CSC384
Knowledge Representation
Part 1

Bahar Aameri & Sonya Allin

Summer 2020

Credits

We gratefully acknowledge those who have contributed to these slides, most recently Bahar Aameri, who merged and augmented slides from Yongmei Liu and a CSC384 slide deck historically developed by Craig Boutilier, Fahiem Bacchus, Sheila McIlraith, Sonya Allin, Hojjat Ghaderi, and others. We also acknowledge the use of material written by Michael Winter, and the use of material originating from slides and the book by Ron Brachman and Hector Levesque.

What is Knowledge Representation and Reasoning (KR&R)?
Symbolic encoding of propositions believed by some agent and their manipulation to produce propositions that are believed by the agent but not explicitly stated.

Why KR&R:

- Large amounts of knowledge are used to understand the world around us.
- **Reasoning** provides compression in the knowledge we need to store.
- **Without** reasoning we would have to store an infeasible amount of information:
  Example: Elephants can't fit into teacups, Elephants can't fit into cars, instead of just knowing that larger objects can't fit into smaller objects.

- **Information:**
  (1) Block A is above block B;
  (2) Block B is above block C.

- **Query:** Is A above C?

Given the information, human can easily draw the conclusion.
How can a **machine** do the same?

Introduction

- Tony, Mike, and John are members of the Alpine Club.
- Every member of the Alpine Club who is not a skier is a mountain climber.
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
- Tony likes rain and snow.
- Is there a member of the Alpine Club who is a mountain climber but not a skier?

Logical representations

- are mathematically precise; thus it's possible to analyze their limitations, properties, and complexity of inferences.
- are formal languages; thus computer programs can manipulate sentences in the language.
- typically, have well-developed proof theories: formal procedures for reasoning to produce new sentences.

In this module we will study **First-Order logic (FOL)**, and a reasoning mechanism called **resolution** that operates on First-Order logic.

Review: Propositional Logic – Syntax

Propositional Variable: A variable which takes only **True** or **False** as values.

The set of all propositional formulas is defined recursively as follows:

- Every propositional variable is a propositional formula;
- If φ is a propositional formula, then so is $\neg\varphi$;
- If φ_1 and φ_2 are propositional formulas, then so are
  - $\varphi_1 \wedge \varphi_2$ (**Conjunction**);
  - $\varphi_1 \vee \varphi_2$ (**Disjunction**);
  - $\varphi_1 \rightarrow \varphi_2$ (**Implication**);
  - $\varphi_1 \leftrightarrow \varphi_2$ (**Bi-implication**).

Truth Assignment: A function τ from the propositional variables into the set of truth values $\{T, F\}$.

Let τ be a truth assignment. The extension $\bar{\tau}$ of τ assigns either T or F to every formula and is defined as follows:

- If $A = x$, where x is a variable, then $\bar{\tau}(A) = \tau(x)$.
- $\bar{\tau}(\neg A) = T$ iff $\bar{\tau}(A) = F$;
- $\bar{\tau}(A \wedge B) = T$ iff $\bar{\tau}(A) = T$ and $\bar{\tau}(B) = T$;
- $\bar{\tau}(A \vee B) = T$ iff $\bar{\tau}(A) = T$ or $\bar{\tau}(B) = T$;
- $\bar{\tau}(A \to B) = F$ iff $\bar{\tau}(A) = T$ and $\bar{\tau}(B) = F$.

Example: Let $V = \{p, r, q\}$ be a set of propositional variables and $\tau_1 : V \to \{T, F\}$ and $\tau_2 : V \to \{T, F\}$ be two truth assignments s.t.:

- $\tau_1(p) = T$, $\tau_1(q) = F$, $\tau_1(r) = F$.
- $\tau_2(p) = F$, $\tau_2(q) = T$, $\tau_2(r) = F$.

Then
$\bar{\tau_1}((\neg p \wedge q) \to r) =$

$\bar{\tau_2}((\neg p \wedge q) \to r) =$

A truth assignment τ **satisfies** a formula A iff $\bar{\tau}(A) = T$.
τ satisfies a set Φ of formulas iff τ satisfies all formula in Φ.

