UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Finite Model Theory

Lecture 4: Locality, 1-0, Fixed Points
8 November, 2017

Foundations of Ontologies and Databases for Information Systems
CS5130 (Winter 17/18)

Recap of Lecture 3

What the lecturer happens to see in the audience—sometimes

https://www.youtube.com/watch?v=IQgAuBhlBT0
Owl video

- 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. combine 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
- Observation
  - Consider a binary query $Q : STRUCT(\sigma) \longrightarrow STRUC(ans)$ to be defined in FOL
  - So, we need a formula ϕ_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 ϕ_Q.
- Different (comparable) locality notions
  - Bounded number of degrees property (BNDP)
  - Gaifman locality
  - Hanf locality

Proving Inexpressibility by Locality

- FOL has a fundamental property: locality
- Observation
  - Consider a binary query $Q : STRUCT(\sigma) \longrightarrow STRUC(ans)$ to be defined in FOL
  - So, we need a formula ϕ_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 ϕ_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

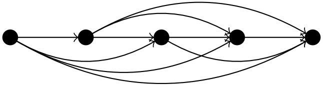
Every FOL query has the BNDP.

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 σ 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 a, b are among those elements and $a_1, \ldots, a_{j-1}$ 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 $\bar{a}$

It's (sometimes) sufficient to Consider Graphs Only

Definition (Gaifman Graph)

For any σ 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 a, b are among those elements and $a_1, \ldots, a_{j-1}$ 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)
- $B_r^{\mathfrak{G}}(\overline{a}) = \{b \in G \mid d(\overline{a}, b) \leq r\}$ (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 $(G, E, \overline{a})$

Gaifman Locality

- $\overline{a} = (a_1, \dots, a_n) \in G^n$ (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 $\vec{a}$ have the same names as in $(G, E, \overline{a})$ (say $c_1, \ldots c_n$) and the 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

- $\overline{a} = (a_1, \dots, a_n) \in G^n$ (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 $(G, E, \overline{a})$

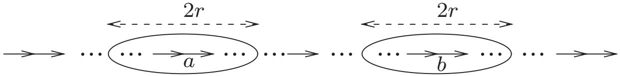
Definition

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 $(b, a) \notin TC(\mathfrak{G})$, $\lightning$

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 $a \in A$:
  $N_r^{\mathfrak{G}}(a) \simeq N_r^{\mathfrak{G}'}(f(a))$

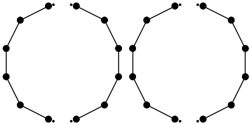
Definition (Hanf locality (formal))

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 $Q(\mathfrak{G}) = Q(\mathfrak{G}')$.

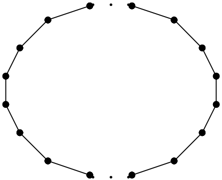
Theorem

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.

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 σ is relational!!
- $STRUC(\sigma, n)$: structures with domain $[n] := \{0, 1, \ldots, n-1\}$ over σ.
- 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.

Let me add a "footnote" on the general strategy of using the 0-1 law ☺

`https://www.youtube.com/watch?v=jWinRjU7Kb0`

Locally stored video

Let me add a “footnote” on the general strategy of using the 0-1 law ☺

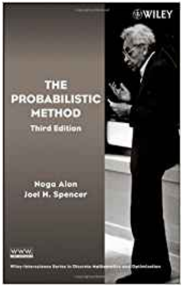
https://www.youtube.com/watch?v=jWinRjU7Kb0

