Tractable Query Answering
Under Probabilistic Constraints

Antoine Amarilli\textsuperscript{1}, Pierre Bourhis\textsuperscript{2}, Pierre Senellart\textsuperscript{1,3}

\textsuperscript{1}Télécom ParisTech

\textsuperscript{2}CNRS-LIFL

\textsuperscript{3}National University of Singapore

September 4th, 2014
Tractable Query Evaluation On Probabilistic Instances

Antoine Amarilli\textsuperscript{1}, Pierre Bourhis\textsuperscript{2}, Pierre Senellart\textsuperscript{1,3}

\textsuperscript{1}Télécom ParisTech
\textsuperscript{2}CNRS-LIFL
\textsuperscript{3}National University of Singapore

September 4th, 2014
Instances and queries

- Given a **relational instance** (\(=\) set of facts, hypergraph)
  \[ I = \{ R(a, b), R(b, c), S(c) \} \]
- Given a **conjunctive query (CQ)** (existentially quantified)
  \[ q : \exists xy \ R(x, y) \land S(y) \]
Instances and queries

- Given a relational instance (= set of facts, hypergraph)
  \[ I = \{ R(a, b), R(b, c), S(c) \} \]
- Given a conjunctive query (CQ) (existentially quantified)
  \[ q : \exists xy \ R(x, y) \land S(y) \]
- Query evaluation (model checking) of \( q \) on \( I \)
Instances and queries

- Given a relational instance ($=$ set of facts, hypergraph)
  \[ I = \{ R(a, b), R(b, c), S(c) \} \]

- Given a conjunctive query (CQ) (existentially quantified)
  \[ q : \exists xy R(x, y) \land S(y) \]

\[ \rightarrow \] Query evaluation (model checking) of \( q \) on \( I \)
 Instances and queries

- Given a **relational instance** $(= \text{set of facts, hypergraph})$
  \[ I = \{ R(a, b), R(b, c), S(c) \} \]

- Given a **conjunctive query (CQ)** (existentially quantified)
  \[ q : \exists xy \ R(x, y) \land S(y) \]

  $\rightarrow$ **Query evaluation** (model checking) of $q$ on $I$
Instances and queries

- Given a **relational instance** (= set of facts, hypergraph)
  \[ I = \{ R(a, b), R(b, c), S(c) \} \]

- Given a **conjunctive query** (CQ) (existentially quantified)
  \[ q : \exists xy \ R(x, y) \land S(y) \]

  → **Query evaluation** (model checking) of \( q \) on \( I \)
  
  → **Data complexity**: \( q \) is fixed
Uncertain and probabilistic instances

- Set of uncertain events
  - $e_{\text{flight}}$: CDG $\rightarrow$ VIE flight AF1756 takes place
  - $e_{\text{bus}}$: Vienna $\rightarrow$ Bratislava buses are running
Uncertain and probabilistic instances

- Set of uncertain events
  
  $e_{\text{flight}}$  CDG $\rightarrow$ VIE flight AF1756 takes place
  $e_{\text{bus}}$  Vienna $\rightarrow$ Bratislava buses are running

- Annotate instance facts with formulae on the events
  
  $\text{IsIn}(\text{AA}, \text{Paris})$  $\neg e_{\text{flight}}$
  $\text{IsIn}(\text{AA}, \text{Vienna})$  $e_{\text{flight}} \land \neg e_{\text{bus}}$
  $\text{IsIn}(\text{AA}, \text{Bratislava})$  $e_{\text{flight}} \land e_{\text{bus}}$
Uncertain and probabilistic instances

- Set of uncertain events
  - Flight \( e_{\text{flight}} \): CDG → VIE flight AF1756 takes place
  - Bus \( e_{\text{bus}} \): Vienna → Bratislava buses are running

- Annotate instance facts with formulae on the events

\[
\begin{align*}
  &\text{IsIn}(\text{AA, Paris}) & \neg e_{\text{flight}} \\
  &\text{IsIn}(\text{AA, Vienna}) & e_{\text{flight}} \land \neg e_{\text{bus}} \\
  &\text{IsIn}(\text{AA, Bratislava}) & e_{\text{flight}} \land e_{\text{bus}}
\end{align*}
\]

→ Semantics: a set of instances (possible worlds).
Uncertain and probabilistic instances

- **Set of uncertain events**
  
  - \( e_{\text{flight}} \): CDG → VIE flight AF1756 takes place
  - \( e_{\text{bus}} \): Vienna → Bratislava buses are running

- **Annotate instance facts with formulae** on the events

<table>
<thead>
<tr>
<th>Instance Fact</th>
<th>Formulae</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsIn(AA, Paris)</td>
<td>( \neg e_{\text{flight}} )</td>
</tr>
<tr>
<td>IsIn(AA, Vienna)</td>
<td>( e_{\text{flight}} \land \neg e_{\text{bus}} )</td>
</tr>
<tr>
<td>IsIn(AA, Bratislava)</td>
<td>( e_{\text{flight}} \land e_{\text{bus}} )</td>
</tr>
</tbody>
</table>

- **Semantics**: a set of instances (possible worlds).

