UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Finite Model Theory

Lecture 3: Motivation, Games, Reduction Tricks
1 November, 2017

Foundations of Ontologies and Databases
for Information Systems
CS5130 (Winter 2017)

Recap of Lecture 2: FOL

FOL as a Representation Language

- FOL provides expressive language with neat semantics to represent assertions relevant for CS
  - System descriptions
  - Desired requirements
  - System behavior description
  - Domain constraints

  - See Exercise 2.1 and Exercise 2.2

Solving Algorithmic Problems in FOL

- Definitions for important semantical properties (satisfaction, satisfiability, entailment) do not tell how to compute them

- Proof calculi to the rescue
- Various FOL calculi exist that have desired properties of being correct and complete
- Prominent ones that are "directed" and hence well implementable: Tableaux and Resolution

- Resolution calculi
  - Refutation calculus (un-satisfiability tester)
  - Data structure: Formula in Clausal Normal Form (see Exercise 2.3)
  - Resolution rule:

$$(A \vee \neg B) \wedge (B \vee C) \vDash_{res} A \vee C$$

Solving Algorithmic Problems in FOL

- No decidability for validity (unsatisfiability, entailment) but semi-decidability
- Hence we will have to consider different variants of FOL
- Undecidability stays when changing to finite model semantics
  It becomes even worse

Theorem (Trakhtenbrot)

Validity of FOL sentences under finite model semantics is not semi-decidable

- Nonetheless FOL has important role (for CS)
  - FOL "open" (has parameters) for restrictions to more feasible fragments: number of variables, predicates, arity of predicates, complex formulae construction, quantifier nesting, quantifier alternation etc.
  - FOL (per se) is useful as a query language on DBs: constant time in data complexity ($\Longrightarrow$ to be discussed today)

End of Recap

Literature Hints

- **Lit:** L. Libkin. The finite model theory toolbox of a database theoretician. In PODS '09: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 65–76, New York, NY, USA, 2009. ACM.
- **Lit:** L. Libkin. Elements Of Finite Model Theory (Texts in Theoretical Computer Science. An Eatcs Series). SpringerVerlag, 2004.
- **Lit:** H. Ebbinghaus and J. Flum. Finite Model Theory. Perspectives in mathematical logic. Springer, 1999.

Aim

Understand: "Finite Model Theory (FMT) is the backbone of database theory"

Finite Model Theory

- Fundamental ideas
  1. Consider DBs as finite FOL structures
  2. Consider FOL as query language over DBs
- Starting with FOL investigate all relevant (algorithmic) problems with finite structure semantics
- These ideas make up an approximative but nonetheless very fruitful theoretical approach to studying DB related problems
  - Showing expressivity bounds for query languages
  - Showing equivalence of DB query languages
  - Showing the inherent complexity of DB query languages

Finite Model Theory

- Fundamental ideas
  1. Consider DBs as finite FOL structures
  2. Consider FOL as query language over DBs
- Starting with FOL investigate all relevant (algorithmic) problems with finite structure semantics
- These ideas make up an approximative but nonetheless very fruitful theoretical approach to studying DB related problems
  - Showing expressivity bounds for query languages
  - Showing equivalence of DB query languages
  - Showing the inherent complexity of DB query languages

FOL as a Query Language

- FOL query formula $\phi(\vec{x})$ (for $\vec{x} = x_1 \ldots, x_n$) over signature σ
  - $\vec{x}$ = distinguished variables, answer variables.

Definition (Answers of a query on a structure)

$$\begin{aligned} ans(\phi(\vec{x}), \mathfrak{A}) &= \mathfrak{A}^{\phi(\vec{x})} \\ &= \{ \vec{d} = (d_1, \ldots, d_n) \mid d_i \in A \text{ and } \mathfrak{A} \models \phi(\vec{x}/\vec{d}) \} \end{aligned}$$

- Set of answers can be considered as a structure with n-ary predicate *ans*
- n-ary query induced by ϕ:

$$Q_\phi : STRUCT(\sigma) \longrightarrow STRUCT(ans)$$

Boolean Queries

- Boolean FOL query formula = FOL formulae without free variables (also called sentences)

- According to definition possible answers are $\{()\}$ (stands for true) and $\emptyset$ (false)

- Boolean queries can be identified with the class of σ structures making them true

Answering (Boolean) FOL queries

- Why is FOL so successful in DB theory?
- E.g., is model checking problem ($\mathfrak{A} \models \phi$) feasible?

