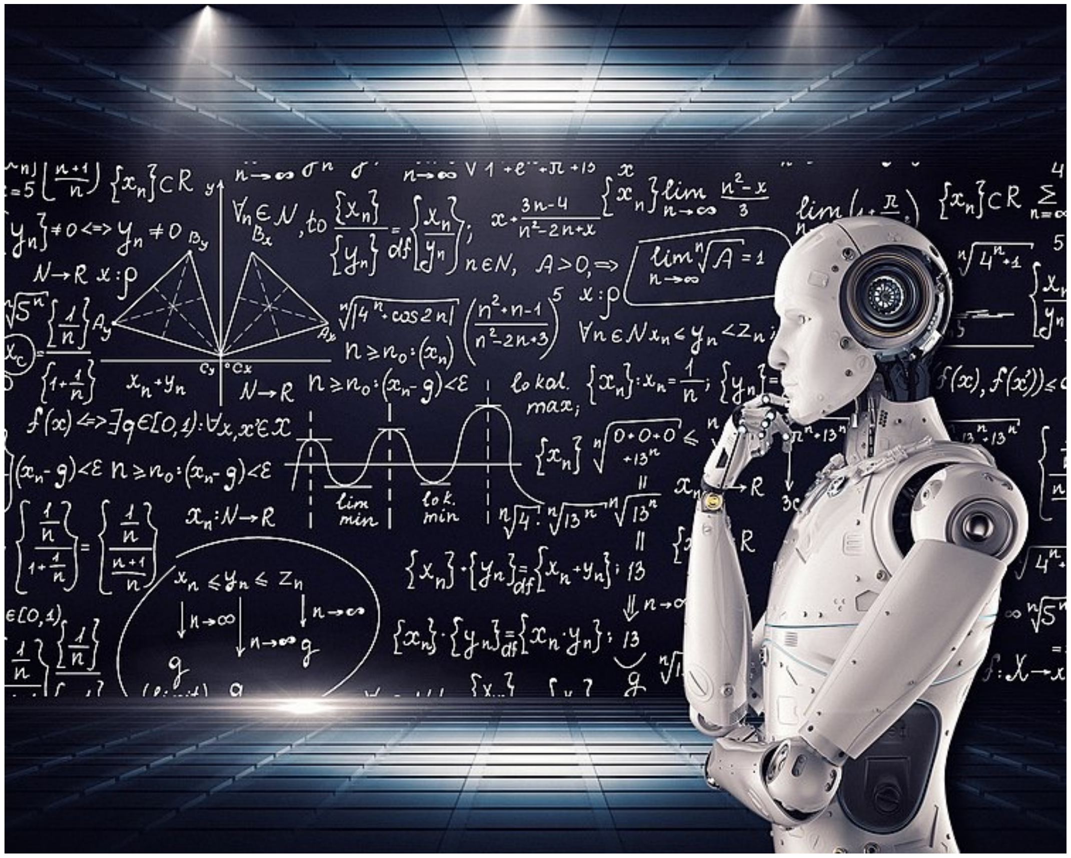
Lecture 15: Chain-of-Thought, AI-assisted Scientific Discovery, and Automatic Theorem-Proving

Mark Hasegawa-Johnson

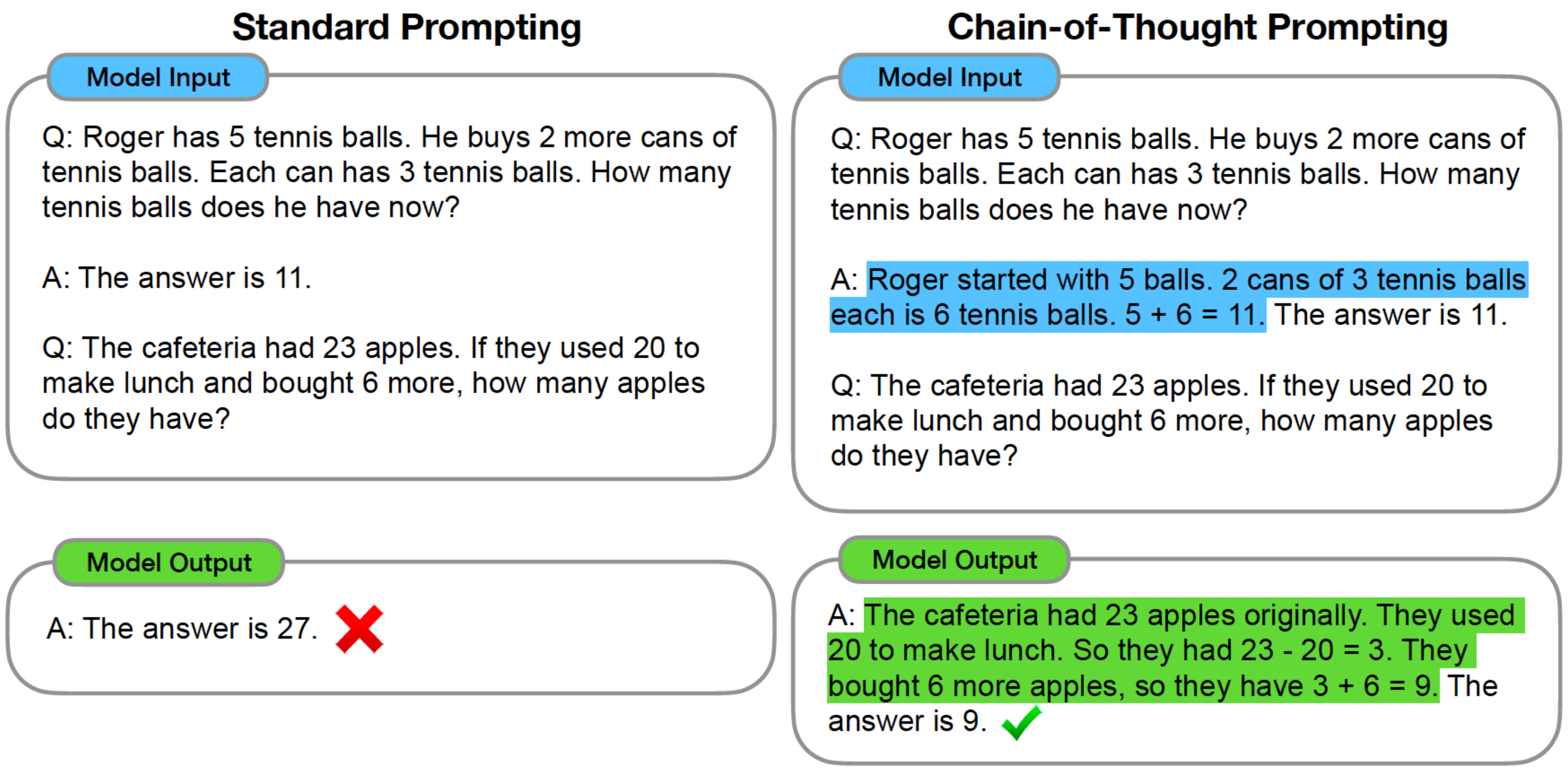


Outline

- Chain-of-thought prompting of large language models
- AI-guided scientific discovery
- Theorem proving as a search problem
- Zeroth-Order and First-Order Logic
- Proving “there exists” vs. “for all” theorems
- Variable normalization, Unification, Forward- & Backward-chaining

Chain of thought prompting

Fig. 1 from Wei, Wang, Schuurmans et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," 2022



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

How are large language models trained?

