UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L.Özçep

# Finite Model Theory

*Lecture 4: Locality, 0-1 law, Fixed Points*
*30 April 2020*

*Informationssysteme CS4130*
*(Summer 2020)*

# Recap of Lecture 3

- Finite Model Theory approach
  - consider DBs as finite structures
  - FOL as query language
- FOL works because
  - Though FOL model checking in PSPACE w.r.t. combined complexity
  - it is in $AC^0$ for data complexity
- Inexpressivity Tools
  - Games as basic tool for proving inexpressivity
  - Reduction tricks

**End of Recap**

# Locality

# Proving Inexpressibility by Locality

- FOL has a fundamental property: locality
- Intuition
  - Binary query $Q : STRUCT(\sigma) \longrightarrow STRUC(ans)$
  - $Q$ to be defined in FOL
  - So, we need a formula $\phi_Q$ in two open variables $x, y$
  - The way how to describe constraints between $x$ and $y$ is restricted by the number of atoms and elements occurring in $\phi_Q$.

# Proving Inexpressibility by Locality

- FOL has a fundamental property: locality
- Intuition
  - Binary query $Q : STRUCT(\sigma) \longrightarrow STRUC(ans)$
  - $Q$ to be defined in FOL
  - So, we need a formula $\phi_Q$ in two open variables $x, y$
  - The way how to describe constraints between $x$ and $y$ is restricted by the number of atoms and elements occurring in $\phi_Q$.
- Different (comparable) locality notions
  - Bounded number of degrees property (BNDP)
  - Gaifman locality
  - Hanf locality

# BNDP

- $in(\mathfrak{G})$ = set of in-degrees of nodes in $\mathfrak{G}$
- $out(\mathfrak{G})$ = set of out-degrees of nodes in $\mathfrak{G}$
- $degs(\mathfrak{G}) = in(\mathfrak{G}) \cup out(\mathfrak{G})$

**Definition**

$Q$ has the bounded number of degrees property (BNDP) iff there is $f_Q : \mathbb{N} \longrightarrow \mathbb{N}$ s.t. for all graphs $\mathfrak{G}$:

If there is $k \in \mathbb{N}$ s.t. $\max(degs(\mathfrak{G})) \leq k$,
then $|degs(Q(\mathfrak{G}))| \leq f_Q(k)$.

- Intuitively: $Q$ disallowed to arbitrarily increase degrees of nodes

**Theorem**

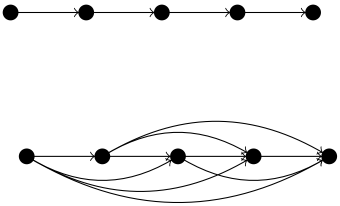
*Every FOL query has the BNDP.*

# Example: TC on Successor Relation Graph

- $\mathfrak{G} = (\{a_0, \ldots, a_n\}, \{E(a_0, a_1), \ldots, E(a_{n-1}, a_n)\})$
- $in(\mathfrak{G}) = out(\mathfrak{G}) = \{0, 1\}$
- $in(TC(\mathfrak{G})) = out(TC(\mathfrak{G})) = \{0, \ldots, n-1\}$

$\mathfrak{G}$

$TC(\mathfrak{G})$



# It's (sometimes) sufficient to Consider Graphs Only

## Definition (Gaifman Graph)

For any $\sigma$ structure $\mathfrak{A}$ one can define the Gaifman graph $\mathfrak{G} = (G, E)$ as follows:

- $G = dom(\mathfrak{A})$
- There is an edge between two elements $a, b$ of $\mathfrak{A}$ iff they co-occur within a relation of $\mathfrak{A}$, formally:

  $(a, b) \in E^{\mathfrak{G}}$ iff $a \neq b$ and there is some ($n$-ary) relation $R^{\mathfrak{A}}$ and a tuple $(a_1, \ldots, a_n)$ such that $a, b$ are among those elements and such that $(a_1, \ldots, a_n) \in R^{\mathfrak{A}}$

# It's (sometimes) sufficient to Consider Graphs Only

## Definition (Gaifman Graph)

For any $\sigma$ structure $\mathfrak{A}$ one can define the Gaifman graph $\mathfrak{G} = (G, E)$ as follows:

- $G = dom(\mathfrak{A})$
- There is an edge between two elements $a, b$ of $\mathfrak{A}$ iff they co-occur within a relation of $\mathfrak{A}$, formally:

  $(a, b) \in E^{\mathfrak{G}}$ iff $a \neq b$ and there is some ($n$-ary) relation $R^{\mathfrak{A}}$ and a tuple $(a_1, \ldots, a_n)$ such that $a, b$ are among those elements and such that $(a_1, \ldots, a_n) \in R^{\mathfrak{A}}$

- $d(a, b)$ = distance between two vertices $a, b$ = path of minimal length between $a, b$
- $d(\bar{a}, b) = min_{a_i \in \bar{a}}\{d(a_i, b)\}$ = distance of vertex $b$ from tuple of vertices $\bar{a}$

# Gaifman locality

Gaifman locality defined here on graphs $\mathfrak{G} = (G, E)$
(can be generalized to arbitrary structures with Gaifman graph)

### Gaifman Locality (Intuitively)

An $m$-ary query $Q$ is Gaifman local iff there is a threshold (radius) $r$ such that for all graphs:

$Q$ cannot distinguish between tuples if their $r$-neighbourhoods in the graph are the same.