- Answer is NO if considering $\mathfrak{A}, \phi$ both as inputs
  $\Longrightarrow$ Combined complexity

Theorem (Stockmeyer 74, Vardi 82)

Model-checking for FOL (and monadic second-order logic MSO) is PSPACE complete.

Lit: L. J. Stockmeyer. The complexity of decision problems in automata theory and logic. PhD thesis, MIT, 1974.

Lit: M. Y. Vardi. The complexity of relational query languages (extended abstract). In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC '82, pages 137–146, New York, NY, USA, 1982. ACM.

Answering (Boolean) FOL queries

- Why is FOL so successful in DB theory?
- E.g., is model checking problem ($\mathfrak{A} \models \phi$) feasible?

- Answer is NO if considering $\mathfrak{A}, \phi$ both as inputs
  $\Longrightarrow$ Combined complexity

Theorem (Stockmeyer 74, Vardi 82)

Model-checking for FOL (and monadic second-order logic MSO) is PSPACE complete.

Lit: L. J. Stockmeyer. The complexity of decision problems in automata theory and logic. PhD thesis, MIT, 1974.

Lit: M. Y. Vardi. The complexity of relational query languages (extended abstract). In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC '82, pages 137–146, New York, NY, USA, 1982. ACM.

Reminder: Complexity Classes

- Encode algorithmic problem Π as a language $\Pi \subseteq \Sigma^*$, i.e., as set of words over an alphabet Σ.
- Decision problem: $w \in \Pi$?

- Example complexity classes
  - PTIME = Problems solvable in polynomial time (w.r.t. the input size) by a deterministic Turing machine
  - PSPACE = Problems solvable in polynomial space (w.r.t. the input size) by a deterministic Turing machine

- Mostly, as computer scientist, you do not refer directly to TMs for getting complexity results
- Instead you (should) train yourself in the art of reducing and learning paradigmatic problems in complexity classes.

Lit: Complexity Zoo: `https://complexityzoo.uwaterloo.ca/Complexity_Zoo`

Lit: M. R. Garey and D. S. Johnson. Computers and Intractability; A Guide to the Theory of NP- Completeness. W. H. Freeman & Co., New York, NY, USA, 1990.

Reminder: Complexity Classes

- Encode algorithmic problem Π as a language $\Pi \subseteq \Sigma^*$, i.e., as set of words over an alphabet Σ.
- Decision problem: $w \in \Pi$?

- Example complexity classes
  - PTIME = Problems solvable in polynomial time (w.r.t. the input size) by a deterministic Turing machine
  - PSPACE = Problems solvable in polynomial space (w.r.t. the input size) by a deterministic Turing machine

- Mostly, as computer scientist, you do not refer directly to TMs for getting complexity results
- Instead you (should) train yourself in the art of reducing and learning paradigmatic problems in complexity classes.

Lit: Complexity Zoo: https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Lit: M. R. Garey and D. S. Johnson. Computers and Intractability; A Guide to the Theory of NP- Completeness. W. H. Freeman & Co., New York, NY, USA, 1990.

Complete Problems

- "Paradigmatic" problems in a complexity class $\mathcal{C}$ called $\mathcal{C}$-complete problems
- $\mathcal{C}$ complete problems (w.r.t. $\mathcal{C}'$ reductions)= Most difficult problems in $\mathcal{C}$ = $\{\Pi \mid \Pi \in \mathcal{C}$ and all other $\mathcal{C}$ problems are $\mathcal{C}'$-reducible to $\Pi\ \}$
- Problem $\Pi \subseteq \Sigma$ is $\mathcal{C}$-reducible to problem $\Pi' \subseteq \Sigma'$, for short: $\Pi \leq_C \Pi'$, iff there is a $\mathcal{C}$-computable function such that for all $w \in \Sigma$: $w \in \Pi$ iff $f(w) \in \Pi'$

