Probabilistic Databases: Other Topics and Conclusion

Antoine Amarilli

Table of contents

Recursive and homomorphism-closed queries

Going to more general queries

The case of UCQs is settled! But what about more expressive queries?

Going to more general queries

The case of ***UCQs*** *is settled! But what about* ***more expressive queries****?*

- Work by [Fink and Olteanu, 2016] about **negation**
- Some work on **ontology-mediated query answering** ([Jung and Lutz, 2012])

Going to more general queries

*The case of **UCQs** is settled! But what about **more expressive queries**?*

- Work by [Fink and Olteanu, 2016] about **negation**
- Some work on **ontology-mediated query answering** ([Jung and Lutz, 2012])

We study the case of **queries closed under homomorphisms**

Going to more general queries

The case of UCQs is settled! But what about more expressive queries?

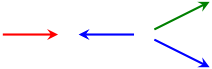
- Work by [Fink and Olteanu, 2016] about negation
- Some work on ontology-mediated query answering ([Jung and Lutz, 2012])

We study the case of queries closed under homomorphisms

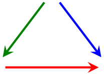
→ We restrict to arity-two signatures (work in progress...)

Homomorphism-closed queries

- A homomorphism from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

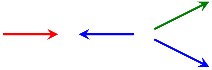
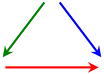


has a homomorphism to



Homomorphism-closed queries

- A homomorphism from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

 has a homomorphism to 

- Homomorphism-closed query Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q

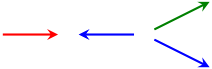
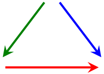
Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.

Homomorphism-closed queries

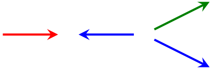
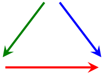
- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

 has a homomorphism to 

- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- Queries with **negations** or **inequalities** are not homomorphism-closed

Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

 has a homomorphism to 

- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- Queries with **negations** or **inequalities** are not homomorphism-closed
- Homomorphism-closed queries can equivalently be seen as **infinite unions of CQs** (corresponding to their models)

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either* Q *is equivalent to a tractable UCQ and* $\mathrm{PQE}(Q)$ *is in PTIME*
- *In all other cases,* $\mathrm{PQE}(Q)$ *is #P-hard*

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either* **Q** *is equivalent to a tractable UCQ and* $\mathrm{PQE}(Q)$ *is in PTIME*
- *In all other cases,* $\mathrm{PQE}(Q)$ *is #P-hard*

- The same holds for RPQs, Datalog queries, etc.

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either **Q** is equivalent to a tractable UCQ and $\mathrm{PQE}(Q)$ is in PTIME*
- *In all other cases, $\mathrm{PQE}(Q)$ is #P-hard*

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ *Q*: $\xrightarrow{} (\xrightarrow{})^* \xrightarrow{}$

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either* Q *is equivalent to a tractable UCQ and* $\mathrm{PQE}(Q)$ *is in PTIME*
- *In all other cases,* $\mathrm{PQE}(Q)$ *is #P-hard*

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ Q: $\xrightarrow{} (\xrightarrow{})^{*} \xrightarrow{}$
  - It is not equivalent to a UCQ: infinite disjunction $\xrightarrow{} (\xrightarrow{})^{i} \xrightarrow{}$ for all $i \in \mathbb{N}$

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either **Q** is equivalent to a tractable UCQ and $\mathrm{PQE}(Q)$ is in PTIME*
- *In all other cases, $\mathrm{PQE}(Q)$ is #P-hard*

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ Q: $\rightarrow (\rightarrow)^* \rightarrow$
  - It is not equivalent to a UCQ: infinite disjunction $\rightarrow (\rightarrow)^i \rightarrow$ for all $i \in \mathbb{N}$
  - Hence, $\mathrm{PQE}(Q)$ is #P-hard

Uniform probabilities

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

- The PQE problem becomes the **uniform reliability** (UR) problem:
  - $\rightarrow$ $\mathrm{UR}(Q)$: given a graph, how many of its subgraphs satisfy Q

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

- The PQE problem becomes the **uniform reliability** (UR) problem:
  - $\rightarrow$ $\mathrm{UR}(Q)$: given a graph, how many of its subgraphs satisfy Q
- The UR problem **reduces** to PQE, but no obvious reduction in the other direction

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$?

- The PQE problem becomes the **uniform reliability** (UR) problem:
  - $\rightarrow$ $\mathrm{UR}(Q)$: given a graph, how many of its subgraphs satisfy Q
- The UR problem **reduces** to PQE, but no obvious reduction in the other direction

We limit to **self-join-free CQs** and extend the "small" Dalvi and Suciu dichotomy to UR:

Theorem (A., Kimelfeld, 2022)

Let Q be a self-join-free CQ:

- *If Q is **hierarchical**, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, even $\mathrm{UR}(Q)$ is #P-hard*

Approximate evaluation

Approximation

- When it's too hard to compute the exact probability, we can approximate it

Approximation

- When it's too hard to compute the exact probability, we can approximate it
- One possibility is to compute a lower bound and upper bound:
  - $\max(\Pr(\phi), \Pr(\psi)) \leq \Pr(\phi \vee \psi) \leq \min(\Pr(\phi) + \Pr(\psi), 1)$
  - $\max(0, \Pr(\phi) + \Pr(\psi) - 1) \leq \Pr(\phi \wedge \psi) \leq \min(\Pr(\phi), \Pr(\psi))$ (by duality)
  - $\Pr(\neg\phi) = 1 - \Pr(\phi)$ (reminder)

Approximation by sampling

Another possibility is to approximate via Monte-Carlo sampling:

