

Sentential vs. First Order Logic

- Sentential Logic:
 - Propositional Letters (A, B, C), Negation (\neg), Conjunction (\wedge), Disjunction (\vee)
- First Order Logic
 - Predicates ($P(a), S(x, y, z)$), Quantification ($\forall x \phi$ and $\exists x \phi$)

Effectiveness

- 1) Exact instructions, finitely long, explaining the procedure and demanding no cleverness.
- 2) No random devices.
- 3) Must produce an answer in a finite number of steps.

There is an effective procedure that can decide
whether $\Sigma \models \tau$

Truth Table

A | B | $\neg(A \vee B)$ | $\neg A \wedge \neg B$

T	 	T	 	F	T	T	T	 	F	T	F	F	T
T	 	F	 	T	T	F	F	 	F	T	T	T	F
F	 	T	 	T	F	F	T	 	T	F	T	F	T
F	 	F	 	T	F	F	F	 	T	F	T	T	F

Disjunctive Normal Form

- $\alpha = \gamma_1 \vee \dots \vee \gamma_k, \quad \gamma_k = \beta_{i_1} \wedge \dots \wedge \beta_{i_n}$
- Any wff φ has an equivalent wff α in disjunctive normal form.
- Both $\{\neg, \vee\}$ and $\{\neg, \wedge\}$ are complete.

Truth and Models

- Structure S defines parameters:
 - Assign to each universal quantifier \forall the universe of S , $|S|$.
 - Assign to each predicate symbol P a relation $P^S \subseteq |S|^n$.
 - Assign to each constant c a member c^S from $|S|$.
 - Assign to each each function f an operation f^S on $|S|$.
- Interpretation: Association between the propositional constants and the truth values True or False.
- Model for T : Structure for T in which all sentences of T are true.

Properties of Sentences

- A sentence is *valid* if and only if every interpretation is a model.
- A sentence is *satisfiable* if and only if some interpretation is a model.
- A sentence is *unsatisfiable* if and only if no interpretation is a model.

A Deductive Calculus

- Deduction of φ from Γ : sequence $\langle \alpha_0, \dots, \alpha_n \rangle$ of formulas so that $\alpha_0 = \varphi$ and for each $i \leq n$ either:
 - a) α_i is in $\Gamma \cup \Lambda$, or
 - b) for some j and $k < i$, α_i is obtained by modus ponens from α_j and α_k .
- There exists a deduction of α from Γ iff α is a theorem of Γ .

A Deductive Calculus

- 1) Tautologies
- 2) $\forall x \alpha \rightarrow \alpha^x_t$
- 3) $\forall x (\alpha \rightarrow \beta) \rightarrow (\forall x \alpha \rightarrow \forall x \beta)$
- 4) $\alpha \rightarrow \forall x \alpha$
- 5) $x \approx x$
- 6) $x \approx y \rightarrow (\alpha \rightarrow \alpha')$

Major Theorems:

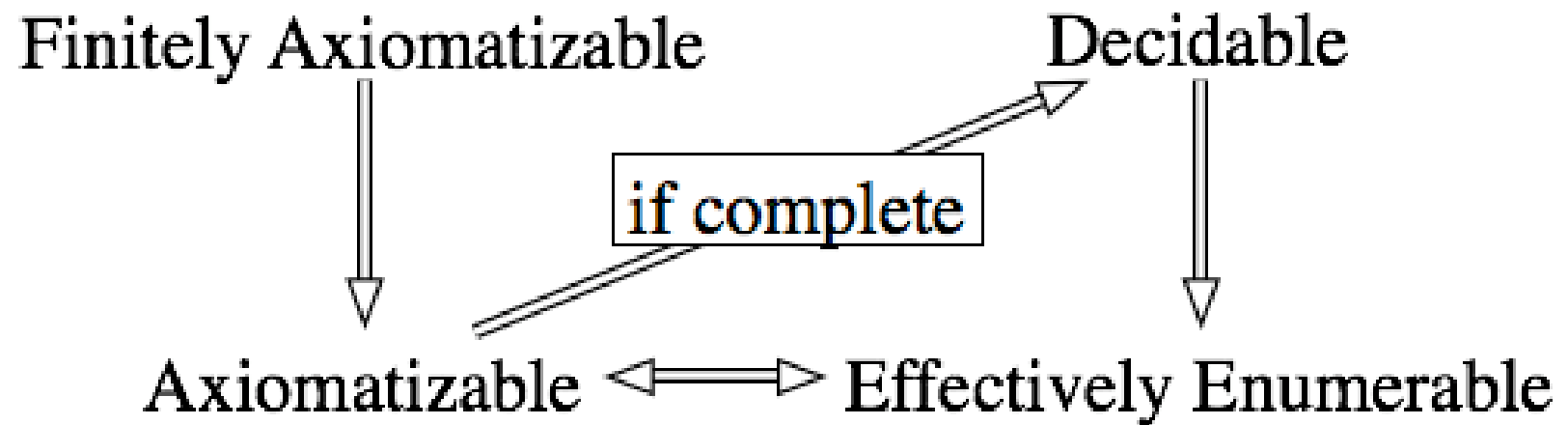
- Soundness: $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$
- Completeness: $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$
- Compactness: $\Gamma \models \varphi \Rightarrow \exists \Gamma_0 \subseteq \Gamma$ finite s.t.
 $\Gamma_0 \models \varphi$