A set Φ of formulas is satisfiable iff some truth assignment τ satisfies Φ.
Otherwise, Φ is unsatisfiable.

Example:

$\Phi_1 = \{r \rightarrow (p \land q), \neg p\}$

$\Phi_2 = \{r \rightarrow (p \land q), r \land \neg p\}$

A formula A is a **logical consequence** of Φ (denoted by $\Phi \models A$) iff for every truth assignment τ, if τ satisfies Φ, then τ satisfies A.

Example: Let $\Phi = \{r \rightarrow ((p \wedge q) \vee s), r \wedge p\}$.

Then $\Phi \models$

Limitations of Propositional Language

- **Only Boolean variables**: Without non-Boolean variables **cross references between individuals** in statements are **impossible**.
  Example: 'If a person has a sibling and that sibling has a child, then the person is an aunt or an uncle.'
  S: a person has a sibling.
  C: a sibling has a child.
  A: a person is an aunt or an uncle.

  $S \wedge C \rightarrow A$

  This approach doesn't work:
  person in S and A are not related.
  sibling in S and C are not related.

- **No quantifiers:** To state a property for all (or some) members of the domain we have to **explicitly list** them.
  Example: 'Every member of the Alpine Club who is not a skier is a mountain climber'

First-Order Logic: Syntax

For **first-order logic** following components are required:

- A set V of variables.
- A set F of function symbols.
- A set P of predicate (relation) symbols.

- **Functions** and **variables** are used to construct terms.
  Terms denote elements of the domain.
- **Predicates** are defined over terms.
  Atomic formulas denote properties and relations that hold about the elements in the domain.
- **Predicates** and **terms** are used to construct formulas.
  Other formulas generate more complex assertions by composing atomic formulas.

A set $\mathcal{L}$ of **function** and **predicate symbols** is called a first-order vocabulary.

First-Order Logic: Syntax

Let $\mathcal{L}$ be a set of function and predicate symbols.

1. Every variable is a term.
2. If f is an n-ary function symbol in $\mathcal{L}$ and $t_1, t_2, ..., t_n$ are $\mathcal{L}$-terms, then $f(t_1, t_2, ..., t_n)$ is a $\mathcal{L}$-term.

Note: 0-ary functions symbols are called **constant symbols**.
Example:

First-Order Logic: Syntax

Let $\mathcal{L}$ be a vocabulary. The set of first-order $\mathcal{L}$-formulas is defined recursively:

1. Atomic Formula: $P(t_1, t_2, ..., t_n)$, where P is an n-ary predicate symbol in $\mathcal{L}$ and $t_1, t_2, ..., t_n$ are $\mathcal{L}$-terms.

2. Negation: $\neg f$, where f is a $\mathcal{L}$-formula.

3. Conjunction: $f_1 \wedge f_2 \wedge ... \wedge f_n$, where $f_1, f_2, ..., f_n$ are $\mathcal{L}$-formulas.

4. Disjunction: $f_1 \vee f_2 \vee ... \vee f_n$, where $f_1, f_2, ..., f_n$ are $\mathcal{L}$-formulas.

5. Implication: $f_1 \rightarrow f_2$, where f_1, f_2 are $\mathcal{L}$-formulas.

6. Existential: $\exists x f$, where x is a variable and f is a $\mathcal{L}$-formula.

7. Universal: $\forall x f$, where x is a variable and f is a $\mathcal{L}$-formula.

Converting English to First-Order Language

- **Individuals**: Constants (0-ary Functions)
  - **tony, mike, john**
    rain, snow
- **Types**: Unary Predicates
  - $AC(x)$: x belongs to Alpine Club.
  - $S(x)$: x is a skier.
  - $C(x)$: x is a mountain climber.
- **Relationships**: Binary Predicates
  - $L(x, y)$: x likes y.

Converting English to First-Order Language

- **Basic Facts**:
  - Tony, Mike, and John belong to the Alpine Club:
    $AC(\textbf{tony}), AC(\textbf{mike}), AC(\textbf{john})$
  - Tony likes rain and snow:
    $L(\textbf{tony, rain}), L(\textbf{tony, snow})$
- **Complex Facts**:
  - Every member of the Alpine Club who is not a skier is a mountain climber.
  - Mountain climbers do not like rain, and anyone who does not like snow is not a skier.

- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.

- Is there a member of the Alpine Club who is a mountain climber but not a skier?

Like variables in programming languages, the variables in FOL have a scope which is determined by the quantifiers.

Lexical scope for variables:

$Animal(x) \land \exists x[Human(x) \lor Women(x)]$.

$\exists x[Animal(x) \rightarrow \neg Human(x)] \land \exists x[Human(x) \lor Women(x)]$

First-Order Logic: Semantic

- In the **propositional logic**, a **truth assignment** provides meaning to a formula.

- In **FOL** we can talk about **(non-Boolean) individuals and elements**.
  So the simple universe of truth values is not rich enough to provide a suitable interpretation for FOL formulas.

- We need **more complicated objects** to give meaning to formulas and terms.

- These objects are called **structures**.

First-Order Structures

Let $\mathcal{L}$ be a first-order vocabulary. An $\mathcal{L}$-**structure** $\mathcal{M}$ consists of the following:

1. A **nonempty set** M called the universe (domain) of discourse.
2. For each n-ary **function symbol** $f \in \mathcal{L}$, an associated function $f^{\mathcal{M}} : M^n \to M$.
   Note: If $n = 0$, then f is a constant symbol and $f^{\mathcal{M}}$ is simply an element of M.
   $f^{\mathcal{M}}$ is called the **extension** of the function symbol f in $\mathcal{M}$.
3. For each n-ary **predicate symbol** $P \in \mathcal{L}$, an associated relation $P^{\mathcal{M}} \subseteq M^n$.
   $P^{\mathcal{M}}$ is called the **extension** of the predicate symbol P in $\mathcal{M}$.

First-Order Structures: Example

Blocks World:

Suppose $\mathcal{L}_{BW}$ includes the following symbols:

- **Function Symbols:**
  - $under(x)$: the block immediately under x if x is not on table; x itself otherwise.
- **Predicate Symbols:**
  - $on(x, y)$: x is place (directly) on y.
  - $above(x, y)$: x is above y.
  - $clear(x)$: no blocks are above x.
  - $ontable(x)$: no blocks are under x.

Suppose $\mathcal{L}_{BW}$ includes the following symbols:

- **Function Symbols:**
  - $under(x)$: the block immediately under x if x is not on table; x itself otherwise.
- **Predicate Symbols:**
  - $on(x, y)$: x is place (directly) on y.
  - $above(x, y)$: x is above y.
  - $clear(x)$: no blocks are above x.
  - $ontable(x)$: no blocks are under x.

$\mathcal{M}_1$ is a $\mathcal{L}_{BW}$-structure such that:

$M_1 = \{A, B, C, D\}$

$on^{\mathcal{M}_1} = \{\langle A, B\rangle, \langle B, C\rangle\}$

$above^{\mathcal{M}_1} = \{\langle A, B\rangle, \langle B, C\rangle, \langle A, C\rangle\}$

$clear^{\mathcal{M}_1} = \{A, D\}$

$ontable^{\mathcal{M}_1} = \{C, D\}$

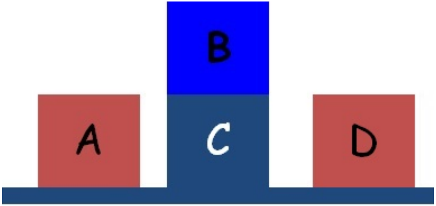
$under^{\mathcal{M}_1}(A) = B$, $under^{\mathcal{M}_1}(B) = C$,

$under^{\mathcal{M}_1}(C) = C$, $under^{\mathcal{M}_1}(D) = D$

Suppose $\mathcal{L}_{BW}$ includes the following symbols:

- **Function Symbols:**
  - $under(x)$: the block immediately under x if x is not on table; x itself otherwise.
- **Predicate Symbols:**
  - $on(x, y)$: x is place (directly) on y.
  - $above(x, y)$: x is above y.
  - $clear(x)$: no blocks are above x.
  - $ontable(x)$: x is placed on the table.

Represent the following configuration by a $\mathcal{L}_{BW}$-structure.



