

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 $\Sigma \models \tau$

Disjunctive Normal Form

$$\alpha = \gamma_1 \vee \dots \vee \gamma_k, \quad \gamma_k = \beta_{i1} \wedge \dots \wedge \beta_{im}$$

Any wff φ has an equivalent wff α in disjunctive normal form.

Both $\{\neg, \vee\}$ and $\{\neg, \wedge\}$ are complete.

Truth & Models

Interpretation: Association between the propositional constants and the truth values True or False.

Structure \mathcal{S} defines parameters:

Assign to each universal quantifier \forall the universe of \mathcal{S} , $|S|$.

Assign to each predicate symbol P a relation $P^{\mathcal{S}} \subseteq |S|^n$.

Assign to each constant c a member $c^{\mathcal{S}}$ from $|S|$.

Assign to each function f an operation $f^{\mathcal{S}}$ on $|S|$.

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 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 .

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) $x \approx x$

5) $\alpha \rightarrow \forall x \alpha$ $x \approx x$

6) $x \approx y \rightarrow (\alpha \rightarrow \alpha')$

Let Γ be a set of sentences in a language \mathcal{L} and φ a sentence in \mathcal{L}

Theorem (Soundness). $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$.

Equivalently, if Γ is satisfiable, then Γ is consistent. This means proofs work correctly: they can only prove statements that are always true.

Theorem (Completeness). $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$.

Equivalently, if Γ is consistent, the Γ is satisfiable. Everything that is logically implied (always true) by Γ is provable, so the proof system is as powerful as it can be. Together with soundness, completeness implies that the syntactic notion of proof corresponds to the semantic notion of entailment.

Theorem (Compactness). $\Gamma \models \varphi \Rightarrow \exists$ a finite set $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \models \varphi$.

Equivalently, if every finite subset Γ_0 of Γ is satisfiable, then Γ is satisfiable. Intuitively, compactness means that all proofs (and counter-examples) must be of finite length. If Γ is not satisfiable, then it must imply a contradiction, in which case there is a proof of a contradiction. Since the proof is only finitely long, it employs only finitely many sentences Γ_0 , so Γ_0 is a finite, unsatisfiable subset of Γ .

Definition (Consequences). If Γ is a set of sentences, then the consequences of Γ , $Cn\Gamma = \{\varphi \mid \Gamma \models \varphi\}$ is the set of all sentences logically implied by Γ

Definition (Theory). A set of sentences \mathcal{T} is a theory if $\mathcal{T} \models \varphi \Rightarrow \varphi \in \mathcal{T}$

\mathcal{T} is closed under entailment, i.e. $\mathcal{T} = Cn\mathcal{T}$

Definition (Theory). If \mathfrak{A} is a structure, then the theory of \mathfrak{A} , $Th\mathfrak{A}$, is the set of all sentences true in \mathfrak{A} , i.e. $\models_{\mathfrak{A}} \varphi \Leftrightarrow \varphi \in Th\mathfrak{A}$.

Definition (Complete Theory). A theory \mathcal{T} is complete if $\forall \varphi, \varphi \in \mathcal{T}$ or $\neg\varphi \in \mathcal{T}$

Note that $Th\mathfrak{A}$ is complete because in any particular structure \mathfrak{A} everything is either true or false

Definition (Enumerable). \mathcal{T} is (effectively/recursively) enumerable if there is an effective procedure/Turing machine that returns "yes" whenever $\varphi \in \mathcal{T}$

Definition (Decidable / Recursive). \mathcal{T} is decidable or recursive if there is an effective procedure that returns "yes" whenever $\varphi \in \mathcal{T}$ and "no" whenever $\varphi \notin \mathcal{T}$

Enumerable theories are not in general decidable because the program may never return if neither the sentence nor its negation is in the theory.

Definition (Axiomatizable). \mathcal{T} is (recursively) axiomatizable if \exists a decidable set Γ such that $\mathcal{T} = Cn\Gamma$.

If Γ is finite, then \mathcal{T} is *finitely axiomatizable*. Axiomatizable implies enumerable, since a program can enumerate the axioms and proofs from the axioms in parallel to enumerate the theory. If the theory is also complete, then it is decidable because the above procedure will always return when it reaches φ or $\neg\varphi$.

