

# AI101

## Logic and AI 2: First-Order Logic

First-order logic allows the use of quantifiers and variables: "there exists  $x$  such that  $x$  is Socrates and  $x$  is a man", where "there exists" is a quantifier, while  $x$  is a variable

WISSENSCHAFTEN IN EINZELDARSTELLUNGEN

BAND XXVII

D. HILBERT † UND W. ACKERMANN

# GRUNDZÜGE DER THEORETISCHEN LOGIK

DRITTE AUFLAGE

Many logicians reckon that the appearance of this book in 1928 marked the birth of first-order logic.



# Recap

## Propositional Logic

Propositional Logic allows to do capture our **knowledge about the world** , and **logically draw conclusions** from it,

- knowledge bases
- Resolution

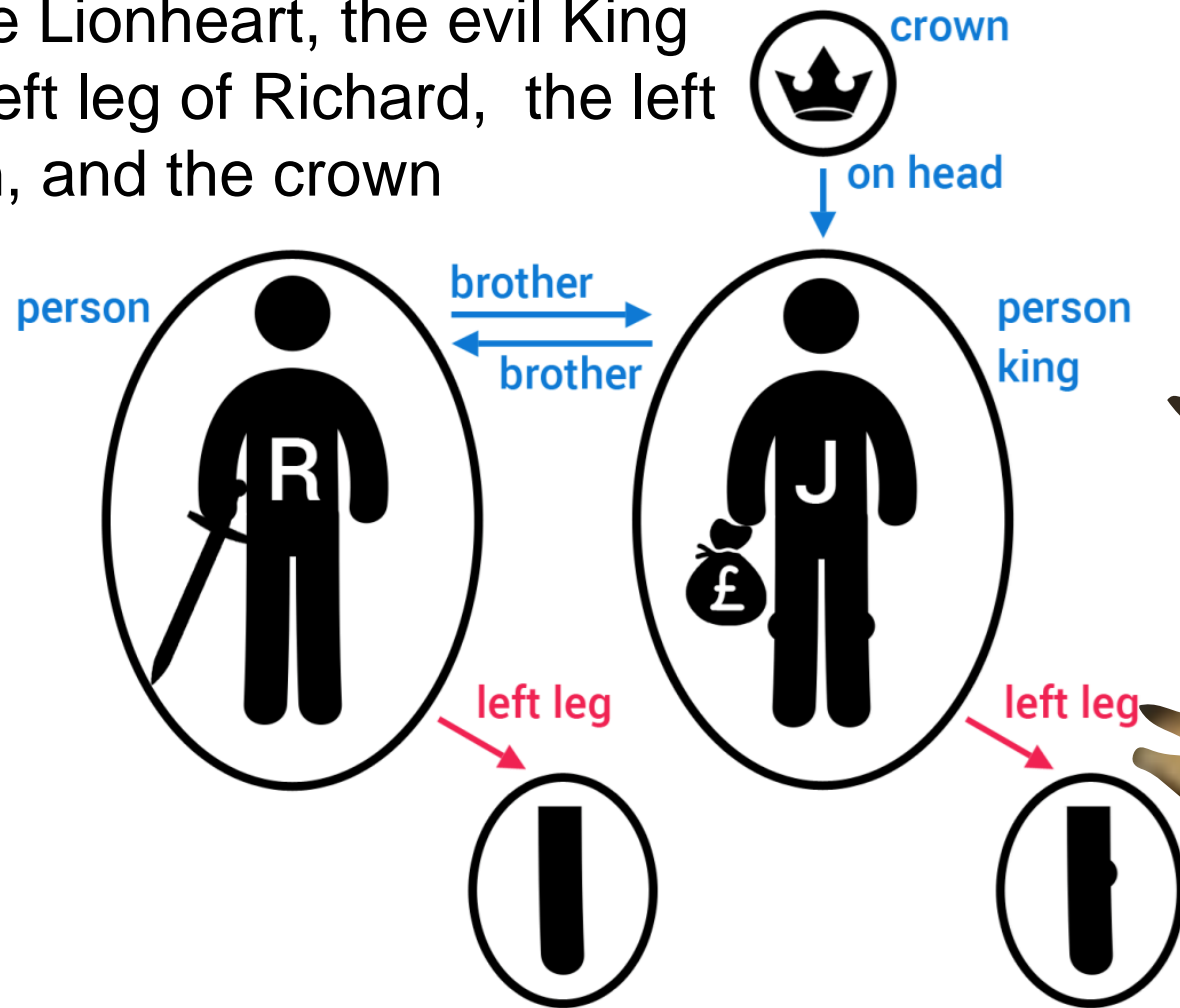
However, there are major limitations. It deals with atomic facts only

- **Has no notion of objects and relations among them**
- **Terms have no formal meaning** only intuition by the reader
  - E.g. RoommateCarryingUmbrella is instructive to us but we could also replace it with “P” without losing anything formally

Thanks to Claudia Chirita from the U. Edingburgh for making her slides available.

