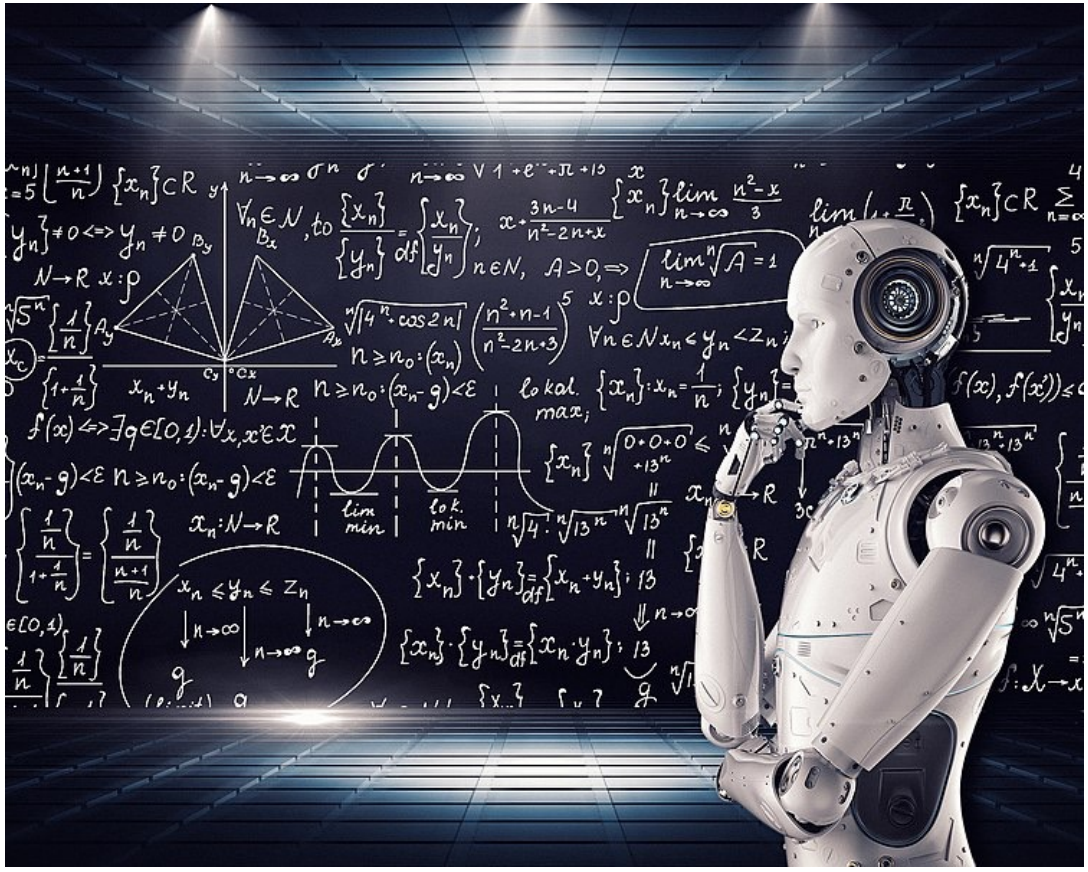
Lecture 27: Automatic Theorem-Proving

Mark Hasegawa-Johnson

4/2024



Outline

- Proving “there exists” vs. “for all” theorems
- Variable normalization
- Unification
- Forward-chaining
- Backward-chaining

Three types of theorems

- Instantiated, e.g., “Colonel West is a criminal”
$$Criminal(west)$$
- Existence theorem, e.g., “There is a spy in the army”
$$\exists x: Spy(x) \wedge Army(x)$$
- Universality theorem, e.g., “All spies are sneaky”
$$\forall x: Spy(x) \Rightarrow Sneaky(x)$$

“Instantiated theorems” are just a special case of “existence theorems”