•

Semantic of First-Order Logic: Intuition

Every $\mathcal{L}$-formula becomes either true or false when interpreted by an $\mathcal{L}$-structure $\mathcal{M}$.

That is, the truth value of a first-order formulas A is evaluated w.r.t to a first-order structure $\mathcal{M}$:

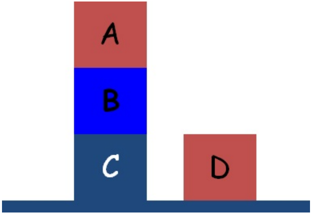
- Terms (variables and functions) of a formula denote elements of the domain. So every term in A must correspond with an element of the universe of $\mathcal{M}$.

- Atomic formulas denote properties and relations that hold about the elements in the domain. $P(t_1, ..., t_n)$ is true in $\mathcal{M}$ if $t_1, ..., t_n$ are related to each other by $P^{\mathcal{M}}$.

- Other formulas generate more complex assertions by composing atomic formulas. Their truth is dependent on the truth of the atomic formulas in them.

Semantic of First-Order Logic: Variable Assignments

Let $\mathcal{M}$ be a structure and X be a set of variables. An **object assignment** σ for $\mathcal{M}$ is a **mapping** from variables in X to the universe of $\mathcal{M}$.



$$X = \{v_1, v_2, v_3, v_4\}$$

$$\sigma(v_1) = D, \qquad \sigma(v_2) = C$$
$$\sigma(v_3) = B, \qquad \sigma(v_4) = A$$

Semantic of First-Order Logic: Variable Assignments

Remember the recursive definition of term:
Let $\mathcal{L}$ be a set of function and predicate symbols.

1. Every variable x is a term.
2. If f is an n-ary function symbol in $\mathcal{L}$ and $t_1, t_2, ..., t_n$ are $\mathcal{L}$-terms, then $f(t_1, t_2, ..., t_n)$ is a $\mathcal{L}$-term.

Let $\mathcal{L}$ be a vocabulary and $\mathcal{M}$ be an $\mathcal{L}$-structure.
The extension $\bar{\sigma}$ of σ is defined recursively:

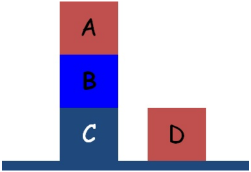
1. for every variable x, $\bar{\sigma}(x) = \sigma(x)$;
2. for every function symbol $f \in \mathcal{L}$, $\bar{\sigma}(f(t_1, ..., t_n)) = f^{\mathcal{M}}(\bar{\sigma}(t_1), ..., \bar{\sigma}(t_n))$.

Semantic of First-Order Logic: Variable Assignments

Let $\mathcal{L}$ be a vocabulary and $\mathcal{M}$ be an $\mathcal{L}$-structure.
The extension $\bar{\sigma}$ of σ is defined recursively:

1. for every variable x, $\bar{\sigma}(x) = \sigma(x)$;

2. for every function symbol $f \in \mathcal{L}$, $\bar{\sigma}(f(t_1, ..., t_n)) = f^{\mathcal{M}}(\bar{\sigma}(t_1), ..., \bar{\sigma}(t_n))$.



$$under^{\mathcal{M}}(A) = B \qquad under^{\mathcal{M}}(B) = C$$
$$under^{\mathcal{M}}(C) = C \qquad under^{\mathcal{M}}(D) = D$$

$X = \{v_1, v_2, v_3, v_4\}$
$\sigma(v_1) = D, \qquad \sigma(v_2) = C$
$\sigma(v_3) = B, \qquad \sigma(v_4) = A$

$\bar{\sigma}(under(under(v_4))) =$

•

First-Order Logic Semantic: Models (Interpretations)

For an $\mathcal{L}$-formula C, $\mathcal{M} \models C[\sigma]$ ($\mathcal{M}$ **satisfies** C under σ, or $\mathcal{M}$ is a **model** of C under σ) is defined recursively on the structure of C as follows (assuming A, B are $\mathcal{L}$-formulas):