### Theorem

*Every FOL-definable query is Gaifman local.*

# Gaifman Locality

- $\overline{a} = (a_1, \ldots, a_n) \in G^n$ (vector of elements)

# Gaifman Locality

- $\overline{a} = (a_1, \ldots, a_n) \in G^n$ (vector of elements)
- $B_r^{\mathfrak{G}}(\overline{a}) = \{b \in G \mid d(\overline{a}, b) \leq r\}$ (radius $r$ ball around $\overline{a}$)

# Gaifman Locality

- $\overline{a} = (a_1, \ldots, a_n) \in G^n$ (vector of elements)
- $B_r^{\mathfrak{G}}(\overline{a}) = \{b \in G \mid d(\overline{a}, b) \leq r\}$ (radius $r$ ball around $\overline{a}$)
- $N_r^{\mathfrak{G}}(\overline{a})$ (r-neighbourhood of $\overline{a}$)
  subgraph induced by $B_r^{\mathfrak{G}}(\overline{a})$ in the structure $(G, E, \overline{a})$

# Gaifman Locality

- $\overline{a} = (a_1, \ldots, a_n) \in G^n$ (vector of elements)
- $B_r^{\mathfrak{G}}(\overline{a}) = \{b \in G \mid d(\overline{a}, b) \leq r\}$ (radius $r$ ball around $\overline{a}$)
- $N_r^{\mathfrak{G}}(\overline{a})$ (r-neighbourhood of $\overline{a}$)
  subgraph induced by $B_r^{\mathfrak{G}}(\overline{a})$ in the structure $(G, E, \overline{a})$
  - Note: $(G, E, \bar{a})$ is a graph where some elements (namely that of $\bar{a}$) are named by constants: they are fixed
  - In $N_r^{\mathfrak{G}}(\overline{a})$ the elements $\overline{a}$ have the same names as in $(G, E, \overline{a})$ (say $c_1, \ldots, c_n$) and there is an edge between a pair of elements $B_r^{\mathfrak{G}}(\overline{a})$ iff there is an edge in $(G, E, \bar{a})$ between them

# Gaifman Locality

- $\overline{a} = (a_1, \ldots, a_n) \in G^n$ (vector of elements)
- $B_r^{\mathfrak{G}}(\overline{a}) = \{b \in G \mid d(\overline{a}, b) \leq r\}$ (radius $r$ ball around $\overline{a}$)
- $N_r^{\mathfrak{G}}(\overline{a})$ (r-neighbourhood of $\overline{a}$)
  subgraph induced by $B_r^{\mathfrak{G}}(\overline{a})$ in the structure $(G, E, \overline{a})$

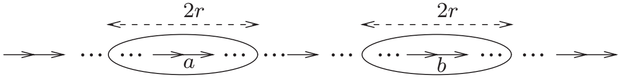
**Definition**

An $m$-ary query $Q$ (with $m > 0$) is Gaifman-local iff:

There exists a radius $r$ s.t. for all $\mathfrak{G}$: If $N_r^{\mathfrak{G}}(\overline{a}) \simeq N_r^{\mathfrak{G}}(\overline{b})$, then $\overline{a} \in Q(\mathfrak{G})$ exactly when $\overline{b} \in Q(\mathfrak{G})$.

# Example: TC is not Gaifman local

$\mathfrak{G}$ :




Proof

- Suppose TC were FOL definable with query $Q$
- Then $Q$ would be Gaifman local with some radius $r$
- $N_r^{\mathfrak{G}}((a,b)) \simeq N_r^{\mathfrak{G}}((b,a))$
  because both subgraphs are disjoint unions of two $2r$-chains
- But $(a,b) \in TC(\mathfrak{G})$ and $(b,a) \notin TC(\mathfrak{G})$, ↯

# Hanf locality

**Definition (Hanf locality (informally))**

A Boolean query $Q$ is Hanf-local iff there is a threshold (radius) $r$ s.t. any pair of graphs $\mathfrak{G}, \mathfrak{G}'$ that can be made pointwise similar w.r.t. $r$-neighbourhoods cannot be told apart by $Q$.

