The Finite Model Theory Toolbox of a Database Theoretician

Leonid Libkin
University of Edinburgh

PODS 2009
Finite Model Theory (FMT)

- The main object of study: logics over finite structures
- Foundational role in the study of the theory of relational databases:
  - relational database = finite relational structure
  - query languages are logic based
- Finite model theory was described as the "backbone of database theory" (Vianu, 1995)
- Connections work both ways: much of the motivation for finite model theory research came from databases.
- Many other applications in CS: verification, AI, constraint satisfaction, algorithms, complexity ...
Finite Model Theory (FMT)

• Mature field; 4 textbooks:
  1. Ebbinghaus-Flum “Finite Model Theory”, 1994
  2. Immerman “Descriptive Complexity”, 1999

• Several surveys, ‘personal perspectives’, etc:
  • Fagin (ICDT 1990, LICS 2000), Kolaitis (PODS 1995, ICDT 1995, LICS 2007), Grohe (LICS 2008), etc

• Surveys they tell you how to prove results, what the main open problems are, etc. – but don’t concentrate on applying FMT techniques.
FMT for a database theoretician: Key problems

1. **Expressiveness** of query languages
   - Query languages have limited expressiveness (e.g., relational calculus = first-order logic)
   - So we need to know what is not expressible in them to add new language constructs
   - Adding language constructs is not free — optimizations!

2. **Complexity** of query languages
   - Can we say something about the complexity of query evaluation by knowing the logical formalism?

3. **Equivalence** of query languages
   - Can we lower the complexity by changing the syntax?
   - Often used in the study of languages for semistructured data.

4. **Satisfiability** (usually, finite satisfiability)
   - Used in static analysis, also when dealing with incomplete information

   - All in the paper; the talk is mostly about Expressiveness and Equivalence.
FMT for a database theoretician cont’d

• Early results: if they involve FMT techniques, there are shown by people actively working in FMT.
• But now the field has built a large arsenal of tools.
• These tools can be used without knowing how they are proved!
• They should be a part of the toolbox of every database theoretician.
• Our goal: present finite-model theory as such a toolbox.
Plan

- **Expressiveness: Tools for first-order logic**
  - Neolithic (stone age) tools – games and tricks.
  - Iron age tools: growing up (locality, 0-1 law).
- **Expressiveness: Tools that work beyond first-order logic**
  - The same arsenal of tools
- **Language equivalence: better query evaluation via the composition method**
  - still growing up: learning how to put stones together
Plan

- **Expressiveness: Tools for first-order logic**
  - Neolithic (stone age) tools – games and tricks.
  - Iron age tools: growing up (locality, 0-1 law).

- **Expressiveness: Tools that work beyond first-order logic**
  - The same arsenal of tools

- **Language equivalence: better query evaluation via the composition method**
  - still growing up: learning how to put stones together
First-Order (FO)

• The most fundamental query language — first-order logic (FO)
• Database people often refer to it as relational calculus.
• The core of SQL (minus aggregation – we’ll address it later).
• A basic question: what are the limitations of FO?
• Intuition: FO cannot express:
  1. nontrivial counting properties, and
  2. queries requiring recursion
• We’ll now see some “canonical” examples of inexpressible queries.
Even cardinality

- **Active domain** = set of all elements stored in a database
- A Boolean query

\[ \text{EVEN}(D) = \text{true} \iff |\text{ActiveDomain}(D)| = 0 \pmod{2} \]
Transitive closure
Transitive closure

\[ \text{trcl}(x, y) \quad :\quad e(x, y) \]
\[ \text{trcl}(x, y) \quad :\quad e(x, z), \text{trcl}(z, y) \]
Same Generation
**Same Generation**

\[
\begin{align*}
sg(x, x) & : \leftarrow \\
sg(x, y) & : \leftarrow e(x', x), e(y', y), sg(x', y')
\end{align*}
\]
Same Generation

\[ sg(x, x) \] :=
\[ sg(x, y) \] := \( e(x', x), e(y', y), sg(x', y') \)
A bit of history

• Classical model theory offers us powerful tools, like compactness.
• But they do not work over finite databases!
  • Exception: compactness argument shows that $\text{EVEN}$ is not FO-definable.
• Fagin 1975: Transitive closure is not FO-expressible over finite graphs. Technique: games.
• Afterwards (1970s, 1980s): more and more advanced game proofs.
• Require nontrivial combinatorial arguments.
• 1990s: proper tools are being developed. They do not require complicated combinatorial proofs.
First-Order Logic (FO)

- Make things simple (for the talk): databases are graphs – they have one binary relation $E(\cdot, \cdot)$.