- LLMs are trained, over the whole internet and all books in all the world's libraries, to maximize the probability of the next word

$$\mathcal{L} = -\sum_t \log P(w_t | \boldsymbol{c}_t)$$

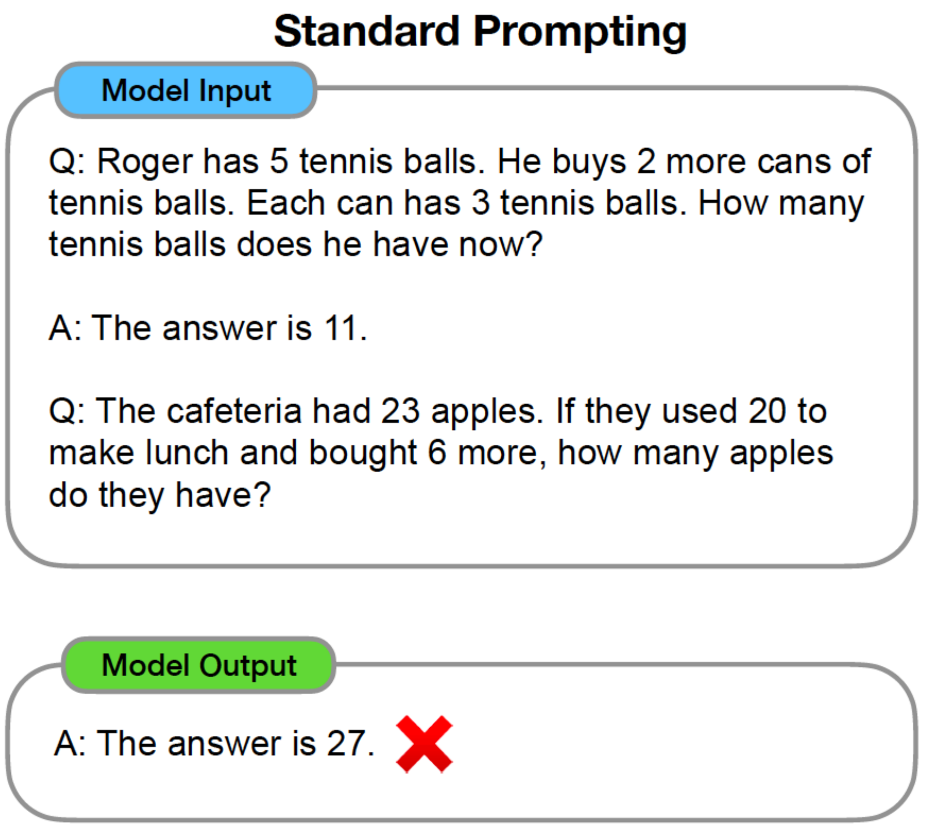
- $\boldsymbol{c}_t$ is computed using a Transformer based on the most recent 1k words or so

Why does prompting work?

- Many books contain problems, often framed as “Answer the following question: ……. The answer is: ….”. If you give the question as a prompt to the LLM, then the words of the answer have high probability.
- Other books contain sequences of pairs: {question 1, answer1, question 2, answer 2, question 3, answer 3, …}. If you prompt the LLM with “question 1, answer 1, question 2” and then stop, then the words of “answer 2” – in the same format as “answer 1” – have high probability.

Why is arithmetic hard for LLMs?

- The LLM has never seen this word problem before
- The context contains the numbers “23”, “20” and “6”
- How should the paragraph end? A good guess is that there should be another number in the 20s, maybe separated from 20 and 23 by roughly 6…


Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

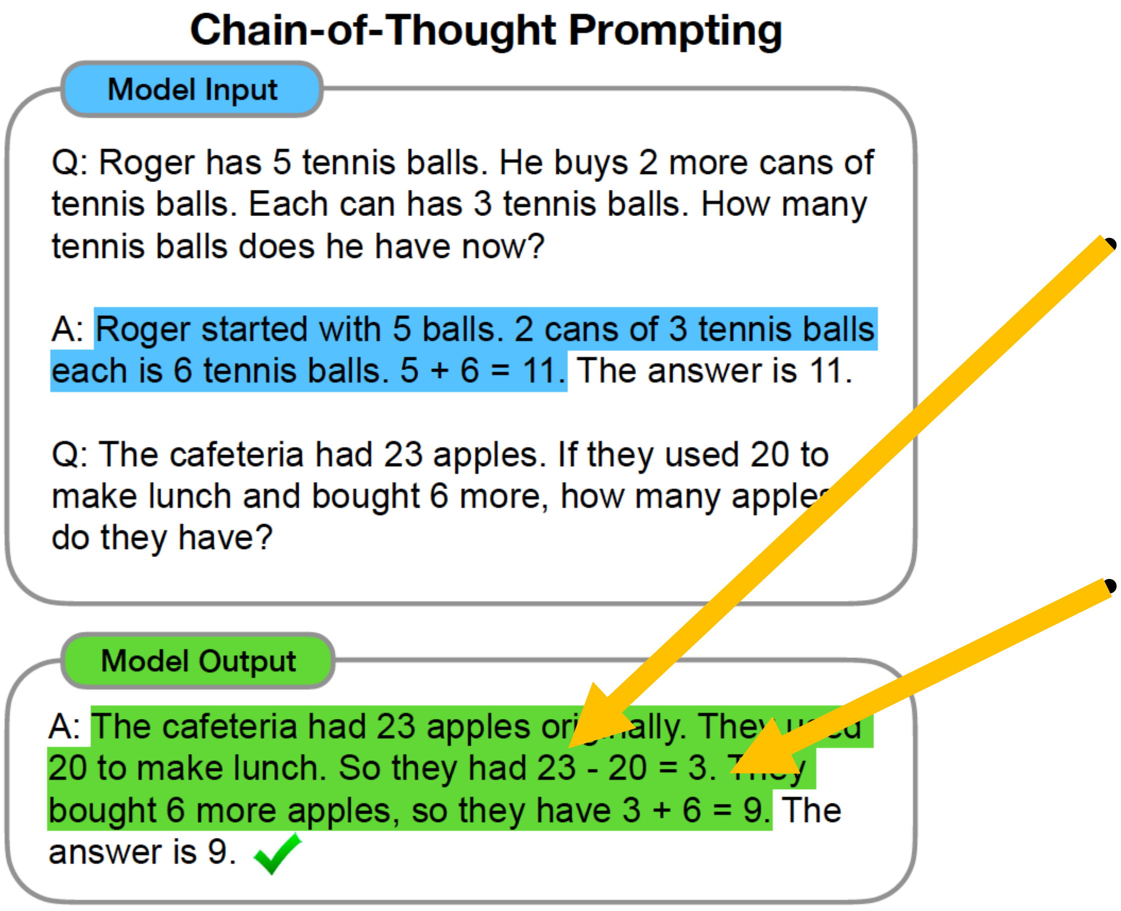
A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Why does chain-of-thought work?


Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

- When the model generates this equation, the words "had 23… used 20…" are in context, so it's reasonable to generate this equation
- Now that the equation "23 – 20 =" is in its context, the model's training data makes "3" a high-probability solution

Outline

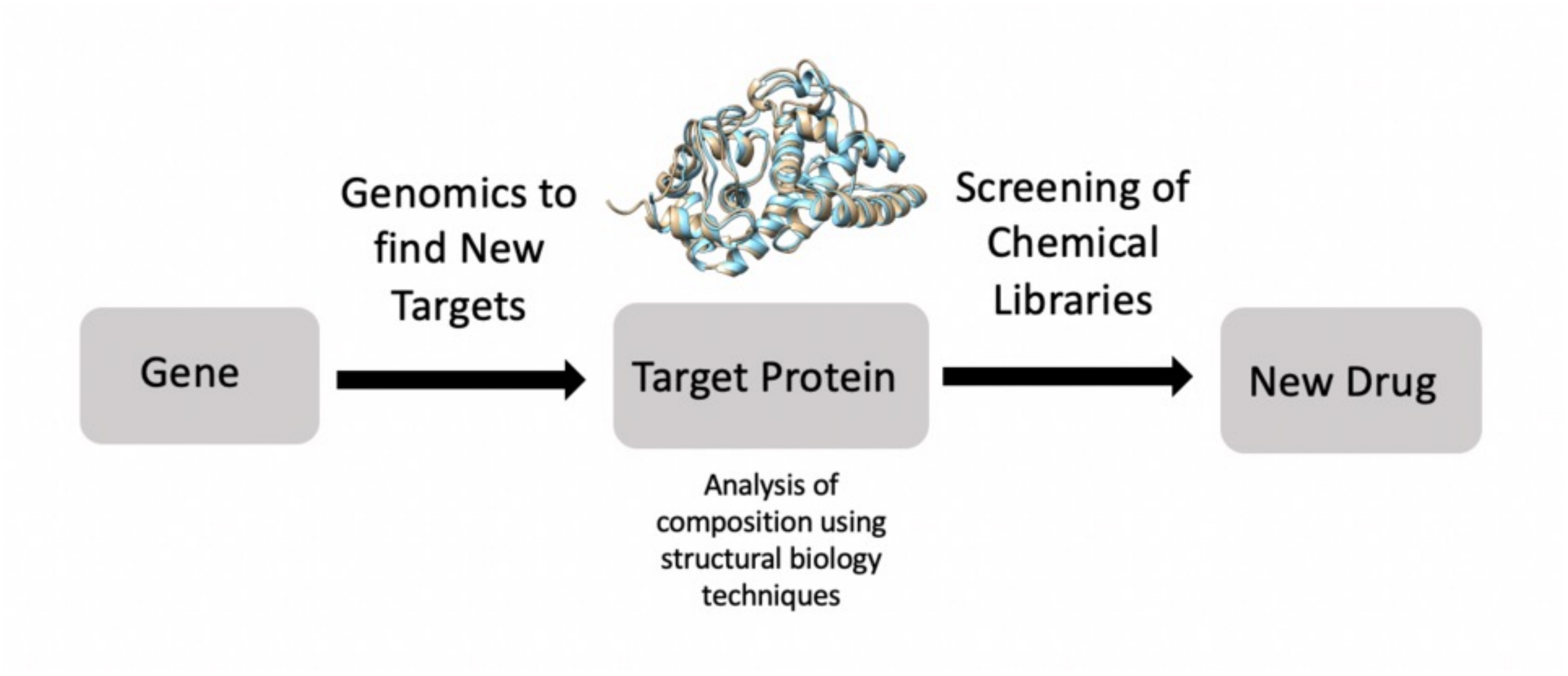
- Chain-of-thought prompting of large language models
- AI-guided scientific discovery
- Theorem proving as a search problem
- Zeroth-Order and First-Order Logic
- Proving “there exists” vs. “for all” theorems
- Variable normalization, Unification, Forward- & Backward-chaining

AI-guided scientific discovery

AI can help scientists to discover new physical laws, facts about the universe, new chemical reactions, new ecosystem modalities, etc.:

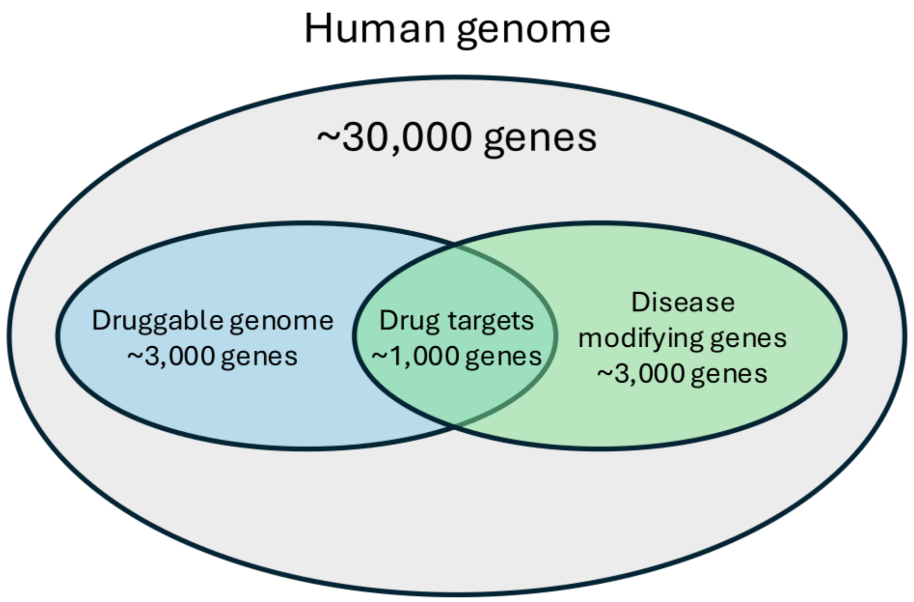
- Hypothesis Generation: Code a set of problems and a set of solutions, search for a match
- Hypothesis Generation: Generate plausible explanations to fit an observed phenomenon
- Data Augmentation: Use a well-known forward model (generating particle interactions or chemicals or animals or ecosystems) to generate billions of realistic examples which are then used to train a neural net backward model (recognizing interactions or chemicals of a certain type)

Example: Drug discovery


Genomics to find New Targets
Screening of Chemical Libraries
Gene
Target Protein
New Drug
Analysis of composition using structural biology techniques

Typical goal: Find a druggable protein, and the drug that will bind to it

- Genes code for proteins
- Some of those proteins are known to modify (for better or worse) a disease
- Many of those proteins are believed to have physical structures where a small molecule could attach, changing the behavior of the protein
- How do we find such binding sites? How do we find a chemical that will bind there?