- Pick a random possible world according to the fact probabilities:
  - → Keep F with probability $\Pr(F)$ and discard it otherwise
  - → Repeat for the other variables

Approximation by sampling

Another possibility is to approximate via Monte-Carlo sampling:

- Pick a random possible world according to the fact probabilities:
  - $\rightarrow$ Keep F with probability $\Pr(F)$ and discard it otherwise
  - $\rightarrow$ Repeat for the other variables
- Evaluate the lineage formula ϕ under this valuation

Approximation by sampling

Another possibility is to approximate via Monte-Carlo sampling:

- Pick a random possible world according to the fact probabilities:
  - → Keep F with probability $\Pr(F)$ and discard it otherwise
  - → Repeat for the other variables
- Evaluate the lineage formula ϕ under this valuation
- Approximate the probability of the formula ϕ as the proportion of times when it was true

Approximation by sampling

Another possibility is to approximate via Monte-Carlo sampling:

- Pick a random possible world according to the fact probabilities:
  - $\rightarrow$ Keep F with probability $\Pr(F)$ and discard it otherwise
  - $\rightarrow$ Repeat for the other variables
- Evaluate the lineage formula ϕ under this valuation
- Approximate the probability of the formula ϕ as the proportion of times when it was true
- Theoretical guarantees: on how many samples suffice so that, with high probability, the estimated probability is almost correct

Other method for a multiplicative approximation: Karp-Luby algorithm

Using external tools

- Specialized software to compute the probability of a formula: weighted model counters
- Examples (ongoing research):
  - c2d: `http://reasoning.cs.ucla.edu/c2d/download.php`
  - d4: `https://www.cril.univ-artois.fr/KC/d4.html`
  - dsharp: `https://bitbucket.org/haz/dsharp`

Repairs

Repairs

- Another kind of uncertainty: we know that the database must satisfy some **constraints** (e.g., functionality)
- The data that we have does **not** satisfy it
- Reason about the ways to **repair** the data, e.g., removing a minimal subset of tuples
- Can we **evaluate queries** on this representation? E.g., is a query true on **every maximal repair**? See, e.g., [Koutris and Wijsen, 2015].

Summary and directions

Summary of what we have seen

- Probabilistic database model: TIDs, facts have independent probabilities
  - $\rightarrow$ Also more expressive models: BIDs, pc-tables

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  - → Extends to a more complex dichotomy on UCQs

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  → Extends to a more complex dichotomy on UCQs
- We can make **all queries in MSO** tractable by bounding the instance **treewidth**
  → And MSO is intractable if you do not bound treewidth (under some conditions)

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  → Extends to a more complex dichotomy on UCQs
- We can make **all queries in MSO** tractable by bounding the instance **treewidth**
  → And MSO is intractable if you do not bound treewidth (under some conditions)
- Extensions: homomorphism-closed queries, uniform reliability...

Other topics of research

- Queries with **negation** [Fink and Olteanu, 2016]
- Queries with **inequalities** [Olteanu and Huang, 2009]
- **Symmetric model counting** [Beame et al., 2015]
- A summary: Dan Suciu, *Probabilistic Databases for All* [Suciu, 2020]

Other topics of research

- Queries with **negation** [Fink and Olteanu, 2016]
- Queries with **inequalities** [Olteanu and Huang, 2009]
- **Symmetric model counting** [Beame et al., 2015]
- A summary: Dan Suciu, *Probabilistic Databases for All* [Suciu, 2020]

And recently:

- **Infinite domains** [Carmeli et al., 2021]
- **PQE under updates** [Berkholz and Merz, 2021]
- **Open-world probabilistic databases** [Ceylan et al., 2021]
- **Active probabilistic databases** [Drien et al., 2022]

Other topics of research

- Queries with **negation** [Fink and Olteanu, 2016]
- Queries with **inequalities** [Olteanu and Huang, 2009]
- **Symmetric model counting** [Beame et al., 2015]
- A summary: Dan Suciu, *Probabilistic Databases for All* [Suciu, 2020]

And recently:

- **Infinite domains** [Carmeli et al., 2021]
- **PQE under updates** [Berkholz and Merz, 2021]
- **Open-world probabilistic databases** [Ceylan et al., 2021]
- **Active probabilistic databases** [Drien et al., 2022]
- (Others? talk to me :))

Future research directions

- **Reusability** of techniques : repairs, Shapley values, graphical models, probabilistic programming, probabilistic constraints...

Future research directions

- **Reusability** of techniques : repairs, Shapley values, graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]

Future research directions

- **Reusability** of techniques : repairs, Shapley values, graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?

Future research directions

- **Reusability** of techniques : repairs, Shapley values, graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?
- Connections to knowledge compilation and **intensional–extensional** conjecture
  - → Can we compute the **provenance** of tractable UCQs in a tractable formalism, e.g., **d-Ds**?

Future research directions

- **Reusability** of techniques : repairs, Shapley values, graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?
- Connections to knowledge compilation and **intensional–extensional** conjecture
  - → Can we compute the **provenance** of tractable UCQs in a tractable formalism, e.g., **d-Ds**?
- Combining the **query-based** and **structure-based** approaches

Future research directions

- **Reusability** of techniques : repairs, Shapley values, graphical models, probabilistic programming, probabilistic constraints…
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?
- Connections to knowledge compilation and **intensional–extensional** conjecture
  - → Can we compute the **provenance** of tractable UCQs in a tractable formalism, e.g., **d-Ds**?
- Combining the **query-based** and **structure-based** approaches

Thanks for your attention!