- **Syntax of FO:**
  - Atomic formulae: $E(x, y)$, $x = y$
  - Boolean combinations: $\varphi \lor \psi$, $\varphi \land \psi$, $\neg \varphi$
  - Quantification: $\exists x \varphi$, $\forall x \varphi$
  - $\varphi(\bar{x})$ means that $\bar{x}$ is the tuple of free variables of $\varphi$

- **Quantifier rank $qr(\varphi)$:**
  - The depth of quantifier nesting in $\varphi$.
  - For example, $qr(\exists x (\forall y E(x, y) \lor \forall z \neg E(x, z)))$ is 2 and not 3.
First-Order Logic – Examples

• There are at least $k$ elements

$$\exists x_1 \ldots \exists x_k \bigwedge_{i,j \leq k, \ i \neq j} \neg (x_i = x_j)$$

• There is a path of length $k$ from $x_0$ to $x_k$

$$\exists x_1 \ldots \exists x_{k-1} \bigwedge_{0 \leq i < k} E(x_i, x_{i+1})$$

• There is no cycle of length $k$

$$\neg \exists x_1 \ldots \exists x_k \ E(x_k, x_1) \land \bigwedge_{i < k} E(x_i, x_{i+1})$$
Ehrenfeucht-Fraïssé games

• The tool of choice for the neolithic period of FMT.
• Played on two databases (graphs) $\mathcal{A}$ and $\mathcal{B}$.
• Two players:
  • The spoiler (the bad guy) tries to show that $\mathcal{A}$ and $\mathcal{B}$ are different.
  • The duplicator (the good guy) tries to show that $\mathcal{A}$ and $\mathcal{B}$ are the same.
• Play for $k$ rounds.
• In each round:
  • the spoiler moves first: selects a database and an element there.
  • the duplicator responds by an element in the other database.
• The duplicator wins if at the end, the played elements form a partial isomorphism between $\mathcal{A}$ and $\mathcal{B}$.
Ehrenfeucht-Fraïssé game - example 1

A

B
Ehrenfeucht-Fraïssé game - example 1
Ehrenfeucht-Fraïssé game - example 1

A

B
Ehrenfeucht-Fraïssé game - example 1

A

B
Ehrenfeucht-Fraïssé game - example 1
Ehrenfeucht-Fraïssé game - example 1
Ehrenfeucht-Fraïssé game - example 1
Ehrenfeucht-Fraïssé game - example 1
Ehrenfeucht-Fraïssé game - example 1

The duplicator wins in 3 rounds.
Ehrenfeucht-Fraïssé game - example 2

\[ \mathcal{A} \quad \mathcal{B} \]
Ehrenfeucht-Fraïssé game - example 2

\[ \mathcal{A} \]

\[ \mathcal{B} \]
Ehrenfeucht-Fraïssé game - example 2

A

B
Ehrenfeucht-Fraïssé game - example 2
Ehrenfeucht-Fraïssé game - example 2

\[ \text{A} \quad \text{B} \]
Ehrenfeucht-Fraïssé game - example 2

A

B

Leonid Libkin

FMT toolbox

16/57
Ehrenfeucht-Fraïssé game - example 2

\[ \mathcal{A} \]

\[ \mathcal{B} \]
Ehrenfeucht-Fraïssé game - example 2

\[\mathcal{A}\]

\[\mathcal{B}\]
Ehrenfeucht-Fraïssé game - example 2

The spoiler wins in 3 rounds.
Ehrenfeucht-Fraïssé games and FO

The duplicator has a winning strategy in the $k$-round Ehrenfeucht-Fraïssé game if he can win in $k$ rounds no matter how the spoiler plays.

**Theorem**

The duplicator has a winning strategy in the $k$-round Ehrenfeucht-Fraïssé game on $\mathfrak{A}$ and $\mathfrak{B}$

$$\iff$$

$\mathfrak{A}$ and $\mathfrak{B}$ cannot be distinguished by FO sentences of quantifier rank up to $k$. 

