華東師範大學
EAST CHINA NORMAL UNIVERSITY

Logic in Computer Science

Lecture 05

Semantical Consequence and Satisfability

Li Qin
Associate Professor
School of Software Engineering

Sentences

Definition

Let S be a signature. An S-sentence is an S-formula φ with $\text{free}(\varphi) = \emptyset$.

Examples:

- $\forall x\, \exists y\, R(x, y)$ is an $\{R\}$-sentence.
- $\exists y\, R(x, y)$ is *not* an $\{R\}$-sentence.

Recall: If φ is an S-sentence and $\mathcal{M}$ an S-structure, then $\mathcal{M} \models \varphi$ means that φ is satisfied in $\mathcal{M}$.

Semantic Consequence: Motivation

Assumption 1: $\mathit{Instructor}(\mathit{john})$

Assumption 2: $\forall x\big(\mathit{Instructor}(x) \to \exists y\,\mathit{Teaches}(x,y)\big)$

Does $\exists z\,\mathit{Teaches}(\mathit{john},z)$ follow from these two sentences?

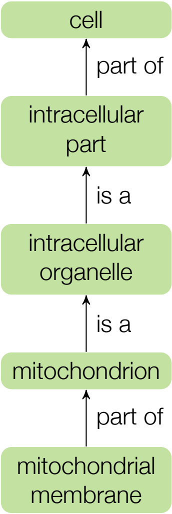
- Intuitively: yes!
- But what does it mean precisely that a sentence follows from a set of sentences?

Semantic Consequence: Motivation

Tiny part of *Gene Ontology*:*

$PartOf(mitochondrial\_membrane, mitochondrion)$
$IsA(mitochondrion, intracellular\_organelle)$
$IsA(intracellular\_organelle, intracellular\_part)$
$PartOf(intracellular\_part, cell)$

$\forall x \forall y \forall z \big((PartOf(x,y) \land IsA(y,z)) \to PartOf(x,z)\big)$
$\forall x \forall y \forall z \big((PartOf(x,y) \land PartOf(y,z)) \to PartOf(x,z)\big)$



Does $PartOf(mitochondrial\_membrane, cell)$ follow from this set of sentences?

What does “follow” mean?

* http://www.geneontology.org/

Semantic Consequence

Definition

Fix a signature S, a set F of S-sentences, and an S-sentence φ.

We say φ follows from F (or φ is a semantic consequence of F) if for all S-structures $\mathcal{M}$:

$$\text{If for all } \psi \in F \text{ we have } \mathcal{M} \models \psi, \text{ then } \mathcal{M} \models \varphi.$$

This is denoted by $F \models \varphi$.

Note: We use $\models$ both for the satisfaction relation and for the semantic consequence relation. The left hand side of $\models$ determines which of the two relations we mean.

Example

Signature: $S = \{P, c\}$

$$\{P(c)\} \models \exists x\, P(x)$$

Proof: We have to show that the following is true for all S-structures $\mathcal{M}$: If $P(c)$ is satisfied in $\mathcal{M}$, then $\exists x\, P(x)$ is satisfied in $\mathcal{M}$.

Let $\mathcal{M}$ be an S-structure, and assume that $\mathcal{M} \models P(c)$. Note that $(\mathcal{M}, a) \models P(x)$, where a is an assignment in $\mathcal{M}$ with $a(x) = c^{\mathcal{M}}$.

Since $a = a[x \mapsto c^{\mathcal{M}}]$, we have $(\mathcal{M}, a[x \mapsto c^{\mathcal{M}}]) \models P(x)$. The definition of the satisfaction relation thus yields $(\mathcal{M}, a) \models \exists x\, P(x)$. Since $\exists x\, P(x)$ is a sentence, we can write this as $\mathcal{M} \models \exists x\, P(x)$. $\square$

Example 2

Signature: $S = \{P, Q, c\}$

$$\{P(c), \forall x(P(x) \to Q(x))\} \models Q(c)$$

Proof: We have to show that the following is true for all S-structures $\mathcal{M}$: If the two sentences in the set on the left-hand side of $\models$ are satisfied in $\mathcal{M}$, then $Q(c)$ is satisfied in $\mathcal{M}$.

