UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Logic, Logic, and Logic

Lecture 2: FOL
15 April 2021

Informationssysteme CS4130
(Summer 2021)

Recap: Role of Logic in CS

Literature Hint: Introductions to Logic

- **Logic for CS**

  Lit: M. Huth and M. Ryan. Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, 2000.

  Lit: M. Ben-Ari. Mathematical Logic for Computer Science. Springer, 2. edition, 2001.

  Lit: U. Schöning. Logik für Informatiker. Spektrum Akademischer Verlag, 5. edition, 2000.

  Lit: M. Fitting. First-Order Logic and Automated Theorem Proving. Graduate texts in computer science. Springer, 1996.

- **Mathematical Logic**

  Lit: H.Ebbinghaus, J.Flum,and W.Thomas. Einführung in die mathematische Logik. Hochschul-Taschenbuch. Spektrum Akademischer Verlag, 2007.

  Lit: D. J. Monk. Mathematical Logic. Springer, 1976.

  Lit: R. Cori and D. Lascar. Mathematical Logic: Propositional calculus, Boolean algebras, predicate calculus. Mathematical Logic: A Course with Exercises. Oxford University Press, 2000.

Recap: First-Order Logic

FOL Structures and Interpretations

- **Structures:** $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots R_n^{\mathfrak{A}}, f_1^{\mathfrak{A}}, \ldots, f_m^{\mathfrak{A}}, c_1^{\mathfrak{A}}, \ldots, c_l^{\mathfrak{A}})$
- Usually: Universe A assumed to be non-empty

  Example: Graphs $\mathfrak{G} = (V, E^{\mathfrak{G}})$

- **Interpretations** $\mathcal{I} = (\mathfrak{A}, \nu)$

  Adds assignments ν for free variables.

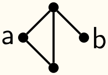
- **Syntax**
  - Terms (Example: $c, f(c, x)$)
  - Atomic formulae (Example: $c = d$, $E(a, d)$)
  - Formulae: (Example: $\exists y \, \exists z \; E(x, y) \wedge E(x, z) \wedge E(y, z)$)

FOL Semantics

- **Semantics** (Satisfaction/truth/modeling $\models$)
  - ...
  - $\mathcal{I} \models \exists x\ \phi$ iff: There is $d \in A$ s.t. $\mathcal{I}_{[x/d]} \models \phi$

Example

$(\mathfrak{G}, x \mapsto a) \models \exists y\ \exists z\ E(x,y) \wedge E(x,z) \wedge E(y,z)$



Alternative notation:
$\mathfrak{G} \models (\exists y\ \exists z\ E(x,y) \wedge E(x,z) \wedge E(y,z))(x/a)$

Definition (Derived Semantic Notions)

- Entailment: $\Phi \models \psi$ ("Φ entails ψ") iff for all interpretations $\mathcal{I}$: if $\mathcal{I} \models \Phi$, then $\mathcal{I} \models \psi$
- ψ is satisfiable iff there is an interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models \psi$
- Φ is satisfiable iff there is an interpretation $\mathcal{I}$ s.t. for all $\psi \in \Phi$: $\mathcal{I} \models \psi$
- $\mathit{Mod}(\Phi) = \{\mathcal{I} \mid \mathcal{I}$ satisfies all $\psi \in \Phi\}$
- ψ is valid iff for all interpretations $\mathcal{I}$: $\mathcal{I} \models \psi$.
- ψ is contradictory (unsatisfiable) iff for all interpretations $\mathcal{I}$: Not $\mathcal{I} \models \psi$

END of recap

FOL: Calculi and Algorithmic Problems

Plan for Today

- We investigate corresponding algorithmic problems for FOL
- Because, e.g., the definition of entailment does not say anything on how to compute that ψ is entailed by Φ
- Moreover, it does not say how much resources (place, time) are needed
- Example algorithmic problems
  - Given a structure $\mathfrak{A}$ and formula ϕ: Decide whether $\mathfrak{A} \models \phi$
  - Given a formula decide whether ϕ is satisfiable (valid, contradictory, resp.)
  - Given Φ, ψ decide whether $\Phi \vDash \psi$.
- Problems are related by reduction (at least for FOL)

Wake-Up Exercise

Show: $\Phi \vDash \psi$ iff $\Phi \cup \{\neg\psi\}$ is unsatisfiable

Remember:

- Entailment: $\Phi \models \psi$ ("Φ entails ψ") iff for all interpretations $\mathcal{I}$: if $\mathcal{I} \models \Phi$, then $\mathcal{I} \models \psi$
- ψ is unsatisfiable (or contradictory) iff for all interpretations $\mathcal{I}$: Not $\mathcal{I} \models \psi$

Challenges of FOL Algorithmic Problems

- First challenge: Domain of structure may be infinite
- But this is not the main problem (as we will see in lecture on finite model theory)

- Second challenge: Number of possible structures is infinite
- We want to tame the infinite by "syntactifying" the problem

A First Step Towards Algorithmization: Proof Calculi

- How to approach entailment problem $\Phi \models \psi$?
- Idea: Break down entailment into smaller entailment steps
  - "Smaller" entailment steps (which are "obvious")
  - Realized by applying finite number of rules $\mathcal{R}$
  - Apply rules to Φ and intermediate results to yield ψ

General derivation procedure

- Input: Φ, ψ
- Output: $\Phi \overset{?}{\models} \psi$
- $DS_0 = Encode(\Phi, \psi)$
- Find derivation $DS_0, \ldots, DS_n$
  where DS_i results from applying a rule from $\mathcal{R}$ to finite set of DS_j with $j < i$.
- Decode(DS_n) into answer to $\Phi \models \psi$

- Differences among calculi regarding: types of rules in $\mathcal{R}$; used data structures DS; proof methodology

Well Known Calculi

Calculus	Rule types	Data structures	Methodology
Hilbert	axioms 2 rules	formulae	direct (premises to conclusion)
Natural deduction	I(ntroduction) and E(limination) rules per constructor	formulae	direct
Gentzen style	axioms + I and E rules per constructor	Entailments	direct
Tableaux	"and", "or" rules	formula in a tree	refutation proofs based on DNF
Resolution	resolution rule	quantifier free formula in CNF in a tree	refutation proofs based on CNF

Resolution

Resolution

- Refutation calculus, i.e., calculus for showing unsatisfiability of a formula

- Steps
  - Data structures: formulas in clausal-normal form (Corresponds to CNF (conjunctive normal form) in propositional logic)
  - One rule: use satisfiability-preserving resolution rule to reduce formulae
  - Iteratively apply until empty clause (means: contradiction) is derived

- There are mature and efficient resolution provers (with many ingenious optimizations)
- Efficient (but nonetheless complete) resolution procedure SLD part of Prolog

Prenex Normal Form

- Idea of normalization
  - Transform formulas into a (syntactically) simpler form
  - preserving as much of the semantics as possible

Definition

A formula of the form $Q_1 x_1, \ldots, Q_n x_n \psi$, where $Q_i \in \{\forall, \exists\}$ and

- ψ, the so-called the matrix, does not contain quantifiers
- no variable occurs free and bounded
- every quantifier bounds a different variable

is said to be in prenex normal form (PNF)

- Here: Simplicity ensured by un-nesting quantifiers (the main reason for un-feasibility)
- Here "preserve semantic" means: Ensure equivalence $\equiv$

$$\phi \equiv \psi \text{ iff } \phi \models \psi \text{ and } \psi \models \phi$$

Existence of Prenex Normal Form

Theorem

Every FOL formula has an equivalent formula in PNF

Propositional Equivalences

- $\neg\neg\phi \equiv \phi$
- $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$
- $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$
- $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$
- $\phi \wedge (\psi \vee \chi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi)$

Equivalence under bounded substitutions

- $\exists x\phi \equiv \exists y(\phi[x/y])$
- where $\phi[x/y]$ is result of substituting every free x with y in ϕ

Quantifier-specific equivalences

- $\forall x\phi \equiv \neg\exists x\neg\phi$
- $\phi \equiv \exists x\phi$ (where x not free in ϕ)
- $(\exists x\phi \wedge \psi) \equiv \exists x(\phi \wedge \psi)$
  (where x not free in ψ)
- $(\exists x\phi \vee \psi) \equiv \exists x(\phi \vee \psi)$
  (x not free in ψ)
- $\exists x\phi \vee \exists x\psi \equiv \exists x(\phi \vee \psi)$
- $\exists x\exists y\phi \equiv \exists y\exists x\phi$

- $\phi \equiv \forall x\, \phi$ (where x not free in ϕ)
- $(\forall x\phi \wedge \psi) \equiv \forall x(\phi \wedge \psi)$
  (where x not free in ψ)
- $(\forall x\phi \vee \psi) \equiv \forall x(\phi \vee \psi)$
  (x not free in ψ)
- $\forall x\phi \wedge \forall x\psi \equiv \forall x(\phi \wedge \psi)$
- $\forall x\forall y\phi \equiv \forall y\forall x\phi$

Substituting with Equivalent Formula

Theorem

Assume $\phi \equiv \psi$ and χ contains ϕ as subformula. If χ' results from substituting ϕ with ψ, then $\chi \equiv \chi'$.