- **Add a probability distribution** on each event
  
  - each event has **probability** \( 0 < p < 1 \) of being true
  - all events are assumed to be **independent**
Uncertain and probabilistic instances

- **Set of uncertain events**
  \[ e_{\text{flight}} \text{ CDG} \rightarrow \text{VIE} \text{ flight AF1756 takes place} \]
  \[ e_{\text{bus}} \text{ Vienna} \rightarrow \text{Bratislava buses are running} \]

- **Annotate instance facts with formulae** on the events

\[
\begin{align*}
\text{IsIn}(\text{AA}, \text{Paris}) & \quad \neg e_{\text{flight}} \\
\text{IsIn}(\text{AA}, \text{Vienna}) & \quad e_{\text{flight}} \land \neg e_{\text{bus}} \\
\text{IsIn}(\text{AA}, \text{Bratislava}) & \quad e_{\text{flight}} \land e_{\text{bus}}
\end{align*}
\]

→ **Semantics**: a set of instances (possible worlds).

- **Add a probability distribution** on each event
  - each event has probability \( 0 < p < 1 \) of being true
  - all events are assumed to be independent

→ **Semantics**: a probability distribution on instances.
Uncertain and probabilistic instances

- **Set of uncertain events**
  
  \[ e_{\text{flight}} \] CDG → VIE flight AF1756 takes place
  
  \[ e_{\text{bus}} \] Vienna → Bratislava buses are running

- **Annotate instance facts with formulae** on the events

  \[
  \begin{align*}
  &\text{IsIn}(AA, \text{Paris}) & &\neg e_{\text{flight}} \\
  &\text{IsIn}(AA, \text{Vienna}) & &e_{\text{flight}} \land \neg e_{\text{bus}} \\
  &\text{IsIn}(AA, \text{Bratislava}) & &e_{\text{flight}} \land e_{\text{bus}}
  \end{align*}
  \]

  \[ \rightarrow \text{ Semantics: a set of instances (possible worlds).} \]

  - **Add a probability distribution** on each event
    
    - each event has probability \( 0 < p < 1 \) of being true
    
    - all events are assumed to be independent

  \[ \rightarrow \text{ Semantics: a probability distribution on instances.} \]

  \[ \rightarrow \text{ Query evaluation: determine the probability of } q \text{ on } \hat{I}. \]
Hardness and tractability

- **With arbitrary annotations**
  - Query evaluation is $\#P$-hard even with a single fact
    (Immediate reduction from $\#SAT$)

- **With simple annotations** (one unique event per tuple)
  - Query evaluation is $\#P$-hard on arbitrary instances
    (Use the instance to do the reduction)
Hardness and tractability

- With arbitrary annotations
  - Query evaluation is \#P-hard even with a single fact
    (Immediate reduction from \#SAT)

- With simple annotations (one unique event per tuple)
  - Query evaluation is \#P-hard on arbitrary instances
    (Use the instance to do the reduction)

- Existing work:
  - Fix a simple annotation scheme
  - Show dichotomy between \#P-hard and PTIME queries
Hardness and tractability

- With arbitrary annotations
  - Query evaluation is $\#P$-hard even with a single fact
    (Immediate reduction from $\#\text{SAT}$)
- With simple annotations (one unique event per tuple)
  - Query evaluation is $\#P$-hard on arbitrary instances
    (Use the instance to do the reduction)

Existing work:
- Fix a simple annotation scheme
- Show dichotomy between $\#P$-hard and PTIME queries

Our approach:
- Find a restriction on the instance and annotations
- Show that many queries are tractable in this case
Bounded treewidth

An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...
Bounded treewidth

An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

instance I
\[ R(a, b) \ R(b, c) \ S(c) \]
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

Instance $I \xrightarrow{R(a, b) \ R(b, c) \ S(c)}$ tree encoding $T_i$

tree decomposition

$O(|I|)$ for fixed width
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[
\begin{align*}
\text{instance } I & \rightarrow \text{ tree encoding } T_I \\
R(a, b) R(b, c) S(c) & \\
\text{tree decomposition} & \\
O(|I|) \text{ for fixed width} & \\
\end{align*}
\]

\[\exists xy \ R(x, y) \land S(y)\]

query \( q \)
An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

\[
\text{instance } I \quad \rightarrow \quad \text{tree encoding } T_I
\]

\[
R(a, b) \quad R(b, c) \quad S(c)
\]

\[
\text{tree decomposition}
\]

\[
O(|I|) \quad \text{for fixed width}
\]

\[
\text{rewriting}
\]

\[
O(1) \quad \text{data complexity}
\]

\[
\exists xy \ R(x, y) \land S(y)
\]

\[
\text{query } q \quad \rightarrow \quad \text{tree automaton } A_q
\]

Linear time data complexity
Bounded treewidth

An idea from instances without probabilities...