To this end, let $\mathcal{M}$ be an S-structure, and assume that $\mathcal{M} \models P(c)$ and

$$\mathcal{M} \models \forall x\big(P(x) \to Q(x)\big).$$

The latter implies

$$(\mathcal{M}, a) \models P(x) \to Q(x) \qquad (\star)$$

for all assignments a in $\mathcal{M}$ and, in particular, for any assignment a with $a(x) = c^{\mathcal{M}}$. Fix such an assignment a. Since $\mathcal{M} \models P(c)$, we have $(\mathcal{M}, a) \models P(x)$. Together with $(\star)$, this implies $(\mathcal{M}, a) \models Q(x)$. Since $a(x) = c^{\mathcal{M}}$, we obtain $\mathcal{M} \models Q(c)$. $\square$

Example 3

Signature: $S = \{I, T, john\}$

$$\{I(john), \forall x\big(I(x) \to \exists y\, T(x,y)\big)\} \models \exists z\, T(john, z)$$

Proof: We have to show that the following is true for all S-structures $\mathcal{M}$: If the two sentences in the set on the left-hand side of $\models$ are satisfied in $\mathcal{M}$, then $\exists z\, T(john, z)$ is satisfied in $\mathcal{M}$.

To this end, let $\mathcal{M}$ be an S-structure, and assume that $\mathcal{M} \models I(john)$ and

$$\mathcal{M} \models \forall x\big(I(x) \to \exists y\, T(x,y)\big).$$

Note that the latter implies

$$(\mathcal{M}, a) \models I(x) \to \exists y\, T(x,y) \qquad (\star)$$

for all assignments a in $\mathcal{M}$ and, in particular, for any assignment a with $a(x) = john^{\mathcal{M}}$. Fix such an assignment a. Since $\mathcal{M} \models I(john)$, we have $(\mathcal{M}, a) \models I(x)$. Together with $(\star)$, this implies $(\mathcal{M}, a) \models \exists y\, T(x,y)$. Since $a(x) = \text{john}^{\mathcal{M}}$, we obtain $\mathcal{M} \models \exists y\, T(john, y)$. $\square$

Example 4

Signature: $S = \{P, Q, c\}$

$$\{P(c) \vee Q(c)\} \not\models P(c)$$

"$P(c)$ does not follow from $P(c) \vee Q(c)$"

Proof: We provide a counter-example for $\{P(c) \vee Q(c)\} \models P(c)$, that is, an S-structure $\mathcal{M}$ such that $\mathcal{M} \models P(c) \vee Q(c)$, but $\mathcal{M} \not\models P(c)$.

One such counter-example is the S-structure $\mathcal{M}$ with $\mathrm{dom}(\mathcal{M}) = \{1\}$, $P^{\mathcal{M}} = \emptyset$, $Q^{\mathcal{M}} = \{1\}$, and $c^{\mathcal{M}} = 1$. Since $c^{\mathcal{M}} = 1 \in Q^{\mathcal{M}}$, we have $\mathcal{M} \models Q(c)$ and hence $\mathcal{M} \models P(c) \vee Q(c)$. On the other hand, since $c^{\mathcal{M}} = 1 \notin P^{\mathcal{M}}$, we have $\mathcal{M} \not\models P(c)$. $\square$

Semantic Equivalence

Definition

Let S be a signature, and φ, ψ two S-sentences.

We say φ and ψ are equivalent if they are satisfied in the same S-structures, i.e., if for all S-structures $\mathcal{M}$:

$$\mathcal{M} \models \varphi \text{ if and only if } \mathcal{M} \models \psi.$$

This is denoted by $\varphi \equiv \psi$.

Observation: $\varphi \equiv \psi$ if and only if $\{\varphi\} \models \psi$ and $\{\psi\} \models \varphi$.

Examples

- $\neg\exists x\, \varphi \equiv \forall x\, \neg\varphi$
  $\neg\forall x\, \varphi \equiv \exists x\, \neg\varphi$

- $\forall x\, \varphi \wedge \forall x\, \psi \equiv \forall x(\varphi \wedge \psi)$
  $\forall x\, \varphi \vee \forall x\, \psi \not\equiv \forall x(\varphi \vee \psi)$

- $\exists x\, \varphi \wedge \exists x\, \psi \not\equiv \exists x(\varphi \wedge \psi)$
  $\exists x\, \varphi \vee \exists x\, \psi \equiv \exists x(\varphi \vee \psi)$

- $\exists x\, \forall y\, \varphi \not\equiv \forall y\, \exists x\, \varphi$

Satisfiability and Tautologies

Definition

Let S be a signature, and φ an S-sentence.

- φ is satisfiable if there is an S-structure $\mathcal{M}$ with $\mathcal{M} \models \varphi$.
- φ is a tautology if for all S-structures $\mathcal{M}$ we have $\mathcal{M} \models \varphi$.

Proposition

Let S be a signature, and let $\varphi, \psi_1, \ldots, \psi_n$ be S-sentences.

1. $\{\psi_1, \ldots, \psi_n\} \models \varphi \iff \psi_1 \wedge \cdots \wedge \psi_n \wedge \neg\varphi$ *is not satisfiable.*
2. $\{\psi_1, \ldots, \psi_n\} \models \varphi \iff (\psi_1 \wedge \cdots \wedge \psi_n) \to \varphi$ *is a tautology.*

Proof of Part 1

"$\Longrightarrow$" Suppose $\{\psi_1, \dots, \psi_n\} \models \varphi$. We have to show that for all S-structures $\mathcal{M}$, we have $\mathcal{M} \not\models \psi_1 \wedge \dots \wedge \psi_n \wedge \neg\varphi$.
To this end, let $\mathcal{M}$ be an S-structure.