## Five Objects:

Richard the Lionheart, the evil King  
John, the left leg of Richard, the left  
leg of John, and the crown



## Compact Knowledge Representation

Example of brothers and kings

# First-Order Logic

## Outline

Can we introduce the notion of an object and use variables (placeholders) in logic to quantify over objects within relations and functions?

### Today

## First Order Logic

- Syntax
- Semantics
- Unification
- Skolem
- Resolution



# Recall: The Roommate Story

You roommate comes home; he/she is completely wet. You know the following things:

- Your roommate is wet
- If your roommate is wet, it is because of rain, sprinklers, or both
- If your roommate is wet because of sprinklers, the sprinklers must be on
- If your roommate is wet because of rain, your roommate must not be carrying the umbrella
- The umbrella is not in the umbrella holder
- If the umbrella is not in the umbrella holder, either you must be carrying the umbrella, or your roommate must be carrying the umbrella
- You are not carrying the umbrella

**How to encode this knowledge with a notion of object at hand?**



WikiMedia Commons, A man walks through heavy rain under an umbrella in Kigali, Rwanda. Emmanuel Kwizera.jpg

# Elements of First Order Logic (FOL)

## Objects:

- can give these names such as Umbrella0, Person0, John, Earth, ...

## Relations:

- $\text{Carrying}(\cdot, \cdot) \rightarrow \text{Carrying}(\text{Person0}, \text{Umbrella0})$ ,
- $\text{IsUmbrella}(\cdot) \rightarrow \text{IsUmbrella}(\text{Umbrella0})$
- Relations with one object = **unary relations** = **properties**

## Functions (referring to objects without a name):

- $\text{Roommate}(\cdot) \rightarrow \text{Roommate}(\text{Person0})$

## Equality:

- $\text{Roommate}(\text{Person0}) = \text{Person1}$

# Things to note about **functions**

- Can be used to encode integers and also data structures.
  - 0, succ(0), succ(succ(0)), ...
  - tree(value2, tree(value2, ...), tree(value3, ...))
- It could be that we have a separate name for Roommate(Person0)
  - E.g., Roommate(Person0) = Person1
  - ... but we do not **need** to have such a name
- A function can be applied to any object  
E.g., Roommate(Umbrella0)

# Well, predicates describe **the truth** in the world



<https://www.youtube.com/watch?v=PAZTIAfaNr8>



# Universal and Existential Quantifier

How do we talk about many objects at once?

We can define variables (placeholders):  $x, y, z$

- Can refer to a single but also to multiple objects

Can we quantify objects like “any”, “some”, “all”,...?

For this we need 2 quantifiers:

1. For all elements
2. There exist one element, that...

## For all

**All lions are cats:**

$\forall x: \text{Lion}(x) \Rightarrow \text{Cat}(x)$

For all  $x$ :  $\text{Lion}(x) \Rightarrow \text{Cat}(x)$

## There exists...

**There is a cat that is not a lion:**

$\exists x: \text{Cat}(x) \Rightarrow \text{NOT}(\text{Lion}(x))$

There exists  $x$ :  $\text{Cat}(x) \Rightarrow \text{NOT}(\text{Lion}(x))$

# English to FOL (1)

“John has Jane’s umbrella”

- $\text{Has}(\text{John}, \text{Umbrella}(\text{Jane}))$

“John has an umbrella”

- there exists  $y$ :  $(\text{Has}(\text{John}, y) \text{ AND } \text{IsUmbrella}(y))$

“Anything that has an umbrella is not wet”

- for all  $x$ :  $((\text{there exists } y: (\text{Has}(x, y) \text{ AND } \text{IsUmbrella}(y))) \Rightarrow \text{NOT}(\text{IsWet}(x)))$

“Any person who has an umbrella is not wet”

- for all  $x$ :  $(\text{IsPerson}(x) \Rightarrow ((\text{there exists } y: (\text{Has}(x, y) \text{ AND } \text{IsUmbrella}(y))) \Rightarrow \text{NOT}(\text{IsWet}(x))))$

# English to FOL (2)

“John has at least two umbrellas”

- there exists  $x$ : (there exists  $y$ : ( $\text{Has}(\text{John}, x) \text{ AND } \text{IsUmbrella}(x) \text{ AND } \text{Has}(\text{John}, y) \text{ AND } \text{IsUmbrella}(y) \text{ AND NOT}(x=y)$ ))