- If an instance has low treewidth then it is almost a tree
- Assume that the instance treewidth is constant...

\[
\text{instance } I \xrightarrow{\text{tree encoding } T_I} \text{tree decomposition} \xrightarrow{O(|I|) \text{ for fixed width}} \text{evaluation linear time} \xrightarrow{\text{query answer}}
\]

\[
\exists x \exists y \ R(x, y) \land S(y) \xrightarrow{\text{rewriting} O(1) \text{ data complexity}} \text{deterministic tree automaton } A_q
\]

\[
R(a, b) \ R(b, c) \ S(c)
\]
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[
\begin{align*}
\text{instance } I & \quad \text{tree encoding } T_I \\
R(a, b) \ R(b, c) \ S(c) & \\
\text{tree decomposition} & \\
O(|I|) \text{ for fixed width} & \\
\text{rewriting} & \\
O(1) \text{ data complexity} & \\
\exists xy \ R(x, y) \land S(y) & \\
\text{query } q & \\
\text{tree automaton } A_q & \\
\end{align*}
\]
Bounded treewidth

An idea from instances without probabilities...

- If an instance has **low treewidth** then it is almost a tree
- Assume that the instance treewidth is **constant**...

\[
\begin{align*}
\text{instance } I & \quad \rightarrow \quad \text{tree encoding } T_I \\
R(a, b) \ R(b, c) \ S(c) & \quad \text{tree decomposition}
\end{align*}
\]

- **Evaluation**
  - Linear time

- **Query Answer**
  - Linear time

- **Rewriting**
  - \(O(1)\) data complexity

- **Deterministic Tree Automaton**
  - \(A_q\)

\[\exists xy \ R(x, y) \land S(y)\]

→ Linear time data complexity
Tractable inference

An idea from probabilities without instances...

- Represent a propositional formula $F$ as a Boolean circuit
- Assume the circuit has constant treewidth

$\rightarrow$ Probability of $F$ can be computed in linear time
(using junction tree algorithm for Bayesian networks)
(assuming constant-time arithmetic operations)
cc-tables

- Boolean circuit for the annotations
Boolean circuit for the annotations

\( R(a, b) \)
\( R(b, c) \)
\( R(c, d) \)
**cc-tables**

- **Boolean circuit** for the annotations

\[
\begin{array}{ccc}
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\
\land & & \\
R(a, b) & & \\
\land & & \\
R(b, c) & & \\
\land & & \\
R(c, d) & & \\
\end{array}
\]

- **Circuit** must have low treewidth
- **Instance** must have low treewidth

→ Need **simultaneous** decomposition
cc-tables

- **Boolean circuit** for the annotations

\[ \begin{array}{c}
1/2 & 1/2 & 1/2 \\
\wedge & & \\
\wedge & & \\
R(a, b) & R(b, c) & R(c, d)
\end{array} \]

- **Circuit** must have low treewidth
- **Instance** must have low treewidth
  - Need *simultaneous* decomposition

**Circuit** must have low treewidth

**Instance** must have low treewidth

→ Need *simultaneous* decomposition
Main result

instance $I$

$\frac{1}{2} \frac{1}{2} \frac{1}{2} \land \land R(a, b) \land R(b, c) \land R(c, d)$
Main result

instance $I$

$\begin{array}{c}
1/2\ & 1/2\ & 1/2\\
\land & R(a, b) & \\
\land & R(b, c) & \\
\land & R(c, d) & \\
\end{array}$

tree encoding $T_i$

$\begin{array}{c}
1/2\\
\land & R(a, b) & \\
1/2\\
\land & R(b, c) & \\
1/2\\
\land & R(c, d) & \\
\end{array}$