Example for PSPACE Complete Problem

- Quantified Boolean Formula (QBF)
  - All propositional symbols p_i are QBF
  - All boolean combinations of QBFs are QBFs
  - If ϕ is a QBF, then so are$\forall p \phi$ and $\exists p \phi$.
  - Semantics: Structures here are truth value assignments

- Theorem: Satisfiability of QBFs is PSPACE complete

Example

- $\exists p \exists q \; p \wedge q$ is satisfiable, because
  there is assignment $\nu(q) = 1$ and $\nu(q) = 1$ making $p \wedge q$ true.
- $\exists p \; p \wedge \neg p$ is not satisfiable

FOL is in PSPACE

Complexity estimation for query answering

Time complexity for checking $\mathfrak{A} \models \phi$ is $O(n^k)$, where

- n = size of input structure $\mathfrak{A}$ and
- k = size of input query ϕ

Note: Size of query k is responsible for exponential blow up

Reminder (Landau-Notation)

- $f \in O(g)$ means: f has function g as upper bound
- Formally: There are constants $c > 0$ and x_0, s.t. for all $x > x_0$:

$$|f(x)| \leq c * |g(x)|$$

FOL is in PSPACE

Complexity estimation for query answering

Time complexity for checking $\mathfrak{A} \models \phi$ is $O(n^k)$, where

- n = size of input structure $\mathfrak{A}$ and
- k = size of input query ϕ

- **Note**: Size of query k is responsible for exponential blow up

- Naive recursive algorithm showing time complexity $(O(n^k))$ and space complexity $(O(k * log(n)))$
  - Atomic formula: Look up in structure
  - Boolean cases: apply semantics of Boolean connectors
  - $\exists x \phi(x)$: Check for all $d \in A$ whether $\mathfrak{A} \models \phi(x/d)$

- PSPACE hardness by reducing QBF satisfiability to FOL model checking

FOL is in AC^0 in Data Complexity

- In practical scenarios DB size n much bigger than query size k
- Therefore: Consider only DB as input; query fixed
  $\Longrightarrow$ data complexity

- This helps a lot, as only query size responsible for exponential complexity, indeed:

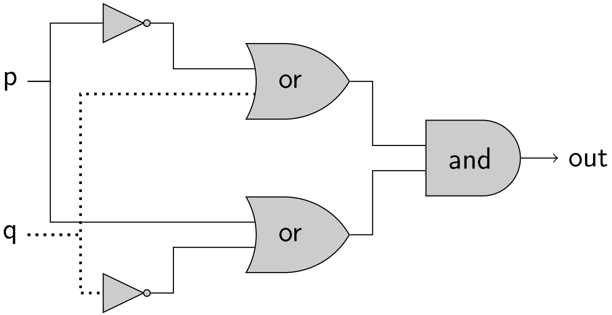
Theorem

Data complexity for FOL query answering is in LOGSPACE and even in AC^0.

- LOGSPACE = Problems solvable in logarithmic space on the read-write tape by a deterministic 2-tape Turing machine
- $AC^0 \subsetneq LOGSPACE$.

The Class AC^0

- Intuitively AC^0 = class of problems solvable in constant time on polynomially many process (in parallel)
- Formally, AC^0 is defined using a computation model based on boolean circuits



Boolean circuit above computes $(\neg p \vee q) \wedge (p \wedge \neg q)$

The class AC^0

- Encode problems as $0/1$ vector inputs
- Computability by circuits: There is family of circuits (for every possible size of input) computing desired boolean function

- In many cases: uniformity condition: family not arbitrarily constructed but computable as output of single TM

Definition

$AC^0 =$ Problems solvable by families of circuits with

- constant depth,
- polynomial size and
- using NOT gates, unlimited-fanin AND gates and OR gates.