- Have to make precise "pointwise similar"

# Hanf locality

- $\mathfrak{G} = (A, E), \mathfrak{G}' = (A', E')$
- $\mathfrak{G} \rightleftarrows_r \mathfrak{G}'$ iff
  there exists bijection $f : A \longrightarrow A'$ s.t. for all $a \in A$:
  $N_r^{\mathfrak{G}}(a) \simeq N_r^{\mathfrak{G}'}(f(a))$

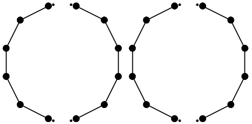
**Definition (Hanf locality (formal))**

A Boolean query $Q$ is Hanf-local iff a radius $r$ exists s.t. for any graphs $\mathfrak{G}, \mathfrak{G}'$ with $\mathfrak{G} \rightleftarrows_r \mathfrak{G}'$ one has $Q(\mathfrak{G}) = Q(\mathfrak{G}')$.

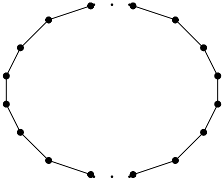
**Theorem**

*Every FOL definable Boolean query is Hanf-local.*

# Example: CONN is not Hanf-local



$\mathfrak{G}$: two cycles of length $m$



$\mathfrak{G}'$: one cycle of length $2m$

Proof

- For contradiction assume CONN is Hanf-local with parameter $r$
- Choose $m > 2r + 1$; $f$ an arbitrary bijection of $\mathfrak{G}$ and $\mathfrak{G}'$
- $r$-neighbourhood of any $a$ the same: $2r$-chain with $a$ in the middle
- Hence $\mathfrak{G} \rightleftarrows_r \mathfrak{G}'$, but: $\mathfrak{G}'$ is connected and $\mathfrak{G}$ is not. ↯

# Comparison of Locality Notions

**Theorem**

*Hanf local* $\models$ *Gaifmann local* $\models$ *BNDP*

# Optional Slide: Adding Order

- Many applications have finite models with a linear order $<$
- Locality conditions in its original form not applicable: $1$-radius already whole structure
- Consider invariant queries

## Definition

A query $Q$ over ordered structures is invariant iff
for all structures $\mathfrak{A}$, all tuples $\overline{b}$ and all linear orders $<_1, <_2$ on $\mathfrak{A}$:
$\overline{b} \in Q((\mathfrak{A}, <_1))$ iff $\overline{b} \in Q((\mathfrak{A}, <_2))$

For an invariant $Q$ define $Q_{inv}$ on arbitrary structures as:
$Q_{inv}(\mathfrak{A}) = Q((\mathfrak{A}, <))$ for arbitrarily chosen $<$.
$Q_{inv}$ called invariant FO-query.

# Optional Slide: Adding Order

- Invariant FO-queries (over finite (!) structures) may still be more expressive than FO-queries (without order)
- Hint
  - The pure existence of an order suffices to talk about evenness
  - Consider Boolean algebras (BA) with even number of atoms.
  - Not axiomatizable in FOL (show using Ehrenfeucht-Fraïsse) but by order invariant FO
  - Axiom states that there is an element in BA containing all atoms in even position and the last one.

# Optional Slide: Adding Order

- Invariant FO-queries (over finite (!) structures) may still be more expressive than FO-queries (without order)
- Hint
  - The pure existence of an order suffices to talk about evenness
  - Consider Boolean algebras (BA) with even number of atoms.
  - Not axiomatizable in FOL (show using Ehrenfeucht-Fraïsse) but by order invariant FO
  - Axiom states that there is an element in BA containing all atoms in even position and the last one.
- Nonetheless, we have

**Theorem**

*Every invariant FOL query is Gaifman-local (and so has BNDP).*

# 0-1 law

# 0-1 law

An inexpressibility tool based on a probabilistic property of FOL queries

## 0-1-law informally

Either almost all finite structures fulfill the property or almost all do not

## Example

Consider the following boolean queries on graphs

- $Q_1 = \forall x, y\ E(x, y)$

  Almost all graphs do not satisfy $Q_1$ (only the complete ones)

- $Q_2 = \forall x \forall y \exists z\ E(z, x) \wedge \neg E(z, y)$

  Almost all graphs satisfy $Q_2$

# Formal definition 0-1 laws

- Here it is important that signature $\sigma$ is relational!!
- $STRUC(\sigma, n)$: structures with domain $[n] := \{0, 1, \dots, n-1\}$ over $\sigma$.
- For a Boolean query $Q$ let

$$\mu_n(Q) = \frac{|\{\mathfrak{A} \in STRUC(\sigma, n) \mid Q(\mathfrak{A}) = true\}|}{|STRUC(\sigma, n)|}$$

- $\mu_n(Q)$ is the probability that a randomly chosen structure on $[n]$ satisfies $Q$
- $\mu(Q) = \lim_{n\to\infty} \mu_n(Q)$ (if limit exists)

**Definition**

A logic has the 0-1-law if for every Boolean query $Q$ expressible in it either $\mu(Q) = 0$ or $\mu(Q) = 1$.