Proof: By structural induction.

Satisfiably Equivalent

- Formulae in PNF are going to be transformed to formula in clausal normal form

- Resulting formula are satisfiably equivalent

$$\phi \equiv_{sat} \psi \text{ iff: } Mod(\phi) \neq \emptyset \text{ iff } Mod(\psi) \neq \emptyset$$

- One cannot guarantee equivalence

Elimination of Exists Quantifiers: Skolemization

- Input a PNF formula $\phi : \forall_1 x_1, \dots \forall_n x_n \exists y \psi$
- Output $\phi' : \forall_1 x_1, \dots \forall_n x_n \psi[y/f(x_1, \dots, x_n)]$
  where f a fresh n-ary function symbol

  ϕ' results from skolemization out of ϕ, f called Skolem function (or Skolem constant if $n = 0$)

- Can be iteratively applied (starting with left-most $\exists$) until all $\exists$ are eliminated. Result is said to be in Skolem form and to be the skolemization of the original formula

Theorem

A formula and its skolemization are satisfiably equivalent.

Example (Skolem Form)

Given formula

$$\phi = \forall x \forall y (P(x,y) \to Q(x)) \to \exists x (\forall y \neg Q(y) \to \exists y \neg P(y,x))$$

transform it to Skolem form

$$\begin{aligned}
& \forall x \forall y (P(x,y) \to Q(x)) \to \exists x (\forall y \neg Q(y) \to \exists y \neg P(y,x)) \\
\equiv \quad & \forall x \forall y (\neg P(x,y) \vee Q(x)) \to \exists x (\neg \forall y \neg Q(y) \vee \exists y \neg P(y,x)) \\
\equiv \quad & \neg \forall x \forall y (\neg P(x,y) \vee Q(x)) \vee \exists x (\neg \forall y \neg Q(y) \vee \exists y \neg P(y,x)) \\
\equiv \quad & \exists x \exists y \neg (\neg P(x,y) \vee Q(x)) \vee \exists x (\exists y \neg\neg Q(y) \vee \exists y \neg P(y,x)) \\
\equiv \quad & \exists x \exists y (\neg\neg P(x,y) \wedge \neg Q(x)) \vee \exists x (\exists y \neg\neg Q(y) \vee \exists y \neg P(y,x)) \\
\equiv \quad & \exists x \exists y (P(x,y) \wedge \neg Q(x)) \vee \exists x (\exists y Q(y) \vee \exists y \neg P(y,x)) \\
\equiv \quad & \exists x_1 \exists y_1 (P(x_1,y_1) \wedge \neg Q(x_1)) \vee \exists x_2 (\exists y_2 Q(y_2) \vee \exists y_3 \neg P(y_3,x_2)) \\
\equiv \quad & \exists x_1 \exists y_1 (P(x_1,y_1) \wedge \neg Q(x_1)) \vee \exists x_2 \exists y_2 (Q(y_2) \vee \exists y_3 \neg P(y_3,x_2)) \\
\equiv \quad & \exists x_1 \exists y_1 (P(x_1,y_1) \wedge \neg Q(x_1)) \vee \exists x_2 \exists y_2 \exists y_3 (Q(y_2) \vee \neg P(y_3,x_2)) \\
\equiv \quad & \exists x_2 \exists y_2 \exists y_3 (\exists x_1 \exists y_1 (P(x_1,y_1) \wedge \neg Q(x_1)) \vee (Q(y_2) \vee \neg P(y_3,x_2))) \\
\equiv \quad & \exists x_2 \exists y_2 \exists y_3 \exists x_1 \exists y_1 ((P(x_1,y_1) \wedge \neg Q(x_1)) \vee (Q(y_2) \vee \neg P(y_3,x_2))) \\
\equiv_{sat} \quad & ((P(d,e) \wedge \neg Q(d)) \vee (Q(b) \vee \neg P(c,a)))
\end{aligned}$$

Clausal Normal Form

Definition

ψ is in clausal normal form (CLNF) iff it is in Skolem form, contains no free variables, and its matrix is in CNF