FOL is in AC^0 data complexity

Proof idea

- Query modelled as boolean circuit family for every possible instance of given DB schema $\mathcal{R}$ and super-domain Dom
- Every ground atom $R(d_1, \ldots, d_n)$ is represented as propositional input symbol
- Gates for every subexpression of query
- Boolean operators in subexpression modelled by corresponding boolean gates
- $\exists$ ($\forall$) quantifier modelled by unbounded fan-in OR (AND) gate

Lit: S. Abiteboul, R. Hull, and V. Vianu. Foundations of Databases. 1995.

Proving Expressivity Bounds for FOL

Expressivity of Languages

- We defined above the query Q_ϕ induced by a formula ϕ (syntax $\rightarrow$ semantics direction)

- For expressivity considerations one goes the other way round (semantics $\rightarrow$ syntax):
  - Given a query

$$Q : STRUC(\sigma) \longrightarrow STRUC(ans)$$

    test whether there is formula ϕ in the given logic s.t. $Q = Q_\phi$
  - In this case one says that Q is definable in the logic (for the given set of structures $STRUC(\sigma)$)

- For Boolean queries definability amounts to:
  Given a class $X \subseteq STRUC(\sigma)$ of structures over a signature σ: there is a sentence ϕ (over the given logic) s.t. $Mod(\phi) = X$

Expressivity of Languages

- We defined above the query Q_ϕ induced by a formula ϕ (syntax $\rightarrow$ semantics direction)

- For expressivity considerations one goes the other way round (semantics $\rightarrow$ syntax):
  - Given a query

    $$Q : STRUC(\sigma) \longrightarrow STRUC(ans)$$

    test whether there is formula ϕ in the given logic s.t. $Q = Q_\phi$
  - In this case one says that Q is definable in the logic (for the given set of structures $STRUC(\sigma)$)

- For Boolean queries definability amounts to:
  Given a class $X \subseteq STRUC(\sigma)$ of structures over a signature σ: there is a sentence ϕ (over the given logic) s.t. $Mod(\phi) = X$

Expressivity of Languages

- We defined above the query Q_ϕ induced by a formula ϕ (syntax $\rightarrow$ semantics direction)

- For expressivity considerations one goes the other way round (semantics $\rightarrow$ syntax):
  - Given a query

    $$Q : STRUC(\sigma) \longrightarrow STRUC(ans)$$

    test whether there is formula ϕ in the given logic s.t. $Q = Q_\phi$
  - In this case one says that Q is definable in the logic (for the given set of structures $STRUC(\sigma)$)

- For Boolean queries definability amounts to:
  Given a class $X \subseteq STRUC(\sigma)$ of structures over a signature σ: there is a sentence ϕ (over the given logic) s.t. $Mod(\phi) = X$

A Very General Notion of Query

- During the discussion of the reduction of LinORD to CONN we discussed a very general notion of a FOL query. Here is the exact definition. (See Immerman: Descriptive Complexity, p. 18)

Definition

Let τ, σ be any two signatures with $\tau = (R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s)$ and k be a fixed natural number. A k-ary first order query $Q : STRUCT(\sigma) \longrightarrow STRUCT(\tau)$ is given by an r+s+1-tuple of σ-formulae $\phi_0, \phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_s$. For each σ structure $\mathfrak{A} \in STRUC(\sigma)$ the formulae describe a τ structure $Q(\mathfrak{A})$

$$Q(\mathfrak{A}) = (\ dom(Q(\mathfrak{A})), R_1^{Q(\mathfrak{A})}, \ldots, R_r^{Q(\mathfrak{A})}, c_1^{Q(\mathfrak{A})}, \ldots c_s^{Q(\mathfrak{A})}\ )$$

with

- $dom(Q(\mathfrak{A})) = \{(b^1, \ldots, b^k) \mid \mathfrak{A} \models \phi_0(b^1, \ldots, b^k)\}$
- $R_i^{Q(\mathfrak{A})} = \{(b_1^1, \ldots, b_1^k), \ldots, (b_i^1, \ldots, b_i^k) \in dom(Q(\mathfrak{A}))^{a_i} \mid \mathfrak{A} \models \phi_i(b_1^1, \ldots, b_{a_i}^k)\}$
- $c_j^{Q(\mathfrak{A})} =$ the unique $(b^1, \ldots, b^k) \in dom(Q(\mathfrak{A}))$ s.t. $\mathfrak{A} \models \psi_j(b^1, \ldots, b^k)$