# Inexpressibility with 0-1 laws

**Theorem**

*FOL has the 0-1-law.*

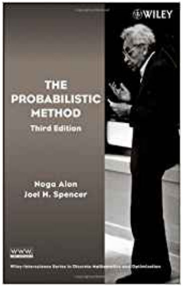
- Helpful for proving inexpressibility of counting properties

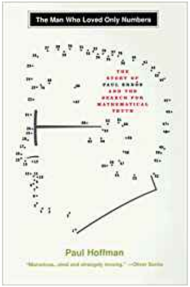
**Example (EVEN is not expressible in FOL)**

$\mu(EVEN)$ not defined because $\mu_n(EVEN)$ alternates between $0$ and $1$.

# Probability and Logic

- The 0-1 law exemplifies a general strategy of using methods for handling uncertainty (probability theory) in order to solve crisp questions (here: FOL expressibility)
- Compare "probabilistic method" as applied to combinatorics
  - Also called "Erdös method"
  - Take a time to learn about the great Hungarian mathematician Erdös, e.g., from biography "The man who loved only numbers" http://www.nytimes.com/books/first/h/hoffman-man.html



# Beyond FOL

# Counting and Aggregation

- Practical languages s.a. SQL allow counting and aggregation.

**Example (Aver. Salary in Depts. with Total Salary > 100,000)**

```
SELECT S1.Dept, AVG(S2.Salary)
FROM S1, S2
WHERE S1.Empl = S2.Empl
GROUP BY S1.Dept
HAVING SUM(S2.Salary) > 100,000
```

Schema: S1(Empl, Dept), S2(Empl, Salary)

- Consider corresponding extensions of FOL
- Some of the tools shown so far still work (when non-ordered structures are considered)

# FOL with counting quantifiers

**Definition (FOL-AllCnt)**

FOL-AllCnt is the extension of FOL with counting quantifiers and counting terms:

- $\exists^{\geq i} x.\phi(x)$: There are at least $i$ elements $x$ fulfilling $\phi$.
- $\sharp\overline{x}.\phi(\overline{x})$: the number of $\overline{x}$ fulfilling $\phi(\overline{x})$.

- Semantics defined w.r.t. 2-sorted FOL structures $\mathfrak{A} = (A, \mathbb{N}, (R^{\mathfrak{A}})_{R\in\sigma}, Arith)$
- Second domain (sort) $\mathbb{N}$ is infinite!
- $Arith$ contains (interpreted) arithmetic predicates and functions

## Example

Parity of a unary predicate symbol U can be expressed by the following formula using counting quantifiers:

$$\exists j \exists i((i + i = j) \wedge \exists^{\geq j} x U(x) \wedge \forall k(\exists^{\geq k} x U(x) \rightarrow k \leq j))$$

"There is an even number ($j$) of $U$s and there are no more than $j$ $U$s"

## Theorem

*FOL+AllCtn queries are Hanf local (and thus Gaifman local and have the BNDP).*

# Aggregation

- $\mathcal{F}$ = aggregate function = family of functions $f_1, f_2, \ldots$ with
- $f_n$ maps $n$-element multisets from $\mathbb{Q}$ to elements from $\mathbb{Q}$.
  E.g.: $SUM = \{s_1, s_2, \ldots,\}$ with $s_k(\{d_1, \ldots, d_k\}) = \Sigma_{i=1}^{k} d_i$

# Aggregation

- $\mathcal{F}$ = aggregate function = family of functions $f_1, f_2, \ldots$ with
- $f_n$ maps $n$-element multisets from $\mathbb{Q}$ to elements from $\mathbb{Q}$.
  E.g.: $SUM = \{s_1, s_2, \ldots, \}$ with $s_k(\{d_1, \ldots, d_k\}) = \Sigma_{i=1}^{k} d_i$

## Definition (FOL-Aggr)

FOL-Aggr = FOL-AllCnt + aggregate terms + $\mathbb{Q}$ instead of $\mathbb{N}$

- Syntax: Terms $t(\overline{x})$ of the form $Aggr_{\mathcal{F}}\overline{y}.(\phi(\overline{x}, \overline{y}), t'(\overline{x}, \overline{y}))$
  - Note the possibility of nesting with term $t'$ (as in SQL)
- Semantics over $\mathfrak{A}$ for tuple $\overline{b}$
  - $t^{\mathfrak{A}}(\overline{b}) = f_{|B|}(\{t'^{\mathfrak{A}}(\overline{b}, \overline{c}) \mid \overline{c} \in B\})$
    where $B := \{\overline{c} \mid \mathfrak{A} \models \phi(\overline{b}, \overline{c})\}$

# Aggregation

- $\mathcal{F}$ = aggregate function = family of functions $f_1, f_2, \ldots$ with
- $f_n$ maps $n$-element multisets from $\mathbb{Q}$ to elements from $\mathbb{Q}$.
  E.g.: $SUM = \{s_1, s_2, \ldots, \}$ with $s_k(\{d_1, \ldots, d_k\}) = \Sigma_{i=1}^{k} d_i$