Theories

- $\text{Cn } \Gamma = \{ \varphi \mid \Gamma \models \varphi \}$
- A set of sentences \mathcal{T} is a theory if $\mathcal{T} \models \varphi \Rightarrow \varphi \in \mathcal{T}$ (I.e. $\mathcal{T} = \text{Cn } \mathcal{T}$)
- If \mathcal{S} is a structure $\text{Th } \mathcal{S}$ is the set of all sentences true in \mathcal{S} ($\models_{\mathcal{S}} \varphi \Leftrightarrow \varphi \in \text{Th } \mathcal{S}$)

Axiomatization

- \mathcal{T} is *complete* if $\forall \varphi, \varphi \in \mathcal{T}$ or $\neg \varphi \in \mathcal{T}$
- \mathcal{T} is *effectively (recursively) enumerable* if \exists an effective procedure that returns “yes” whenever $\varphi \in \mathcal{T}$
 - \mathcal{T} is *decidable (recursive)* if it also returns “no” whenever $\varphi \notin \mathcal{T}$
- \mathcal{T} is *(recursively) axiomatizable* if \exists a decidable set Γ such that $\mathcal{T} = \text{Cn } \Gamma$
 - \mathcal{T} is *finitely axiomatizable* if Γ is finite

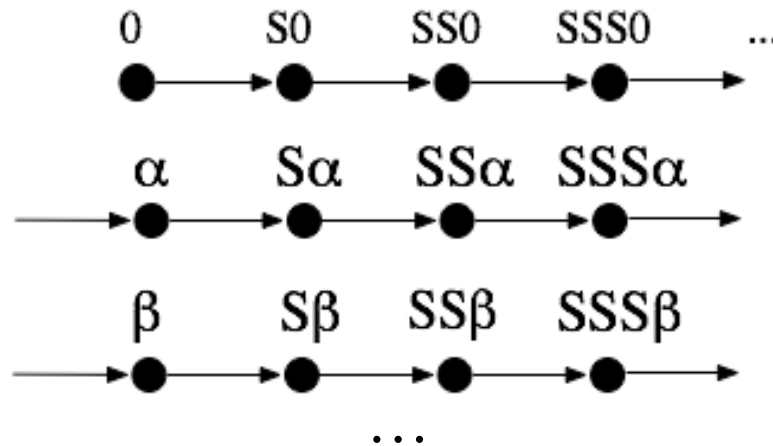


Infinite Models

- Assume the language \mathcal{L} is countable:
 - Los-Vaught Test: If \mathcal{T} is \aleph -categorical (all models of cardinality \aleph are isomorphic) for some infinite \aleph , then \mathcal{T} is complete
 - Löwenheim-Skolem Theorem: If Γ is satisfiable then it is satisfiable in a countable model
- The class of all finite structures cannot be expressed in first-order logic

Number Theory

- Let \mathcal{N} be the structure $(\mathbb{N}; 0, S, <, +, \cdot, E)$ with the usual interpretations
- \mathbb{N} cannot be described up to isomorphism by first-order logic
 - “ $\forall x \exists n (x = S^n 0)$ ” cannot be expressed
 - “ $\forall x (x = 0 \vee x = S0 \vee x = SS0 \vee \dots)$ ” requires an infinite sentence