Definition

A quantifier-free formula is in conjunctive normal form (CNF) iff it is a conjunction of clauses

- Clause: Disjunction of literals
- Literal: atomic FOL formula or negated atomic FOL formula

Example CNF: $\underbrace{(R(a,x) \vee \neg P(x))}_{\textit{clause}} \wedge \underbrace{(\neg P(b) \vee Q(y))}_{\textit{clause}}$

Theorem

For every ψ there exists a satisfiably equivalent ψ' in CLNF

Resolution Idea

- Observation used for resolution:

$$(\alpha \vee \phi) \wedge (\neg\alpha \vee \psi) \wedge \chi \equiv_{sat} (\phi \vee \psi) \wedge \chi$$

  where
  - $\{\alpha, \neg\alpha\}$ is a pair of complementary literals
  - ϕ, ψ, χ arbitrary formulae

- Apply this equivalence iteratively on the matrix of formula in CLNF until empty clause (i.e. contradiction) is derived

- More convenient set notation for clauses
  - Clause $L_1 \vee \cdots \vee L_n$ written as set $\{L_1, \ldots, L_n\}$
  - $\overline{L_i}$ is complement of L_i
    E.g.: $\overline{R(a)} = \neg R(a)$, $\overline{\neg R(a)} = R(a)$

Lazy Proof Strategy by Unification

- Want to identify literals as complementary using unification
- Substitution σ: function from variables to terms

- σ unifies literals L_1, L_2 iff $L_1\sigma = L_2\sigma$
- Example
  - $L_1 = P(x, y)$, $L_2 = P(g(z), a)$
  - $\sigma_1 = [x/g(z), y/a]$

- Laziness: Find a most general unifier (mgu)
  - σ_1 more general than $\sigma_2 = [x/g(a), y/a, z/a]$.
  - σ is an mgu iff for all unifiers σ' there is substitution σ'' such that $\sigma' = \sigma \circ \sigma''$.

Theorem (Robinson)

Every unifyable finite set of literals has a mgu.

Resolution Step

Definition

Given clauses Cl_1, Cl_2, the clause RCl is a resolvent of Cl_1, Cl_2 iff

1. There are variable renamings σ_1, σ_2 s.t. $Cl_1\sigma_1$ and $Cl_2\sigma_2$ contain different variables.
2. There is a literal $L_1 \in Cl_1\sigma_1$ and $L'_1 \in Cl_2\sigma_2$ s.t. $\{L_1, \overline{L'}_1\}$ unifiable with mgu σ
3. $RCl = (CL_1\sigma_1 \setminus \{L_1\} \cup CL_2\sigma_2 \setminus \{L'_1\})\sigma$

A convenient graphical notation

$$\begin{tikzcd}
Cl_1 \arrow[dr, "\sigma"'] & & Cl_2 \arrow[dl, "\sigma"] \\
& RCl &
\end{tikzcd}$$

Example (Resolution)

$$
\begin{tikzcd}
\{\neg P(x_1, y_1), Q(x_1)\} \arrow[dr, "{[x_1/y_2]}"'] & \{\neg Q(y_2)\} \arrow[d] & \{P(y_3, x_2)\} \arrow[ddl, "{[y_3/y_2][x_2/y_1]}"] \\
& \{\neg P(y_2, y_1)\} \arrow[d] & \\
& \square &
\end{tikzcd}
$$

Correctness and Completeness

Definition

A calculus C is

- correct w.r.t. entailment iff: Whenever $\Phi \vdash_C \psi$, then $\Phi \models \psi$
- complete w.r.t. entailment iff: Whenever $\Phi \models \psi$, then $\Phi \vdash_C \psi$

- Correctness means: you can prove entailments only that really hold
- Completeness means: Whenever an entailment holds then there is also a proof for it. (Proved by ingenious Gödel)

Theorem

All aforementioned calculi are correct and complete

Resolution Theorem

- Let ψ be a clause set
- $Res(\psi) = \psi \cup \{RCl \mid RCl$ is a resolvent of clauses in $\psi\}$
- $R^{i+1}(\psi) = Res(Res^i(\psi))$
- $Res^*(\psi) = \bigcup Res^i(\psi)$

Theorem

Every ϕ in CLNF with matrix ψ is unsatisfiable iff $\Box \in Res^*(\psi)$ (or equivalently: if there is a derivation graph ending in $\Box$.)