How to prove that a property $\mathcal{P}$ is not expressible in FO

Find families of graphs $\mathcal{A}_k$ and $\mathcal{B}_k$, for $k \in \mathbb{N}$ so that:

1. All $\mathcal{A}_k$ have property $\mathcal{P}$;
2. None of $\mathcal{B}_k$ has property $\mathcal{P}$;
3. The duplicator has a winning strategy in the $k$-round Ehrenfeucht-Fraïssé game on $\mathcal{A}_k$ and $\mathcal{B}_k$

Because: if $\mathcal{P}$ were expressible by a sentence $\varphi$ of quantifier rank $k$, by 3. we would have that $\mathcal{A}_k$ and $\mathcal{B}_k$ must agree on $\varphi$, but 1. and 2. say that they do not.
A surprisingly powerful example

Let $L_n$ be a linear ordering of length $n$.

**Theorem**

If $m, n \geq 2^k$, then the duplicator has a winning strategy in the $k$-round Ehrenfeucht-Fraïssé game on $L_n$ and $L_m$. 
A surprisingly powerful example

Let $L_n$ be a linear ordering of length $n$.

**Theorem**

If $m, n \geq 2^k$, then the duplicator has a winning strategy in the $k$-round Ehrenfeucht-Fraïssé game on $L_n$ and $L_m$.

**Corollary**

Query \textsc{Even} is not expressible over linear orderings.

Because: take $\mathcal{A}_k$ to be $L_{2^k}$ and $\mathcal{B}_k$ to be $L_{2^k+1}$.
The early 1980s: the era of tricks

The result about $\text{EVEN}$ shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.
The early 1980s: the era of tricks

The result about $\text{EVEN}$ shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor;
The early 1980s: the era of tricks

The result about $\text{EVEN}$ shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor;
The early 1980s: the era of tricks

The result about \textsc{Even} shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor;
The early 1980s: the era of tricks

The result about \texttt{EVEN} shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor; from the last to the 2nd element;
The early 1980s: the era of tricks

The result about **EVEN** shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor; from the last to the 2nd element; from the penultimate to the 1st.
The early 1980s: the era of tricks

The result about \textbf{EVEN} shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor; from the last to the 2nd element; from the penultimate to the 1st.
The early 1980s: the era of tricks

The result about \textbf{EVEN} shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Edges: to the 2nd successor; from the last to the 2nd element; from the penultimate to the 1st.
The early 1980s: the era of tricks

The result about \texttt{EVEN} shows that none of the following is definable in FO: Graph connectivity; graph acyclicity; transitive closure.

Connected if \texttt{EVEN} is false; disconnected if \texttt{EVEN} is true.
The era of tricks cont’d

- A similar trick (with one backedge instead of two) shows that acyclicity is not FO-expressible.
- Since connectivity is not FO-expressible, neither is the transitive closure query:
  - the symmetric-transitive closure of $G$ is a complete graph iff $G$ is connected.
The 1990s: the era of tools

• The iron age of FMT.
• For more complicated problems, stone (pebble) tools – games – become very hard to use.
• More and more complicated winning conditions are used:
  • Fagin, Ajtai, Vardi, Stockmeyer, Schwentick, Kolaitis, Väänänen, etc
• Fagin, Stockmeyer, Vardi, 1993: Let’s build a library of winning strategies for the duplicator.
• Key idea: locality.
  • Already present in earlier work by Gaifman 1980 and Hanf 1965.
Transitive closure revisited

Degrees of nodes: 0, 1
Transitive closure revisited

Degrees of nodes: $0, 1, \ldots, n$ – depends on the input.
Transitive closure revisited

Degrees of nodes: $0, 1, \ldots, n$ – depends on the input.

This cannot happen for FO queries!
A useful property

- A query $Q$ from graphs to graphs has the **bounded number of degrees property (BNDP)** if there is a function $f_Q : \mathbb{N} \to \mathbb{N}$ such that:

  \[
  \text{all degrees in } G \text{ are bounded by } k \\
  \downarrow \\
  \text{the number of different degrees in } Q(G) \text{ is at most } f_Q(k)
  \]

- We’ve just seen that transitive closure violates the BNDP.
A useful property

• A query $Q$ from graphs to graphs has the bounded number of degrees property (BNDP) if there is a function $f_Q : \mathbb{N} \rightarrow \mathbb{N}$ such that:

\[
\begin{align*}
\text{all degrees in } G \text{ are bounded by } k \\
\downarrow \\
\text{the number of different degrees in } Q(G) \text{ is at most } f_Q(k)
\end{align*}
\]

• We’ve just seen that transitive closure violates the BNDP.

Theorem (Dong, L., Wong’95, L. ’97)

Every FO query has the BNDP.