Example: Reductio of linear order to connectivity
$Q_{red} : LinOrd \rightarrow CONN$

- $\tau = E$, $\sigma =<$, $r = 1$, $s = 0$
- $k = 1$, $\phi_0 =$ an arbitrary tautology
- $\phi_1 =$ see Exercise 3.3

Need for New Proof Techniques

- Main classical techniques used for classical FOL do not work
- Because corresponding theorems do not hold for FMT

- Reminder: Main properties of FOL
  - Compactness (Comp)
  - Löwenheim-Skolem (Lösko)
- These properties characterize FOL for arbitrary structures: Lindström theorems

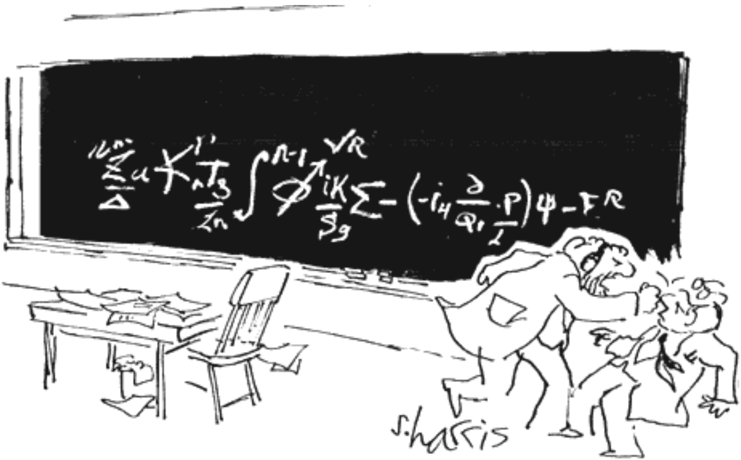
Finite Compactness Pendant?

Fin-Comp

If every finite subset of Φ has a finite model, then Φ has a finite model.

- The finite version of compactness (Fin-Comp) does not hold for FOL.
- Falsifier
  - $\lambda_n := \exists x_1, \ldots, x_n \bigwedge_{i \neq j} \neg(x_i = x_j)$
    (says: "There are at least n elements")
  - $\{\lambda_n \mid n \in \mathbb{N}\}$ has not finite model though every subset has (Compare Exercise 2.2.3)

What's the Right "Proof Technique"?



"You want proof? I'll give you proof!"

We Prefer to Proof/Argue ... without Being a Poser ...

Pinguin Video

URL: `https://www.youtube.com/watch?v=7iDn5d9q9Y8`

Convention for the Following

Assume all structures are relational, i.e., there are no function symbols other than constants—unless stated otherwise

Games

Games as Essence of Being a Human

"Der Mensch spielt nur, wo er in voller Bedeutung des Wortes Mensch ist, und er ist nur da ganz Mensch, wo er spielt"

(F. Schiller, Briefe über die ästhetische Erziehung des Menschen (1795))

Games as a CS Tool

- In logic, Fraïssé games are an important proof tool
- Different variations (w.r.t. rules, winning strategies)
- We will consider a basic game type and show how to use it.

- But: games have high "cognitive complexity" even for non-trivial problems
- Therefore: Use games for simple but generic problems and reduce others to these

Ehrenfeucht-Fraïssé Games

- **Notation:** $G_n(\mathfrak{A}, \mathfrak{B})$
  n-round game played for structures on same signature
- **Input:** structures $\mathfrak{A}, \mathfrak{B}$
- **Players:** spoiler and duplicator
- **Output:** a function relating elements from $\mathfrak{A}, \mathfrak{B}$
- **Rules:** see next slide
- Spoiler's aim: show $\mathfrak{A}, \mathfrak{B}$ are "different"
- Duplicator's aim: show $\mathfrak{A}, \mathfrak{B}$ are "same"

Rules of the Game

- In turn, spoiler choose structure and element i in it and
- duplicator chooses other structure and element in it

- After n rounds: n elements $a_1, \ldots, a_n$ from $\mathfrak{A}$ and n elements $b_1, \ldots, b_n$ from $\mathfrak{B}$ are chosen.