“John has at most two umbrellas”

- for all  $x, y, z$ : ( $(\text{Has}(\text{John}, x) \text{ AND } \text{IsUmbrella}(x) \text{ AND } \text{Has}(\text{John}, y) \text{ AND } \text{IsUmbrella}(y) \text{ AND } \text{Has}(\text{John}, z) \text{ AND } \text{IsUmbrella}(z)) \Rightarrow (x=y \text{ OR } x=z \text{ OR } y=z)$ )

Some open examples for you at home

- “TUDa’s basketball team defeats any other basketball team”
- “Every team defeats some other team”

# Relation between Universal and Existential

## For all

**All lions are cats:**

For all  $x$ :  $\text{Lion}(x) \Rightarrow \text{Cat}(x)$

## There exists...

**There is no lion that is not a cat**

$\text{NOT}(\text{there exists } x: \text{Lion}(x) \text{ AND } \text{NOT}(\text{Cat}(x)))$

Or in general:

**For all  $x$ :  $a$  is equivalent to  $\text{NOT}(\text{there exists } x: \text{NOT}(a))$**

# Axioms and Theorems

## Or FOL and Academic Writing

Before showing how to reason within FOL, let us clarify two common terms (that you can also find in many scientific papers)

### Axioms

Basic facts about the domain, our “initial” knowledge base

### Theorems

Statements that are logically derived from axioms

## ACADEMIC WRITING

### Definition

Academic writing or Scholarly writing is clear, concise, focused, structured and backed up by evidence. Its purpose is to aid the reader's understanding.

### PURPOSE

The purpose is to present information in order to display a clear understanding of a subject.

### CHARACTERISTICS

Clear and Concise Focus  
Logical Structure  
Evidence  
Formal tone and style

### FEATURES

Complexity - Formality - Precision -  
Objectivity - Explicitness - Accuracy -  
Hedging - Responsibility -  
Organisation - Planning

### AREAS

Essay writing - Report writing -  
Speech writing - Dissertation writing -  
Thesis writing - Research paper  
writing - Term paper writing -  
Assignment writing - Homework  
writing

### ACADEMIC WRITING SERVICE

MyAssignmentHelp is a reliable and professional academic writing service. It is recommended by millions of netizens and students studying across the globe.

### LEGALITY

It is that if you submit the provided solution under the impression that it is yours. It is always recommended to the students that they study the solution

### CUSTOMER SUPPORT

Around the clock customer support service through online chat/SMS/email. You can also get assistance over the call 24\*7. SMS notifications for all updates about your order.



# Substitution

Anyway, how do we arrive at a systematic inference approach?

Since we can now talk about sets of objects, we must be able to talk about smaller, more specific sets:

## SUBST(itution)

SUBST replaces one or more variables with something else in a sentence

### Example:

- $\text{SUBST}(\{x/\text{John}\}, \text{IsHealthy}(x) \Rightarrow \text{NOT}(\text{HasACold}(x)))$  gives us
- $\text{IsHealthy}(\text{John}) \Rightarrow \text{NOT}(\text{HasACold}(\text{John}))$

# Instantiating quantifiers with SUBST

From

**for all  $x$ :  $a$**

we can obtain for any  $g$

**$\text{SUBST}(\{x/g\}, a)$**

we may add to the KB any instantiation of that sentence  $a$ , where the logic variable  $x$  is replaced by a concrete **ground term**  $g$ :

## Important:

- $k$  is a constant that does not appear elsewhere in the knowledge base (**Skolem constant**)
- We don't need the original sentence anymore
- Be careful when instantiate existentials after universals  
→ see next slide

From

**there exists  $x$ :  $a$**

we can obtain for one  $k$

**$\text{SUBST}(\{x/k\}, a)$**

we may add a single instantiation of that sentence  $a$  to the KB, where the logic variable  $x$  is replaced by a **Skolem constant**  $k$  which must not appear elsewhere in the knowledge base

# Instantiating quantifiers with SUBST

If we want to instantiate existentials after universals, we have to be a bit more careful

Let's assume **for all x: there exists y: IsParentOf(x,y)**

- ~~for all x: IsParentOf(x,k)~~
- **for all x: IsParentOf(x, k(x))**

**The Skolem constant depends on x, i.e., it becomes a function**

Intuition:

1. When we only have an existential quantifier, replace a variable y with a constant k
2. In the case above we say “for every x there is a y such that...”. This y does not have to be the same for each x. So we have to find a specific y for every x:

$$k : x \rightarrow y = k(x)$$

we replace the existential quantifier by defining a function, mapping x to y

3. This is identical to “There exist a function k, mapping every x into a y such that for every x IsParentOf(x, k(x)) holds”.