Case 1: $\mathcal{M} \models \psi_i$ for all $i \in \{1, \dots, n\}$.
Since $\{\psi_1, \dots, \psi_n\} \models \varphi$, this implies $\mathcal{M} \models \varphi$, hence $\mathcal{M} \not\models \neg\varphi$.
In particular, $\mathcal{M} \not\models \psi_1 \wedge \dots \wedge \psi_n \wedge \neg\varphi$.

Case 2: $\mathcal{M} \not\models \psi_i$ for some $i \in \{1, \dots, n\}$.
This immediately yields $\mathcal{M} \not\models \psi_1 \wedge \dots \wedge \psi_n \wedge \neg\varphi$.

Proof of Part 1

"$\Longleftarrow$" Suppose $\mathcal{M} \not\models \psi_1 \wedge \cdots \wedge \psi_n \wedge \neg\varphi$ for all S-structures $\mathcal{M}$. We have to show that $\{\psi_1, \ldots, \psi_n\} \models \varphi$.

To this end, let $\mathcal{M}$ be an S-structure such that $\mathcal{M} \models \psi_i$ for all $i \in \{1, \ldots, n\}$. In other words, $\mathcal{M} \models \psi_1 \wedge \cdots \wedge \psi_n$. Since $\mathcal{M} \models \psi_1 \wedge \cdots \wedge \psi_n$, but $\mathcal{M} \not\models \psi_1 \wedge \cdots \wedge \psi_n \wedge \neg\varphi$, we must have $\mathcal{M} \not\models \neg\varphi$. The latter is equivalent to $\mathcal{M} \models \varphi$.

We have thus shown that for all S-structures $\mathcal{M}$, if $\mathcal{M} \models \psi_i$ for all $i \in \{1, \ldots, n\}$, then $\mathcal{M} \models \varphi$. This proves $\{\psi_1, \ldots, \psi_n\} \models \varphi$. $\square$

Satisfiability: Examples

1. $\exists x \big(P(x) \land \neg P(x)\big)$ is not satisfiable.
2. $\forall x\, \exists y\, R(x, y) \land \neg \forall u\, \exists v\, R(v, u)$ is satisfiable.
3. $\forall x\, P(x) \land \exists x\, \neg P(x)$ is not satisfiable.

Proof of 2

Claim: $\forall x\, \exists y\, R(x, y) \wedge \neg \forall u\, \exists v\, R(v, u)$ is satisfiable.

Proof: We have to show that there is an $\{R\}$-structure $\mathcal{M}$ such that the sentence is satisfied in $\mathcal{M}$.

To this end, let $\mathcal{M}$ be the $\{R\}$-structure with:

- domain $\{1, 2\}$,
- $R^{\mathcal{M}} = \{(1, 1), (2, 1)\}$.

Then,

- $\mathcal{M} \models \forall x\, \exists y\, R(x, y)$: For all $d \in \{1, 2\}$, there exists an element $d' \in \{1, 2\}$ with $(d, d') \in R^{\mathcal{M}}$;
- $\mathcal{M} \not\models \forall u\, \exists v\, R(v, u)$: For $d' = 2$, there does not exist an element $d \in \{1, 2\}$ with $(d, d') \in R^{\mathcal{M}}$.

Hence, $\mathcal{M} \models \forall x\, \exists y\, R(x, y) \wedge \neg \forall u\, \exists v\, R(v, u)$. $\square$

Proof of 3

Claim: $\forall x\, P(x) \land \exists x\, \neg P(x)$ is not satisfiable.

Proof: By contradiction. Suppose that the sentence is satisfiable. Then, there is a $\{P\}$-structure $\mathcal{M}$ with $\mathcal{M} \models \forall x\, P(x) \land \exists x\, \neg P(x)$. Recall that the latter denotes:

$$(\mathcal{M}, a) \models \forall x\, P(x) \land \exists x\, \neg P(x) \qquad (\star)$$

for an arbitrary assignment a in $\mathcal{M}$. Now, $(\star)$ implies:

1. $(\mathcal{M}, a) \models \forall x\, P(x)$ and
2. $(\mathcal{M}, a) \models \exists x\, \neg P(x)$.