Theorem (Löwenheim-Skolem). *If Γ is a satisfiable set of sentences in a countable language, then Γ has a countable model.*

Definition (Categorical). *\mathcal{T} is κ -categorical if all models of \mathcal{T} of cardinality κ are isomorphic.*

Theorem (Loś-Vaught Test). *If \mathcal{T} is a theory in a countable language, and \mathcal{T} has no finite models and \mathcal{T} is \aleph -categorical for some infinite cardinal \aleph , then \mathcal{T} is complete.*

In general, theories are not κ -categorical. Speaking very fuzzily, first-order logic tends to have difficulty capturing infinite structures up to isomorphism, distinguishing between sizes of infinity, etc. For instance, the class of all finite structures is not definable in first-order logic.

The structure of \mathbb{N} also cannot be defined up to isomorphism. Suppose we attempt to build \mathbb{N} using the constant 0, the successor function S , and a suitable set of axioms. We can create a theory which \mathbb{N} models, but we cannot specify that these are the only elements because we cannot write " $\forall x \exists n (S^n 0 = x)$ " or " $\forall x (x = 0 \vee x = S0 \vee x = SS0 \vee \dots)$ ". Neither can be expressed in first order logic with finite sentences, so there are always other structures that the theory models.

Let $\mathfrak{N} = (\mathbb{N}; 0, S, +, \times, E)$ be the usual structure of number theory, where S means successor, and E means exponentiation.

Theorem (Gödel's First Incompleteness Theorem). *$Th\mathfrak{N}$ is not recursively axiomatizable.*

The proof involves developing a method to make statements about the language within the language itself and then constructing a sentence stating "this sentence is not provable", which must be true but not provable. The subset of \mathfrak{N} consisting of $(\mathbb{N}; 0, S, <, +)$ is decidable, but adding \times to the structure makes it undecidable.

Theorem (Gödel's Second Incompleteness Theorem). *If \mathcal{T} is a "sufficiently strong" axiomatizable theory, then $\mathcal{T} \vdash Cons\mathcal{T} \Leftrightarrow \mathcal{T}$ is inconsistent.*

\mathfrak{N} is sufficiently strong, so it proves its own consistency iff it is inconsistent.

CNF and Skolemization

Definition (Conjunctive normal form (CNF)). A sentence is in conjunctive normal form (CNF) if it is of the form $(c_{11} \vee c_{12} \vee \dots) \wedge (c_{21} \vee c_{22} \vee \dots) \wedge (c_{31} \vee c_{32} \vee \dots)$ with each c_{ij} being a literal. Literals are atomic sentences or negations thereof.

Algorithm (Converting to CNF and removing quantifiers).

1. Eliminate implications.
2. Move negations \neg inwards.
3. Standardize variables: give each variable a different name for each quantifier scope it is in.
4. Skolemize: replace each existentially quantified variable by a function of the universally quantified variables in whose scope it is.
5. Drop universal quantifiers (and understand that they are still there: namely implicitly at the beginning of the sentence).
6. Distribute \wedge over \vee .

Unification

Definition (Unification). Unification means finding a substitution that makes different logical expressions look identical.

Definition (Unifier). A unifier for the pair of expressions (ϕ, ψ) is a variable substitution θ such that $\phi^\theta = \psi^\theta$.

Definition (Generality of unifiers). 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 $\theta_1 \sqsupset \eta$ to refer to the substitution composition $\eta \circ \theta_1$.)

Theorem (Most General Unifier (MGU)). If a pair of expressions can be unified, they have a most general unifier (MGU) which is unique up to renaming of variables.

Algorithm (Finding an MGU). From AIMA, Figure 9.1¹.

```
function UNIFY( $x, y, \theta$ ) returns a substitution to make  $x$  and  $y$  identical
  inputs:  $x$ , a variable, constant, list, or compound
            $y$ , a variable, constant, list, or compound
  if  $\theta = \text{failure}$  then return failure
  else if  $x = y$  then return  $\theta$ 
  else if VARIABLE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )
  else if VARIABLE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )
  else if COMPOUND?( $x$ ) and COMPOUND?( $y$ ) then
    return UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))
  else if LIST?( $x$ ) and LIST?( $y$ ) then
    return UNIFY(REST[ $x$ ], REST[ $y$ ], UNIFY(FIRST[ $x$ ], FIRST[ $y$ ],  $\theta$ ))
  else return failure
function UNIFY-VAR( $var, x, \theta$ ) returns a substitution
```

¹ Slightly modified: Leslie Kaelbling pointed out that OCCUR-CHECK? has to take x^θ instead of x as argument.

inputs: *var*, a variable
x, any expression
θ, the substitution built up so far
if $\{var/val\} \in \theta$ **then return** UNIFY(*val*, *x*, *θ*)
else if $\{x/val\} \in \theta$ **then return** UNIFY(*var*, *val*, *θ*)
else if OCCUR-CHECK?(*var*, x^θ) **then return** failure
else return add $\{var, x\}$ to *θ*

Well-founded relations

Definition (Well-founded relation). 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 $(\forall u \in \mathbf{Z}: obj(t_u) \wedge t_{u+1} < t_u)$.

