Semantic Theory
Entailment & Deduction

Manfred Pinkal
Stefan Thater

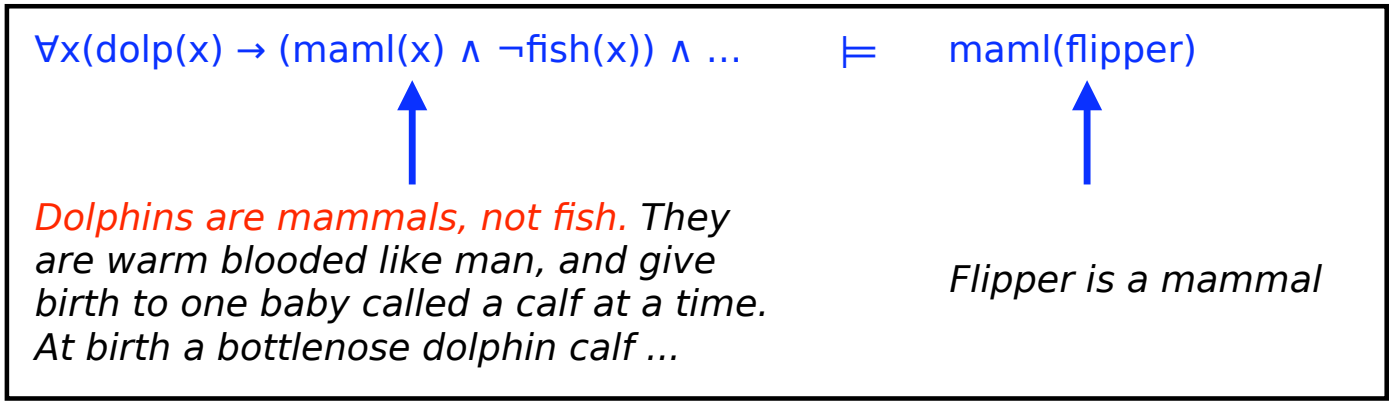
Universität des Saarlandes

2009-05-05

Why Logic?

- Logic supports precise, consistent and controlled meaning representation via truth-conditional interpretation.
- Logic provides deduction systems to model inference processes, controlled by a formal entailment concept.
- Logic supports uniform modelling of the semantic composition process (⇒ type theory)

Entailment & Deduction



(1) ∀x(dolp(x) → (maml(x) ∧ ¬fish(x))

(2) dolp(flipper) → (maml(flipper) ∧ ¬fish(flipper))

(3) dolp(flipper)

(4) maml(flipper) ∧ ¬fish(flipper)

(5) maml(flipper)

Entailment

- A formula Φ is true in a model structure M iff $[\![\Phi]\!]^{M,g} = 1$ for every variable assignment g.
- A formula Φ is satisfiable iff there is at least one model structure M such that φ is true in M.
- A set of formulas Γ is (simultaneously) satisfiable iff there is a model structure M s.t. every formula in Γ is true in M.
  - We also say that M satisfies Γ, or M is a model of Γ
- A set of formulas Γ entails a formula Φ ($\Gamma \models \Phi$) iff Φ is true in every model structure that satisfies Γ.

Entailment & Deduction

- A set of formulas Γ entails a formula Φ ($\Gamma \models \Phi$) iff Φ is true in every model structure that satisfies Γ.
- A deduction system is a collection of rules that derive formulas from sets of formulas (premisses).
- A formula φ is derivable (deducible) from a set of formulas Γ ($\Gamma \vdash \varphi$) in a given deduction system, iff one can obtain φ starting from Γ, by using deduction rules and possibly axioms of that deduction system.
- A formula φ is provable ($\vdash \varphi$) iff $\varnothing \vdash \varphi$
- A set of formulas Γ is inconsistent iff there is a formula φ such that $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$. Otherwise, it is consistent.