Winning condition

Duplicator wins iff

$(a_1, \ldots, a_n)$ plays in $\mathfrak{A}$ the same role as $(b_1, \ldots, b_n)$ in $\mathfrak{B}$

Partial Isomorphism

Formalize sameness of tuples' roles by notion of partial isomorphism

Definition (Partial Isomorphism)

For structures $\mathfrak{A}$, $\mathfrak{B}$ over signature σ, let $f : A \longrightarrow B$ a function with domain $dom(f)$. f is called a partial isomorphism iff

- f is injective
- For every constant c: $c^{\mathfrak{A}} \in dom(f)$ and $f(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$
- For all relation symbols R (including identity) and $a_1, \ldots, a_n \in dom(f)$
  $R^{\mathfrak{A}}(a_1, \ldots, a_n)$ iff $R^{\mathfrak{B}}(f(a_1), \ldots, f(a_n))$

If f is total and bijective, then f is called an isomorphism, and $\mathfrak{A}, \mathfrak{B}$ are said to be isomorphic, for short $\mathfrak{A} \simeq \mathfrak{B}$

Winning Condition Formalized

- After n rounds: n elements $a_1, \ldots, a_n$ from $\mathfrak{A}$ and n elements $b_1, \ldots, b_n$ from $\mathfrak{B}$ are chosen.
- Winning condition
  Duplicator wins iff
  $f : a_i \mapsto b_i$ is a partial isomorphism of $\mathfrak{A}$ and $\mathfrak{B}$.
- Game equivalence
  $\mathfrak{A} \sim_{G_n} \mathfrak{B}$ iff: Duplicator has a winning strategy in $G_n(\mathfrak{A}, \mathfrak{B})$
  ($\mathfrak{A}$ and $\mathfrak{B}$ are the same w.r.t. n-round games)

Quantifier Rank

- How do we use games for proving in-expressivity?
- We need two more technical notions
  1. to capture nesting depth of quantifiers
  2. to capture property that two structures model the same sentences (up to some syntactical complexity)

Quantifier Rank

- How do we use games for proving in-expressivity?
- We need two more technical notions
  1. to capture nesting depth of quantifiers
  2. to capture property that two structures model the same sentences (up to some syntactical complexity)

Definition (Quantifier Rank $qr(\phi)$)

- $qr(\phi) = 0$ for atoms ϕ
- $qr(\phi \vee \psi) = qr(\phi \wedge \psi) = qr(\phi \rightarrow \psi) = max\{qr(\phi), qr(\psi)\}$
- $qr(\neg\phi) = qr(\phi)$
- $qr(\exists x\ \phi) = qr(\forall x\ \phi) = qr(\phi) + 1$
- Example: $qr(\ \forall x[\exists w(P(x, w)) \wedge \exists y \exists z R(x, y, z)]\ ) = 3$

Quantifier Rank

- How do we use games for proving in-expressivity?
- We need two more technical notions
  1. to capture nesting depth of quantifiers
  2. to capture property that two structures model the same sentences (up to some syntactical complexity)

Definition (Equivalence Up to Rank n)

$\mathfrak{A} \equiv_n \mathfrak{B}$ iff $\mathfrak{A}$ and $\mathfrak{B}$ agree on all FOL sentences of quantifier rank up to n.

How to Use Games?

Theorem

$\mathfrak{A} \sim_{G_n} \mathfrak{B}$ *iff* $\mathfrak{A} \equiv_n \mathfrak{B}$

This gives a non-FOL-expressibility tool

- Aim: Show Q not expressible in FOL
- Construct families of structure $\mathfrak{A}_n, \mathfrak{B}_n$ s.t.
  1. All $\mathfrak{A}_n$ satisfy Q
  2. No $\mathfrak{B}_n$ satisfies Q
  3. $\mathfrak{A}_n \sim_{G_n} \mathfrak{B}_n$
- Assume Q expressible as FOL formula ϕ of quantifier rank n. Then $\mathfrak{A}_n \models \phi$ and $\mathfrak{B}_n \models \neg\phi$, but $\mathfrak{A}_n \sim_{G_n} \mathfrak{B}_n$ $\lightning$

