UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Logic, Logic, and Logic

Lecture 2: FOL

28 October, 2015

Foundations of Ontologies and Databases for Information Systems
CS5130 (Winter 2015)

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:** $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots R_n^{\mathfrak{A}}, f_1^{\mathfrak{A}}, \dots, f_m^{\mathfrak{A}}, c_1^{\mathfrak{A}}, \dots, c_l^{\mathfrak{A}})$
- 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.

Because dealing with variables is non-trivial...

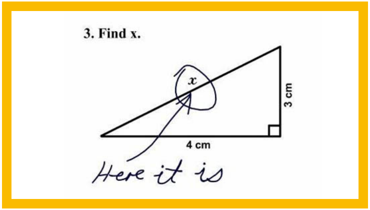
- **Syntax**
  - Terms (Example: $c, f(c, x)$)
  - Atomic formulae (Example: $c = d$, $E(a, d)$)
  - Formulae: (Example: $\exists y \, \exists z \, E(x, y) \wedge E(x, z) \wedge E(y, z)$)

FOL Structures and Interpretations

- **Structures:** $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots R_n^{\mathfrak{A}}, f_1^{\mathfrak{A}}, \dots, f_m^{\mathfrak{A}}, c_1^{\mathfrak{A}}, \dots, c_l^{\mathfrak{A}})$
- 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.

Because dealing with variables is non-trivial...



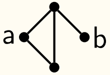
- **Syntax**
  - Terms (Example: $c, f(c, x)$)
  - Atomic formulae (Example: $c = d$, $E(a, d)$)
  - Formulae: (Example: $\exists y \, \exists z \, E(x, y) \wedge E(x, z) \wedge E(y, z)$)

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. $\mathcal{I} \models \phi$
- Φ is **satisfiable** iff there is an interpretation $\mathcal{I}$ s.t. for all $\phi \in \Phi$: $\mathcal{I} \models \psi$

- $\mathit{Mod}(\Phi) = \{\mathcal{I} \mid \mathcal{I} \text{ satisfies all } \psi \in \Phi\}$

- 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)

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

- 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 **contradictory (unsatisfiable)** iff for all interpretations $\mathcal{I}$: Not $\mathcal{I} \models \psi$

Challenges of FOL Algorithmic Problems

- First challenge: Structure domains 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 $\Phi \vDash \psi$?
- **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
- Common derivation procedure for all calculi
  - Input: Φ, ψ
  - Output: $\Phi \overset{?}{\vDash} \psi$
  - $DS_0 = Encode(\Phi, \psi)$
  - 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) to answer to $\Phi \vDash \psi$
- Differences regarding
  - the rule types/numbers $\mathcal{R}$
  - used data structures DS
  - and the used proof methodology

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

- Common derivation procedure for all calculi
  - 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) to answer to $\Phi \models \psi$

- Differences regarding
  - the rule types/numbers $\mathcal{R}$
  - used data structures DS
  - and the used proof methodology

A First Step Towards Algorithmization: Proof Calculi

- How to approach entailment problem $\Phi \vDash \psi$?
- **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

- Common derivation procedure for all calculi
  - Input: Φ, ψ
  - Output: $\Phi \overset{?}{\vDash} \psi$
  - $DS_0 = Encode(\Phi, \psi)$
  - 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) to answer to $\Phi \vDash \psi$

- Differences regarding
  - the rule types/numbers $\mathcal{R}$
  - used data structures DS
  - and the used proof methodology

Well known Calculi

calculus	rule types	data structures	methodology
Hilbert	axioms 2 rules	formula	direct (premises to conclusion)
Natural deduction	introduction and elimination 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 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

Resolution

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

- Steps
  - Data structures: formulas in **clausal-normal form** (Corresponds to CNF 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

Resolution

- **Refutation calculus**, i.e., calculus for showing unsatisfiability of a formula
- Steps
  - Data structures: formulas in **clausal-normal form** (Corresponds to CNF 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 $Q_1x_1, \ldots, Q_nx_n\psi$, where $Q_i \in \{\forall, \exists\}$ and

- ψ (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: Simple form 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

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

Quantifier-specific equivalences

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

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

Existence of Prenex Normal Form

Theorem

Every FOL formula has an equivalent formula in PNF

Propositional Equivalences

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

Quantifier-specific equivalences

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

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

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

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

Existence of Prenex Normal Form

Theorem

Every FOL formula has an equivalent formula in PNF

Propositional Equivalences

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

Quantifier-specific equivalences

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

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

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 may be **satisfiably equivalent** only

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

Elimination of Exists Quantifiers: Skolemization

- Input a PNF formula $\phi : \forall_1 x_1, \ldots \forall_n x_n \exists y \psi$
- Output $\phi' : \forall_1 x_1, \ldots \forall_n x_n \psi[x/f(y_1, \ldots, y_n)]$
  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

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

transform it to Skolem form

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

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

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 notation
  - 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.

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.