• Corollary: transitive closure is not FO-definable.
Another application of BNDP – Same Generation

Degrees: 0, 1, 2
Another application of BNDP – Same Generation
Another application of BNDP – Same Generation

Degrees: \(1, 2, 4, 8, \ldots, 2^{\text{depth}(G) - 1}\)  
Number of degrees: \(\text{depth}(G)\)
Another application of BNDP – Same Generation

Violates the BNDP — Hence same-generation is not FO-definable
What makes the BNDP work?

- Locality of FO.
- There are two tools based on locality:
  1. Gaifman-locality (Gaifman 1982)
- Key concept: neighborhood.
- A neighborhood of radius $r$ of $\bar{a}$ in a graph $G$ is denoted by $N^G_r(\bar{a})$.
- It is the subgraph induced by all the nodes of distance $\leq r$ from one of the nodes in $\bar{a}$.
- Nodes $\bar{a}$ are distinguished:
  - if we have an isomorphism $h : N^G_r(a_1, \ldots, a_n) \rightarrow N^{G'}_r(b_1, \ldots, b_n)$ then $h(a_1) = b_1$, $\ldots$, $h(a_n) = b_n$. 
Gaifman-locality

Theorem (Gaifman 1982, bound from L., ’98)

For every FO formula \( \varphi(\bar{x}) \) of quantifier rank \( k \) and for every graph \( G \):

\[ N_{2k}^G(\bar{a}) \text{ and } N_{2k}^G(\bar{b}) \text{ are isomorphic} \implies G \models \varphi(\bar{a}) \iff \varphi(\bar{b}) \]
Theorem (Gaifman 1982, bound from L., ’98)

For every FO formula $\varphi(\bar{x})$ of quantifier rank $k$ and for every graph $G$:

$$N^G_{2k}(\bar{a}) \text{ and } N^G_{2k}(\bar{b}) \text{ are isomorphic} \Rightarrow G \models \varphi(\bar{a}) \iff \varphi(\bar{b})$$

Application: Transitive closure is not definable in FO.
Gaifman-locality

**Theorem (Gaifman 1982, bound from L., '98)**

For every FO formula $\varphi(\bar{x})$ of quantifier rank $k$ and for every graph $G$:

$N^G_{2k}(\bar{a})$ and $N^G_{2k}(\bar{b})$ are isomorphic $\Rightarrow G \models \varphi(\bar{a}) \leftrightarrow \varphi(\bar{b})$

**Application:** Transitive closure is not definable in FO.

If $\varphi(x, y)$ is of quantifier rank $k$ and $r = 2^k$ then both $\varphi(a, b)$ and $\varphi(b, a)$ are true, or both are false.
Hanf-locality

- Write $G \iff_r G'$ if there exists a bijection $f : G \to G'$ such that $N^G_r(a)$ and $N^{G'}_r(f(a))$ are isomorphic for every node $a$.
- Locally two graphs look the same, up to a bijection $f$. 
Hanf-locality

- Write $G \xleftrightarrow{r} G'$ if there exists a bijection $f : G \to G'$ such that $N_r^G(a)$ and $N_r^{G'}(f(a))$ are isomorphic for every node $a$.
- Locally two graphs look the same, up to a bijection $f$.

**Theorem (Fagin, Stockmeyer, Vardi, ’93, bound from L.’98)**

For every FO sentence $\varphi$ of quantifier rank $k$,

$$if \ G \xleftrightarrow{2k} G' \ then \ G \models \varphi \iff \ G' \models \varphi$$

- Can be extended to arbitrary queries, but most often this notion is used for Boolean queries.
Hanf-locality: application

- If $m > 2r + 1$ then $G \Leftrightarrow_r G'$ (all $r$-neighborhoods are the same).
- Hence no FO sentence $\varphi$ defines connectivity: as long as $m > 2^{qr(\varphi)} + 1 + 1$, graphs $G$ and $G'$ cannot be distinguished by $\varphi$. 
Counting: towards 0-1 laws

- How to prove that nontrivial counting properties are not expressible?
- **Even**: roughly half of databases have the property, and half don’t.
- FO cannot exhibit such a behavior.
Counting: towards 0-1 laws

- How to prove that nontrivial counting properties are not expressible?
- **Even**: roughly half of databases have the property, and half don’t.
- FO cannot exhibit such a behavior.