## Definition (FOL-Aggr)

FOL-Aggr = FOL-AllCnt + aggregate terms + $\mathbb{Q}$ instead of $\mathbb{N}$

- Syntax: Terms $t(\overline{x})$ of the form $Aggr_{\mathcal{F}}\overline{y}.(\phi(\overline{x}, \overline{y}), t'(\overline{x}, \overline{y}))$
  - Note the possibility of nesting with term $t'$ (as in SQL)
- Semantics over $\mathfrak{A}$ for tuple $\overline{b}$
  - $t^{\mathfrak{A}}(\overline{b}) = f_{|B|}(\{t'^{\mathfrak{A}}(\overline{b}, \overline{c}) \mid \overline{c} \in B\})$
    where $B := \{\overline{c} \mid \mathfrak{A} \models \phi(\overline{b}, \overline{c})\}$

Correspondence to SQL:

- $\overline{x}$ = grouping attributes
- $\phi(\overline{x}, \overline{y})$ = HAVING clause

# Locality for FOL+Aggr

**Theorem**

*FOL-Aggr queries are Hanf-local (and thus Gaifmann-local and have the BNDP).*

- If order is added, then locality is lost

# Higher-Order Logics

- Second order logic (SO): Allow quantification over relations
- Vocabulary: FOL vocabulary + predicate variables $X, Y, \ldots$
- Syntax: FOL syntax +
  - $Xt_1 \ldots t_n$ is a formula (for $n$-ary relation variable $X$ and terms $t_i$)
  - If $\phi$ is a formula, then so are $\exists X \phi$, $\forall X \phi$

# Higher-Order Logics

- Second order logic (SO): Allow quantification over relations
- Vocabulary: FOL vocabulary + predicate variables $X, Y, \ldots$
- Syntax: FOL syntax +
  - $Xt_1 \ldots t_n$ is a formula (for $n$-ary relation variable $X$ and terms $t_i$)
  - If $\phi$ is a formula, then so are $\exists X \phi$, $\forall X \phi$

- Higher-order quantification adds expressivity, e.g.,
- $EVEN(\sigma)$ (for any signature $\sigma$, in particular for $\sigma = \{\}$) expressible. (Exercise)

# Pormint example: MSO

- Monadic Second order logic (MSO): SO with second order quantifiers over unary predicates
- (Finite) words/strings $w$ over alphabet $\Sigma$ as (finite) structures over signature $Str = \{<, P_a\}_{a\in\Sigma}$
  - Domain = $[n] = \{0, 1, \dots, n-1\}$ = positions in word of length $n$
  - For each symbol $a \in \Sigma$ unary predicate $P_a$ of positions at wich $a$ occurs
  - Binary order $<$ on positions
  - Example: $w = abba$ is structure $([0, 1, 2, 3], <, \{0, 3\}, \{1, 2\})$

**Theorem (Regular languages = MSO)**

*The regular languages are exactly those definable by MSO sentences.*

# Fixed Point Logics (FPLs)

- Reachability queries call for extension of FOL with "iteration" mechanism
- FPLs use a well-behaved self-referential process/bootstrapping
  - Fixed points as limits of this process
  - Different fixed points may exist

# Fixed Point Logics (FPLs)

- Reachability queries call for extension of FOL with “iteration” mechanism
- FPLs use a well-behaved self-referential process/bootstrapping
  - Fixed points as limits of this process
  - Different fixed points may exist

- Different fixed point logics exist (e.g. largest, least)
- Most prominent in DB theory: Datalog

# Example: Compute the Transitive Closure

- $E(x, y) =$ edge of graph $\mathfrak{G}$,
- $R(x, y) =$ transitively closed relation between vertices

$$\forall x, \forall y\, R(x, y) \quad \leftrightarrow \quad E(x, y) \vee (\exists z. E(x, z) \wedge R(z, y))$$

# Example: Compute the Transitive Closure

- $E(x, y) =$ edge of graph $\mathfrak{G}$,
- $R(x, y) =$ transitively closed relation between vertices

$$\forall x, \forall y\, \boxed{R}(x, y) \quad \leftrightarrow \quad E(x, y) \vee (\exists z. E(x, z) \wedge \boxed{R}(z, y))$$

- For all graphs $\mathfrak{G}$ find extension $\mathfrak{G}' = (\mathfrak{G}, R^{\mathfrak{G}'})$ s.t. lhs and rhs evaluate to the same relation. (*)

# Example: Compute the Transitive Closure

- $E(x, y) =$ edge of graph $\mathfrak{G}$,
- $R(x, y) =$ transitively closed relation between vertices

$$\forall x, \forall y\, R(x, y) \quad \leftrightarrow \quad E(x, y) \vee (\exists z. E(x, z) \wedge R(z, y))$$