Incompleteness

- First Incompleteness Theorem: $\text{Th } \mathcal{N}$ is not recursively axiomatizable
 - $(\mathbb{N}; 0, S, <, +)$ is decidable
 - $(\mathbb{N}; 0, S, <, +, \cdot)$ is undecidable
- Second Incompleteness Theorem: If \mathcal{T} is a “sufficiently strong” recursively axiomatizable theory then $\mathcal{T} \vdash \text{Cons } \mathcal{T} \Leftrightarrow \mathcal{T}$ is *inconsistent*
 - If $\text{Th } \mathcal{N}$ is consistent then $\text{Th } \mathcal{N} \not\vdash \text{Cons Th } \mathcal{N}$

Conjunctive normal form for first-order logic

- the first-order resolution rule requires that sentences be in conjunctive normal form (CNF): $(c_{11} \vee c_{12} \vee \dots) \wedge (c_{21} \vee c_{22} \vee \dots) \wedge (c_{31} \vee c_{32} \vee \dots)$
- However in AI it is often implied that that a sentence that is in CNF also has its “quantifier removed”.

- ex.: $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x,y)] \Rightarrow \exists y \text{ Loves}(y,x)$

becomes $[\text{Animal}(F(x)) \vee \text{Loves}(G(x),x)] \wedge [\neg \text{Loves}(x,F(x)) \vee \text{Loves}(G(x),x)]$

- Removal of quantifiers isn't magic: it simply means that a sentence is converted to a form in which
 - every free variable is actually universally quantified by an invisible quantifier at the beginning and
 - every originally existentially quantified variable is represented as a function of the universally quantified variables in whose scope it was (this part is called **Skolemization**).

Converting to CNF using Skolemization

- How does one convert a sentence to CNF and remove quantifiers? (Recall: removing existential quantifiers is called Skolemization.)
- ex.: $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x,y)] \Rightarrow \exists y \text{ Loves}(y,x)$
- 1. eliminate implications:
$$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x,y)] \vee \exists y \text{ Loves}(y,x)$$
- 2. move \neg inwards:
$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee \exists y \text{ Loves}(y,x)$$
- 3. standardize variables (give each variable a different name for each quantifier scope it is in):
$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee \exists z \text{ Loves}(z,x)$$
- 4. Skolemize:
$$\forall x [\text{Animal}(F(x)) \wedge \neg \text{Loves}(x,F(x))] \vee \text{Loves}(G(x),x)$$
- 5. drop universal quantifiers:
$$[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x,F(x))] \vee \text{Loves}(G(x),x)$$
- 6. distribute \wedge over \vee :
$$[\text{Animal}(F(x)) \vee \text{Loves}(G(x),x)] \wedge [\neg \text{Loves}(x,F(x)) \vee \text{Loves}(G(x),x)]$$

Unification

- Inference rules like resolution require finding substitutions that make different logical expressions look identical; this is called **unification**.
- ex.:
$$\frac{[\text{Animal}(F(x)) \vee \text{Loves}(G(x),x)] \wedge [\neg \text{Loves}(u,v) \text{ or } \vee \neg \text{Kills}(u,v)]}{[\text{Animal}(F(x)) \vee \neg \text{Kills}(G(x),x)]}$$
- Here, the unifier is $\theta = \{u/G(x), v/x\}$.
- Let θ_1 and θ_2 both be unifiers for the pair of expressions (ϕ, ψ) . θ_1 is said to be *more general* than θ_2 if there is a substitution η such that $\theta_2 = \eta \circ \theta_1$. (Note of caution: Manna/Waldinger confusingly uses the notation $\theta_1 \circ \eta := \eta \circ \theta_1$.)
- ex.: For $(\text{Knows}(\text{John}, x), \text{Knows}(y, z))$, the unifier $(y/\text{John}, x/z)$ is more general than $(y/\text{John}, x/\text{John}, z/\text{John})$.
- *If* a pair of expressions can be unified, they have a **most general unifier (MGU)** which is *unique* (up to renaming of variables).
- Finding the MGU can be done by inspection; the exact algorithm (on handout) does nothing but recursively going into deeper and deeper levels of the expression and aligning corresponding arguments, trying to most-generally-unify each pair of arguments.
- How do you find out whether two expressions are not unifiable? Simply do the MGU algorithm and abort if at some level you have to unify a variable with an expression *containing it*, for instance: $\text{unify}(x, f(x)) \rightarrow \text{abort!}$