- Notice that an **instantiated theorem** is just a special type of **existence theorem**: “Colonel West is a criminal” is the same statement as “There is a person who is Colonel West, and who is a Criminal”

$$Criminal(west)$$
$$\exists x: x = west \wedge Criminal(x)$$

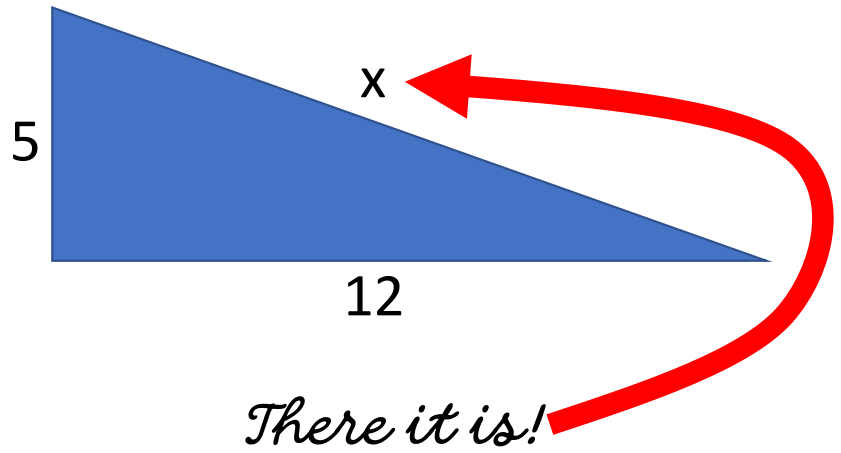
- Notice that these two types of theorem are proven in the same way:
  - $Criminal(west)$: Find an x such that x=west, and Criminal(x)
  - $\exists x: Spy(x) \wedge Army(x)$: Find an x such that Spy(x) and Army(x)

Proving and disproving theorems

An existence theorem:

- … can be **proven** by finding any x that satisfies the conditions.
- In order to **disprove** the statement $\exists x: F(x)$, you must prove that $\forall x: \neg F(x)$

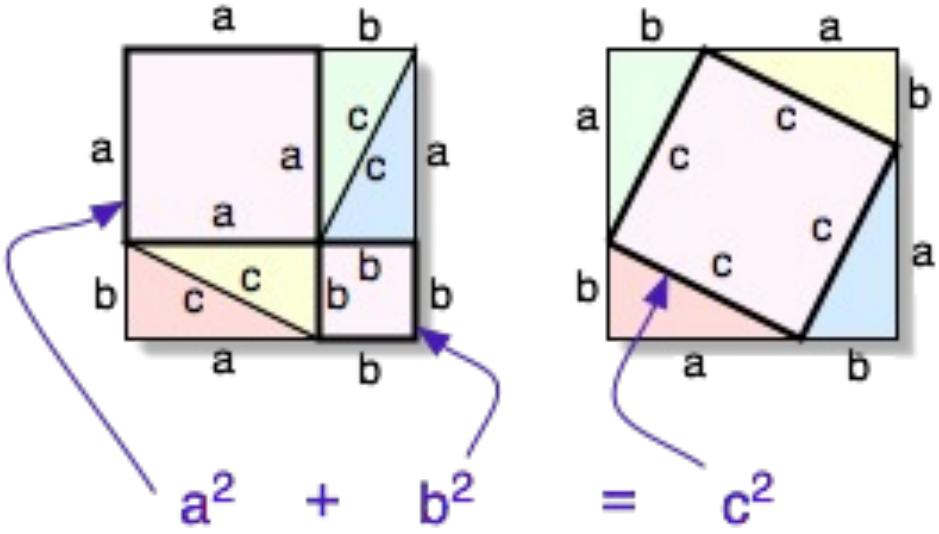
Find x:



Proving and disproving theorems

A universality theorem:

- To **disprove** the statement $\forall x: F(x)$, you just need to find a counterexample, i.e., you just need to prove that $\exists x: \neg F(x)$.
- To **prove** a universality theorem, you need to show that the existence of an x that violates the theorem contradicts known true propositions.



Proof that, for any right triangle with hypotenuse c and sides a and b, $a^2 + b^2 = c^2$. The existence of any right triangle violating this theorem would violate the proposition that the area of a rectangle with sides a and b is ab. Public domain image, https://commons.wikimedia.org/wiki/File:Pythagorean_proof.png

Two types of proofs

- In order to **prove an existence theorem**, or disprove a universality theorem, you just need to find an x that satisfies the statement.
  - This is done using forward-chaining or backward-chaining with ***unification***.
  - I will spend the rest of today's lecture talking about this.
- In order to disprove an existence theorem, or **prove a universality theorem**, you need to prove that the existence of any such x would contradict known true propositions.
  - This is done using a proof method called ***resolution***.
  - We will not cover it in this course, and you don't need to know it for the exam. It is interesting, but even more difficult than unification, so we will not learn it this semester.