tree decomposition $O(|I|)$ for fixed width
Main result

$$\exists x y \ R(x, y) \land S(y)$$

query $$q$$
Main result

instance $I$

$\exists xy \ R(x, y) \land S(y)$

query $q$

deterministic tree automaton $A_q$

rewriting

O(1) data complexity

O($|I|$) for fixed width

tree encoding $T_I$

Background
Ideas
Results
Consequences
Main result

Instance $I$

Tree encoding $T_I$

Tree decomposition $O(|I|)$ for fixed width

Tree encoding $T_I$

Bounded treewidth circuit $C$

Instrumentation linear time

Rewriting $O(1)$ data complexity

Query $q$

Deterministic tree automaton $A_q$

Existential query $\exists xy \ R(x, y) \land S(y)$
Main result

instance $I$

$\exists xy \ R(x, y) \land S(y)$

query $q$

$\forall I \ 1/2 1/2 1/2 \land R(a, b) \land R(b, c) \land R(c, d)$

tree decomposition $O(|I|)$ for fixed width

rewriting $O(1)$ data complexity

$\exists xy \ R(x, y) \land S(y)$

query $q$

$\forall I \ 1/2 1/2 1/2 \land R(a, b) \land R(b, c) \land R(c, d)$

tree encoding $T_I$

bounded treewidth circuit $C$

instrumentation linear time

probabilistic inference $O(|C|)$ for fixed width

probability $p$
Main result

Instance $I$

Tree encoding $T_I$

Tree decomposition $O(|I|)$ for fixed width

Rewriting $O(1)$ data complexity

Query $q$

Deterministic tree automaton $A_q$

Bounded treewidth circuit $C$

Instrumentation linear time

Probabilistic inference $O(|C|)$ for fixed width

Probability $p = 0.42$
Consequences

- For queries representable as deterministic automata ...
  - CQs
  - Monadic second-order
  - Guarded second-order
Consequences

- For queries *representable as deterministic automata* ...
  - CQs
  - Monadic second-order
  - Guarded second-order

- ... on various *probabilistic models* ...
  - Tuple-independent tables
  - Block-independent disjoint tables
  - pc-tables (presented before)
  - Probabilistic XML
Consequences

- For queries representable as deterministic automata ...
  - CQs
  - Monadic second-order
  - Guarded second-order
- ... on various probabilistic models ...
  - Tuple-independent tables
  - Block-independent disjoint tables
  - pc-tables (presented before)
  - Probabilistic XML
- ... assuming bounded treewidth (for reasonable definitions) ...
Consequences

- For queries **representable as deterministic automata** ...
  - → CQs
  - → Monadic second-order
  - → Guarded second-order

- ... on various **probabilistic models** ...
  - → Tuple-independent tables
  - → Block-independent disjoint tables
  - → pc-tables (presented before)
  - → Probabilistic XML

- ... assuming **bounded treewidth** (for reasonable definitions) ...
  - → ... probability of fixed $q$ can be computed in $O(\hat{|I|})$!
We can combine the following techniques:

- Computing tree decompositions
- Encoding problems to automata on tree encodings of instances
- Evaluating probabilities on bounded-treewidth circuits
We can combine the following techniques:
- Computing tree decompositions
- Encoding problems to automata on tree encodings of instances
- Evaluating probabilities on bounded-treewidth circuits

Applications:
- Tractable probabilistic query evaluation in practice?
- Reasoning under uncertain rules
  (hence the bait-and-switch on the title...)
Conclusion

- We can **combine** the following techniques:
  - Computing **tree decompositions**
  - Encoding problems to **automata** on tree encodings of instances
  - Evaluating **probabilities** on bounded-treewidth circuits

- **Applications:**
  - Tractable probabilistic query evaluation in **practice**?
  - Reasoning under **uncertain rules**
    (hence the bait-and-switch on the title...)

- **Questions:**
  - Other **semirings** than Boolean AND/OR?
  - Other tasks than **probabilistic inference**?
Conclusion

- We can combine the following techniques:
  - Computing tree decompositions
  - Encoding problems to automata on tree encodings of instances
  - Evaluating probabilities on bounded-treewidth circuits

- Applications:
  - Tractable probabilistic query evaluation in practice?
  - Reasoning under uncertain rules
    (hence the bait-and-switch on the title...)

- Questions:
  - Other semirings than Boolean AND/OR?
  - Other tasks than probabilistic inference?

What are bounded-treewidth circuits good for?

http://cstheory.stackexchange.com/q/25624

modified aug 28 at 13:05 a3nm 1,432
Conclusion

- We can combine the following techniques:
  - Computing tree decompositions
  - Encoding problems to automata on tree encodings of instances
  - Evaluating probabilities on bounded-treewidth circuits

- Applications:
  - Tractable probabilistic query evaluation in practice?
  - Reasoning under uncertain rules
    (hence the bait-and-switch on the title...)

- Questions:
  - Other semirings than Boolean AND/OR?
  - Other tasks than probabilistic inference?

Thanks for your attention!