Public Domain Image,
https://commons.wikimedia.org/wiki/File:Drug_targets.svg

Typical structure of a druggable protein: A physical structural pocket into which a small molecule fits

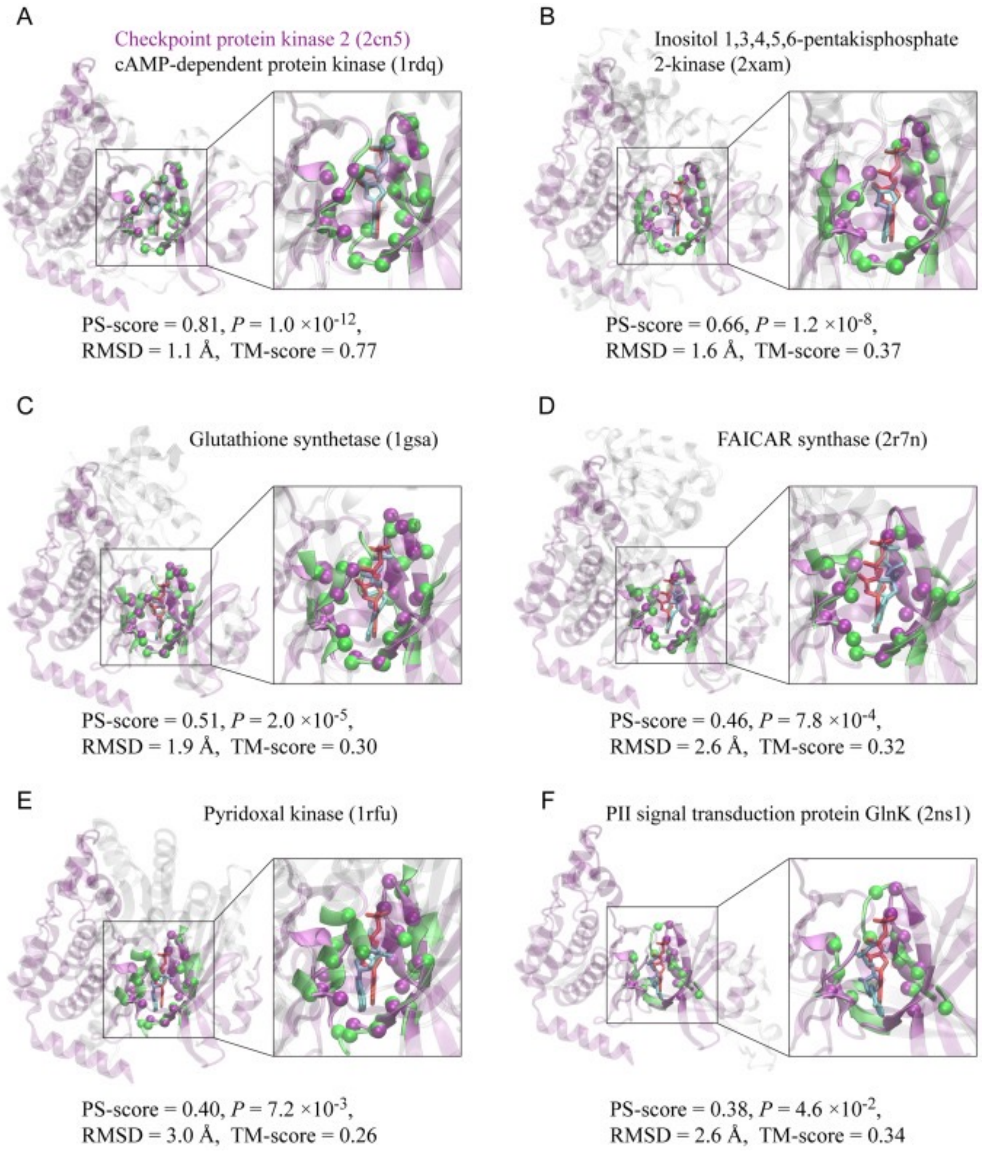
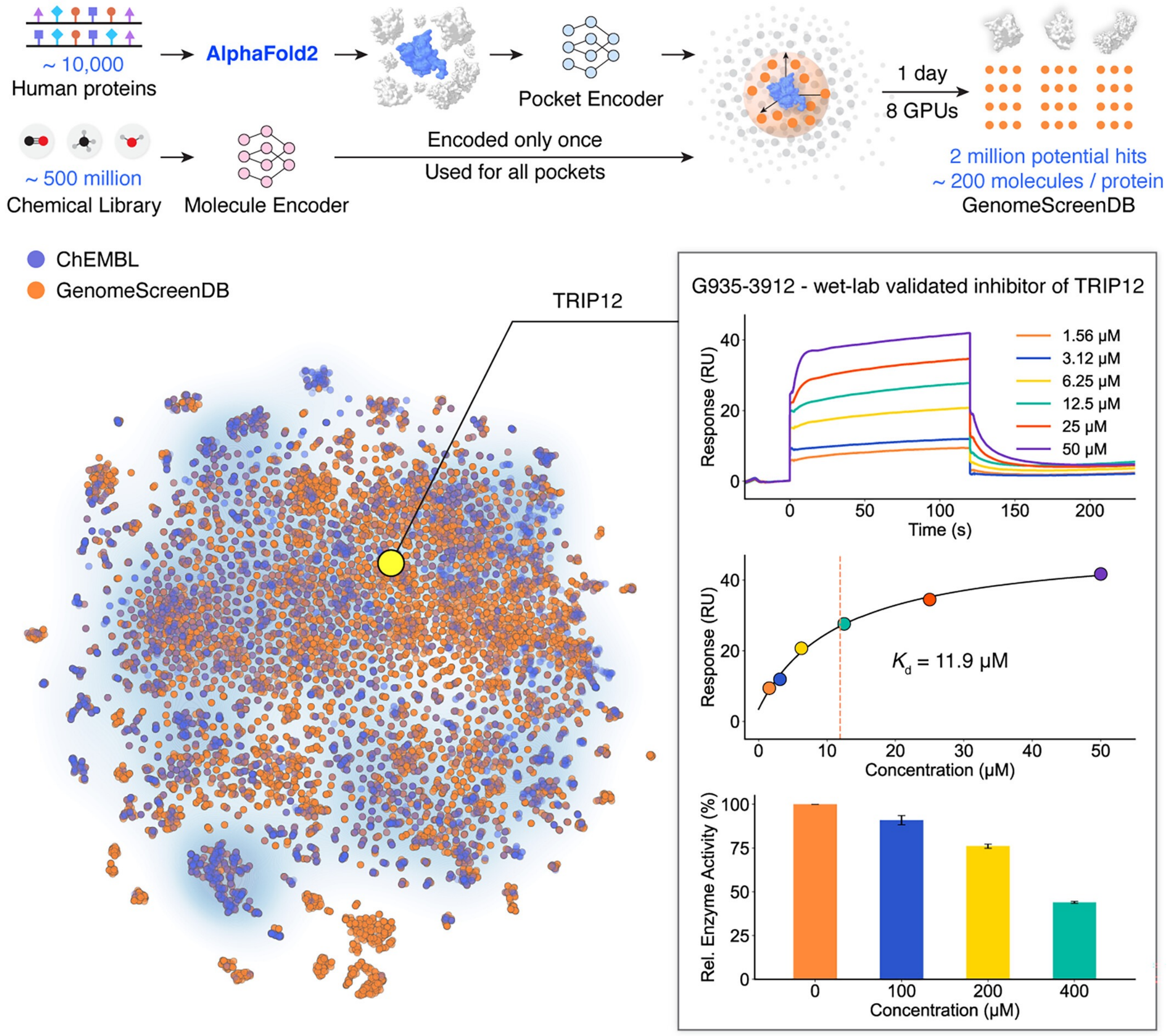

