycles...

... but also initial chasing to force "generic neighborhoods"

\[ \mathcal{R}[2] \subseteq \mathcal{R}[1], \quad \mathcal{R}[2] \to \mathcal{R}[1] \]

\[ \mathcal{I} \]

\[ \mathcal{R}(\bullet) \]

\[ \mathcal{A}(\bullet) \]
Infinite functional paths...
Infinite functional paths...

Infinite functional paths...
Blueprint

- Infinite functional paths...
- ... but only within a cycle

Infinite functional paths...
... but only within a cycle

Infinite functional paths...
... but only within a cycle

\[ \begin{align*}
    \text{R}[2] & \subseteq \text{R}[1] \quad \text{R}[2] \rightarrow \text{R}[1]
\end{align*} \]
- Infinite functional paths...
- ... but only within a cycle
- Infinite functional paths...

... but only within a cycle
Blueprint

- Infinite functional paths...
- ... but only within a cycle
- Connect it back (match elements)
Infinite functional paths...
... but only within a cycle
Connect it back (match elements)
- Infinite functional paths...
- ... but only within a cycle
- Connect it back (match elements)
- More complex if many positions of many relations are involved...
Blueprint

- Infinite functional paths...
- ... but only within a cycle
- Connect it back (match elements)
- More complex if many positions of many relations are involved...
- Uses cardinality along cycles...
Infinite functional paths...
... but only within a cycle
Connect it back (match elements)
More complex if many positions of many relations are involved...
Uses cardinality along cycles...
... but also initial chasing to force “generic neighborhoods”
Dependency graph

- Build a DAG on the dependency cycles
Build a DAG on the dependency cycles

R(\* \* \*)
S(\* \* \*)
Dependency graph

- Build a DAG on the dependency cycles
Build a DAG on the dependency cycles
We never create fresh elements for a higher dependency in the DAG
Dependency graph

- Build a DAG on the dependency cycles
- We never create fresh elements for a higher dependency in the DAG
- Satisfy cycles along a topological sort
Build a **DAG** on the dependency **cycles**

- We never create **fresh elements** for a **higher dependency** in the **DAG**
- Satisfy cycles along a **topological sort**

$\Rightarrow$ **Finite extension** that satisfies UID$s$/UFD$s$ with a **homomorphism** to the quotient
Higher-arity FDs

- **Ignored** so far
- May only be triggered at non-dangerous reuses
- **Idea**: if non-dangerous but dangerous for higher-arity FD then no unary key

\[ R(\text{non-danger} \rightarrow \text{export}) \]
Higher-arity FDs

- Ignored so far
- May only be triggered at non-dangerous reuses
- Idea: if non-dangerous but dangerous for higher-arity FD then no unary key

![Diagram of R( ) with non-danger non-danger export export]
Higher-arity FDs

- Ignored so far
- May only be triggered at non-dangerous reuses
- Idea: if non-dangerous but dangerous for higher-arity FD then no unary key

![Diagram](attachment:higherarityfds.png)

- Idea:
  - create many reuse candidates
  - combine them in different patterns

⇒ Lemma: if no UKD then $O(n^{>1})$ patterns for $O(n)$ elements
Higher-arity FDs
Higher-arity FDs
Higher-arity FDs

- **O(n) elements**
- **O(n^{>1}) patterns**

reuse candidates
Higher-arity FDs

- $O(n)$ elements
- $O(n^{>1})$ patterns
- $\ldots$
- $\ldots$
- Model completion

Reused candidates
Higher-arity FDs

- O(n) elements
- O(n^{>1}) patterns

reuse candidates

model completion
Higher-arity FDs

- **O(n) elements**
- **O(n^{>1}) patterns**
- **Reuse candidates**
- **Model completion**
Higher-arity FDs

O(n) elements
O(n>1) patterns

reuse candidates

model completion

R(○) R(□) R(△)
R(●) R(●) R(●)
R(●) R(■) R(●)

...
Higher-arity FDs

- O(n) elements
- O(n^{>1}) patterns
- Reuse candidates
- Model completion
Higher-arity FDs