Well-founded relations

- In a given theory, a binary relation $<$ is said to be **well-founded** over a unary predicate symbol obj if:
 - There are no infinite sequences $\langle t_0, t_1, \dots \rangle$ of objects such that $(\forall u \in \mathbf{Z}: obj(t_u) \wedge t_{u+1} < t_u)$.
- ex.: $<$ is not well-founded over *integer*; $>$ is not well-founded over *natural-number*; $<$ is well-founded over *natural-number*.
- A well-founded relation is *irreflexive* and *asymmetric*, but not necessarily *transitive*.
- Given two relations, their **union relation** is defined as follows:
$$\forall obj\ x,y: [x <_{\text{union}} y : \Leftrightarrow (obj_1(x) \wedge obj_2(y)) \vee (obj_1(x) \wedge obj_1(y) \wedge x <_1 y) \vee (obj_2(x) \wedge obj_2(y) \wedge x <_2 y)]$$
- **Theorem:** If obj_1 and obj_2 characterize disjoint classes of objects and are both well-founded, then their union relation is well-founded too.
- Well-founded relations are needed for **well-founded induction**, which is a generalization of (ordinary) induction. Well-founded induction is not on the Logic Comp, though, but well-founded relations are.

Polarity

- For any given sentence, we can annotate its subexpressions with **polarities**: $+$, $-$, \pm (there is no distinction between \pm and \mp).
- recursive rules:
 - The sentence S itself has positive polarity: S^+
 - $[\neg F]^\pi \Rightarrow [\neg F^{-\pi}]$
 - $[F \wedge G]^\pi \Rightarrow [F^\pi \wedge G^\pi]$; same principle for \vee
 - $[F \Rightarrow G]^\pi \Rightarrow [F^{-\pi} \Rightarrow G^\pi]$
 - $[\text{if } F \text{ then } G \text{ else } H]^\pi \Rightarrow [\text{if } F^\pm \text{ then } G^\pi \text{ else } H^\pi]$
 - $[F \Leftrightarrow G]^\pi \Rightarrow [F^\pm \Leftrightarrow G^\pm]$
- ex.: $[[[\neg P^+]^- \vee Q^-]^- \Rightarrow [Q^+ \wedge [P^\pm \Leftrightarrow [\neg Q^\pm]^\pm]^+]^+]^+$
- An occurrence of a variable that is $+$ or \pm is positive, one that is $-$ or \pm is negative, one that is $+$ is strictly positive, and one that is $-$ is strictly negative.
- **Theorem:** If we replace at least one strictly positive occurrence E^+ of an expression within a sentence S with a “truer” sentence F (that is, $E \Rightarrow F$), the whole sentence becomes “truer” (that is, $S(E^+) \Rightarrow S(F^+)$).
- **Theorem:** If we replace at least one strictly negative occurrence E^- of an expression within a sentence S with a “truer” sentence F (that is, $E \Rightarrow F$), the whole sentence becomes “falsier” (that is, $S(F^-) \Rightarrow S(E^-)$).

Deductive tableaux

- A sentence (both in propositional logic and in first-order logic) can be represented as a **tableau**.
- left column: **assertions**;
right column: **goals**
- The tableau on the right is defined to be equivalent to the sentence
 $(Q \vee R) \Rightarrow (\neg P \vee (P \Rightarrow Q))$.

assertions	goals
	G1. $\neg P$
A2. $Q \vee R$	
	G3. $P \Rightarrow Q$

- In general, a tableau with assertions A_1, A_2, \dots, A_m and goals G_1, G_2, \dots, G_n is defined to be equivalent to the sentence
 $(A_1 \wedge A_2 \wedge \dots \wedge A_m) \Rightarrow (G_1 \vee G_2 \vee \dots \vee G_n)$.
- You can move a sentence from one column of a tableau to the other (yielding an equivalent tableau) if you negate it.

Resolution rule for tableaux

- Proofs can be done using tableaux by representing the sentence-to-be-proven as a tableau and then applying *sound* tableaux deduction rules (that is, rules that yield equivalent tableaux) until we get an obviously valid tableau.
- On the Logic Comp, you might need the *resolution rule* for tableaux: If a tableau contains the assertions $A_1[P]$ and $A_2[P]$ ($A_1[P]=A_2[P]$ is allowed), you may add $(A_1[\text{false}] \vee A_2[\text{true}])$ (replace *all* occurrences of P within A_1 and A_2 by *false* and *true*, respectively) as an assertion to the tableau.
- This was the AA-version (assertion-assertion) of the resolution rule; for the AG, GA, and GG versions, refer to the handout. You also need *many* more tricks for being able to play around with tableaux effectively (see Manna/Waldinger), but the resolution rule is the most crucial.