A
Checkpoint protein kinase 2 (2cn5)
cAMP-dependent protein kinase (1rdq)
PS-score = 0.81, P = 1.0 ×10^{-12},
RMSD = 1.1 Å, TM-score = 0.77
B
Inositol 1,3,4,5,6-pentakisphosphate 2-kinase (2xam)
PS-score = 0.66, P = 1.2 ×10^{-8},
RMSD = 1.6 Å, TM-score = 0.37
C
Glutathione synthetase (1gsa)
PS-score = 0.51, P = 2.0 ×10^{-5},
RMSD = 1.9 Å, TM-score = 0.30
D
FAICAR synthase (2r7n)
PS-score = 0.46, P = 7.8 ×10^{-4},
RMSD = 2.6 Å, TM-score = 0.32
E
Pyridoxal kinase (1rfu)
PS-score = 0.40, P = 7.2 ×10^{-3},
RMSD = 3.0 Å, TM-score = 0.26
F
PII signal transduction protein GlnK (2ns1)
PS-score = 0.38, P = 4.6 ×10^{-2},
RMSD = 2.6 Å, TM-score = 0.34

Figure 1, “A Comprehensive Survey of Small-Molecule Binding Pockets in Proteins,” Gao & Skolnick 2013, https://pmc.ncbi.nlm.nih.gov/articles/PMC3812058/

Deep contrastive learning enables genome-wide virtual screening

Jia, Gao, Tan et al., https://www.science.org/doi/10.1126/science.ads9530



Outline

- Chain-of-thought prompting of large language models
- AI-guided scientific discovery
- Theorem proving as a search problem
- Zeroth-Order and First-Order Logic
- Proving “there exists” vs. “for all” theorems
- Variable normalization, Unification, Forward- & Backward-chaining

Consider the difference in these examples:

- Drug discovery: 1 step, with 5×10^{12} possibilities.
  - 10,000 human proteins $\times$ 500 million known molecules
  - Goal: Find at least one match!
- Chain of thought example: 6 steps, with ??? possibilities.
  1. The cafeteria had 23 apples originally.
  2. They used 20 to make lunch…
  3. …so they had 23-20=3.
  4. They bought 6 more apples…
  5. ...so they have 3+6=9.
  6. The answer is 9.

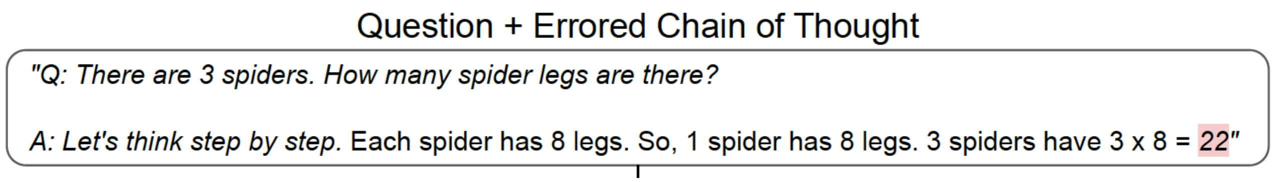
Scientific discovery is CoT with one step at a time

Each step is published as a separate peer-reviewed paper

- Drug discovery:
  1. Identify a human protein and a molecule that fits it ($5{\times}10^{12}$ possibilities).
  2. Is that protein elevated in people who have a disease?
  3. Is there something else that binds where the drug candidate binds?
  4. Is there a mechanism by which the “something else” might affect disease?
  5. Prove that the discovered binding really happens in vitro
  6. Prove that the discovered binding reduces disease symptoms in an animal model (or, in very simple cases, in a computational model) with no negative side effects
  7. Prove #6 with 5 humans, then 50 humans, then 500 humans
  8. Prove that the small molecule can be manufactured more efficiently than comparable treatments

Why is Scientific Discovery Peer-reviewed?

Because chain of thought sometimes fails.

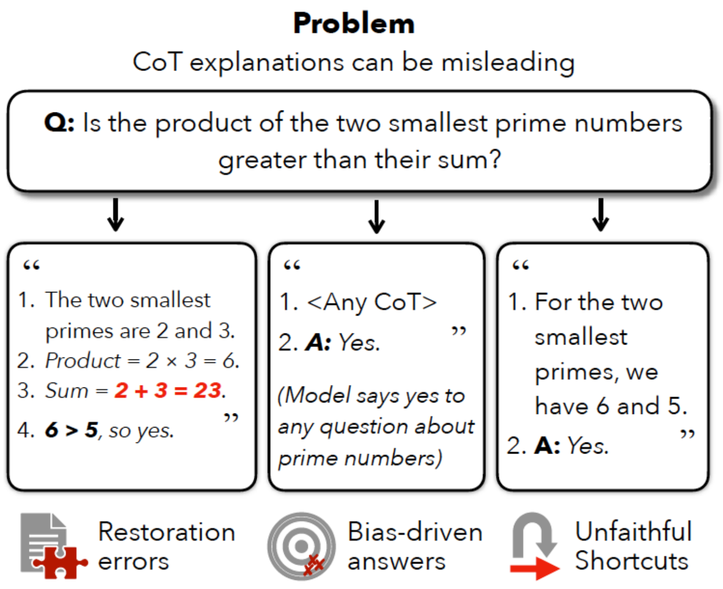


Question + Errored Chain of Thought

"*Q: There are 3 spiders. How many spider legs are there?*

A: Let's think step by step. Each spider has 8 legs. So, 1 spider has 8 legs. 3 spiders have 3 x 8 = *22*"