Outline

- Proving “there exists” vs. “for all” theorems
- Variable normalization
- Unification
- Forward-chaining
- Backward-chaining

Variable Normalization

- When we are given a database of facts, and a theorem to prove, often we will see the same variable name used for different purposes
- For example, consider the statements

$$Sweet(chocolate)$$
$$\forall x: Sweet(x) \Rightarrow Likes(jack, x)$$
$$\exists x, y: Likes(x, y)$$

- The first two statements prove the third statement.
- In order to prove it, however, we need to normalize variables.

Public domain image, https://commons.wikimedia.org/wiki/File:Tomando_chocolate_1892_Gumersindo_Pardo_Reguera.jpg

Variable Normalization

- Variable normalization replaces the old variable names with new variable names such that:
  1. If the same variable name occurs in different rules, change it so that ***each rule uses a different set of variable names***
  2. If the same variable occurs multiple times in one rule, its multiple instances still have the same name
- For example, the example on the previous page could be normalized to:

$$Sweet(chocolate)$$
$$Sweet(x_1) \Rightarrow Likes(jack, x_1)$$
$$\exists x_2, y_1 : Likes(x_2, y_1)$$

Public domain image, https://commons.wikimedia.org/wiki/File:Tomando_chocolate_1892_Gumersindo_Pardo_Reguera.jpg

Outline

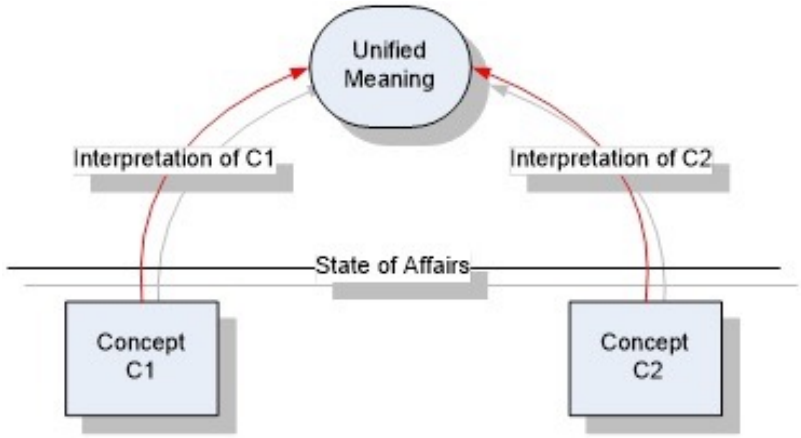
- Proving “there exists” vs. “for all” theorems
- Variable normalization
- Unification
- Forward-chaining
- Backward-chaining

Unification

Given a proposition P written in terms of the variables $\mathcal{V}_P$ and constants C, and a proposition Q written in terms of the variables $\mathcal{V}_Q$ and constants C, ***unification*** finds a substitution S that ***unifies the propositions P and Q***, in the sense that:

- Find a substitution $S: \{\mathcal{V}_P, \mathcal{V}_Q\} \rightarrow \{\mathcal{V}_Q, C\}$ such that
- $S(P) = S(Q) = U$

… or prove that no such substitution exists.



Unification Example

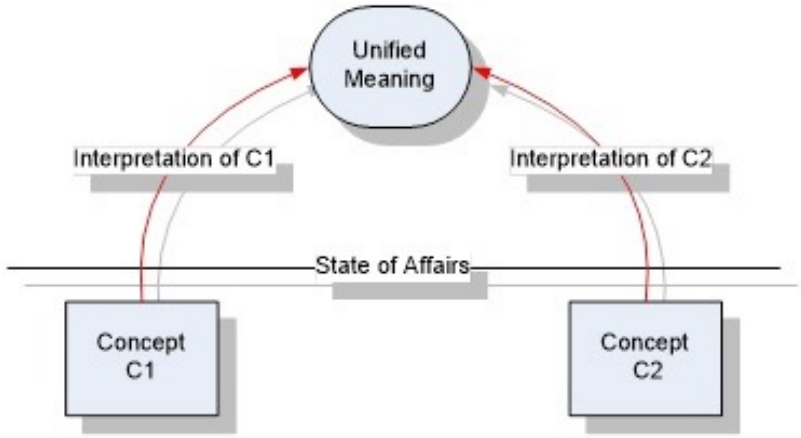
- Consider the two propositions

$$P = Likes(jack, x_1)$$
$$\mathcal{V}_P = \{jack, x_1\}$$
$$Q = Likes(x_2, y_1)$$
$$\mathcal{V}_Q = \{x_2, y_1\}$$