- This shows correctness and completeness w.r.t. unsatisfiability testing
- But entailment can be reduced to it (remember wake-up question).
- Possible proof based on Herbrand models

Optional Slide: Completeness and Correctness for Resolution

- Herbrand structures blur syntax-semantic distinctions.

- Given ψ in Skolem form.
- Herbrand terms $HT(\psi)$: all possible closed terms from function symbols (and constants) in ψ
- Herbrand structure $HS(\psi)$
  - Domain: $HT(\psi)$
  - Interpretation of function symbols: $f^{HS(\psi)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
  - Relation symbols arbitrarily

Theorem

A formula is satisfiable iff it (its CLNF) has a Herbrand model

- Construction of Herband model: Interpret relation symbols R as $R^{HS(\psi)}(t_1, \ldots, t_n)$ if $\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \in R^{\mathcal{I}}$ for satisfying $\mathcal{I}$.

Optional Slide: Herbrand Expansion

- Given ψ in Skolem form $\forall x_1, \ldots, \forall x_n \phi$
- $HE(\psi)$: All "groundings" of the matrix with Herbrand terms

$$\{\psi[x_1/t_1, \ldots, x_n/t_n] \mid t_i \in HS(\psi)\}$$

Theorem (Herbrand)

Skolem formula ψ is satisfiable iff a finite subset of $HE(\psi)$ is satisfiable

Proof idea

- Show that ψ is satisfiable iff it has a Herbrand model
- Show that ψ has a Herbrand model iff $HE(\psi)$ is satisfiable
- Use compactness of propositional logic (discussed later)

But wait....

- We have shown completeness of calculi
- Doesn't this mean that we have a decision procedure for entailment (unsatisfiability)?
  - NO!

Theorem

Deciding validity (unsatisfiability, entailment) is un-decidable

  - But semi-decidability holds:
    if formula is valid you will eventually find a derivation; if formula not valid you won't know

Turing Machines

- One of the first precise computation models are Turing machines (TMs)
- Specifies precisely what it means to solve a problem algorithmically
  - Starting from a finite input (encoding)
  - give after a (finite number) of discrete steps
  - an encoding of the desired output
- Other alternative computation models: recursive functions, lambda calculus, register machines
- These computation models have been shown to be equivalent

Church Turing Thesis

What is intuitively computable is computable by a Turing machine

VIDEO: A Lego™ Turing machine

https://www.youtube.com/watch?v=FTSAiF9AHN4

Semi-decidability

Theorem

FOL entailment is semi-decidable, i.e., there is a TM s.t.

- *If Φ and ψ are inputs with $\Phi \vDash \psi$, then TM stops with yes*
- *otherwise it stops with no or it does not stop.*

Proof sketch:

- Given a calculus C with derivation relation $\vdash_C$ complete and correct for entailment
- The possible inferences starting from Φ make up a tree (with finite set of children for every node)
  - The root (level 0) is $\mathit{Encode}(\Phi, \psi)$
  - The finitely many children at level $n+1$ are those D_i that are generated from children at level up to n
  - Do a breadth first search until $\mathit{Encode}(\Phi \vDash \psi)$ appears

Why is FOL so Important?

Why is FOL so Successful (w.r.t.) CS

- Theoretical Answer: FOL is most expressive logic w.r.t. relevant properties (Lindström Theorems)
  $\Longrightarrow$ today

- Practical Answer: Has proven useful for query answering on SQL DBs and much more
  $\Longrightarrow$ next lectures

Compactness in Topology

"Ah, Kompaktheit, eine wundervolle Eigenschaft" (Jaenich 2008, S.24)

- Compactness notion stems from mathematical field topology
- Topologies $\mathfrak{T} = (X, \mathcal{O})$
  - Domain X and open sets $\mathcal{O} \subseteq Pot(X)$ with
  - Every union of open sets is open
  - Every finite intersection is open
  - X and $\emptyset$ are open
- Open covering of X
  Family of open sets $\{U_i\}_{i \in I}$ with $U_i \in \mathcal{O}$ and $\bigcup_{i \in I} U_i = X$

Lit: K. Jänich. Topologie. Springer, 8th edition, 2008.

Compactness in Topology

Definition

$(X, \mathcal{O})$ is compact iff every open covering of X has a finite sub-covering.

- How compactness is used to infer global properties from local properties
  - Let P be a property such that if open U, V have it, then also $U \cup V$ has it.
  - Then: If for every point $a \in X$ there is an open U_a having P, then X has P.