Soundness & Completeness

- A deduction system is sound, if we can derive only semantically entailed formulae from a set of premisses:
  - if $\Gamma \vdash_S \varphi$, then $\Gamma \models \varphi$
- A deduction system complete, if it provides derivations for all entailed formulae from a set of premisses.
  - if $\Gamma \models \varphi$, then $\Gamma \vdash_S \varphi$
- For sound and complete deduction systems, the proof-theoretic and semantic concepts coincide:
  - Provability ↔ Validity
  - Derivability / Deducibility ↔ Entailment
  - Consistency ↔ Satisfiability

Axioms and deduction rules

- Deduction calculi are typically made up of
  - axioms (which can be unconditionally used in every proof)
  - deduction rules
- Example for a frequently used axiom:
  - Tertium non datur: $A \vee \neg A$
- Example for a frequently used deduction rule:
  - Modus Ponens: $A \rightarrow B, A \vdash B$

Deduction systems

- Theorem provers (in the narrow sense) typically with
  - Refutation method
  - Resolution proof procedure
  - Examples: Vampire, SPASS, BLIKSEM, OTTER
- Model generators
  - Check consistency: output is YES, if the hypothesis is consistent with the premisses
    - … plus a model for $\Gamma \cup \{\varphi\}$ as an important side effect.
  - Examples: MACE, KIMBA

Resolution – Preliminaries

- A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses.
  - A literal is an atomic formula, or its negation.
  - A clause is a disjunction of literals.
- Every formula can be converted into an equivalent CNF-formula.

Resolution (Propositional Logic)

- Single deduction rule:

$$\frac{(A_1 \vee \ldots \vee A_i \vee \ldots \vee A_n), \qquad (B_1 \vee \ldots \vee B_j \vee \ldots \vee B_m)}{A_1 \vee \ldots \vee A_{i-1} \vee A_{i+1} \vee \ldots \vee A_n \vee B_1 \vee \ldots \vee B_{j-1} \vee B_{j+1} \vee \ldots \vee B_m}$$

- where A_i is the complement of B_j
  - $A_i = p$ and $B_j = \neg p$ or vice versa (p a propositional variable)
- The clause produced by the resolution rule is called the resolvent of the two input clauses
- The dividing line stands for "$\vdash$"

Resolution (Propositional Logic)

- Single deduction rule:

$$\frac{(A_1 \vee \ldots \vee A_i \vee \ldots \vee A_n), \qquad (B_1 \vee \ldots \vee B_j \vee \ldots \vee B_m)}{A_1 \vee \ldots \vee A_{i-1} \vee A_{i+1} \vee \ldots \vee A_n \vee B_1 \vee \ldots \vee B_{j-1} \vee B_{j+1} \vee \ldots \vee B_m}$$

- If R is the resolvent of two clauses in some CNF-formula F, then F and F ∧ R are equivalent.
  - In other words: R is entailed by F.
- A CNF-formula F is unsatisfiable iff the empty clause (□) can be deduced from F using the resolution rule.

Resolution Algorithm

- Convert all sentences to CNF
- Negate the desired conclusion (converted to CNF)
- Apply resolution rule until
  - a contradiction (empty clause) is derived.
  - rule cannot be applied

$(A \vee B) \wedge (A \rightarrow C) \wedge (B \rightarrow C) \vdash C$

(1)	$A \vee B$	premise
(2)	$\neg A \vee C$	premise
(3)	$\neg B \vee C$	premise
(4)	$\neg C$	negated conclusion
(5)	$C \vee B$	1, 2
(6)	$\neg A$	2, 4
(7)	$\neg B$	3, 4
(8)	C	5, 7
(9)	$\square$	4, 8

Predicate Logic (Variant)

- Terms:
  - $t ::= x \mid f_k(t_1, \ldots, t_k)$
  - Terms are either variables or an *n*-ary function symbol applied to *n* terms. Constants are zero-place function symbols.
- Interpretation of terms:
  - $V_M(f_k) = n$-ary function $U_M^n \rightarrow U_M$
  - $[\![f_k(t_1, \ldots, t_k)]\!]^{M,g} = V_M(f_k)([\![t_1]\!]^{M,g}, \ldots, [t_k]\!]^{M,g})$

Prenex Normal Form

- Formulas in prenex normal form: $Q_1x_1 \ldots Q_nx_n\ F$
  - $Q_i \in \{\forall, \exists\}$
  - F contains no quantifier symbol
- Every first-order formula is logically equivalent to some formula in prenex normal form.
  - $\neg\forall x\varphi \Leftrightarrow \exists x\neg\varphi$
  - $\neg\exists x\varphi \Leftrightarrow \forall x\neg\varphi$
  - $(\forall x\varphi \wedge \Psi) \Leftrightarrow \forall x(\varphi \wedge \Psi)$
  - $(\exists x\varphi \rightarrow \Psi) \Leftrightarrow \forall x(\varphi \rightarrow \varphi)$
  - ...

