Overview

1. Introduction | Relational data model
2. First-order queries
3. Complexity of query answering
4. Complexity of FO query answering
5. Conjunctive queries
6. Tree-like conjunctive queries
7. Query optimisation
8. Conjunctive Query Optimisation / First-Order Expressiveness
9. First-Order Expressiveness / Introduction to Datalog
10. Expressive Power and Complexity of Datalog
11. Optimisation and Evaluation of Datalog
12. Evaluation of Datalog (2)
13. Graph Databases and Path Queries
14. Outlook: database theory in practice

See course homepage [⇒ link] for more information and materials

Markus Krötzsch, 12 May 2016

Review

Conjunctive queries (CQs) are simpler than FO-queries:
- \( \text{NP} \) combined and query complexity (instead of \( \text{PSpace} \))
- data complexity remains in \( \text{AC}^0 \)

CQs become even simpler if they are tree-shaped:
- GYO algorithm defines acyclic hypergraphs
- acyclic hypergraphs have join trees
- join trees can be evaluated in \( \text{P} \) with Yannakakis' Algorithm

This time:
- Find more general conditions that make CQs tractable
  \( \leadsto \) “tree-like” queries that are not really trees
- Play some games

Is Yannakakis’ Algorithm Optimal?

We saw that tree queries can be evaluated in polynomial time, but we know that there are much simpler complexity classes:

\[
\text{NC}^0 \subset \text{AC}^0 \subset \text{NC}^1 \subset \text{L} \subset \text{NL} \subset \text{AC}^1 \subset \ldots \subset \text{NC} \subset \text{P}
\]

Indeed, tighter bounds have been shown:

**Theorem (Gottlob, Leone, Scarcello: J. ACM 2001)**

Answering tree BCQs is complete for \( \text{LOGCFL} \).

**LOGCFL**: the class of problems \( \text{LOGSPACE} \)-reducible to the word problem of a context-free language:

\[
\text{NC}^0 \subset \text{AC}^0 \subset \text{NC}^1 \subset \text{L} \subset \text{NL} \subset \text{LOGCFL} \subset \text{AC}^1 \subset \ldots \subset \text{NC} \subset \text{P}
\]

\( \leadsto \) highly parallelisable
Generalising Tree Queries

In practice, many queries are tree queries, but even more queries are “almost” tree queries, but not quite . . .

How can we formalise this idea?

Several attempts to define “tree-like” queries:
- Treewidth: a way to measure tree-likeness of graphs
- Query width: towards tree-like query graphs
- Hypertree width: adoption of treewidth to hypergraphs

How to recognise trees . . .
. . . from quite a long way away:

Tree Decompositions

Idea: if we can group the edges of a graph into bigger pieces, these pieces might form a tree structure

**Definition**

Consider a graph $G = (V, E)$. A tree decomposition of $G$ is a tree structure $T$ where each node of $T$ is a subset of $V$, such that:

- The union of all nodes of $T$ is $V$.
- For each edge $(v_1 \rightarrow v_2) \in E$, there is a node $N$ in $T$ such that $v_1, v_2 \in N$.
- For every vertex $v \in V$, the set of nodes of $T$ that contain $v$ form a subtree of $T$; equivalently: if two nodes contain $v$, then all nodes on the path between them also contain $v$ (connectedness condition).

Nodes of a tree decomposition are often called bags (not related to the common use of “bag” as a synonym for “multiset”)

Tree Decompositions: Example
Treewidth

The treewidth of a graph defines how “tree-like” it is:

**Definition**

The width of a tree decomposition is the size of its largest bag minus one. The treewidth of a graph $G$, denoted $\text{tw}(G)$, is the smallest width of any of its tree decompositions.

Simple observations:

- If $G$ is a tree, then we can decompose it into bags that contain only one edge $\leadsto$ trees have treewidth 1
- Every graph has at least one tree decomposition where all vertices are in one bag $\leadsto$ max. treewidth = number of vertices

**More Examples**

What is the treewidth of the following graphs?