- For all graphs $\mathfrak{G}$ find extension $\mathfrak{G}' = (\mathfrak{G}, R^{\mathfrak{G}'})$ s.t. lhs and rhs evaluate to the same relation. (*)
- Read equivalence as a iteratively applied rule from right to left

$$\boxed{X_{new}}(x, y) \quad \leftarrow \quad \underbrace{E(x, y) \vee (\exists z. E(x, z) \wedge X_{old}(z, y))}_{\phi(x, y, X_{old})}$$

# Example: Compute the Transitive Closure

- $E(x, y) =$ edge of graph $\mathfrak{G}$,
- $R(x, y) =$ transitively closed relation between vertices

$$\forall x, \forall y\, R(x, y) \quad \leftrightarrow \quad E(x, y) \lor (\exists z. E(x, z) \land R(z, y))$$

- For all graphs $\mathfrak{G}$ find extension $\mathfrak{G}' = (\mathfrak{G}, R^{\mathfrak{G}'})$ s.t. lhs and rhs evaluate to the same relation. (*)
- Read equivalence as a iteratively applied rule from right to left

$$\boxed{X_{new}}(x, y) \quad \leftarrow \quad \underbrace{E(x, y) \lor (\exists z. E(x, z) \land X_{old}(z, y))}_{\phi(x, y, X_{old})}$$

- Induces a step(-jump)-operator $F$ on the semantical side
- For $X \subseteq G \times G$:

$$F : X \mapsto \boxed{\{(d_1, d_2) \mid (\mathfrak{G}, X, x/d_1, y/d_2) \models \phi(x, y, X)\}}$$

## Example: Compute the Transitive Closure

- $E(x, y) =$ edge of graph $\mathfrak{G}$,
- $R(x, y) =$ transitively closed relation between vertices

$$\forall x, \forall y\, R(x, y) \quad \leftrightarrow \quad E(x, y) \vee (\exists z. E(x, z) \wedge R(z, y))$$

- For all graphs $\mathfrak{G}$ find extension $\mathfrak{G}' = (\mathfrak{G}, R^{\mathfrak{G}'})$ s.t. lhs and rhs evaluate to the same relation. (*)
- Read equivalence as a iteratively applied rule from right to left

$$\boxed{X_{new}}(x, y) \quad \leftarrow \quad \underbrace{E(x, y) \vee (\exists z. E(x, z) \wedge X_{old}(z, y))}_{\phi(x,y,X_{old})}$$

- Induces a step(-jump)-operator $F$ on the semantical side
- For $X \subseteq G \times G$:

$$F : X \mapsto \boxed{\{(d_1, d_2) \mid (\mathfrak{G}, X, x/d_1, y/d_2) \models \phi(x, y, X)\}}$$

- Condition (*) reread: find fixed point $R$, i.e., $F(R) = R$

# Constructing Least Fixed Points

- Start with extension $\emptyset$ (seed) and proceed iteratively
- Progress schema: $\emptyset$, $F(\emptyset)$, $F(F(\emptyset))$, $F^3(\emptyset)$, $F^4(\emptyset), \ldots$
- In our example
  - $X^0 = \text{seed} = \emptyset$
  - $X^1 = E^{\mathfrak{G}} = \text{direct edges}$
  - $X^2 = F(X^1) = X^1 \cup \{(x, y) \mid \exists z.E(x, z) \land X^1(z, y)\} =$ direct edges or paths of length 2
  - $\ldots$
  - $R^{\mathfrak{G}'} = \bigcup_{i \in \mathbb{N}} X^i$
- The fixed point here is the least fixed point.

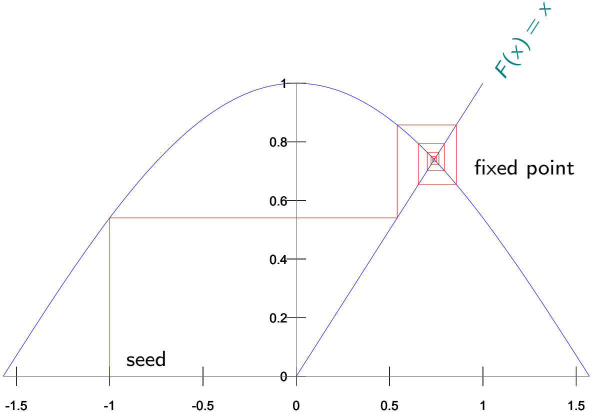
# Constructing Least Fixed Points

- Start with extension $\emptyset$ (seed) and proceed iteratively
- Progress schema: $\emptyset$, $F(\emptyset)$, $F(F(\emptyset))$, $F^3(\emptyset)$, $F^4(\emptyset), \ldots$
- In our example
  - $X^0 =$ seed $= \emptyset$
  - $X^1 = E^{\mathfrak{G}} =$ direct edges
  - $X^2 = F(X^1) = X^1 \cup \{(x, y) \mid \exists z.E(x, z) \wedge X^1(z, y)\} =$ direct edges or paths of length 2
  - $\ldots$
  - $R^{\mathfrak{G}'} = \bigcup_{i \in \mathbb{N}} X^i$