Wake-Up Exercise

Prove the correctness of this type of reasoning from local to global within compact spaces!

Proof

- Assume that if open U, V have P, then also $U \cup V$ has it. (*)
- Assume further that for all a there is U_a having P.
- $\{U_a\}_{a \in X}$ is a covering of X.
- Because of compactness there is a finite covering $U_{a_1} \cup \cdots \cup U_{a_n} = X$.
- Because of (*) it follows that $U_{a_1}, \ldots, U_{a_n}$ has P, i.e., X has P.

Definition ((Logical) Compactness)

A logic $\mathcal{L}$ has the compactness property if the following holds: For all sets Φ of formulae in $\mathcal{L}$: If every finite subset of Φ has a model, then Φ has a model.

- Equivalent definition:

  If $\Phi \vDash \psi$, then already $\Phi_0 \vDash \psi$ for a finite Φ_0
- Intuitively: Infiniteness adds not additional expressive power for FOL

Theorem

FOL has the compactness property.

- Logical compactness derived from topological notion
- FOL compactness is a corollary of Tychonoff's Theorem ("Any product of compact topological spaces is compact")

Application: Reachability is not FOL Expressible

Query Q_{reach}: List all cities reachable from Hamburg!

$$
\begin{aligned}
Q_{reach}(x) \quad = \quad & Flight(Hamburg, x) \vee \\
& \exists x_1 Flight(Hamburg, x_1) \wedge Flight(x_1, x) \vee \\
& \exists x_1, x_2 Flight(Hamburg, x_2) \wedge Flight(x_2, x_1) \wedge Flight(x_1, x) \vee \ldots
\end{aligned}
$$

Theorem

Reachability is not expressible in FOL.

Proof

- For contradiction assume there is FOL $\phi_{reach}(x, y)$ expressing reachability over edges E
- Consider FOL formulae ϕ_n: "There is an n-path from c to c'"
- Let $\Psi = \{\neg\phi_i \mid i \in \mathbb{N}\} \cup \{\phi_{reach}(c, c')\}$
- Ψ is unsatisfiable, but every finite subset is satisfiable ↯

Application: Infinitesimal Probabilities

- Over continuous domains "low-dimensional" events have probability 0
- Conditional probability $P(B|A)$ undefined for $P(A) = 0$
- But $P($ point on east hemisphere $|$ point on equator$)$ should be $1/2$ (and not undefined)

  $\Longrightarrow$ Need infinitesimal positive probability weights
- Consider $T = Th(\mathbb{R}) \cup \{a < \Omega \mid a \text{ is name of a real number}\}$
- Every finite subset of T satisfiable; with compactness T is satisfiable
- $1/\Omega$ infinitesimal element

  Lit: J. Weisberg. Varieties of bayesianism. In D. M. Gabbay, S. Hartmann, and J. Woods, editors, Inductive Logic, volume 10 of Handbook of the History of Logic, pages 477–551. North-Holland, 2011.

  Lit: A. Robinson. Non-standard Analysis. Princeton Landmarks in Mathematics. Princeton University Press, 1996.

FOL has the Löwenheim-Skolem-Property

Theorem (Downward Löwenheim-Skolem-Property)

Every satisfiable, countable set of FOL sentences (theory) has a countable model.

- Intuitively: If you can talk with countably many sentences about structures, then there is a countable model verifying this fact.
- Can be shown by Herbrand expansions
- Leads to Skolem's paradox
  - You can formalize mathematics within countable FOL theory, namely, Zermelo-Fränkel Set Theory (ZFC)
  - ZFC $\models$ "there are uncountable sets".

Why FOL is so Important: Lindström Theorems

Theorem (First Lindström Theorem)

There is no (regular) logic that is more expressive than FOL and fulfills compactness and Löwenheim-Skolem Property

- Meta theorem
- Intuitively: FOL is the most expressive (regular) logic fulfilling compactness and the Löwenheim-Skolem Property

- Regularity of logic
  - Contains boolean operators
  - Allows relativizing formula to domains
  - Allows substituting constants and function symbols by relation symbols

Limits of FOL

- Positive: FOL can be used for effective query answering on <u>one</u> model (in data complexity)!
- Negative
  - Entailment problem, satisfiability etc. not decidable
    $\Longrightarrow$ Calls for restriction to feasible fragments
  - Expressivity not sufficient (no recursion)
    $\Longrightarrow$ Calls for extensions (and restrictions)