References i

- Abiteboul, S., Kimelfeld, B., Sagiv, Y., and Senellart, P. (2009).
  On the expressiveness of probabilistic XML models.
  VLDB Journal, 18(5).
- Amarilli, A., Bourhis, P., and Senellart, P. (2015).
  Provenance circuits for trees and treelike instances.
  In *ICALP*.
- Amarilli, A., Bourhis, P., and Senellart, P. (2016).
  Tractable lineages on treelike instances: Limits and extensions.
  In *PODS*.

References ii

Amarilli, A. and Ceylan, I. I. (2020).
A dichotomy for homomorphism-closed queries on probabilistic graphs.
In *ICDT*.

Amarilli, A. and Kimelfeld, B. (2022).
Uniform Reliability of Self-Join-Free Conjunctive Queries.
Under review.

Beame, P., Van den Broeck, G., Gribkoff, E., and Suciu, D. (2015).
Symmetric weighted first-order model counting.
In *PODS*.

Benedikt, M., Kharlamov, E., Olteanu, D., and Senellart, P. (2010).
Probabilistic XML via Markov chains.
PVLDB, 3(1).

Berkholz, C. and Merz, M. (2021).
Probabilistic databases under updates: Boolean query evaluation and ranked enumeration.
In *PODS*.

Carmeli, N., Grohe, M., Lindner, P., and Standke, C. (2021).
Tuple-independent representations of infinite probabilistic databases.
In *PODS*.

Ceylan, I. I., Darwiche, A., and Van den Broeck, G. (2021).
Open-world probabilistic databases: Semantics, algorithms, complexity.
Artificial Intelligence, 295.

Cohen, S., Kimelfeld, B., and Sagiv, Y. (2009).
Running tree automata on probabilistic xml.
In *PODS*.

Dalvi, N., Ré, C., and Suciu, D. (2009).
Probabilistic databases: Diamonds in the dirt.
Communications of the ACM, 52(7).

Dalvi, N. N. and Suciu, D. (2004).
Efficient query evaluation on probabilistic databases.
In *VLDB*.

Drien, O., Freiman, M., and Amsterdamer, Y. (2022).
ActivePDB: Active probabilistic databases.
Working draft.

References v

Fink, R. and Olteanu, D. (2016).
Dichotomies for queries with negation in probabilistic databases.
ACM Transactions on Database Systems, 41(1).

Imielinski, T. and Lipski, W. (1984).
Incomplete information in relational databases.
Journal of the ACM, 31(4).

Jung, J. C. and Lutz, C. (2012).
Ontology-based access to probabilistic data with OWL QL.
In *ISWC*.

Kenig, B. and Suciu, D. (2021).
A dichotomy for the generalized model counting problem for unions of conjunctive queries.
In *PODS*.

Koutris, P. and Wijsen, J. (2015).
The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints.
In *SIGMOD*.

Olteanu, D. and Huang, J. (2009).
Secondary-storage confidence computation for conjunctive queries with inequalities.
In *SIGMOD*.

Suciu, D. (2020).
Probabilistic databases for all.
In *PODS*.

Widom, J. (2005).
Trio: A system for integrated management of data, accuracy, and lineage.
In *CIDR*.

Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

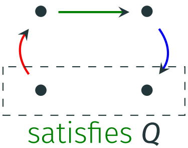
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges $\rightarrow$ $\rightarrow$ $\rightarrow$ such that:



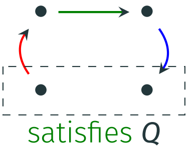
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges $\rightarrow$ $\rightarrow$ $\rightarrow$ such that:



but

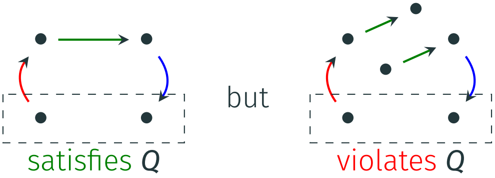
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

Idea: find a tight pattern, i.e., a graph with three distinguished edges $\rightarrow$ $\rightarrow$ $\rightarrow$ such that:



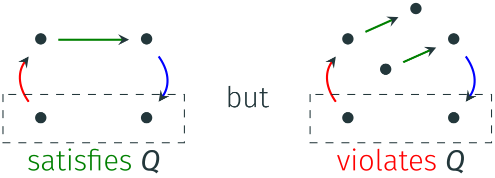
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ is #P-hard

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges $\rightarrow$ $\rightarrow$ $\rightarrow$ such that:

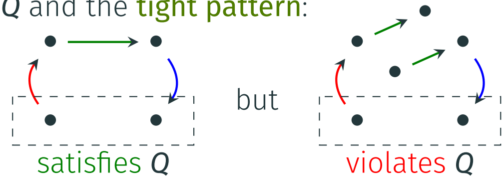


Theorem

Any unbounded query closed under homomorphisms has a tight pattern

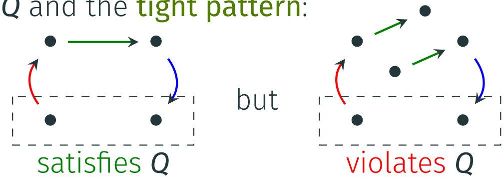
Using tight patterns to show hardness of PQE

- Fix the query Q and the tight pattern:



Using tight patterns to show hardness of PQE

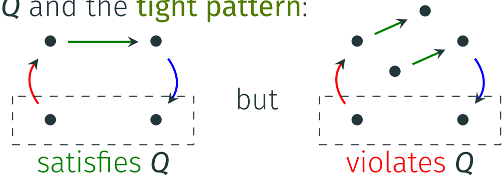
- Fix the query Q and the **tight pattern**:



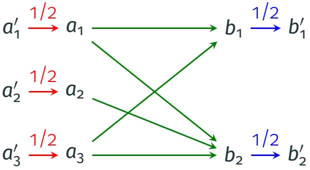
- We reduce from PQE for the **intractable** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$