Example: Inexpressibility of EVEN

- *EVEN*(σ): structures over signature σ with domain of even cardinality
- The signature is relevant for the proofs
- Simpel case: $\sigma = \emptyset \quad \Longrightarrow \quad$ structures are sets

Proposition

EVEN$(\emptyset)$ *is not expressible in FOL*

Proof

- Choose $\mathfrak{A}_n$ as $2n$-element set, $\mathfrak{B}_n$ as $2n+1$-element set.
- $\mathfrak{A}_n \in EVEN(\emptyset)$ and $\mathfrak{B}_n \notin EVEN(\emptyset)$
- $\mathfrak{A}_n \sim_{G_n} \mathfrak{B}_n$: Duplicator plays already played element in the other set iff spoiler does

Example: Inexpressibility of EVEN

- *EVEN*(σ): structures over signature σ with domain of even cardinality
- The signature is relevant for the proofs
- Simpel case: $\sigma = \emptyset \quad \Longrightarrow \quad$ structures are sets

Proposition

EVEN($\emptyset$) *is not expressible in FOL*

Proof

- Choose $\mathfrak{A}_n$ as $2n$-element set, $\mathfrak{B}_n$ as $2n+1$-element set.
- $\mathfrak{A}_n \in EVEN(\emptyset)$ and $\mathfrak{B}_n \notin EVEN(\emptyset)$
- $\mathfrak{A}_n \sim_{G_n} \mathfrak{B}_n$: Duplicator plays already played element in the other set iff spoiler does

Inexpressibility of EVEN(σ) with Games

- What about EVEN(σ) for non-empty σ?
- Consider: $\sigma = \{<\}$ and class of structures = linear orders
- L_n: n-element total ordering on some set

Theorem

For every $m, k \geq 2^n$: $L_m \sim_{G_n} L_k$.

- In particular $EVEN(<)$ not expressible over linear orders: take $\mathfrak{A}_n = L_{2^n}$, $\mathfrak{B}_n = L_{2^n+1}$.

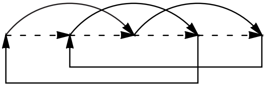
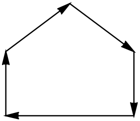
Proving Inexpressivity: Reduction Tricks (not Tools)

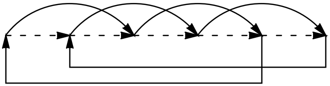
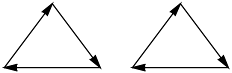
- Showing FOL inexpressibility of
  - graph connectivity CONN
  - acyclicity ACYCL
  - transitive closure TC

  by reduction of EVEN($<$) to each of them

Reduce EVEN($<$) to Graph Connectivity

linear order is odd iff graph connected

 $\Rightarrow$ 

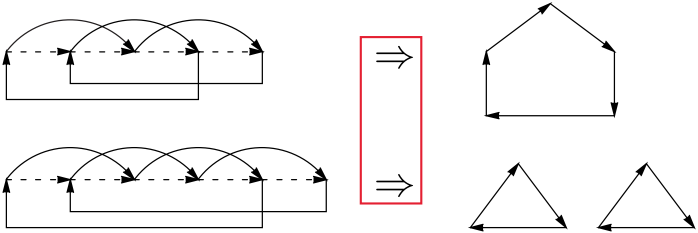
 $\Rightarrow$ 

linear order is even iff graph is disconnected

- Construction of graph from linear order is expressible as an FOL query $Q_{red} : \mathit{LinOrd} \longrightarrow \mathit{GRAPH}$ (see Exercise 3.3)

Reduce EVEN($<$) to Graph Connectivity

linear order is odd iff graph connected



linear order is even iff graph is disconnected

- Construction of graph from linear order is expressible as an FOL query $Q_{red} : \mathit{LinOrd} \longrightarrow \mathit{GRAPH}$ (see Exercise 3.3)

ACYCL and TC are not FOL expressible

- ACYCL: Reduction EVEN $\Rightarrow$ ACYCL as above but with one back edge from last node to first node

- Reduction for TC: CONN $\Rightarrow$ TC
  - Add edge $E(x, y)$ for every edge $E(y, x)$
  - Compute TC on resulting graph
  - Test whether graph is complete

Solutions to Exercise 2 (15 Points)

Solution to Exercise 2.1 (6 Points)

Formulate the following English sentences in FOL— preserving as much as possible the logical structure.