Theorem (Properties of well-founded relations). Any well-founded relation is irreflexive and asymmetric (but not necessarily transitive).

Definition (Union relation). Given two (not necessarily well-founded) relations, their union relation is defined as follows:

$$\forall obj\ x,y: x <_{\text{union}} y : \Leftrightarrow (obj_1(x) \wedge obj_2(y)) \\ \vee (obj_1(x) \wedge obj_1(y) \wedge x <_1 y) \\ \vee (obj_2(x) \wedge obj_2(y) \wedge x <_2 y)$$

That is, both $<_1$ and $<_2$ are preserved, but in addition all *obj₁*-objects are smaller than all *obj₂*-objects.

Theorem (Well-foundedness of union relation). If *obj₁* and *obj₂* characterize disjoint classes of objects and are both well-founded, then their union relation is well-founded too.

Polarity

Definition (Polarity). Each subsentence of a sentence *S* has a polarity: +, −, ± (there is no distinction between ± and ∓). These are the recursive rules for assigning polarities:

- The sentence *S* itself has positive polarity: S^+
- $[\neg F]^\pi \Rightarrow [\neg F^\pi]$
- $[F \wedge G]^\pi \Rightarrow [F^\pi \wedge G^\pi]$
- $[F \vee G]^\pi \Rightarrow [F^\pi \vee G^\pi]$
- $[F \Rightarrow G]^\pi \Rightarrow [F^\pi \Rightarrow G^\pi]$
- $[\text{if } F \text{ then } G \text{ else } H]^\pi \Rightarrow [\text{if } F^\pm \text{ then } G^\pi \text{ else } H^\pi]$
- $[F \Leftrightarrow G]^\pi \Rightarrow [F^\pm \Leftrightarrow G^\pm]$

Example (Polarity). $\left[\left[[\neg P^+]^- \vee Q^- \right]^- \Rightarrow \left[Q^+ \wedge [P^\pm \Leftrightarrow [\neg Q^\pm]^\pm]^\pm \right]^\pm \right]^\pm$

Definition ((Strict) positivity/negativity). An occurrence of a variable that is + or ± is a positive occurrence, and one that is − or ∓ is a negative occurrence. In addition, one that is + is a strictly positive occurrence, and one that is − is a strictly negative occurrence.

Theorem (Replacement of strictly positive occurrences). 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 (Replacement of strictly negative occurrences). 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, $S(E^-) \Rightarrow S(F^-)$).

Deductive tableaux

Definition (Tableau). A tableau is a table of sentences with two columns, assertions on the left and goals on the right. There is either an assertion or a goal in each row, but not both.

Example (Tableau).

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

Definition (Semantics of tableaux). 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 $(A_1 \wedge A_2 \wedge \dots \wedge A_m) \Rightarrow (G_1 \vee G_2 \vee \dots \vee G_n)$.

Theorem (Tableaux). You can move a sentence from one column of a tableau to the other (yielding an equivalent tableau) if you negate it.

Definition (Sound deduction rule for tableaux). A sound deduction rule for tableaux is one that transforms a tableau into one that is equivalent to the original.

Theorem (Resolution rule for tableaux). The resolution rule for tableaux is a sound rule.

- **AA-version (assertion-assertion):** 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.
- **GG-version (goal-goal):** If a tableau contains the goals $G_1[P]$ and $G_2[P]$ ($G_1[P]=G_2[P]$ is allowed), you may add $(G_1[\text{false}] \wedge G_2[\text{true}])$ (replace *all* occurrences of P within G_1 and G_2 by *false* and *true*, respectively) as a goal to the tableau.
- **AG-version (assertion-goal):** If a tableau contains the assertion $A[P]$ and the goal $G[P]$, you may add $(\neg A[\text{false}] \wedge G[\text{true}])$ (replace *all* occurrences of P within A and G by *false* and *true*, respectively) as a goal to the tableau.
- **GA-version (goal-assertion):** If a tableau contains the goal $G[P]$ and the assertion $A[P]$, you may add $(G[\text{false}] \wedge \neg A[\text{true}])$ (replace *all* occurrences of P within G and A by *false* and *true*, respectively) as a goal to the tableau.