Using tight patterns to show hardness of PQE

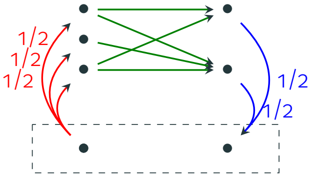
- Fix the query Q and the **tight pattern**:



- We reduce from PQE for the **intractable** CQ: $Q_0 : x \rightarrow y \rightarrow z \rightarrow w$

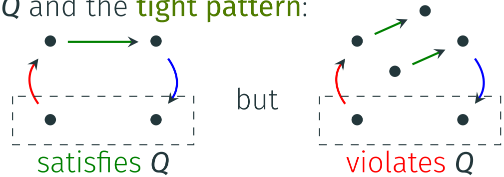


is coded as

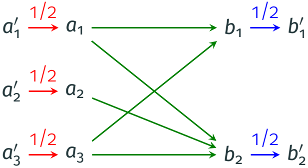


Using tight patterns to show hardness of PQE

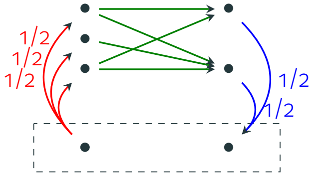
- Fix the query Q and the **tight pattern**:



- We reduce from PQE for the **intractable** CQ: $Q_0 : x \xrightarrow{} y \xrightarrow{} z \xrightarrow{} w$



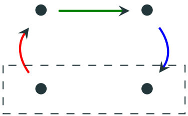
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

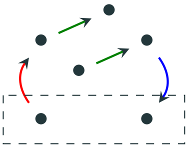
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:



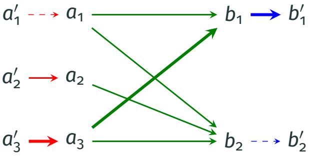
satisfies Q

but

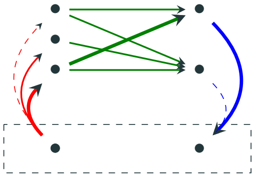


violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \to y \to z \to w$



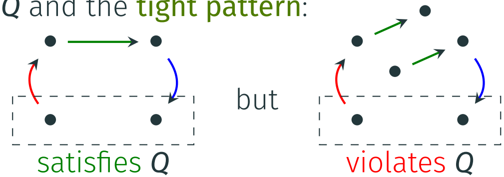
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0 iff the corresponding possible world of the TID at the **right** satisfies the query Q...

Using tight patterns to show hardness of PQE

- Fix the query Q and the tight pattern:

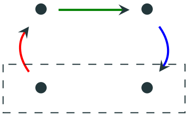


- We reduce from PQE for the intractable CQ: $Q_0 : x \rightarrow y \rightarrow z \rightarrow w$

Idea: possible worlds at the left have a path that matches Q_0
iff the corresponding possible world of the TID at the right satisfies the query Q...

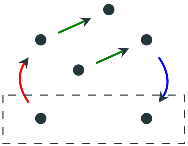
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:



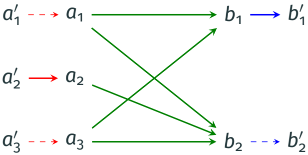
satisfies Q

but

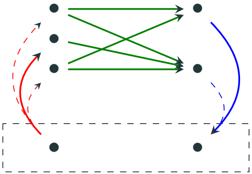


violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \rightarrow y \rightarrow z \rightarrow w$



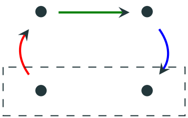
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...
... except we need **more** from the tight pattern!

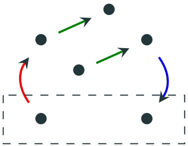
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:



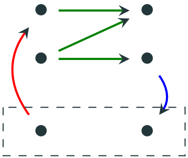
satisfies Q

but



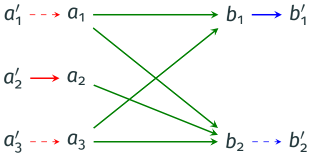
violates Q

AND

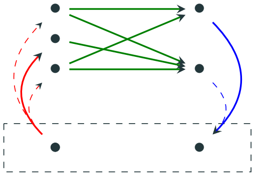


violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \rightarrow y \rightarrow z \rightarrow w$



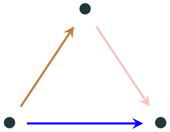
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...
... except we need **more** from the tight pattern!

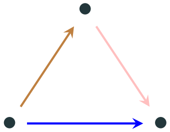
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:

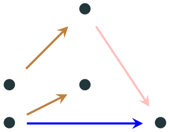


Why can we always find tight patterns?

- Unbounded queries have arbitrarily large minimal models
- Take a large minimal model D and disconnect its edges:

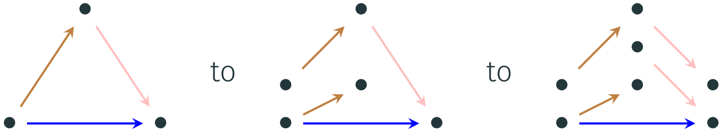


to



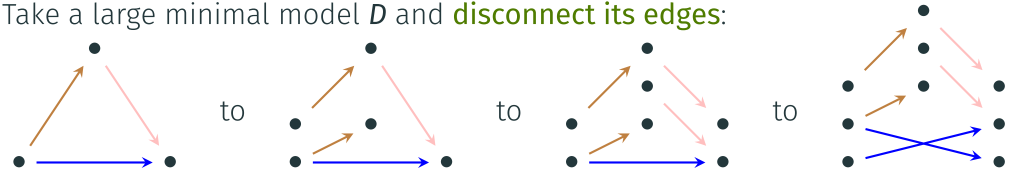
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