Skolem Normal Form

- A first order formula is in Skolem normal form if it is in conjunctive prenex normal form with only universal first-order quantifiers.
- Every first-order formula can be converted into Skolem normal form while not changing its satisfiability:
  - replace every occurrence of an existentially quantified variable y with a term $f(x_1, \ldots, x_k)$, where
    - f is a new function symbol
    - $x_1, \ldots, x_k$ are the variables that are universally quantified and whose quantifiers precede that of y.
- Example: $\exists x \forall y \exists z \forall w\ P(x, y, z, w) \Rightarrow \forall y \forall w\ P(a, y, f(y), w)$

Skolem Normal Form

- Every first-order formula can be converted into Skolem normal form while not changing its satisfiability.
- The resulting formula is not necessarily equivalent to the original one, …
- … but is equisatisfiable with it: it is satisfiable if and only if the original one is.
- Formulas in Skolem normal form can be represented by their matrix (without quantifiers).

Substitutions

- If t is a term and φ is a formula (possibly) containing the variable x, then φ[t/x] is the result of replacing all free occurrences of x in φ by t.
- This replacement results in a formula that logically follows the original one provided that no free variable of t becomes bound in this process.
- A variable y is free for x in φ iff no free occurrence of y in φ falls in the scope of a quantifier ∀x or ∃x.
- Renaming: ∀xφ is equivalent to ∀yφ[y/x]
  - provided x free for y in φ

Substitutions & Unifiers

- Expressions A and B are unifiable iff there exists a substitution (unifier) σ such that Aσ = Bσ
- g is a most general unifier of A and B iff for all unifiers σ of A and B, there exists a substitution σ’ such that
  - Aσ = (A g) σ’ and
  - Bσ = (B g) σ’

Resolution (Sketch)

- While there are clauses that can be resolved:
  - Select 2 clauses K_1 and K_2 such that K_1 and K_2 have no variables in common
    - rename variables if necessary
  - Resolve K1 and K2 (after unification) and apply the unification substitution to the result.

A Simple Example

(1)	$\neg H(x) \vee M(x)$	premise
(2)	$H(s)$	premise
(3)	$\neg M(s)$	negated conclusion
(4)	$M(s)$	1, 2, $\sigma = [s/x]$
(5)	$\square$	3, 4

$\forall x(H(x) \rightarrow M(x)$	“all man are mortal”
$H(s)$	“Socrates is a man”
$M(s)$	“Socrates is mortal”

Another Example [⇒ Whiteboard]

(1) $\neg S(x) \vee A(x)$

(2) $\neg T(x, y) \vee \neg A(y) \vee A(x)$

(3) $S(f)$

(4) $T(b, f)$

(5) $\neg A(b)$

(6) ...