Treewidth: Example

$\leadsto$ tree decomposition of width 2 = treewidth of the example graph

Treewidth and Conjunctive Queries

Treewidth is based on graphs, not hypergraphs $\leadsto$ treewidth of CQ = treewidth of primal graph of query hypergraph

Query graph and corresponding primal graph:

$\leadsto$ Treewidth 3

Observation: acyclic hypergraphs can have unbounded treewidth!
Exploiting Treewidth in CQ Answering

Queries of low treewidth can be answered efficiently:

**Theorem** (Dechter/Chekuri+Rajamaran ‘97/Kolaitis+Vardi ‘98/Gottlob & al. ‘98)

Answering BCQs of treewidth $k$ is possible in time $O(n^k \log n)$, and thus in polynomial time if $k$ is fixed.

The problem is also complete for LOGCFL.

Checking for low treewidths can also be done efficiently:

**Theorem** (Bodlaender ‘96)

Given a graph $G$ and a fixed number $k$, one can check in linear time if $\text{tw}(G) \leq k$, and the corresponding tree decomposition can also be found in linear time.

Warning: neither CQ answering nor tree decomposition might be practically feasible if $k$ is big

---

Seymour and Thomas [1993] gave an alternative characterisation of treewidth:

**The Cops-and-Robber Game**

- The game is played on a graph $G$
- There are $k$ cops and one robber, each located at one vertex
- In each turn:
  - the cops can fly to an arbitrary vertex in the graph
  - the robber can run along the edges of the graph, as far as she likes, as long as she does not pass through any vertex that was occupied by a cop before or after the turn (the robber can run to a place where a cop was before the turn, but not pass through such a place)
- The goal of the cops is to catch the robber; the goal of the robber is never to be caught

**Cops and Robbers: Example**

<table>
<thead>
<tr>
<th>Cops and Robbers: Example</th>
<th>Cops &amp; Robbers and Treewidth</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Theorem (Seymour and Thomas)**

A graph $G$ is of treewidth $\leq k - 1$ if and only if $k$ cops have a winning strategy in the cops & robber game on $G$.

Intuition: the cops together can block even the widest branch and still move in on the robber
Treewidth via Logic

Kolaitis and Vardi [1998] gave a logical characterisation of treewidth

Bounded treewidth CQs correspond to certain FO-queries:
- We allow FO-queries with $\exists$ and $\land$ as only operators
- But operators can be nested in arbitrary ways (unlike in CQs)
- Theorem: A query can be expressed with a CQ of treewidth $k$ if and only if it can be expressed in this logic using a query with at most $k + 1$ distinct variables

Intuition: variables can be reused by binding them in more than one $\exists$

$\leadsto$ Apply a kind of “inverted prenex-normal-form transformation”

$\leadsto$ Variables that occur in the same atom or in a “tightly connected” atom must use different names

$\leadsto$ minimum number of variables $\iff$ treewidth ($+$1)

Advantages:
- Bounded treewidth is easy to check
- Bounded treewidth CQs are easy to answer

Disadvantages:
- Even families of acyclic graphs may have unbounded treewidth
- Loss of information when using primal graph (cliques might be single hyperedges – linear! – or complex query patterns – exponential!)

$\leadsto$ Are there better ways to capture “tree-like” queries?

Query Width

Idea of Chekuri and Rajamaran [1997]:
- Create tree structure similar to tree decomposition
- But consider bags of query atoms instead of bags of variables
- Two connectedness conditions:
  1. Bags that refer to a certain variable must be connected
  2. Bags that refer to a certain query atom must be connected

Query width: least number of atoms needed in bags of a query decomposition

Theorem (Gottlob et al. 1999)
Deciding if a query has query width at most $k$ is $\text{NP}$-complete.

In particular, it is also hard to find a query decomposition

$\leadsto$ Query answering complexity drops from $\text{NP}$ to $\text{P}$ . . .