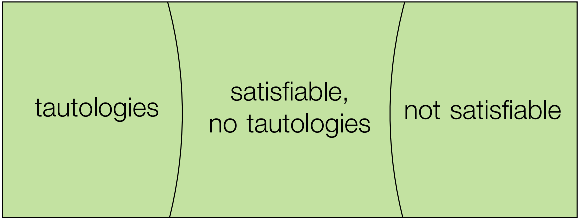
The second item states that there is a $d \in \mathrm{dom}(\mathcal{M})$ with $(\mathcal{M}, a[x \mapsto d]) \not\models P(x)$. This implies $(\mathcal{M}, a) \not\models \forall x\, P(x)$, and hence contradicts the first item above. $\square$

Satisfiability vs Tautologies

Proposition

Let S be a signature, and let φ be an S-sentence. Then, the following are equivalent:

1. *φ is a tautology.*
2. *$\neg\varphi$ is not satisfiable.*



Satisfiability Problem for Predicate Logic

Satisfiability Problem (for Predicate Logic)

Input: a signature S, and an S-sentence φ

Task: If φ is satisfiable, output "yes", otherwise "no"

Recall: The analogous problem for propositional logic can be solved, e.g., via truth tables or the tableau algorithm.

Theorem

The satisfiability problem for predicate logic is undecidable:

There is no algorithm that solves the satisfiability problem for predicate logic.

Corollaries

The following problems are also undecidable:

Input:	a signature S, and an S-sentence φ
Task:	If φ is a tautology, output "yes", otherwise "no"

Input:	a signature S, and S-sentences $\psi_1, \ldots, \psi_n$, φ
Task:	If $\{\psi_1, \ldots, \psi_n\} \models \varphi$, output "yes", otherwise "no"

Proof: An algorithm for any of these problems could be used to solve the satisfiability problem (for predicate logic):

- φ is satisfiable $\iff$ $\neg\varphi$ is no tautology;
- φ is satisfiable $\iff$ $\emptyset \not\models \neg\varphi$.

But the satisfiability problem is undecidable. $\square$

Undecidability of Satisfiability – Overview

Theorem

The satisfiability problem for predicate logic is undecidable:

There is no algorithm that solves the satisfiability problem for predicate logic.

Proof Steps:

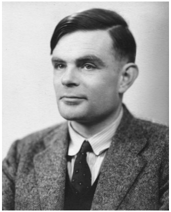
1. Introduce an auxiliary problem, the Halting problem, and prove that it is undecidable.
2. Show that if there was an algorithm for the satisfiability problem, then it could be used to solve the Halting problem (which is impossible by step 1).

Algorithms and Computations

For the proof, we need a precise definition of the concept of algorithm (or computation).

1. Usually: based on Turing machines

   (will be covered in detail in COMP218)



Alan Turing
1912–1954

2. Here (due to time-constraints):

   An algorithm is a program written in some standard programming language. For definiteness, we choose Java.

Decidability

- A decision problem is a problem where the task is to decide if a given input has a certain property.

  Example: Satisfiability problem

- A decision problem is undecidable if there is no algorithm that solves it (and terminates on every input).

  Example: To show that the satisfiability problem is undecidable, we have to rule out the existence of an algorithm that takes as input

  - a signature S and
  - an S-sentence φ

  and outputs "yes" or "no" depending on whether φ is satisfiable.

The Halting Problem

Halting Problem

Input: a Java program P and an input x for P

Task: output "yes" if P terminates when it is run with x as input; otherwise output "no"

- The Halting Problem is a decision problem.
- We assume that every Java program takes only one input (say, the string of command line arguments).

Theorem

The Halting Problem is undecidable.

Proof Sketch

1. Assume the Halting Problem is *not* undecidable. Then, there is a Java program P_{Halt} that solves it.

2. Let P_{new} be a new Java program working as follows:

> Input: a Java program Q
>
> - out $=$ the output of P_{Halt} when it is started with $P := Q$ and $x := Q$
> - **if** out$==$"no" **then** output "yes" **else** run forever

3. P_{new} cannot be a Java program:

$$
\begin{aligned}
& P_{\text{new}} \text{ terminates when it is started with input } P_{\text{new}} \\
\iff & P_{\text{Halt}} \text{ outputs "no" when started with } P := P_{\text{new}} \text{ \& } x := P_{\text{new}} \\
\iff & P_{\text{new}} \text{ does not terminate when it is started with input } P_{\text{new}}.
\end{aligned}
$$

This is a contradiction, so the assumption in 1 is wrong.

Undecidability of the Satisfiability Problem

- For any Java program P and input x, we can construct an S-sentence φ (for a suitable signature S) such that

  $$\varphi \text{ is satisfiable } \iff P \text{ terminates when given } x \text{ as input.}$$

  Intuition: Construct φ in such a way that it is satisfied in an S-structure $\mathcal{M}$ precisely if $\mathcal{M}$ describes an execution of P on input x that finishes after a finite number of steps (i.e., calls `System.exit(...)` at some point).

- Since the Halting Problem is undecidable, we conclude that the satisfiability problem is undecidable.