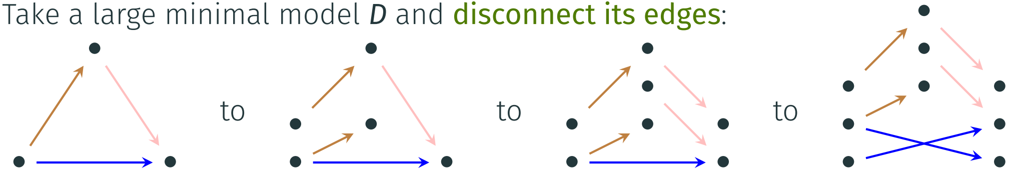
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model *D* and **disconnect its edges**:



Why can we always find tight patterns?

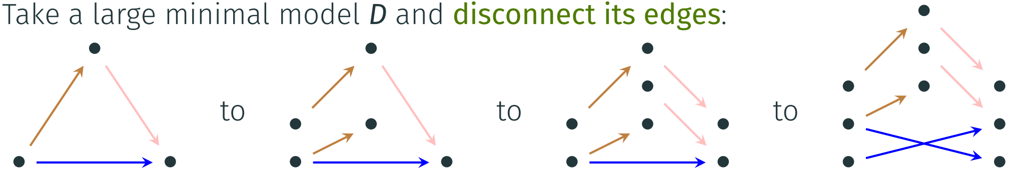
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**

Why can we always find tight patterns?

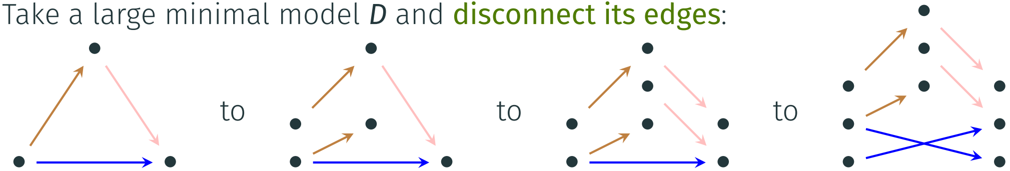
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**

Why can we always find tight patterns?

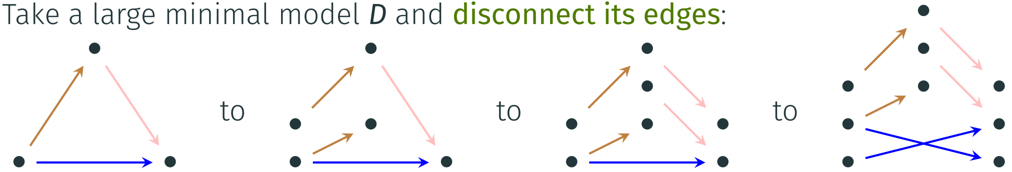
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'

Why can we always find tight patterns?

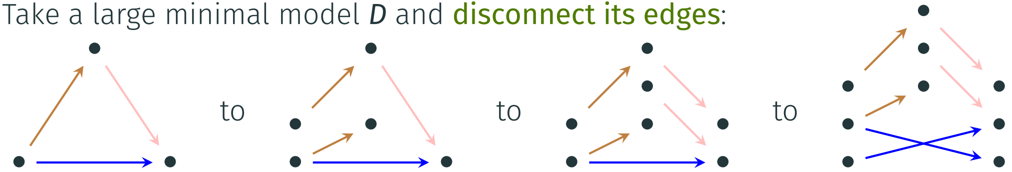
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'
  - It is **homomorphically equivalent** to a constant-sized D'' satisfying Q

Why can we always find tight patterns?

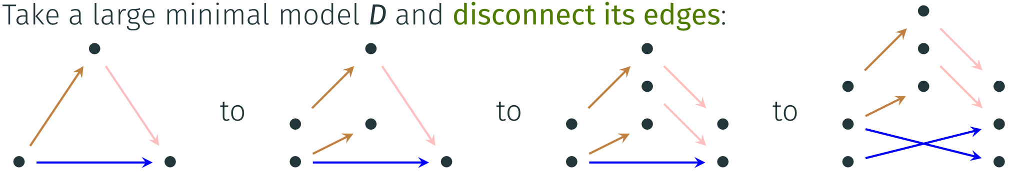
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'
  - It is **homomorphically equivalent** to a constant-sized D'' satisfying Q
  - D'' has a **homomorphism** back to D

Why can we always find tight patterns?

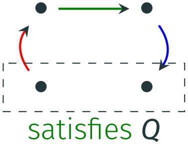
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



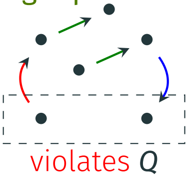
- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'
  - It is **homomorphically equivalent** to a constant-sized D'' satisfying Q
  - D'' has a **homomorphism** back to D
  - This contradicts the **minimality** of the large D

Rescuing the proof

We know that we have a **tight pattern**:

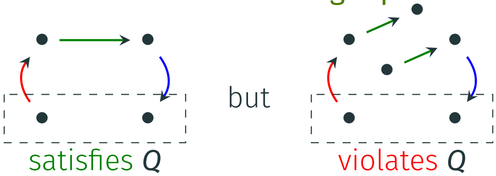


but



Rescuing the proof

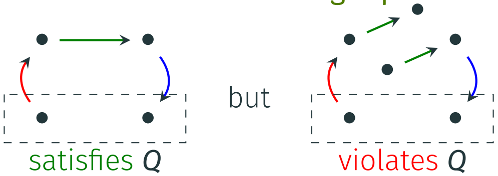
We know that we have a **tight pattern**:



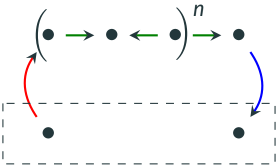
Consider its **iterates**

Rescuing the proof

We know that we have a **tight pattern**:

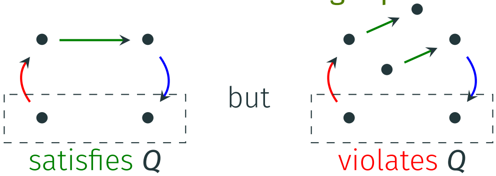


Consider its **iterates** for each $n \in \mathbb{N}$:

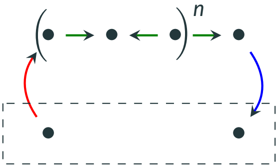


Rescuing the proof

We know that we have a **tight pattern**:

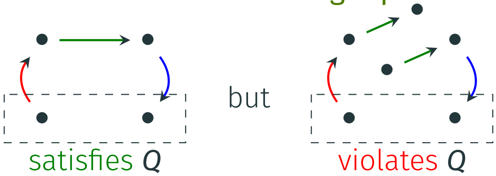


Consider its **iterates** for each $n \in \mathbb{N}$:

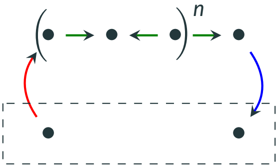


Rescuing the proof

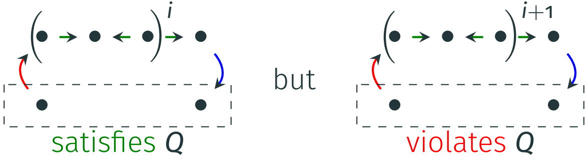
We know that we have a **tight pattern**:



Consider its **iterates** for each $n \in \mathbb{N}$:

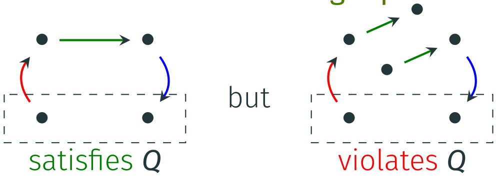


Case 1: some iterate **violates** the query:

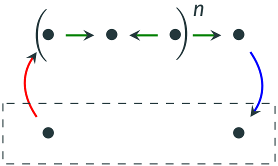


Rescuing the proof

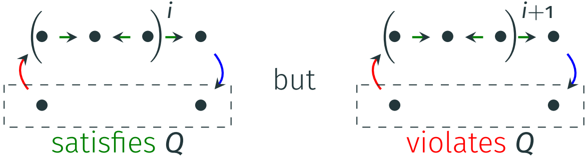
We know that we have a **tight pattern**:



Consider its **iterates** for each $n \in \mathbb{N}$:



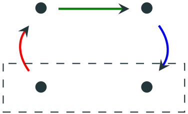
Case 1: some iterate **violates** the query:



$\rightarrow$ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

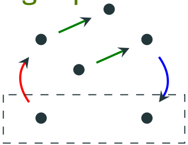
Rescuing the proof

We know that we have a **tight pattern**:



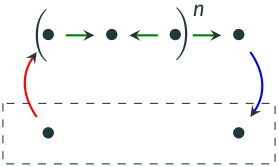
satisfies Q

but

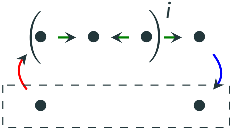


violates Q

Consider its **iterates** for each $n \in \mathbb{N}$:

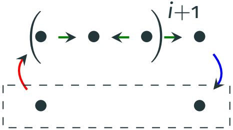


Case 1: some iterate **violates** the query:



satisfies Q

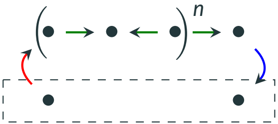
but



violates Q

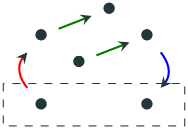
$\rightarrow$ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

Case 2: all iterates **satisfy** the query:



satisfies Q for all $n \in \mathbb{N}$

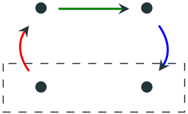
but



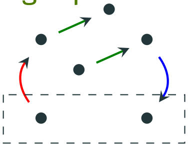
violates Q

Rescuing the proof

We know that we have a **tight pattern**:

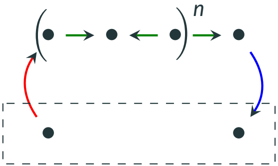


but

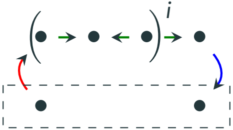


satisfies Q violates Q

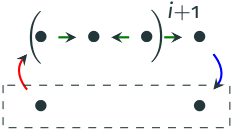
Consider its **iterates** for each $n \in \mathbb{N}$:



Case 1: some iterate **violates** the query:



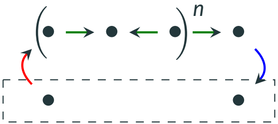
but



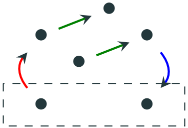
satisfies Q violates Q

→ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

Case 2: all iterates **satisfy** the query:



but

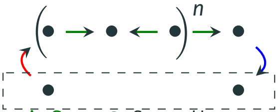


satisfies Q for all $n \in \mathbb{N}$ violates Q

→ Call this an **iterable pattern**

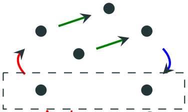
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

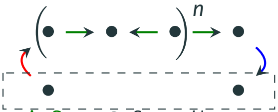
but



violates Q

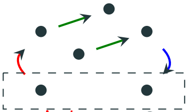
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

but



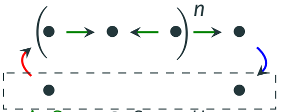
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?