- Pick a database “at random”.
- Check if it satisfies a property $\mathcal{P}$.
- What’s the probability of that?
- If $\mathcal{P}$ is FO-definable, it is 0 or 1: **0-1 law**.
- Need to formalize: ‘pick a database at random’.
Towards 0-1 laws

- For each $n$ look at graphs with nodes $1, \ldots, n$.
- For a property $\mathcal{P}$, let

$$\mu_n(\mathcal{P}) = \frac{|\{\text{graphs on } 1, \ldots, n \text{ that satisfy } \mathcal{P}\}|}{|\{\text{graphs on } 1, \ldots, n\}|}$$

- Proportion of graphs on $1, \ldots, n$ satisfy $\mathcal{P}$, or
- Probability that a randomly picked graph on $1, \ldots, n$ — with respect to the uniform distribution — satisfies $\mathcal{P}$.
- Asymptotic probabilities:

$$\mu(\mathcal{P}) = \lim_{n \to \infty} \mu_n(\mathcal{P})$$
Asymptotic probabilities: examples

• $\mu(\text{EVEN})$ – does not exist: $\mu_n(\text{EVEN}) = \begin{cases} 1, & \text{if } n \text{ is even} \\ 0, & \text{if } n \text{ is odd} \end{cases}$

• $\mu(\text{exists isolated node}) = 0.$

• $\mu(\text{diameter} \leq 2) = 1.$

• $\mu(\text{graph is connected}) = 1.$

• Two sets $A$ and $B$ with $B \subseteq A$.

  Parity is true iff $|B|$ is even.

  $\mu(\text{Parity}) = \frac{1}{2}.$
0-1 law

Theorem (Fagin 1976)

If $\mathcal{P}$ is FO-definable, then $\mu(\mathcal{P})$ exists and equals 0 or 1.
0-1 law

**Theorem (Fagin 1976)**

If $\mathcal{P}$ is FO-definable, then $\mu(\mathcal{P})$ exists and equals 0 or 1.

- If you like truly beautiful proofs, this is the one for you!
- Immediate corollaries: **EVEN** and **Parity** are not FO-definable.
- **Warning**: the result does not hold when we consider specific classes of structures.
- For example, 0-1 law fails over **ordered graphs**:
- $\mu(\text{there is an edge between the first and the last element}) = \frac{1}{2}$. 
Plan

- **Expressiveness:** Tools for first-order logic
  - Neolithic (stone age) tools – games and tricks.
  - Iron age tools: growing up (locality, 0-1 law).

- **Expressiveness:** Tools that work beyond first-order logic
  - The same arsenal of tools

- **Language equivalence:** better query evaluation via the composition method
  - still growing up: learning how to put stones together
FO extensions

- **Ordering**: elements stored in a database are typically ordered, and order comparisons can be used in queries.

- **Counting and aggregation**: we all know it from SQL; a very common feature in database queries.

- **Fixed points**: for many years a popular topic in database research (Datalog). Now also part of SQL-3.

- **Interpreted operations**: e.g., arithmetic operations such as $x^2 + y \leq x \cdot z$ in queries.
Ordering on the domain

- Can transitive closure be expressed over ordered graphs? What about connectivity? acyclicity? etc.
- We know that **EVEN** is not expressible.
- Queries such as transitive closure do not refer to ordering.
- **Order-invariant queries**: can use an ordering, but it does not matter which ordering is used.
- **Order-invariant formulae over graphs**: $\varphi(\bar{x})$ over $E(\cdot, \cdot), <$ so that

$$
(G, <_1) \models \varphi(\bar{a}) \iff (G, <_2) \models \varphi(\bar{a})
$$

for every two orderings $<_1$ and $<_2$ on the nodes.
- Defines an order-invariant query $Q_\varphi$:

$$
\bar{a} \in Q_\varphi(G) \iff (G, <) \models \varphi(\bar{a}) \text{ for some ordering } <
$$
Order-invariant queries

• A mysterious class:
  • only makes sense in the finite;
  • a non-r.e. class of queries;
  • locality techniques do not seem to help: with $<$ everything is a neighborhood of radius 1.

• But quite remarkably:

\[\textbf{Theorem (Grohe, Schwentick, 2000)}\]

Order-invariant queries are Gaifman-local and have the BNDP.