- The unification of these two propositions is the substitution

$$S = \{x_2 : jack, x_1 : y_1\}$$
$$U = Likes(jack, y_1)$$

- Every ***constant*** that appears in either P or Q should also appear in U
- ***Variables*** appear in U only if they came from Q, not P



Bidirectional Unification

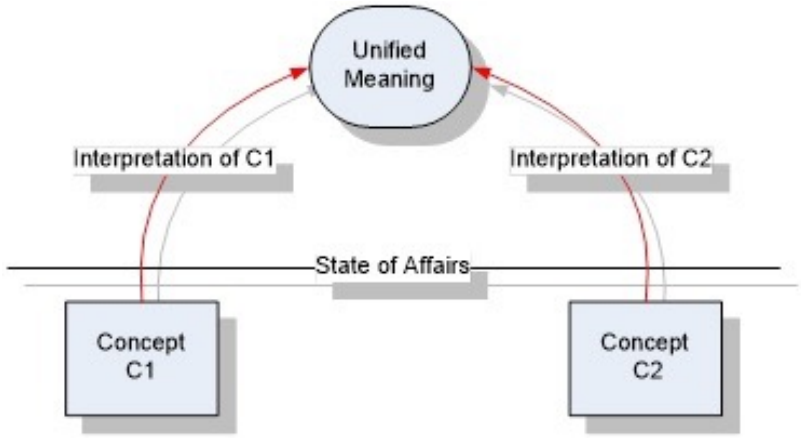
- Notice that unification can draw constants from both P and Q
- Consider the two propositions

$$P = Likes(jack, x_1)$$
$$Q = Likes(x_2, chocolate)$$

- The unification of these two propositions is the substitution

$$S = \{x_2 : jack, x_1 : chocolate\}$$
$$U = Likes(jack, chocolate)$$

- Every constant that appears in either P or Q should also appear in U



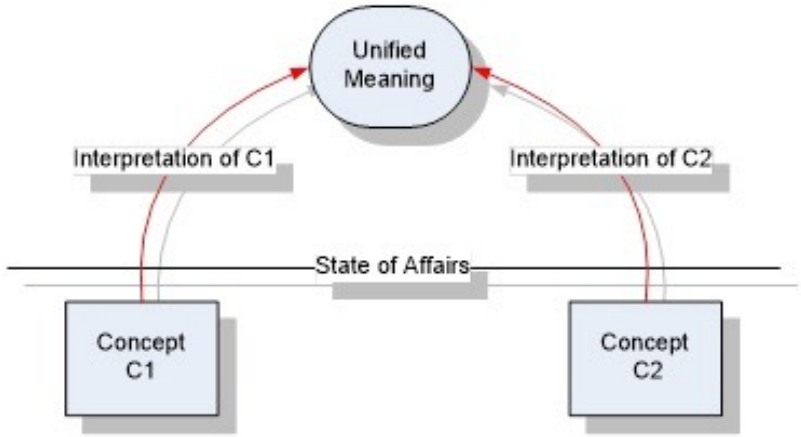
When does unification fail?

- Unification fails if we can't make a unified proposition that implies both P and Q
- For example, unification fails if P and Q have different predicates:

$$P = Likes(jack, x_1)$$
$$Q = Eats(x_2, y_1)$$

- Unification also fails if a particular argument is a constant, but P and Q have different constants, e.g.,

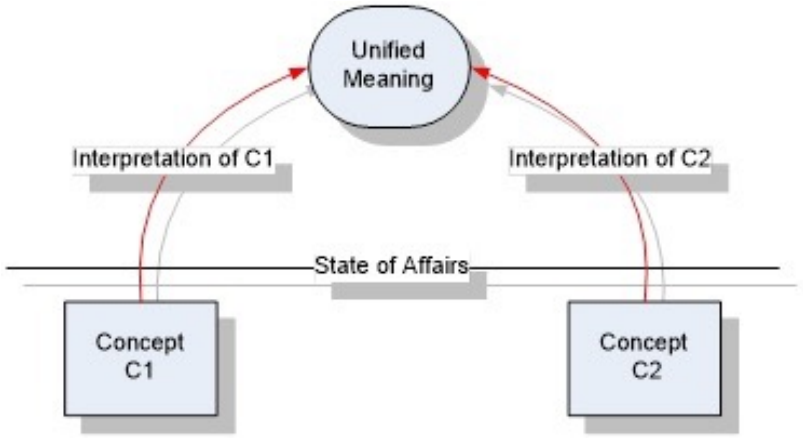
$$P = Likes(jack, x_1)$$
$$Q = Likes(jill, x_3)$$



Unification in more general terms

The word “unification” is more generally defined as:

- …mapping of two source expressions
- … onto a single target expression, with some standardized format, such that
- … the target expression implies both of the source expressions.