- The fixed point here is the least fixed point.
- Nota bene
  - A fixed point may not exist
  - There may be many fixed points
  - There may not be a least fixed point. (Exercise)

# Fixed Point Construction Graphically

- Fixed point for $F(x) = cos(x)$.
- Attractor



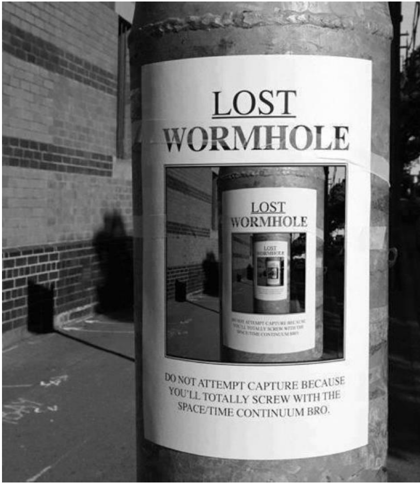
# Recursive Humor

- Wiki entry **Recursive humor** (last access 27 April 2020).

  It is not unusual for such books to include a joke entry in their glossary along the lines of: Recursion, see Recursion.[6]

  [...] An alternative form is the following, from Andrew Plotkin: "If you already know what recursion is, just remember the answer. Otherwise, find someone who is standing closer to Douglas Hofstadter than you are; then ask him or her what recursion is."

  **Lit:** D. Hofstadter. Gödel, Escher, Bach: An Eternal Golden Braid.Vintage Books, 1979.




- Blog Recursively Recursive
  `https://recursivelyrecursive.wordpress.com/category/recursive-humour/page/2/`

# Datalog

- Developed around 1980s
- Renaissance (not only as proof tool but) as industrially applied tool
- EXPTIME-complete in combined complexity; PTIME-complete data complexity
- Simple evaluation strategy for positive fragment (no negation)
- Negation calls for hierarchical evaluation (stratification)
- Different fragments; optimizations ...

# Datalog

- General Logic Programm: Finite set of rules of the form

$$\underbrace{\alpha}_{\textit{head}} \leftarrow \underbrace{\beta_1, \ldots, \beta_n}_{\textit{body}}$$

  - $\alpha$ atomic formula; $\beta_i$ are literals
  - Free variables $\forall$ quantified; comma , read as $\wedge$
  - Fact = rule with empty body.
  - Intensional relation: occurs in some head
  - Extensional relation: not in head (unless rule is a fact)

# Datalog

- General Logic Programm: Finite set of rules of the form

$$\underbrace{\alpha}_{head} \leftarrow \underbrace{\beta_1, \ldots, \beta_n}_{body}$$

  - $\alpha$ atomic formula; $\beta_i$ are literals
  - Free variables $\forall$ quantified; comma , read as $\wedge$
  - Fact = rule with empty body.
  - Intensional relation: occurs in some head
  - Extensional relation: not in head (unless rule is a fact)

- Datalog program = logic program with
  - no function symbols
  - no intensional relation negated in body
  - Sometimes additionally safety constraints:
    - all free variables in head also in body
    - all variables in negated atoms (or arithmetical expressions such as identity) also in non-negated atom in body
  - Semantics for datalog programs: by step-operator used in parallel for intensional relations

# Datalog example: ancestors of Mary

$$
\begin{aligned}
\mathit{ans}(x) &\leftarrow \mathit{ancestor}(x, \mathit{mary}) \\
\mathit{ancestor}(x, y) &\leftarrow \mathit{parentOf}(x, y) \\
\mathit{ancestor}(x, y) &\leftarrow \mathit{parentOf}(x, z), \mathit{ancestor}(z, y)
\end{aligned}
$$

In FOL notation:

$$
\begin{aligned}
\forall x\ \mathit{ancestor}(x, \mathit{mary}) \rightarrow \mathit{ans}(x) \\
\forall x \forall y\ \mathit{parentOf}(x, y) \rightarrow \mathit{ancestor}(x, y) \\
\forall x, \forall y\ (\exists z\ \mathit{parentOf}(x, z), \mathit{ancestor}(z, y)) \rightarrow \mathit{ancestor}(x, y)
\end{aligned}
$$

# SQL 3 Recursion example

```
%Find Mary's ancestors from ParentOf(parent,child)
   WITH RECURSIVE Ancestor(anc,desc) AS
          ( (SELECT parent as anc, child as desc FROM ParentOf)
            UNION
            (SELECT Ancestor.anc, ParentOf.child as desc
             FROM Ancestor, ParentOf
             WHERE Ancestor.desc = ParentOf.parent) )
   SELECT anc FROM Ancestor WHERE desc = "Mary"
```