- **O(n)** elements
- **O(n^{>1})** patterns

Reuse candidates

**Model completion**
Higher-arity FDs

- \( O(n) \) elements
- \( O(n^{>1}) \) patterns
- \( O(n) \) reuses

Model completion
Blowing up cycles

- Usually: product with a group of high girth [Otto, 2002]
- Ensures all non-instance cycles are large
- We cannot do it directly as it would violate the FDs
Blowing up cycles

- Usually: product with a group of high girth [Otto, 2002]
- Ensures all non-instance cycles are large
- We cannot do it directly as it would violate the FDs
- However, intuition:
  - FD violations can only appear on non-dangerous reuses...
  - ... and those facts are mapped to the same quotient fact
  - ... so cycles on them are self-homomorphic in the quotient model $M$

```
R( ○ □ △ )
R( ○ □ △ )
```

non-danger reuses
Blowing up cycles

- Usually: product with a group of high girth [Otto, 2002]
- Ensures all non-instance cycles are large
- We cannot do it directly as it would violate the FDs
- However, intuition:
  - FD violations can only appear on non-dangerous reuses...
  - ... and those facts are mapped to the same quotient fact
  - ... so cycles on them are self-homomorphic in the quotient

\[
\begin{align*}
R(\text{non-danger reuses}) & \overset{h}{\rightarrow} R(\text{same quotient fact}) \\
\text{model } M \overset{\text{chase/}\equiv_k}{\rightarrow} R(\text{self-homomorphic in the quotient})
\end{align*}
\]
Blowing up cycles

- Usually: product with a group of high girth [Otto, 2002]
- Ensures all non-instance cycles are large
- We cannot do it directly as it would violate the FDs
- However, intuition:
  - FD violations can only appear on non-dangerous reuses...
  - ... and those facts are mapped to the same quotient fact
  - ... so cycles on them are self-homomomorphic in the quotient

\[ R(\bullet\square\triangle) \quad h \quad R(\circ\square\triangle) \]
\[ \text{model } M \quad \text{chase}/\equiv_k \quad \text{non-danger reuses} \]

⇒ Blow up cycles, but not those cycles
Blowing up cycles via the product

\[ \text{chase/} \equiv_k \text{-} \text{k-acyclic-universal} \]
Blowing up cycles via the product

satisfies $\Sigma^*$
safe overlaps

$\mathcal{M} \xrightarrow{\text{hom}} \text{chase/}\equiv_k \text{ k-acyclic-universal}$
Blowing up cycles via the product

\[ \text{satisfies } \Sigma^* \]
\[ \text{safe overlaps} \]

\[ M \]
\[ \text{prod} \]

\[ M \times G \]
\[ \text{satisfies } \Sigma^* \]

\[ \text{hom} \]

\[ \text{k-acyclic-universal} \]

\[ \text{chase/}\equiv_k \]

\[ \text{prod} \]

\[ \text{chase/}\equiv_k \]
\[ \times G \]
\[ \text{k-universal} \]
Blowing up cycles via the product

satisfies $\Sigma^*$
safe overlaps

\[ M \]
\[ \text{prod} \]
\[ \text{hom} \]

\[ M \times G \]
\[ \text{satisfies } \Sigma^* \]
k-universal

k-acyclic-
universal

\[ \text{chase/eq}_k \]
\[ \text{prod} \]
\[ \text{hom} \]

\[ M \times G \]
\[ \text{k-universal} \]
# Table of Contents

1. Introduction
2. Existing approaches
3. Result
4. Proof ideas
5. Conclusion
Summary

- UIDs/FDs finitely controllable up to closure

- **Main differences with arity-two:**
  - Clusters for non-dangerous reuses
  - Combinations for higher-arity FDs
  - More complex reconnexions along cycles
  - More elaborate cycle elimination (via the quotient)

⇒ **Generalize to richer unary languages for high arity?**
  - Need construction for finite implication
  - Does the finite model construction adapt?
Summary

- UIDs/FDs finitely controllable up to closure
- Main differences with arity-two:
  - Clusters for non-dangerous reuses
  - Combinations for higher-arity FDs
  - More complex reconnexions along cycles
  - More elaborate cycle elimination (via the quotient)

⇒ Generalize to richer unary languages for high arity?
  - Need construction for finite implication
  - Does the finite model construction adapt?

Thanks for your attention!


