Ron Fagin and Acyclic Hypergraphs

Why Hypergraphs?
Interesting Properties
Fagin’s Hierarchy

Jeffrey D. Ullman
Stanford University
Hypergraphs

- Nodes + *(hyper)*edges*  that are sets of any number of nodes.
Hypergraphs as Schemas

- Nodes = attributes.
- Hyperedges = relation schemas.
- Hypergraph = database schema.

= \{ABC, BCD, BDE, DEF\}
Hypergraphs as Natural Joins

- Nodes = attributes.
- Edges = schemas of relations being joined.
  - Any equijoin can be so represented if we rename equated attributes from different relations.

\[
\begin{align*}
&= ABC \otimes BCD \otimes BDE \otimes DEF \\
\end{align*}
\]
Beeri, Fagin, Maier, Mendelzon, U, Yannakakis (STOC, 1981) looked at hypergraphs primarily as database schemas.

At that time, the “universal-relation wars” were raging.

- Could you ask queries about attributes only and allow the system to figure out the proper join to connect these attributes?

- Identified a class of schemas (“acyclic”) with certain properties that made sense as a universal relation.
It turns out there is a simple way to tell whether a hypergraph is acyclic, so we won’t bother with the original definition.

Due to Graham and Yu-Oszoyoglu independently.

“Reduce” the hypergraph using the following two rules:

- Eliminate a node in only one hyperedge.
- Eliminate a hyperedge contained in another.

If you get down to one empty edge, then the hypergraph is acyclic.
Example: GYO Reduction

A B C D E F

B C D E F

C B D E F

C B D E

B D E

D
Previously, Phil Bernstein and his students Chiu, Goodman, and Shmueli had looked at a seemingly unrelated question: when does a join have a \textit{full reducer}? 

- = finite sequence of semijoins that is guaranteed to eliminate from the relations all tuples that dangle in the complete join.
A related formulation: when does *local consistency* imply *global consistency*? 

- = the join of any two relations has no dangling tuples
- = there are no dangling tuples in any relation when the join of all the relations is taken.

It turns out “exists a full reducer” = “local consistency implies global consistency” = “acyclic.”
These three relations are locally consistent. But the join of all three relations is empty. Hence not globally consistent.
Example: Semijoin Reduction

Now, semijoin reduction will make each relation empty. But the number of steps needed depends on the number of tuples.
1. $AB \bowtie CA$ eliminates only $(0, 1)$.
2. Then $BC \bowtie AB$ eliminates only $(1, 2)$.
3. And so on...

Notice the change
Monotone Joins

- A join of two relations is *monotone* if it has no dangling tuples.

- **Important consequence**: the output of a monotone join is at least as large each of its arguments.
  - If implemented properly, the time taken by the join is proportional to input size + output size.

- **Note**: “local consistency” = “joins of two database relations are monotone,” but “monotone” applies to intermediate joins also.
This line of research had a very different view of the condition under which full reducers exist (and under which local consistency = global consistency).

If and only if you can build a tree with:

- Nodes = relation schemas.
- For every attribute, the set of nodes containing that attribute is connected.
Example: Tree View of Acyclicity
Example: Tree View of Acyclicity
Example: Tree View of Acyclicity
Example: A Cyclic Join

By symmetry, all trees look like this. Notice A is at disconnected nodes.
Theorem

- From Beeri, Fagin, Maier, and Yannakakis (J. ACM, 1983).

- A hypergraph is acyclic if and only if its hyperedges form a tree whose nodes containing any given attribute are connected.

- Therefore, acyclic hypergraphs, and only acyclic hypergraphs, have:
  1. Full reducers.
  2. Local consistency = global consistency.
  3. Local consistency => monotone join sequences guaranteed to exist.
While the tree-based definition of acyclicity is generally less convenient to use than the GYO definition, it yielded an important generalization.

Tree width = maximum number of elements (= relation schema or attribute) at a tree node, where all attributes are in connected set of nodes.

Finite tree width yields several useful properties shared with acyclic hypergraphs.
Example: Tree Width

Now, the A’s are at a connected set of nodes, and the tree width = 2, since the root has two members.
In his seminal paper “Degrees of Acyclicity for Hypergraphs and Relational Database Schemes” (J. ACM, 1983), Ron defined four different notions of acyclicity.

- Berge acyclicity, and γ-, β-, and α–acyclicity.
- $\alpha$-acyclic = what we have been calling “acyclic.”
In the leading graph-theory text of the time, Berge defined a cycle in a hypergraph to be a sequence of distinct nodes $n_1, n_2, \ldots, n_k$ such that there are distinct hyperedges containing each consecutive pair of nodes in the end-around sense: $\{n_1, n_2\}, \{n_2, n_3\}, \ldots, \{n_k, n_1\}$.

Exactly what you want for (ordinary) graphs.

But weird for hypergraphs.

Example: $\text{A} - \text{B} - \text{C} - \text{D}$ has a cycle $\text{B}, \text{C}$. 
The other three notions of acyclicity each have many equivalent definitions and properties.

One simple hierarchy of distinctions is (assuming the relations are locally consistent):

- $\alpha$-acyclic = the join of all the relations in the hypergraph has a sequence of monotone joins.
- $\beta$-acyclic = the join of any connected subset of the relations has a sequence of monotone joins.
- $\gamma$-acyclic = any join sequence for any connected subset of the relations is monotone.
1. The four notions of acyclicity are distinct and are contained as follows: Berge acyclic $\subset \gamma$-acyclic $\subset \beta$-acyclic $\subset \alpha$-acyclic.
2. Each of the definitions has a polynomial-time test.
3. For each there is an appropriate notion of a “cycle” analogous to that used by Berge.
Example: $\alpha$-acyclic, Not $\beta$-acyclic

$\alpha$-acyclic.
Remove D, E, F.
Resulting hyperedges are contained in ABC.

But ... remove ABC, and the result is an $\alpha$-cyclic hypergraph.
Hence, original is not $\beta$-acyclic.
A former student, Anand Rajaraman, returned for his PhD after founding a startup, Junglee.

The Junglee folks had developed techniques for examining Web pages and figuring out what data was connected to what.

- **Example**: Help-wanted pages. To which job(s) did a location or salary refer?

- **Thesis question**: what HTML structures allowed Junglee methods to work.

- **Answer**: the $\beta$-acyclic hypergraphs.