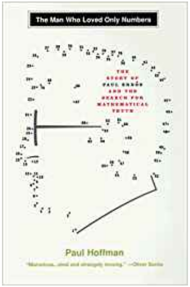
Locally stored video

OK, now the real footnote—on the next slide

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, Aggregation

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

Example (Departments with Average 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) > 1000
```

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 ϕ.
- $\sharp\overline{x}.\phi(\overline{x})$: 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 Us and there are no more than j Us"

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 $s_k(\{d_1, \ldots, d_k\}) = \Sigma_{i=1}^{k} d_i$

Definition (FOL-Aggr)

FOL-Aggr same as FOL+AllCnt but with aggregate terms (and $\mathbb{Q}$ instead of $\mathbb{N}$).

- Syntax: Terms $t(\overline{x})$ of the form $Aggr_{\mathcal{F}}\overline{y}.(\phi(\overline{x}, \overline{y}), t'(\overline{x}, \overline{y}))$
  - Not eh 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 ϕ is a formula, then so are $\exists X \phi$, $\forall X \phi$

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

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

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

- 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

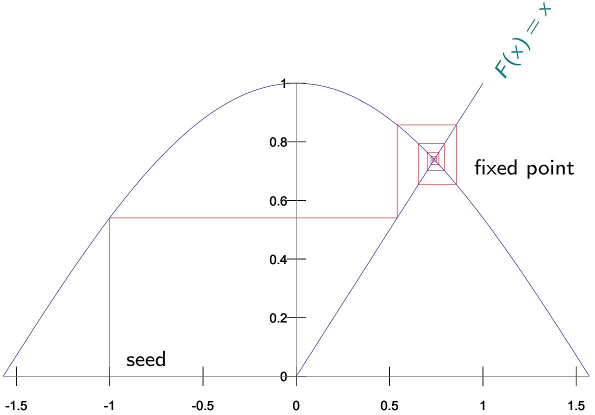
- Start with extension $\emptyset$ (seed) and proceed iteratively
- Progress schema: $\emptyset$, $F(\emptyset)$, $F(F(\emptyset))$, $F^3(\emptyset)$, $F^4(\emptyset), \ldots$
- In our example
  - $X^0 = \text{seed} = \emptyset$
  - $X^1 = E^{\mathfrak{G}} = \text{direct edges}$
  - $X^2 = F(X^1) = X^1 \cup \{(x, y) \mid \exists z. E(x, z) \wedge X^1(z, y)\} =$ 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)

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
  - $X^0 = \text{seed} = \emptyset$
  - $X^1 = E^{\mathfrak{G}} = \text{direct edges}$
  - $X^2 = F(X^1) = X^1 \cup \{(x,y) \mid \exists z.E(x,z) \wedge X^1(z,y)\} =$ 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

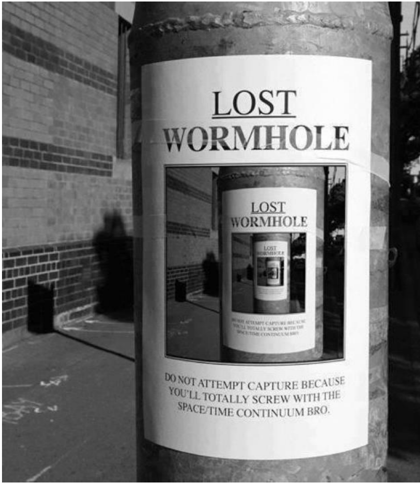


Recursive Humor

- Wiki entry **Recursive humor**.
  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

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

  - α atomic formula; β_i are literals
  - Free variables $\forall$ quantified; comma , read as $\wedge$
  - Intensional relation: Relation symbol occurring in some head
  - Extensional relation: occurring only in body

- 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

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

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

  - α atomic formula; β_i are literals
  - Free variables $\forall$ quantified; comma , read as $\wedge$
  - Intensional relation: Relation symbol occurring in some head
  - Extensional relation: occurring only in body

- 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

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

In FOL notation:

$$\forall x\ ancestor(x, mary) \rightarrow ans(x)$$
$$\forall x \forall y\ parentOf(x, y) \rightarrow ancestor(x, y)$$
$$\forall x, \forall y\ (\exists z\ parentOf(x, z), ancestor(z, y)) \rightarrow ancestor(x, y)$$

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 (subkutane)
- 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 ϕ 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 ϕ in $\mathfrak{A}$"
  - Restriction: X has to occur positively (i.e. after an even number of $\neg$ in ϕ)
    (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
- (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.

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

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 ...

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

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)

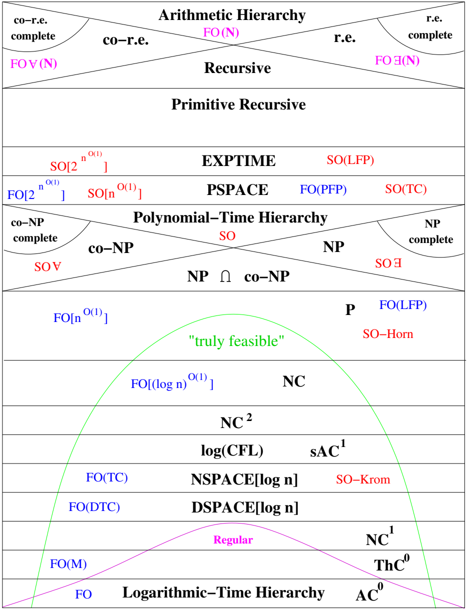
Existential second order logic ($SO\exists$) captures the class of problems verifiable in polynomial time (NP)

$SO\exists$ = *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 σ with $<\in \sigma$ and classes of structures $K \subseteq STRUC(\sigma)$:

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



The Descriptive World