. . . but we need to solve another $\text{NP}$-hard problem first!
Generalised Hypertree Width

Gottlob, Leone, and Scarcello had another idea on defining tree-like hypergraphs:

Intuition:
- Combine key ideas of tree decomposition and query decomposition
- Start by looking at a tree decomposition
- But define the width based on query atoms:
  How many atoms do we need to cover all variables in a bag?

\[\leadsto\] Generalised hypertree width

\[\leadsto\] A technical condition is needed to get a simpler-to-check notion

Markus Krötzsch, 12 May 2016

Database Theory

slide 21 of 28

Hypertree Width

Definition

Consider a hypergraph \(G = \langle V, E \rangle\). A hypertree decomposition of \(G\) is a tree structure \(T\) where each node \(n\) of \(T\) associated with a bag of variables \(B_n \subseteq V\) and with a set of edges \(G_n \subseteq E\), such that:

- \(T\) with \(B_n\) yields a tree decomposition of the primal graph of \(G\).
- For each node \(n\) of \(T\):
  1. the vertices used in the edges \(G_n\) are a superset of \(B_n\),
  2. if a vertex \(v\) occurs in an edge of \(G_n\) and this vertex also occurs in \(B_m\) for some node \(m\) below \(n\) in \(T\), then \(v \in B_n\).

The width to \(T\) is the largest number of edges in a set \(G_n\).

The hypertree width of \(G\), \(hw(G)\), is the least width of its hypertree decompositions.

\((2)\) is the "special condition": without it we get the generalised hypertree width

Markus Krötzsch, 12 May 2016

Database Theory

slide 22 of 28

Hypertree Width: Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>F</td>
<td>4</td>
<td>B</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>G</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Special condition violated \(\leadsto\) no hypertree decomposition

\[\leadsto\] But generalised hypertree decomposition of width 2

Markus Krötzsch, 12 May 2016

Database Theory

slide 23 of 28

Hypertree Width: Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>F</td>
<td>4</td>
<td>B</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>G</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Special condition satisfied \(\leadsto\) hypertree decomposition of width 3

Markus Krötzsch, 12 May 2016

Database Theory

slide 24 of 28
Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width \( \leq \) hypertree width \( \leq \) query width
  (both inequalities might be \( < \) in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width \( < k \)" is NP-complete
- Deciding "generalised hypertree width \( < 4 \)" is NP-complete
- Deciding "hypertree width \( < k \)" is polynomial (LOGCFL)
- Hypertree decompositions can be computed in polynomial time if \( k \) is fixed

**Theorem**
For a BCQ of (generalised) hypertree width \( k \), query answering can be decided in polynomial time, and is complete for LOGCFL.

... but the degree of the polynomial time bound is greater than \( k \)

Hypertree Width via Games

There is also a game characterisation of (generalised) hypertree width.

**The Marshals-and-Robber Game**
- The game is played on a hypergraph
- There are \( k \) marshals, each controlling one hyperedge, and one robber located at a vertex
- Otherwise similar to cops-and-robber game
- Special condition: Marshals must shrink the space that is left for the robber in every turn!

Hypertree width \( \leq k \) if and only if \( k \) marshals have a winning strategy
\( \leadsto \) hypergraph is acyclic iff 1 marshal has a winning strategy

Hypertree Width via Logic

There is also a logical characterisation of hypertree width.

**Loosely \( k \)-Guarded Logic**
- Fragment of FO with \( \exists \) and \( \land \)
- Special form for all \( \exists \) subexpressions:
  \[
  \exists x_1, \ldots, x_n (G_1 \land \ldots \land G_k \land \varphi)
  \]
  where \( G_i \) are atoms ("guards") and every variable that is free in \( \varphi \) occurs in one such atom \( G_i \).

A query has hypertree width \( \leq k \) if and only if it can be expressed as a loosely \( k \)-guarded formula
\( \leadsto \) tree queries correspond to loosely 1-guarded formulae
("loosely 1-guarded" logic is better known as guarded logic and widely studied)

Summary and Outlook

Besides tree queries, there are other important classes of CQs that can be answered in polynomial time:
- Bounded treewidth queries
- Bounded hypertree width queries

General idea: decompose the query in a tree structure

Other possible characterisations via games and logic

Next topics:
- What else is there besides query answering? \( \leadsto \) optimisation
- Measure expressivity rather than just complexity