• Corollary: Transitive closure, connectivity, etc are not expressible even with order.
Adding counting and aggregation to the language

- Standard SQL feature.
- Assume domain of 2 sorts:
  - usual database entries (graph nodes);
  - numbers (for examples, $\mathbb{Q}$).
- Add counting terms and operations:
  - $\#\bar{x} . \varphi$ – how many $\bar{x}$ satisfy $\varphi$.
  - $P_{\text{property}}(\cdot)$ testing the property of numbers.
- Examples:
  - $\exists x \ P_{\text{even}}(\#y . E(x, y))$ – there is a node of even degree.
Adding counting and aggregation to the language

- aggregates and grouping by example: sum up all even degrees in a graph
  - in SQL: \[\text{SELECT} \ \text{SUM}(R.C) \ \text{FROM} \ \text{(SELECT} \ E.A, \ \text{COUNT}(E.B) \ \text{AS C} \ \text{FROM} \ E \ \text{GROUPBY} \ E.A \ \text{HAVING} \ \text{MOD(COUNT}(E.B),2) = 0) \ \text{R}\]
  - in logic: \(\text{Aggr}_{\text{SUM}} \times (P_{\text{even}}(\#y.E(x,y)), \ #y.E(x,y))\)
Adding counting and aggregation to the language

- aggregates and grouping by example: sum up all even degrees in a graph
  - in SQL:  
    ```
    SELECT SUM(R.C)
    FROM (SELECT E.A, COUNT(E.B) AS C
           FROM E
           GROUP BY E.A
           HAVING MOD(COUNT(E.B),2) = 0) R
    ```
  - in logic:  
    ```
    Aggr_{SUM} \times (P_{even}(\#y.E(x,y)), \#y.E(x,y))
    ```
  - Formally: \( \mathcal{F} \) is an aggregate (e.g., SUM, COUNT...)

- aggr_term(\( \bar{x} \)) = Aggr_{\mathcal{F}}\bar{y} (\varphi(\bar{x},\bar{y}), t(\bar{x},\bar{y}))

- Semantics:
  - Find all \( \bar{y}_1, \ldots, \bar{y}_k \) so that \( \varphi(\bar{x},\bar{y}_i) \) holds
  - Calculate \( v_i = t(\bar{x},\bar{y}_i) \)
  - aggr_term(\( \bar{x} \)) is \( \mathcal{F}(\{v_1, \ldots, v_k\}) \)
Expressiveness of aggregation

- Which arithmetic predicates and aggregate functions to add?
- Let’s be generous: add them all.
- But still look at queries over graph nodes (e.g., transitive closure).

**Theorem (Hella, L., Nurmonen, Wong’99, improved L.’01)**

Queries expressed in the aggregate language with arbitrary arithmetic and aggregates are local:
i.e., Hanf-local, Gaifman-local, and have the BNDP.

- In particular, the usual SQL (select-from-where-groupby-having) cannot express transitive closure.
Aggregation and order

- What if we have an order on graph nodes? Can we recover locality?
- **No**, even in a minimalistic setting:
  - Arithmetic: $<, +, \times$
  - Aggregation: \text{SUM}
- If such an aggregate language cannot express transitive closure over ordered graphs, then some complexity classes are separated:
  - \text{TC}^0 \text{ and NLOGSPACE}
  - big open problem in complexity theory
Recursion and Datalog

- Have seen it already:
  - transitive closure:

    \[
    \text{trcl}(x, y) \quad :\quad e(x, y) \\
    \text{trcl}(x, y) \quad :\quad e(x, z), \text{trcl}(z, y)
    \]

  - same-generation:

    \[
    \text{sg}(x, x) \quad :\quad \\
    \text{sg}(x, y) \quad :\quad e(x', x), e(y', y), \text{sg}(x', y')
    \]

- Now available in the latest SQL standard: WITH RECURSIVE.
  - But without negation.
  - With negation, several semantics exist.
Datalog: expressive power

- Without negation, queries are monotone.
- Even with negation and inflationary semantics:

  \[ \text{Theorem (Blass, Kozen, Gurevich, 1985)} \]

  \[ \text{Datalog has the 0-1 law.} \]

- This is without order. What if order is added?
- Then Datalog (with negation) captures PTIME.
- To prove bounds, one needs to separate complexity classes again.
- But without order, it can be separated from NP: 3-colorability is not expressible in Datalog with negation (Dawar, ’98).
  - A useful result (recent application in the work on schema mappings)
Plan

- Expressiveness: Tools for first-order logic
  - Neolithic (stone age) tools – games and tricks.
  - Iron age tools: growing up (locality, 0-1 law).