Outline

- Proving “there exists” vs. “for all” theorems
- Variable normalization
- Unification
- Forward-chaining
- Backward-chaining

Forward-chaining

Forward-chaining is a method of proving a theorem, T:

- Starting state: a database of known true propositions, $\mathcal{D} = \{P_1, P_2, \ldots\}$
- Actions: the set of possible actions is defined by a set of rules, where each rule has the form $P \Longrightarrow Q$.
- Neighboring states: if P_1 unifies to P creating $\mathrm{S(P)} = \mathrm{S}(P_1)$, then create the new database $\mathcal{D}' = \{P_1, P_2, \ldots, S(Q)\}$
- Termination: search terminates when we find a database containing T

Example of forward-chaining

Database: $\mathcal{D} = \{Sweet(chocolate)\}$

Rule: $Sweet(x_1) \Rightarrow Likes(jack, x_1)$

Theorem: $\exists x_2, y_1 : Likes(x_2, y_1)$

Proof:

1. Unify $Sweet(x_1)$ to $Sweet(chocolate)$. Apply the substitution to $Likes(jack, x_1)$. Result:
   $\mathcal{D}' = \{Sweet(chocolate), Likes(jack, chocolate)\}$

2. Unify $Likes(x_2, y_1)$ to $Likes(jack, chocolate)$. Result:
   $\mathcal{D}'' = \begin{Bmatrix} Sweet(chocolate), Likes(jack, chocolate), \\ \exists jack, chocolate : Likes(jack, chocolate) \end{Bmatrix}$

Public domain image, https://commons.wikimedia.org/wiki/File:Tomando_chocolate_1892_Gumersindo_Pardo_Reguera.jpg

Forward-chaining

- **What's Special About Theorem Proving**:
  - A state, at level n, can be generated by the combination of several states at level n-1.
- **Definition: Forward Chaining** is a search algorithm in which each action
  - generates a new proposition,
  - …and adds it to the database of known propositions.

Outline

- Proving “there exists” vs. “for all” theorems
- Variable normalization
- Unification
- Forward-chaining
- Backward-chaining

Backward-chaining

Backward-chaining is a method of proving a result, R:

- Starting state: a set of "goals" containing only one goal, the result to be proven, $\mathcal{G} = \{R\}$
- Actions: the set of possible actions is defined by
  1. A set of rules of the form $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Longrightarrow Q$, and
  2. A set of known true propositions.
- Neighboring states: if Q unifies with some $Q' \in \mathcal{G}$ then
  - Remove Q' from $\mathcal{G}$
  - Replace it with $P_1 \wedge P_2 \wedge \cdots \wedge P_n$
- Termination: search terminates if all propositions in the goalset are known to be true.

Example of backward-chaining

Theorem: $\mathcal{G} = \{\exists x_2, y_1 : Likes(x_2, y_1)\}$

Rules:

$$\mathbb{T} \Rightarrow Sweet(chocolate)$$
$$Sweet(x_1) \Rightarrow Likes(jack, x_1)$$

Proof step 1:

Unify $Likes(jack, x_1)$ to $Likes(x_2, y_1)$. Apply the substitution to $Sweet(x_1)$. Result:

$$\mathcal{G}' = \{Sweet(x_1)\}$$

Proof step 2:

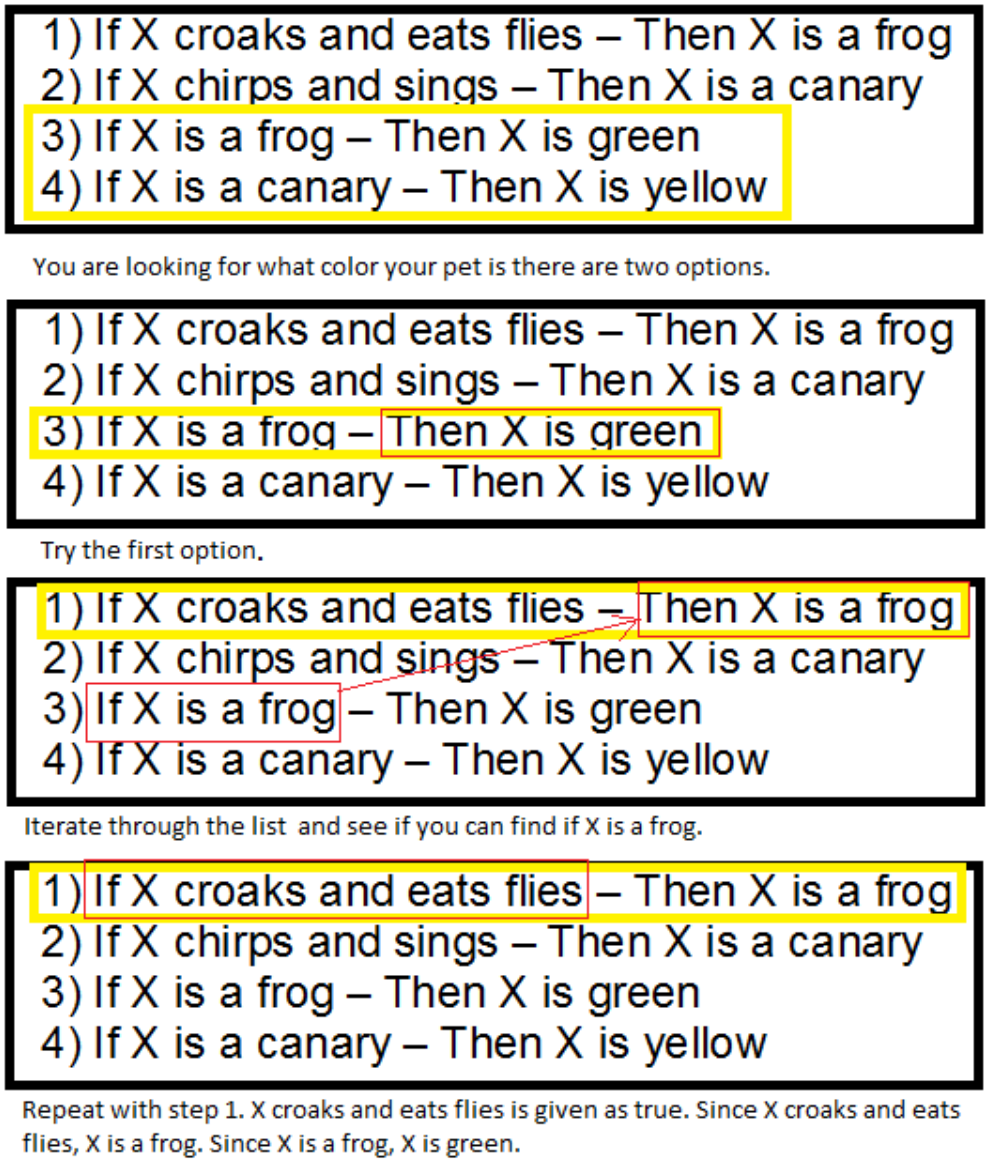
Unify $Sweet(x_1)$ to $Sweet(chocolate)$. Apply the substitution to $\mathbb{T}$ (no effect). Result:

$$\mathcal{G}'' = \{\mathbb{T}\}$$

Public domain image, https://commons.wikimedia.org/wiki/File:Tomando_chocolate_1892_Gumersindo_Pardo_Reguera.jpg

Another example (from Wikipedia)

- My friend wants to give me Fritz
- Fritz croaks and eats flies
- I want to know what color Fritz is
- I have two possibilities to consider: either $\mathcal{G} = \{Green(fritz)\}$, or $\mathcal{G} = \{Yellow(fritz)\}$
- Backward-chaining follows the steps shown here to prove that Fritz is green.