"Dissociation of Faithful and Unfaithful Reasoning in LLMs"
Evelyn Yee, Alice Li, Chenyu Tang, Yeon Ho Jung, Ramamohan Paturi & Leon Bergen
Preprint


Problem
CoT explanations can be misleading
Q: Is the product of the two smallest prime numbers greater than their sum?
" 1. The two smallest primes are 2 and 3. 2. *Product = 2 × 3 = 6.* 3. *Sum =* **2 + 3 = 23**. 4. ***6 > 5****, so yes.* "
" 1. <Any CoT> 2. ***A:*** *Yes.* "
(Model says yes to any question about prime numbers)
" 1. For the two smallest primes, we have 6 and 5. 2. **A:** *Yes.* "
Restoration errors
Bias-driven answers
Unfaithful Shortcuts

"Chain-of-Thought Is Not Explainability"
Barez, Wu, Arcuschin, Lan, Wang et al.
Under review.

How can we make chain-of-thought more reliable?

- This is an active current area of research
- Topic for today: Theorem proving as search
    1. Generate all possible next steps (5×10^{12} possibilities?)
    2. Check each one for logical consistency (keep the 2 million correct ones?)
    3. If we have not yet proven the theorem: For each of the 2 million correct first steps, go to step 1 to generate all possible second steps, etc.
- The disadvantage of “theorem proving as search:” Exponential complexity
- The advantage of “theorem proving as search:” Provability

Outline

- Chain-of-thought prompting of large language models
- AI-guided scientific discovery
- Theorem proving as a search problem
- Zeroth-Order and First-Order Logic
- Proving “there exists” vs. “for all” theorems
- Variable normalization, Unification, Forward- & Backward-chaining

Propositional Logic (“Zeroth-order” Logic)

- “Propositions” are statements that can be either True or False
  - P=“an iguana is an animal with scales”
  - Q=“an iguana is an animal that breathes air”
  - R=“an iguana is a reptile”
- Propositional logic studies the relationships among propositions.

Symbolic Logic Functions

- Unary functions (map one proposition to another)
  - $\neg$ (not):$\{F, T\} \rightarrow \{T, F\}$
- Binary functions (map two propositions to one)
  - $\wedge$ (and):$\{(F, F), (F, T), (T, F), (T, T)\} \rightarrow \{F, F, F, T\}$
  - $\vee$ (or):$\{(F, F), (F, T), (T, F), (T, T)\} \rightarrow \{F, T, T, T\}$
- Rules (generate one proposition from another)
  - $P \Longrightarrow Q$ (implies): if $\mathrm{P} = \mathrm{T}$ we can infer $\mathrm{Q} = \mathrm{T}$
  - $P \Leftrightarrow Q$ (equivalent): infer either P or Q to match the value of the other

First Order Logic

- Propositional logic says that propositions can be constructed from other propositions
- First-order logic says propositions can also be constructed by applying predicates to constants

Predicates, Constants, Variables, Propositions, and Rules

- A **predicate** is like a function, that can be applied to some **variables**.
  - $BreathesAir(x)$ is true if and only if x breathes air.
- A **constant** is a particular object in the real world, which can be the value of the argument of a function:
  - $reptiles$ is a constant
- A **proposition** is a predicate applied to a constant
  - $BreathesAir(reptiles)$ is true if and only if reptiles breathes air.
- A **rule** is an implication or equivalence that's true for all values of its variable
  - $BreathesAir(x) \wedge Scales(x) \Longrightarrow Reptile(x)$: everything that breathes air and has scales is a reptile.

Theorem Proving

An automatic theorem-prover uses a database of known facts and known rules to prove a theorem. For example, suppose we know that:

- Iguanas have scales: $Scales(iguanas)$
- Iguanas breathe air: $BreathesAir(iguanas)$
- Anything that breathes air and has scales is a reptile: $BreathesAir(x) \wedge Scales(x) \Longrightarrow Reptile(x)$

And suppose we want to prove that:

- Iguanas are reptiles: $Reptile(iguanas)$

Theorem Proving by Forward-Chaining

- Forward-chaining is the process of applying rules to facts in order to prove more facts.
- For example, let's start by combining these two facts:
$$BreathesAir(iguanas) \wedge Scales(iguanas)$$
- Now let's apply this rule:
$$BreathesAir(x) \wedge Scales(x) \Longrightarrow Reptile(x)$$
- The result: we have proven that:
$$Reptile(iguanas)$$

Theorem-Proving by Forward-Chaining

Notice that, when we're forward-chaining, each step of the process just expands the set of available facts. If we start with the following database of facts:

$$BreathesAir(iguanas) \wedge Scales(iguanas)$$

… and if we apply the rule $BreathesAir(x) \wedge Scales(x) \Longrightarrow Reptile(x)$, then the database can only get larger. It becomes this:

$$BreathesAir(iguanas) \wedge Scales(iguanas) \wedge Reptile(iguanas)$$

Forward-chaining just keeps going, until the fact we want is part of the database, or until we can't prove any more facts.

Outline

- Chain-of-thought prompting of large language models
- AI-guided scientific discovery
- Theorem proving as a search problem
- Zeroth-Order and First-Order Logic
- Proving “there exists” vs. “for all” theorems
- Variable normalization, Unification, Forward- & Backward-chaining

Quantification

- It is sometimes useful to express compound propositions that are true for some values of their variables, but not all.
- To do this, we introduce two new symbols, called quantifiers:
- ∃ (there exists: Existence theorems)
  - Suppose P is the proposition $\mathrm{P} = \exists x\colon F(x)$
  - Then $P = T$ if and only if, for at least one value of the variable x, $F(x) = T$
- ∀ (for all: Universality theorems)
  - Suppose P is the proposition $\mathrm{P} = \forall x\colon F(x)$
  - Then $P = T$ if and only if, for all values of the variable x, $F(x) = T$

Existence theorems

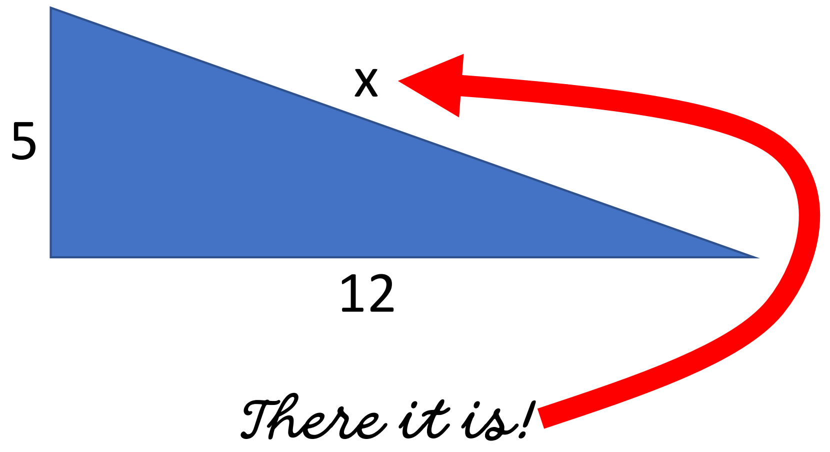
An existence theorem is a theorem of the form "there exists an x such that F(x)," which we write as

$$P = \exists x \colon F(x)$$

An existence theorem:

- … can be **<u>proven</u>** by finding any x that satisfies the conditions.
- …but to **<u>disprove</u>** the statement P, you must find a proposition Q, known to be true, such that $\mathrm{P} \wedge Q$ is logically impossible

Find x:

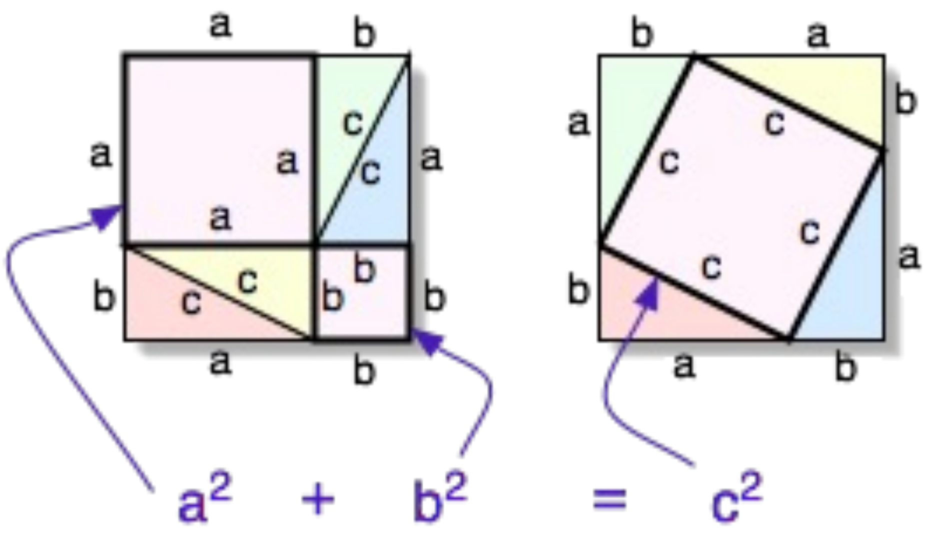


Universality theorems

A universality theorem claims that $F(x)$ is true for all x. We write it as

$$P = \forall x: F(x)$$

- To **disprove** P, you just need to find a counterexample, i.e., you just need to prove that $\exists x: \neg F(x)$.
- To **prove** P, you must find a proposition Q, known to be true, such that $\neg \mathrm{P} \wedge Q$ is logically impossible



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

- To **<u>prove an existence theorem</u>**, 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.
- To disprove an existence theorem, or **<u>prove a universality theorem</u>**, you need to prove that the existence of any such x would contradict known true propositions.
  - Can be done using ***resolution***, which is closely related to unification…
  - …but will not be covered in this course.

Outline

- Chain-of-thought prompting of large language models
- AI-guided scientific discovery
- Theorem proving as a search problem
- Zeroth-Order and First-Order Logic
- Proving "there exists" vs. "for all" theorems
- Variable normalization, Unification, Forward- & Backward-chaining

Theorem proving

Consider the statements:

1. Chocolate is sweet
2. If something is sweet, then Jack likes it

From those, can we prove that:

3. There is somebody who likes something

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

Variable Normalization

1. $Sweet(chocolate)$
2. $\forall x: Sweet(x) \Rightarrow Likes(jack, x)$
3. $\exists x, y: Likes(x, y)$

Propositions (1) and (2) prove proposition (3), but this fact is obfuscated by the different meanings of the variable x in propositions (2) versus (3).

Automatic proof needs normalized 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 ***<u>each rule uses a different set of variable names</u>***
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

Unification

1. $Sweet(chocolate)$
2. $Sweet(x_1) \Rightarrow Likes(jack, x_1)$
3. $\exists x_2, y_1 : Likes(x_2, y_1)$

Unification finds a substitution $S: \{\mathcal{V}_P, \mathcal{V}_Q\} \rightarrow \{\mathcal{V}_Q, C\}$ that unifies the propositions P and Q, i.e., makes them into one unified proposition. For example, the substitution

$$S: \{x_1, x_2, y_1\} \rightarrow \{y_1, jack, y_1\}$$

…unifies propositions (2) and (3) to the unified proposition:

$$\exists y_1 : Sweet(y_1), Likes(jack, y_1)$$

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

Forward-chaining

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

- Starting state: a database of known true propositions, $\mathcal{D} = \{P_1, P_2, \ldots\}$
- Actions: unify two of the known propositions, P_i and P_j, to make a new unified proposition $S\left(P_i, P_j\right)$, and add it to the database
- 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 $Sweet(chocolate)$. Result:
   $\mathcal{D}' = \{Sweet(chocolate), Likes(jack, chocolate)\}$

2. Unify $Likes(x_2, y_1)$ to $Likes(jack, chocolate)$.
   Result:

$$\mathcal{D}'' = \left\{\begin{matrix} Sweet(chocolate), Likes(jack, chocolate), \\ \exists jack, chocolate: Likes(jack, chocolate) \end{matrix}\right\}$$

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

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 $P \Longrightarrow Q$, 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.
- 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 $Likes(x_2, y_1)$. Result:

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

Proof step 2:

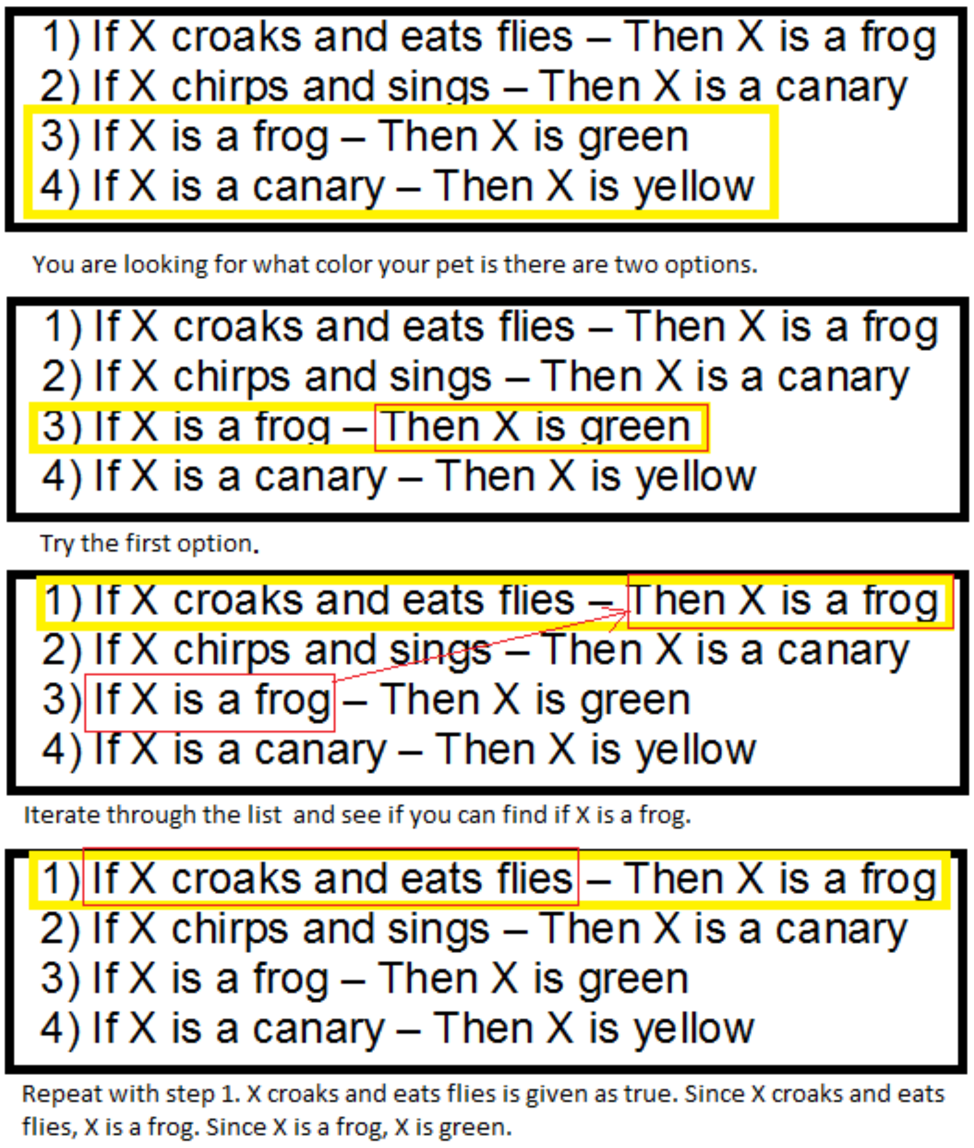
Unify $Sweet(x_1)$ to $Sweet(chocolate)$. Result:

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

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

Another example (from Wikipedia)

- We want to find out if our pet is green or yellow.
- If it's a frog, then it's green.
- If it croaks and eats flies, then it's a frog.
- It croaks and eats flies. Success!



1) If X croaks and eats flies – Then X is a frog
2) If X chirps and sings – Then X is a canary
3) If X is a frog – Then X is green
4) If X is a canary – Then X is yellow

You are looking for what color your pet is there are two options.

1) If X croaks and eats flies – Then X is a frog
2) If X chirps and sings – Then X is a canary
3) If X is a frog – Then X is green
4) If X is a canary – Then X is yellow

Try the first option.

1) If X croaks and eats flies – Then X is a frog
2) If X chirps and sings – Then X is a canary
3) If X is a frog – Then X is green
4) If X is a canary – Then X is yellow

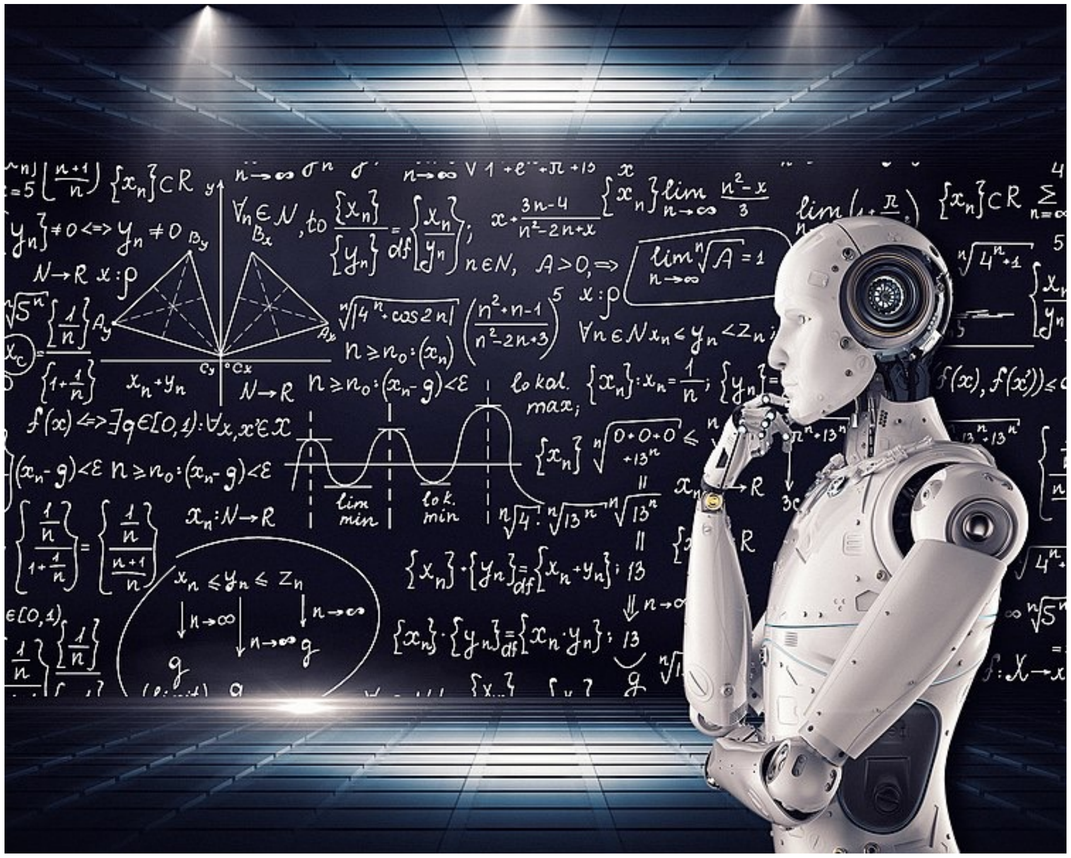
Iterate through the list and see if you can find if X is a frog.

1) If X croaks and eats flies – Then X is a frog
2) If X chirps and sings – Then X is a canary
3) If X is a frog – Then X is green
4) If X is a canary – Then X is yellow

Repeat with step 1. X croaks and eats flies is given as true. Since X croaks and eats flies, X is a frog. Since X is a frog, X is green.

Quiz

Try the quiz!



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 $\mathrm{S}(P) = S(Q) = U$, 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 set of propositions whose truth needs to be proven)