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

Solutions to Exercise 3 (12 Points)

Ad Exercise 3.1 (4 Points)

Give at least two aspects of real DBs for which the approach of identifying DBs with finite FOL structures is not sufficient or adequate.

- DBs may have NULL values (but structures are not incomplete). So one has to argue with completions of DBs. (See lecture on data exchange)
- Domain of the structure corresponding to a DB is not explicitly specified
  - Natural (as in FOL) vs. active domain semantics (consider only those constants occurring in a DB as potential element of the domain)
  - Safety considerations needed for FOL (not the case for relational calculus/SQL)
- One can show: FOL under active domain semantics the same as SQL
- Nonetheless: It means dependency on domain.

Ad Exercise 3.2 (4 Points)

Argue why the usual restriction in FMT to consider only relational structures (i.e., no function symbols allowed) is not problematic. That is more formally: How can formulae with function symbols be represented by formulae containing only relation symbols (in particular the identity relation)?

- For every n-ary functional symbol f introduce $n+1$-ary relation symbol R_f and state that R_f is a function:

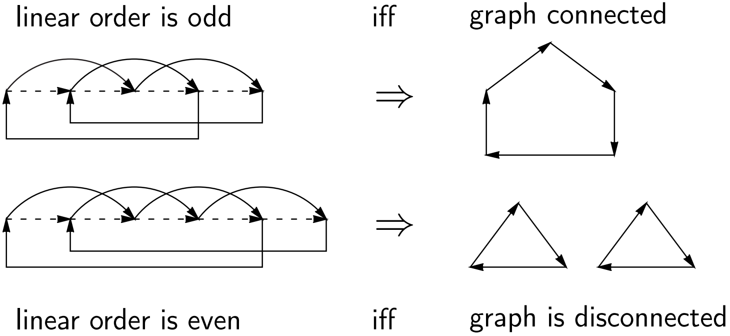
$$\forall x_1, \ldots \forall x_{n-1} \exists y_1 R_f(x_1, \ldots, x_n, y_1) \wedge$$
$$\forall y_1, y_2 R(x_1, \ldots, x_n, y_1) \wedge R(x_1, \ldots, x_n, y_2) \rightarrow y_1 = y_2$$

- Then recursively eliminate all terms by substituting atoms of the form
  - $f(\vec{t_1}) = t_2$ with $R_f(\vec{t_1}, t_2)$
  - $S(f_1(\vec{t_1}), \vec{t_2}, \ldots, \vec{t_n})$ with $\exists x S(x, \vec{t_2}, \ldots, \vec{t_n}) \wedge R_f(t_1, x)$

  and so on.

Ad Exercise 3.3 (4 Points): Reduce EVEN($<$) to Graph Connectivity

Formalize the reduction query $Q_{red} : LinOrd \rightarrow GRAPH$ from linear orders to graphs by describing a query formula inducing Q_{red}.

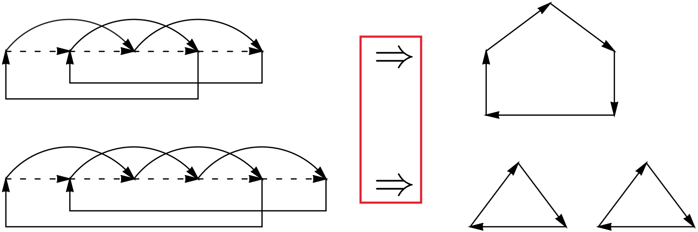


- Construction of graph from linear order expressible as an FOL query $Q_{red} : LinOrd \longrightarrow GRAPH$

Ad Exercise 3.3 (4 Points): Reduce EVEN(<) to Graph Connectivity

Formalize the reduction query $Q_{red} : LinOrd \to GRAPH$ from linear orders to graphs by describing a query formula inducing Q_{red}.

linear order is odd iff graph connected



linear order is even iff graph is disconnected

- Construction of graph from linear order expressible as an FOL query $Q_{red} : LinOrd \longrightarrow GRAPH$

Ad Exercise 3.3 (4 Points)

- Helper formulae
  - $succ(x, y) : x < y \wedge \neg\exists z.x < z \wedge z < y$
  - $last(x) : \neg\exists z.x < z$
  - $first(x) : \neg\exists z.z < x$
- Define Q_{red}: $LinOrd \longrightarrow GRAPH$ as

$$
\begin{aligned}
E(x, y) \quad = \quad & \psi(x, y) = \\
& (\exists z(succ(x, z) \wedge succ(z, y))) \vee \\
& (last(x) \wedge \exists z(first(z) \wedge succ(z, y))) \vee \\
& (\exists z(last(z) \wedge succ(x, z) \wedge first(y)))
\end{aligned}
$$

- Assume that CONN is expressible as FOL query ϕ_{conn} over signature $\{E\}$ for graphs.
- Then EVEN($<$) would be FOL expressible as:
  $\phi_{conn}[E/\psi]$ ↯
  (Note: $\phi_{conn}[E/\psi]$ is shorthand for replacing every occurrence of atom $E(u, w)$ by formula $\psi(u, w)$ in ϕ_{conn}.)

Exercise 4 (16 Points)

Exercise 4.1 (6 Points)

Use Hanf locality in order to proof that the following boolean queries are not FOL-definable.

1. Is a given graph acyclic?
2. Is a given graph a tree?

.

Exercise 4.2 (4 Points)

Show that $EVEN(\sigma)$ can be defined within second-order logic **for any** σ.

Hint: formalize "There is a binary relation which is an equivalence relation having only equivalence classes with exactly two elements" and argue why this shows the axiomatizability.

Exercise 4.3 (2 Points)

Argue why (in particular within the DB community) one imposes safety conditions for Datalog rules.

Exercise 4.4 (4 points)

Give examples of general program rules for which

1. No fixed point exists at all (Hint: “This sentence is not true”)
2. Has two minimal fixed points (Hint: “The following sentence is false. The previous sentence is false.”)