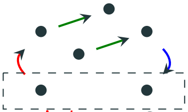
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

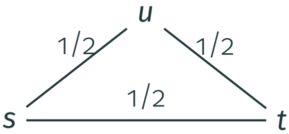
but



violates Q

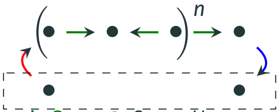
Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



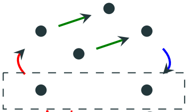
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

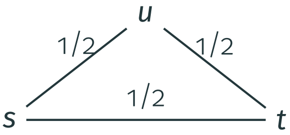
but



violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

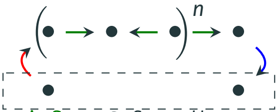
- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as

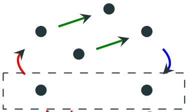
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

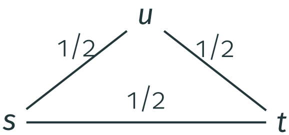
but



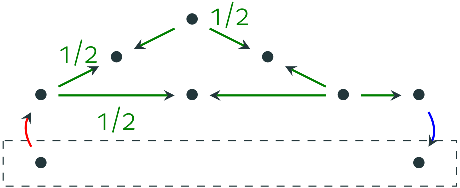
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?

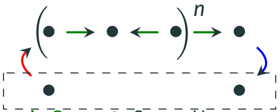


is coded as



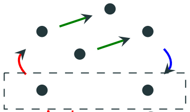
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

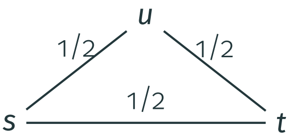
but



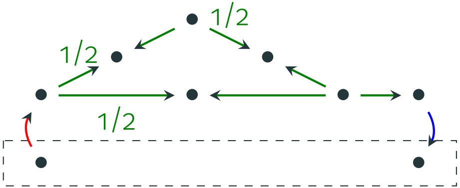
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $\mathbf{1/2}$
- Output: what is the **probability** that the source and target are **connected**?



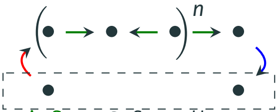
is coded as



Idea: There is a **path connecting s and t** in a possible world of the graph at the left iff the query Q is **satisfied** in the corresponding possible world of the TID at the right

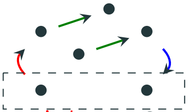
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

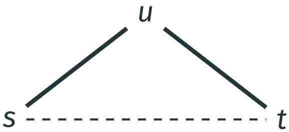
but



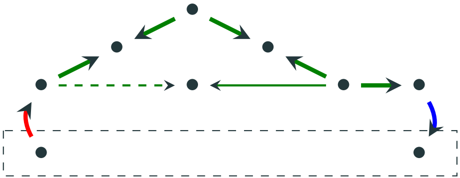
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



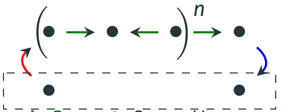
is coded as



Idea: There is a **path connecting s and t** in a possible world of the graph at the left iff the query Q is **satisfied** in the corresponding possible world of the TID at the right

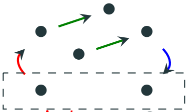
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

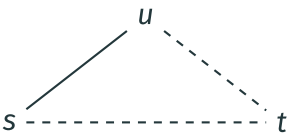
but



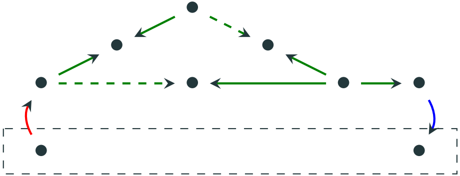
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as

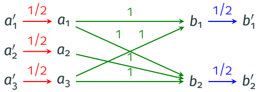


Idea: There is a **path connecting s and t** in a possible world of the graph at the left iff the query Q is **satisfied** in the corresponding possible world of the TID at the right

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \xrightarrow{} y \xrightarrow{} z \xrightarrow{} w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:

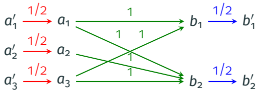


Task: count the number X of **red-blue edge subsets** that **violate** Q

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



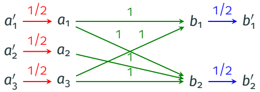
Task: count the number X of **red-blue edge subsets** that **violate** Q

- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



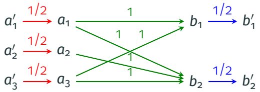
Task: count the number X of **red-blue edge subsets** that **violate** Q

- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$
- Create unweighted copies of G modified with some **parameterized gadgets**
  - $\rightarrow$ Call the **oracle** for $\mathrm{SC}(\mathrm{Q})$ on each to get answers $N_1, \ldots, N_k$

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



Task: count the number X of **red-blue edge subsets** that **violate** Q

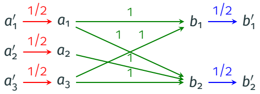
- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$
- Create unweighted copies of G modified with some **parameterized gadgets**
  - $\rightarrow$ Call the **oracle** for $\mathrm{SC(Q)}$ on each to get answers $N_1, \ldots, N_k$
- Show that each N_i is a **linear function** of $X_1, \ldots, X_k$, so:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