$\mathcal{M} \models P(t_1, ..., t_n)[\sigma]$	iff	$\langle \bar{\sigma}(t_1), ..., \bar{\sigma}(t_n) \rangle \in P^{\mathcal{M}}$.
$\mathcal{M} \models (s = t)[\sigma]$	iff	$\bar{\sigma}(s) = \bar{\sigma}(t)$.
$\mathcal{M} \models \neg A[\sigma]$	iff	$\mathcal{M} \not\models A[\sigma]$.
$\mathcal{M} \models (A \vee B)[\sigma]$	iff	$\mathcal{M} \models A[\sigma]$ or $\mathcal{M} \models B[\sigma]$.
$\mathcal{M} \models (A \wedge B)[\sigma]$	iff	$\mathcal{M} \models A[\sigma]$ and $\mathcal{M} \models B[\sigma]$.
$\mathcal{M} \models (\forall x A)[\sigma]$	iff	$\mathcal{M} \models A[\sigma(m/x)]$ for all $m \in M$.
$\mathcal{M} \models (\exists x A)[\sigma]$	iff	$\mathcal{M} \models A[\sigma(m/x)]$ for some $m \in M$.

First-Order Logic Semantic: Models (Interpretations)

For an $\mathcal{L}$-formula C, $\mathcal{M} \models C[\sigma]$ ($\mathcal{M}$ **satisfies** C under σ, or $\mathcal{M}$ is a **model** of C under σ) is defined recursively on the structure of C as follows (assuming A, B are $\mathcal{L}$-formulas):

$\mathcal{M} \models P(t_1, ..., t_n)[\sigma]$	iff	$\langle \bar{\sigma}(t_1), ..., \bar{\sigma}(t_n) \rangle \in P^{\mathcal{M}}$.
$\mathcal{M} \models (s = t)[\sigma]$	iff	$\bar{\sigma}(s) = \bar{\sigma}(t)$.
$\mathcal{M} \models \neg A[\sigma]$	iff	$\mathcal{M} \not\models A[\sigma]$.
$\mathcal{M} \models (A \vee B)[\sigma]$	iff	$\mathcal{M} \models A[\sigma]$ or $\mathcal{M} \models B[\sigma]$.
$\mathcal{M} \models (A \wedge B)[\sigma]$	iff	$\mathcal{M} \models A[\sigma]$ and $\mathcal{M} \models B[\sigma]$.
$\mathcal{M} \models (\forall x A)[\sigma]$	iff	$\mathcal{M} \models A[\sigma(m/x)]$ for all $m \in M$.
$\mathcal{M} \models (\exists x A)[\sigma]$	iff	$\mathcal{M} \models A[\sigma(m/x)]$ for some $m \in M$.

Note: $\sigma(m/x)$ is an object assignment function exactly like σ, but maps the variable x to the individual $m \in M$. That is:

For $y \neq x : \sigma(m/x)(y) = \sigma(y)$

For x: $\sigma(m/x)(x) = m$

Models: Example

Let $\mathcal{M}_3$ be a structure such that:
$M_3 = \{A, B, C, D\}$
$on^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle\}$
$above^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle, \langle A, C\rangle\}$
$clear^{\mathcal{M}_3} = \{A, D\}$
$ontable^{\mathcal{M}_3} = \{C, D\}$

Does $\mathcal{M}_3$ satisfy
$\forall x \forall y (on(x, y) \rightarrow above(x, y))$

Let $\mathcal{M}_3$ be a structure such that:
$M_3 = \{A, B, C, D\}$
$on^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle\}$
$above^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle, \langle A, C\rangle\}$
$clear^{\mathcal{M}_3} = \{A, D\}$
$ontable^{\mathcal{M}_3} = \{C, D\}$

Does $\mathcal{M}_3$ satisfy
$\forall x \forall y (above(x, y) \rightarrow on(x, y))$

Let $\mathcal{M}_3$ be a structure such that:
$M_3 = \{A, B, C, D\}$
$on^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle\}$
$above^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle, \langle A, C\rangle\}$
$clear^{\mathcal{M}_3} = \{A, D\}$
$ontable^{\mathcal{M}_3} = \{C, D\}$

Does $\mathcal{M}_3$ satisfy
$\forall x \exists y (clear(x) \vee On(y, x))$

Let $\mathcal{M}_3$ be a structure such that:
$M_3 = \{A, B, C, D\}$
$on^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle\}$
$above^{\mathcal{M}_3} = \{\langle A, B\rangle, \langle B, C\rangle, \langle A, C\rangle\}$
$clear^{\mathcal{M}_3} = \{A, D\}$
$ontable^{\mathcal{M}_3} = \{C, D\}$

Does $\mathcal{M}_3$ satisfy
$\exists y \forall x (clear(x) \vee On(y, x))$

First-Order Logic Semantic: Models

An occurrence of x in A is **bounded** iff it is in a sub-formula of A of the form $\forall x B$ or $\exists x B$. Otherwise the occurrence is **free**.