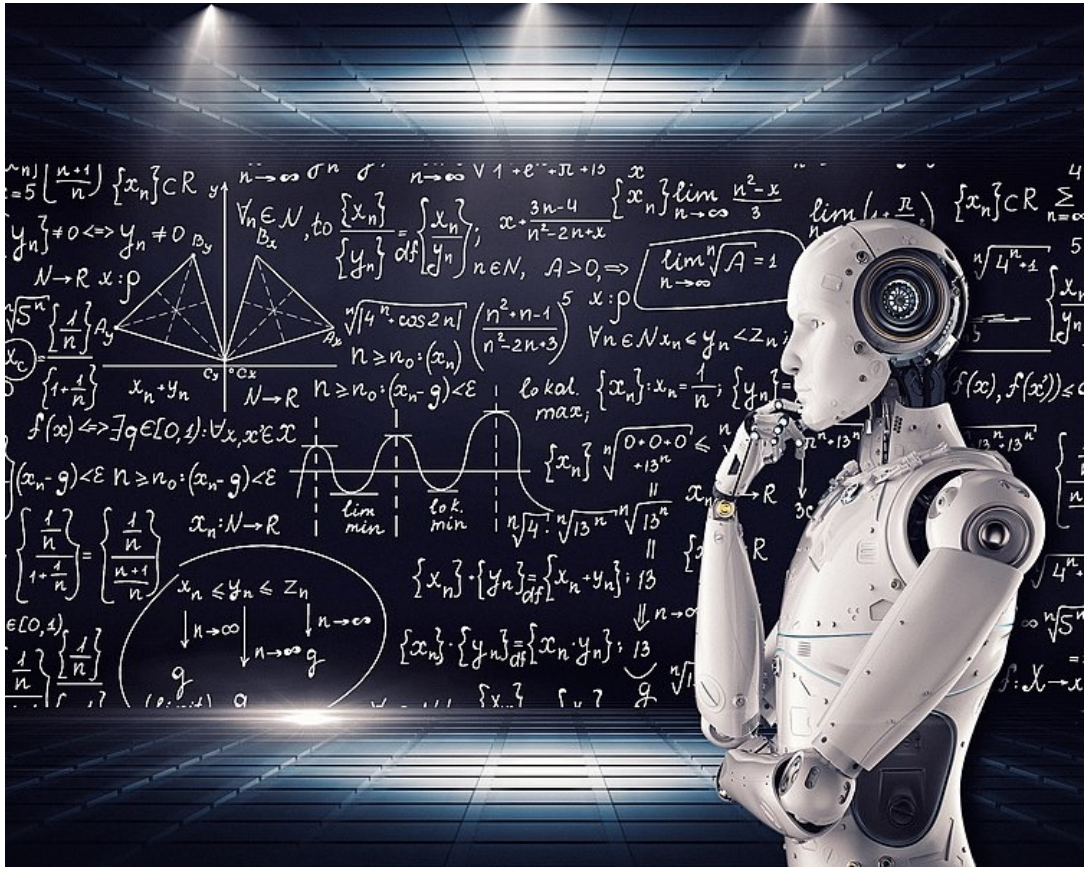


Backward-chaining

- **What Else is Special About Theorem Proving:**
  - The "goal set" is a set of propositions that need to be proven.
- **Definition: Backward Chaining** is a search algorithm in which
  - State = {goal set}
  - Action = apply a known rule, backward: replace the goal's *consequent* (its RHS) with its *antecedents* (its LHS)
  - Termination = the goalset contains nothing but truth

Quiz

- Try the quiz! https://us.prairielearn.com/pl/course_instance/147925/assessment/2411228



CC-SA 2.0, https://commons.wikimedia.org/wiki/File:Artificial_Intelligence_%26_AI_%26_Machine_Learning_-_30212411048.jpg

Comparison of forward-chaining and backward-chaining

Forward-chaining:

- Time complexity: $\mathcal{O}\{b^d\}$, where b is the number of rules that can be applied at any step, and d is the number of steps necessary to prove the theorem
- Space complexity: in order to make it easy to retrieve the database for each state, each state should save a complete copy of the database!

Backward-chaining:

- Time complexity: $\mathcal{O}\{b^d\}$, where b is the number of rules that can be applied at any step, and d is the number of steps necessary to prove the theorem
- Space complexity: each state only needs to save a copy of the goalset, which is usually much smaller than the database.

Summary

- Proving “there exists” theorems: find an x that satisfies the statement
- Variable normalization: each rule uses a different set of variable names
- Unification: Find a substitution S: $\{\mathcal{V}_P, \mathcal{V}_Q\} \rightarrow \{\mathcal{V}_Q, C\}$ such that $S(P) = S(Q) = U$, or prove that no such substitution exists
- Forward-chaining: Search problem in which each action is a unification, and the state is the set of all known true propositions
- Backward-chaining: Search problem in which each action is a unification, and the state is the goal (the proposition whose truth needs to be proven)