Task: count the number X of **red-blue edge subsets** that **violate** Q

- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$
- Create unweighted copies of G modified with some **parameterized gadgets**
  - $\rightarrow$ Call the **oracle** for $\mathrm{SC}(\mathrm{Q})$ on each to get answers $N_1, \ldots, N_k$
- Show that each N_i is a **linear function** of $X_1, \ldots, X_k$, so:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- Show **invertibility** of this matrix to recover the X_i from the N_i

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

→ If the matrix is invertible, then we have succeeded

Using the equation system

We have obtained the system:

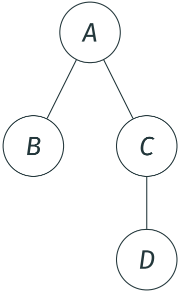
$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

→ If the matrix is invertible, then we have succeeded

We can choose gadgets and parameters to get a Vandermonde matrix, and show invertibility via several arithmetical tricks

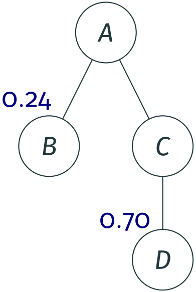
The semistructured model and XML




```
<a>
  <b>...</b>
  <c>
    <d>...</d>
  </c>
</a>
```

- Tree-like structuring of data
- No (or less) schema constraints
- Allow mixing tags (structured data) and text (unstructured content)
- Particularly adapted to tagged or heterogeneous content

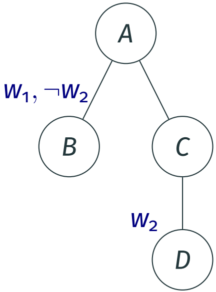
Simple probabilistic annotations



- Probabilities associated to tree nodes
- Express parent/child dependencies
- Impossible to express more complex dependencies
- ⇒ some sets of possible worlds are not expressible this way!

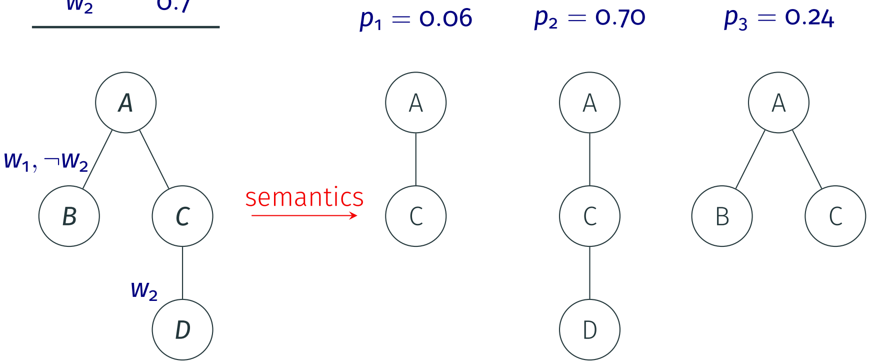
Annotations with event variables

Event	Prob.
w_1	0.8
w_2	0.7



Annotations with event variables

Event	Prob.
w_1	0.8
w_2	0.7



- Expresses arbitrarily complex dependencies

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]
- **#P-hard** in the general model

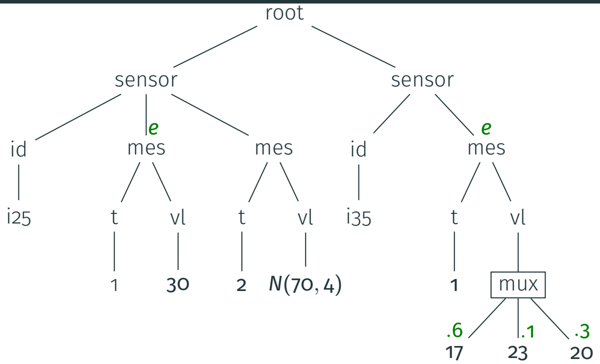
Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]
- **#P-hard** in the general model
- This generalizes to PQE for **MSO** on **relational databases (TID)** when assuming that the **treewidth** is bounded [Amarilli et al., 2015]

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]
- **#P-hard** in the general model
- This generalizes to PQE for **MSO** on **relational databases (TID)** when assuming that the **treewidth** is bounded [Amarilli et al., 2015]
- Bounding the treewidth is **necessary** for tractability in a certain sense [Amarilli et al., 2016]

A general probabilistic XML model [Abiteboul et al., 2009]



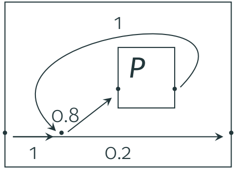
- *e*: event "it did not rain" at time 1
- mux: mutually exclusive options
- $N(70, 4)$: normal distribution

- Compact representation of a **set of possible worlds**
- Two kinds of dependencies: global (*e*) and local (mux)
- Generalizes **all previously proposed models** of the literature

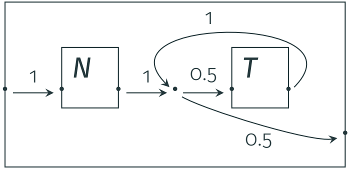
Recursive Markov chains [Benedikt et al., 2010]

```
<!ELEMENT directory (person*)>
<!ELEMENT person (name,phone*)>
```

D: `directory`



P: `person`



- Probabilistic model that **extends** PXML with local dependencies
- Generate documents of **unbounded** width or depth

C-tables [Imielinski and Lipski, 1984]

Patient	Examin. 1	Examin. 2	Diagnosis	Condition
A	23	12	α	
B	10	23	$\bot_1$	
C	2	4	γ	
D	$\bot_2$	15	$\bot_1$	
E	$\bot_3$	17	β	$18 < \bot_3 < \bot_2$

- NULLs are labeled, and can be **reused** inside and across tuples
- **Arbitrary correlations** across tuples
- **Closed** under the relational algebra
- Every set of possible worlds can be represented as a database with c-tables