$\forall x(S(x) \rightarrow A(x))$

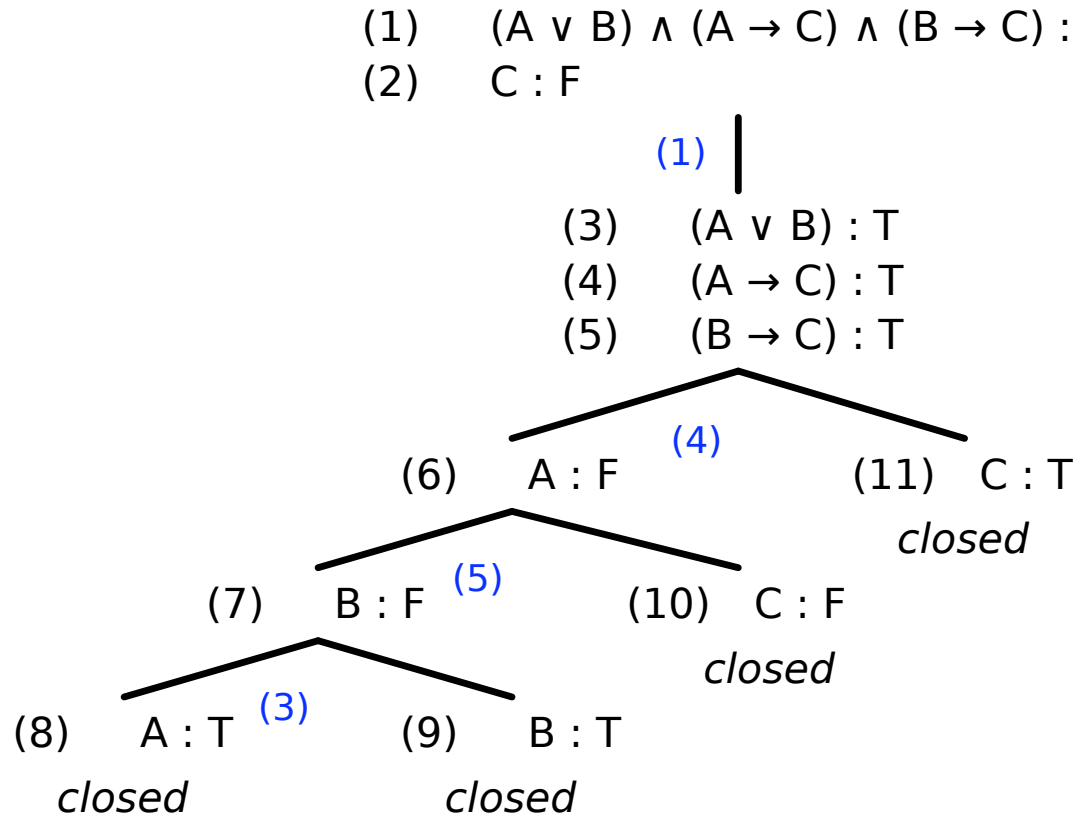
$\forall x((\exists y T(x,y) \wedge A(y)) \rightarrow A(x))$

$S(f)$

$T(b, f)$

$A(b)$

Semantic Tableaux



Semantic Tableaux

- Initialize tableaux:
  - initial branch: premise and negated conclusion
- Expand tableaux:
  - choose a branch and a formula in it
  - expand tableaux applying rule to the selected rule
- Terminate:
  - all rules have been applied, or
  - tableaux is closed
- A branch is closed if it contains both A : T and A : F for some atom A.
- A tableaux is closed if all branches are closed.

Expansion Rules

$A \wedge B : T$	$\Longrightarrow$	$A : T, B : T$	
$A \wedge B : F$	$\Longrightarrow$	$A : F \mid B : F$	
$A \vee B : T$	$\Longrightarrow$	$A : T \mid B : T$	
$A \vee B : F$	$\Longrightarrow$	$A : F, B : F$	
$A \rightarrow B : T$	$\Longrightarrow$	$A : F \mid B : T$	
$A \rightarrow B : F$	$\Longrightarrow$	$A : T, B : F$	
$\neg A : T$	$\Longrightarrow$	$A : F$	
$\neg A : F$	$\Longrightarrow$	$A : T$	
$\forall x\ A : T$	$\Longrightarrow$	$A[x/a] : T$	for arbitrary a
$\forall x\ A : F$	$\Longrightarrow$	$A[x/a] : F$	for new a
$\exists x\ A : T$	$\Longrightarrow$	$A[x/a] : T$	for new a
$\exists x\ A : F$	$\Longrightarrow$	$A[x/a] : F$	for arbitrary a

Examples [⇒ Whiteboard]

- $\forall x(F(x) \rightarrow G(c)) \models \exists x F(x) \rightarrow G(c)$
- $\forall x(F(x) \rightarrow \neg G(x)) \wedge \neg \forall x \neg F(x) \models \exists x \neg G(x)$
- $\exists x F(x) \not\models \forall x F(x)$
- $\exists x \forall y R(x,y) \not\models \forall x R(x,x)$

Some Remarks

- Propositional logic is decidable, but NP-complete. A proof can require exponential time in the number of clauses.
- FOL is undecidable
  - provable / valid formulas are recursively enumerable
- Implemented deduction systems – originally mostly developed for purposes of mathematical theorem proving
  - allow very efficient derivability / entailment checks.
- Theorem provers for fragments of FOL are yet more efficient.
  - horn-clause logic
  - description logics