# FOL with Least Fixed Points

- Datalog extends FOL w.r.t. the semantics (subcutaneous)
- There are different extensions of FOL with fixed point operators available in the syntax

- Example $\exists FO(LFP)$: existential fragment of FOL extended with relation variables and with least fixed point operator $[LFP_{\vec{y},Y}\phi]$

# $\exists FO(LFP)$

- Syntax: $FORM_{\exists FO(LFP)}$ = set of $\exists FO(LFP)$ formulae
  - Every second-order atomic formula is in $FORM_{\exists FO(LFP)}$
  - $\neg\phi$ for $\phi$ an atomic FOL formula
  - $\phi \wedge \psi \in FORM_{\exists FO(LFP)}$
  - $\phi \vee \psi \in FORM_{\exists FO(LFP)}$
  - $\exists x \phi \in FORM_{\exists FO(LFP)}$ (only (existential) quantification over first-order variables)
  - $[LFP_{\vec{x},X}\phi]\vec{t}$
- Semantics
  - ...
  - $\mathfrak{A} \models [LFP_{\vec{x},X}\phi]\vec{t}$ iff
    "For $X$ chosen as least fixed point, $\vec{t}$ fulfills $\phi$ in $\mathfrak{A}$"
  - Restriction: $X$ has to occur positively (i.e. after an even number of $\neg$) in $\phi$
    (Needed to guarantee existence of lfp)

**Theorem**

*Existential fragment of $\exists FO(LFP)$ is equivalent to Datalog.*

# 0-1 law for Datalog

**Theorem**

*Datalog (without negation and ordering) has the 0-1 law.*

- In particular you can not express EVEN

# 0-1 law for Datalog

**Theorem**

*Datalog (without negation and ordering) has the 0-1 law.*

- In particular you can not express EVEN
- (Adding negation allows to express EVEN, which does not fulfill 0-1 law)
- In fact a successor relation together with min- and max-predicates is sufficient.

$$\begin{aligned} odd(x) &\leftarrow min(x) \\ odd(x) &\leftarrow S(x,y), even(y) \\ even(x) &\leftarrow S(y,x), odd(y) \\ EVEN &\leftarrow max(x), even(x) \end{aligned}$$

# What we Did not Cover

Very many FMT topics were not covered in these two lectures, in particular ...

- Proving equivalence of languages (using types)
- Descriptive Complexity
- Algorithmic Model Theory (Infer meta-theorems on algorithmic properties by constraining some input parameters (parameterized complexity) )

# Descriptive Complexity

- There is a close relationship between complexity classes and logics (queries expressible in a logic)
- Hints to astonishing correspondences between prima facie two different worlds
- The world of representation (what?) and of calculation (how?)
- Results talk about data complexity (!)
- Results mainly for ordered structures

# Fagin lays the foundations

- One of the first insights which founded descriptive complexity goes back to Fagin

**Theorem ($SO\exists$ captures NPTIME)**

*Existential second order logic ($SO\exists$) captures the class of problems verifiable in polynomial time (NP)*

$SO\exists$ *= second order logic where second order quantifiers are restricted to* $\exists$

**Definition**

A logic $\mathcal{L}$ captures a complexity class $\mathcal{C}$ iff for all $\sigma$ with $<\in \sigma$ and classes of structures $K \subseteq STRUC(\sigma)$:

$$K \in \mathcal{C} \text{ iff } K \text{ is axiomatizable in } \mathcal{L}$$

# If you want to become famous ....

- ... prove or disprove the following:

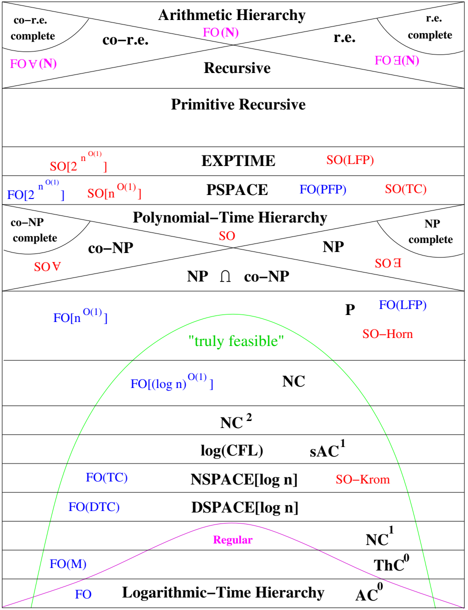
  There is a “natural” logic characterizing *PTIME* over non-ordered finite structures.

# If you want to become famous ....

- ... prove or disprove the following:

  There is a "natural" logic characterizing *PTIME* over non-ordered finite structures.

- If you can show there is no such logic, then $NP \neq PTIME$



The Descriptive World

N.Immerman: Descriptive Complexity, ACM SIGACT NEWS, vol. 34, no. 3, 2003, p.5)