1. Every graduate course is a course.

   $\forall x(\mathit{GraduateCourse}(x) \rightarrow \mathit{Course}(x))$
2. No Student is a tutor of himself.

   $\forall x(\mathit{Student}(x) \rightarrow \neg \mathit{isTutorOf}(x, x))$
3. A person is a student if and only if he takes some graduate course

   $\forall x(\mathit{Person}(x) \rightarrow (\mathit{Student}(x) \leftrightarrow \exists y(\mathit{GraduateCourse}(y) \wedge \mathit{takes}(x, y)))$
4. Every student has exactly one Identity number.

   $\forall x(\mathit{Student}(x) \rightarrow \exists y \mathit{hasID}(x, y) \wedge (\forall z(\mathit{hasID}(x, z) \rightarrow z = y)))$
5. No course was attended by no student.

   $\neg \exists x(\mathit{Course}(x) \wedge \neg \exists y(\mathit{Student}(y) \wedge \mathit{attended}(y, x)))$
6. There are courses that were not attended by all students.

   $\exists x(\mathit{Course}(x) \wedge \neg \forall y(\mathit{Student}(y) \rightarrow \mathit{attended}(y, x)))$

Solution to Exercise 2.2 (3 Points)

1. $\phi_1 = \exists x, y, z(x \neq y \wedge x \neq z \wedge y \neq z)$
2. $\phi_2 = \exists x \forall y, z(x = y \vee x = z \vee y = z)$
   (actually: usually one presuppose a nonempty domain. So instead of $\exists x$ one could have used $\forall x$)
3. Can not be formulated in FOL. Assume, for contradiction, it were defined by a formula ϕ_{fin}. Consider formulae ϕ_i saying that there are at least i elements and the set $X = \{\phi_i \mid i \in \mathbb{N}\} \cup \{\phi_{fin}\}$. Every finite subset of X has a model. Due to compactness X must have a model ↯.

Solution to Exercise 2.3 (6 Points)

- Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \forall x P(x)$. This means that for all elements in the domain of $\mathcal{I}$ they must fall into the extension of P. As we may assume that the domain is not empty it follows that there must be an element which is in $\mathcal{I}(P)$.
-

$$
\begin{aligned}
& \forall x P(x,a) \rightarrow (\exists x Q(f(x)) \vee P(a,b) \vee \forall y Q(f(y))) \\
\equiv \quad & \forall x P(x,a) \rightarrow (\exists z Q(f(z)) \vee P(a,b) \vee \forall y Q(f(y))) \\
\equiv \quad & \neg \forall x P(x,a) \vee (\exists z Q(f(z)) \vee P(a,b) \vee \forall y Q(f(y))) \\
\equiv \quad & (\exists x \neg P(x,a)) \vee (\exists z Q(f(z)) \vee P(a,b) \vee \forall y Q(f(y))) \\
\equiv \quad & \exists x (\neg P(x,a) \vee (\exists z Q(f(z)) \vee P(a,b) \vee \forall y Q(f(y)))) \\
\equiv \quad & \exists x \exists z (\neg P(x,a) \vee (Q(f(z) \vee P(a,b) \vee \forall y Q(f(y)))) \\
\equiv \quad & \exists x \exists z \forall y (\neg P(x,a) \vee (Q(f(z) \vee P(a,b) \vee Q(f(y)))) \\
\equiv_{sat} \quad & \forall y (\neg P(c,a) \vee (Q(f(d) \vee P(a,b) \vee Q(f(y))))
\end{aligned}
$$

Exercise 3 (12 Points)

Hand in your exercise as one pdf File in Moodle by November 6, 23:55h

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.

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)?

Exercise 3.3 (4 Points)

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