Example:

$P(x) \wedge \exists x[P(x) \vee Q(x)]$

In a structure $\mathcal{M}$, formulas with **free variables** might be **true for some** object assignments to the free variables and **false for others**.

Example: Consider the formula $P(x, y) \wedge P(y, x)$ and the following structure $\mathcal{M}$:

$$M = \{a, b\} \qquad P^{\mathcal{M}} = \{\langle a, a \rangle\}$$

First-Order Logic Semantic: Models

A formula A is **closed** if it contains no free occurrence of a variable.
A **closed formula** is called a **sentence**.
Example:
$P(x) \wedge \exists x[P(x) \vee Q(x)]$.

$\forall x P(x) \wedge \exists x[P(x) \vee Q(x)]$

If σ and σ' agree on the **free variables** of A, then $\mathcal{M} \models A[\sigma]$ iff $\mathcal{M} \models A[\sigma']$.
Proof: Structural induction on A.

Corollary: If A is a **sentence**, then for any object assignments σ and σ',

$$\mathcal{M} \models A[\sigma] \qquad \text{iff} \qquad \mathcal{M} \models A[\sigma']$$

So, if A is a **sentence** (no free variables), σ is **irrelevant** and we omit mention of σ and simply write $\mathcal{M} \models A$.

Let Φ be a **set of sentences**.

- $\mathcal{M}$ satisfies Φ (denoted by $\mathcal{M} \models \Phi$) if for **every** sentence $A \in \Phi$, $\mathcal{M} \models A$.
- If $\mathcal{M} \models \Phi$, we say $\mathcal{M}$ is a model of Φ.
- We say that Φ is satisfiable if there is a structure $\mathcal{M}$ such that $\mathcal{M} \models \Phi$.

Unintended Models: Example

Let Φ_1 be a set containing the following sentences ($\boldsymbol{c_1}, \boldsymbol{c_2}$ are constant symbols, we use **bold** font to distinguish constant symbols from variables):

- $on(\boldsymbol{c_1}, \boldsymbol{c_2})$
- $clear(\boldsymbol{c_1})$
- $above(\boldsymbol{c_1}, \boldsymbol{c_2})$

Consider a model of Φ_1 with **size three** (i.e., the size of the domain of the model is three).

$M_1 = \{A, B, C\}$

$\boldsymbol{c_1}^{\mathcal{M}_1} = A \qquad \boldsymbol{c_2}^{\mathcal{M}_1} = B$

$on^{\mathcal{M}_1} = \{\langle A, B\rangle, \langle B, C\rangle\}$

$clear^{\mathcal{M}_1} = \{A, C\}$

$above^{\mathcal{M}_1} = \{\langle A, B\rangle\}$

Eliminating Unintended Models: Example

Let Φ_2 be a set containing the following sentences($\boldsymbol{c_1, c_2}$ are constant symbols):

- $\forall x(clear(x) \rightarrow \neg \exists y(on(y, x)))$
- $\forall x \forall y(on(x, y) \rightarrow above(x, y))$
- $\forall x \forall y \forall z((above(x, y) \wedge above(y, z)) \rightarrow above(x, z))$
- $on(\boldsymbol{c_1, c_2})$
- $clear(\boldsymbol{c_1})$
- $above(\boldsymbol{c_1, c_2})$

Construct **two models** of Φ_2 with **size three** (i.e., the size of the domain of each model must be three).

•

Logical Satisfiability: Practice Question

Example: is $\{\forall x(P(x) \rightarrow Q(x)), P(\mathbf{a}), \neg Q(\mathbf{a})\}$ satisfiable?