- Expressiveness: Tools that work beyond first-order logic
  - The same arsenal of tools

- Language equivalence: better query evaluation via the composition method
  - still growing up: learning how to put stones together
Language equivalence: games come back

• The focus of FMT applications in databases switches in the 21st century from inexpressibility results to proving language equivalence.

• Goal: start with a benchmark of expressiveness, and find a language with good complexity of query evaluation.

• Usually in the context of data that comes with a nice structure.

• XML data model – labeled trees with some nodes carrying data values.

• For the talk, use words – to keep pictures and notations simple – but the paper deals with trees as well.
Language equivalence: games come back

- Key technique: composing Ehrenfeucht-Fraïssé games.
- Such composition tells us how queries on substructures combine for evaluating queries on the whole structure.
- For words/trees, benchmark expressiveness is typically **MSO**:
  - Monadic Second Order Logic, adds quantification over sets.
  - \( \exists X_1 \forall X_2 \ldots \alpha(X_1, X_2, \ldots) \) where \( \alpha \) is FO.
  - Has the power of **automata** on words and trees.
  - But **problematic** complexity of query evaluation.
- Question: is it possible to achieve better complexity simply by syntactic manipulations?
Changing syntax to lower complexity

There are known examples, e.g., **FO** and **LTL** (linear temporal logic):

- Have the same power over words (Kamp, 1969)
- Checking whether $w \models \varphi$ with **linear data complexity** (i.e., $O(|w|)$) requires the following **query complexity**:
  - **non-elementary** for **FO** (a stack of exponentials, Frick, Grohe, '03)
  - **linear** for **LTL**
- **FO** or **MSO** evaluation over trees and words with **linear data complexity** implies **non-elementary query complexity**.
- Need another – but equivalent – logic!
Words as databases

A word $w$ over $\Sigma = \{a_1, \ldots, a_m\}$ is a database with relations $E(\cdot, \cdot), L_1(\cdot), \ldots, L_m(\cdot)$:

- $E$ is the ordering of positions;
- $L_i$'s define labelings.

$w = a_1a_2a_1a_2$:

positions 0, 1, 2, 3; positions 0,2,3 labeled $a_1$; position 1 labeled $a_2$

$$E = \begin{array}{c|c}
0 & 1 \\
1 & 2 \\
2 & 3 \\
0 & 2 \\
1 & 3 \\
0 & 3 \\
\end{array}$$

$$L_1 = \begin{array}{c}
0 \\
2 \\
3 \\
\end{array}$$

$$L_2 = \begin{array}{c}
1 \\
\end{array}$$
MSO over words

Each MSO sentence $\varphi$ defines a language

$$\mathcal{L}(\varphi) = \{ w \in \Sigma^* | w \models \varphi \}$$

Theorem (Büchi, Elgot, Trakhtenbrot 1960)

$\text{MSO-definability} = \text{Regular languages}$

A similar result holds for trees as well – both binary and unranked.

We now show how to go from MSO to automata.
Types

• Look at FO (or MSO) sentences of quantifier rank \( k \)
• Only finitely many distinct ones – up to logical equivalence
• **Rank-\( k \) type**: set of all sentences of quantifier rank \( k \) true in a database. Notation: \( tp_k(D) \)
• Types are finite objects, definable in the logic.
• Each sentence is a disjunction of types.
• Another way of looking at Ehrenfeucht-Fraïssé games:
  \[
  tp_k(D) = tp_k(D') \iff \text{the duplicator has a winning strategy in the \( k \)-round game.}
  \]
• For MSO, the game is slightly more complex:
  • both players can play sets and points
  • but all the results remain true.
From MSO to automata: automata compute types

If $tp_k(w) = tp_k(w')$, then $tp_k(w \cdot a) = tp_k(w' \cdot a)$: compose games!
From MSO to automata: automata compute types

If $tp_k(w) = tp_k(w')$, then $tp_k(w \cdot a) = tp_k(w' \cdot a)$: compose games!

$w$

duplicator wins

in $k$ rounds

$w'$
From MSO to automata: automata compute types

If $tp_k(w) = tp_k(w')$, then $tp_k(w \cdot a) = tp_k(w' \cdot a)$: compose games!
If \( tp_k(w) = tp_k(w') \), then \( tp_k(w \cdot a) = tp_k(w' \cdot a) \): compose games!

 duplicator still wins in \( k \) rounds
From MSO to automata: automata compute types

If $tp_k(w) = tp_k(w')$, then $tp_k(w \cdot a) = tp_k(w' \cdot a)$

Deterministic Automaton for sentence $\varphi$:

- **States** are rank-$k$ types;
- **Initial state**: the type of the empty word;
- **Final states**: those types whose disjunction forms $\varphi$.
- **Transition** $\delta(\tau, a)$: the uniquely determined type of $w \cdot a$ if the type of $w$ is $\tau$.