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, RCl, 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 $L'_1 \in CL_2$ s.t. $\{L_1, \overline{L'}_1\}$ unifyable 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

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

Resolution Example

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

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 only prove entailments 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
- $Res(\psi) = \psi \cup \{RCl \mid RCl \text{is a resolvent of clauses in } \psi\}$
- $R^{i+1} = Res(Res^i(\psi))$
- $Res^*(\psi) = \bigcup Res^i(\psi)$

Theorem

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

- 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

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 $R^{HS(\psi)}(t_1, \ldots, t_1)$ if $\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \in P^{\mathcal{I}}$ for satisfying $\mathcal{I}$.

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 $E(\psi)$ is satisfiable

Proof idea

- Show that ψ is satisfiable iff it has a Herbrand model
- Show that ψ has a Herbrand model iff $E(\psi)$ 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)?

Theorem

Deciding validity (unsatisfiability, entailment) is un-decidable

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

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^{TM} Turing machine

Undecidability of Validity

- Shown by **Reduction** of Post Correspondence Problem to Validity problem
- Reduction strategy widely used by relying on library of known results (also for proving complexity bounds)

Post Correspondence Problem (PCP)

- Input: Finite list of word pairs $(x_1, y_1), \ldots, (x_k, y_k)$ with $x_i, y_i \in \{0, 1\}^+$
- Output: Is there list of indices $i_1, \ldots, i_n \in \{1, \ldots k\}$ with $n \geq 1$ s.t. $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

Wake-up Exercise

Show that the following PCP instance has a solution.

Undecidability of Validity

- Shown by **Reduction** of Post Correspondence Problem to Validity problem
- Reduction strategy widely used by relying on library of known results (also for proving complexity bounds)

Post Correspondence Problem (PCP)

- Input: Finite list of word pairs $(x_1, y_1), \ldots, (x_k, y_k)$ with $x_i, y_i \in \{0, 1\}^+$
- Output: Is there list of indices $i_1, \ldots, i_n \in \{1, \ldots k\}$ with $n \geq 1$ s.t. $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

Wake-up Exercise

Show that the following PCP instance has a solution.

Solution: choose index $(i_1, \ldots, i_4) = (1, 3, 2, 3)$

Undecidability of Validity

- Given PCP instance $K = ((x_1, y_1), \dots, (x_k, y_k))$, produce formula ϕ_k such that
  K has a solution iff ϕ_K is valid.
- Use two function symbols f_0 and f_1 to mimic 1 and 0
- $f_{i_1,\dots i_l}(x)$ abbreviates $f_{i_1}(f_{i_2}(\dots f_{i_l}(x)\dots))$ (string $i_1 \dots i_l$)
- Consider formula

$$\phi_K : (\phi_1 \land \phi_2 \to \phi_3)$$

with

- $\phi_1 : \bigwedge_{i=1}^{k} P(f_{x_i}(a), f_{y_i}(a))$
- $\phi_2 : \forall u \forall v (P(u, v) \to \bigwedge_{i=1}^{k} P(f_{x_i}(u), f_{y_i}(v))$
- $\phi_3 : \exists z P(z, z)$

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 false 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 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: Most expressive language 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!

Wake-Up Exercise

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

Solution

- 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 as 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")

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 as 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!

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

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

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 contable verifying model.
- 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 $\vDash$ "there are uncountable sets".

Wake-up Question

Argue why Skolem's paradox is only of a psychological nature, i.e., it does <u>not</u> prove the inconsistency of ZFC.

Wake-up Question

Argue why Skolem's paradox is only of a psychological nature, i.e., it does not prove the inconsistency of ZFC.

- In ZFC one can express that there are is an uncountable set US.
- Uncountable means that there does not exist an injective function f from US to the natural numbers.
- So in the countable domain of a model $\mathfrak{A}$for ZFC there there is no injective function $f : US \to \mathbb{N}$ though clearly you may find (in another richer model) such an injective function.

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

Limits of FOL

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

Exercise 2 (15 Points)

Send your solutions in one pdf file as presentation by Tuesday evening, 3 November, 2015 to oezcep@ifis.uni-luebeck.de.

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.
2. No Student is a tutor of himself.
3. A person is a student if and only if he takes some graduate course
4. Every student has exactly one Identity number.
5. No course was attended by no student.
6. There are courses that were not attended by all students.

Exercise 2.2 (3 points)

State FOL sentences ϕ_i over a given signature whose models $M_i = Mod(\phi_i)$ are the following ones—if possible. Otherwise argue why this is not possible.

1. $M_1 = \{$ All structures with at least 3 elements$\}$
2. $M_2 = \{$ All structures with at least one element and at most 2 elements$\}$
3. $M_3 = \{$ All structures with finitely many elements$\}$

Exercise 2.3 (6 points)

1. Show that the formula $\forall x\ P(x) \rightarrow \exists y\ P(y)$ is valid—using only the definition of the satisfaction relation $\models$. (2 points)
2. Transfer the following formula into clausal normal form

$$\forall x P(x, a) \rightarrow (\exists x Q(f(x)) \vee P(a, y) \vee \forall y Q(f(y)))$$

(4 points)