After reading $w$, the state of the automaton is $tp_k(w)$. 
Language for extracting positions in words?

- Why? We really need a language for extracting positions in trees:
  - Information extraction from XML document;
  - Work by Gottlob, Koch, and colleagues; Lixto system
- We demonstrate the idea on words; it works perfectly well for trees as well.
Language for extracting positions in words?

- Why? We really need a language for extracting positions in trees:
  - Information extraction from XML document;
  - Work by Gottlob, Koch, and colleagues; Lixto system
- We demonstrate the idea on words; it works perfectly well for trees as well.
- Key composition ideas:

  1) \( tp_k(w) = tp_k(w') \)
     \( tp_k(u) = tp_k(u') \) \( \Rightarrow \) \( tp_k(w \cdot a \cdot u) = tp_k(w' \cdot a \cdot u') \)

  2) \( tp_k(u) = tp_k(w) \) \( \Rightarrow \) \( tp_k(w^{-1}) = tp_k(u^{-1}) \)
Language for extracting positions in words

- How to express MSO (or FO) $\varphi(x)$ over words?
- **Idea:** for $w \cdot a \cdot u$, compute
  1. $tp_k(w)$ going forward from the first position;
  2. $tp_k(u^{-1})$ going backwards from the last position;
  3. These types tell us whether the $a$ position is selected.

- Express this in **Datalog**.
- Compute $tp_k(w)$ going forward – use predicates $U_\tau$ for types:

  \[
  U_{\tau_a}(x) \; : = \; \text{First}(x), L_a(x); \quad a \in \Sigma \\
  U_{\tau'}(x) \; : = \; \text{Succ}(y, x), L_a(x), U_\tau(y); \quad a \in \Sigma, \; \delta(\tau, a) = \tau'
  \]
Datalog program cont’d

- Types going backwards $V_{\tau}$: symmetric.
- Answer – for all triples $(\tau, a, \tau')$ saying that $a$ is selected, add:
  
  $$\text{Answer}(x) :– U_{\tau}(y), \text{Succ}(y, x), P_{a}(x), \text{Succ}(x, z), V_{\tau'}(z)$$

- We used **Monadic Datalog**: all idb predicates are monadic.
  - Edb predicates: successor, labelings, First and Last.
- It captures MSO over words.
- Complexity of evaluating program $P$ on $w$: $O(||P|| \cdot |w|)$. 
Review of the journey

- Composition technique suggested expressing MSO and FO properties in monadic datalog.
- Monadic datalog captures MSO and has very good complexity bounds.
- The approach works for trees and yields many XML languages:
  - Monadic datalog captures MSO for trees, with the same complexity – one needs to add predicates for the root, leaves, first and last children of nodes (Gottlob, Koch, '01)
  - ETL – Efficient tree logic (Neven, Schwentick, '00)
  - Temporal logics with good query evaluation properties (Schlingloff '92, Marx '04, Barceló, L., '05)
  - Dialects of XPath (Marx '04)
Conclusion

• Quite likely, if you need to use FMT in database research, you need one of the techniques described in this tutorial (paper).
• No need to read textbooks/proofs – just use the toolbox!
  • Unless you want to work in FMT.
Conclusion

- Quite likely, if you need to use FMT in database research, you need one of the techniques described in this tutorial (paper).
- No need to read textbooks/proofs – just use the toolbox!
  - Unless you want to work in FMT.
- But if you want to read an FMT book, ask me after the talk and I’ll tell you which one to buy.
The future of FMT

- We’ve learned how to work with both stone and iron age tools.
- Now it’s time to go back to games and start putting stones together.
- Middle Ages tools: actively developed by Ben Rossman (the rest of the world is trying to catch up). Solved 3 long-standing open problems recently:
  1. successor-invariance
  2. preservation under homomorphisms in the finite
  3. strictness of the $FO^k$ hierarchy over ordered structures.
- Next step, $N$ years away: modern era tools.
  - Main application: separating complexity classes.
- Good news – most of the database theory tasks are easily doable with stone and iron age tools.