Albert Thoralf Skolem



# Moving on to reasoning: Generalized modus ponens

**for all x: Loves(John, x)**

- John loves every thing

**for all y: (Loves(y, Jane)  $\Rightarrow$  FeelsAppreciatedBy(Jane, y))**

- Jane feels appreciated by every thing that loves her

Can infer from this: **FeelsAppreciatedBy(Jane, John)**

Here, we used the substitution  $\{x/\text{Jane}, y/\text{John}\}$  and first get rid of existential quantified (moving them after the all quantifiers and then skolemize)

- Note we used different variables for the different sentences

General UNIFY algorithms for finding a good substitution (not touched here)

# UNIFY

Keeping things as general as possible in unification

Consider EdibleByWith

- e.g., EdibleByWith(Soup, John, Spoon) – John can eat soup with a spoon

**for all x: for all y: EdibleByWith(Bread, x, y)**

- Anything can eat bread with anything

**for all u: for all v: (EdibleByWith(u, v, Spoon) => CanBeServedInBowlTo(u,v))**

- Anything that is edible with a spoon by something can be served in a bowl to that something

Substitution: {x/z, y/Spoon, u/Bread, v/z}

Gives: **for all z: CanBeServedInBowlTo(Bread, z)**

Alternative substitution {x/John, y/Spoon, u/Bread, v/John} would only have given CanBeServedInBowlTo(Bread, John), which is not as general



# Resolution for FOL

**for all x: (NOT(Knows(John, x)) OR IsMean(x) OR Loves(John, x))**

- John loves everything he knows, with the possible exception of mean things

**for all y: (Loves(Jane, y) OR Knows(y, Jane))**

- Jane loves everything that does not know her

What can we unify? What can we conclude?

Use the substitution: {x/Jane, y/John}

Get: **IsMean(Jane) OR Loves(John, Jane) OR Loves(Jane, John)**

Complete (i.e., if not satisfiable, will find a proof of this), if we can remove literals that are duplicates after unification

- Also need to put everything in **canonical form** first

# First-order CNF (our canonical form used for resolution)

1. Convert to **Negation Normal Form**, i.e., negation symbols only occur immediately before predicate symbols
2. If variable names are used twice within scopes of different quantifiers, **rename** one of the such that the name is not used anywhere else.
3. **Skolemize the statements:**
  1. Move quantifiers out so that we get ForAll  $X_1, X_2, \dots$  Exists  $Y_1, Y_2, \dots$  Formula. Quantifiers only appear in the initial prefix but never inside NOT, AND, or OR.
  2. Replace existentially quantified variables by Skolem functions
  3. Discard all universal quantifiers
4. **Convert** into clause set

# First-order CNF (our canonical form used for resolution)

## Example

### Example

1. Anything anyone eats and is not killed, is food.
2. For all x: For all y:  $\text{eats}(x,y) \text{ AND } \text{NOT}(\text{killed}(x)) \Rightarrow \text{food}(y)$
3. For all x: For all y:  $\text{NOT}(\text{eats}(x,y) \text{ AND } \text{NOT}(\text{killed}(x))) \text{ OR } \text{food}(y)$
4. For all x: For all y:  $\text{NOT}(\text{eats}(x,y)) \text{ OR } \text{killed}(x) \text{ OR } \text{food}(y)$
5.  $\text{NOT}(\text{eats}(x,y)) \text{ OR } \text{killed}(x) \text{ OR } \text{food}(y)$

### Steps to CNF (see previous slide)

1. Text to FOL
2. Eliminate all implications
3. Move Negations inwards
4. Rename variables
5. Eliminate existential quantifiers (Skolemization)
6. Drop universal quantifiers
7. Convert into clauses

We skipped steps 4 since we do not have duplication in the variables, step 5 since we do not have existential quantifiers and step 7 since we have already clauses.

# Notes on inference in FOL

Deciding whether a sentence is entailed is **semidecidable**:

- there are algorithms that will eventually produce a proof of any entailed sentence

It is **not** **decidable**:

- we cannot always conclude that a sentence is not entailed

# Nevertheless, still a lot of fun to Program in Logic (Prolog)

The above shows an ordinary constraint between two variables:

Person and Food

Prolog makes you name this constraint.

(Person, Food)


Here's a program that defines it:

- `eats(sam, dal).`                      `eats(josie, samosas).`
- `eats(sam, curry).`                    `eats(josie, curry).`
- `eats(rajiv, burgers).`                `eats(rajiv, dal). ...`

Now it acts like a subroutine! At the Prolog prompt you can type

- `eats(Person1, Food1).`    % constraint over two variables
- `eats(Person2, Food2).`    % constraint over two **other** variables

Person	Food
sam	dal
sam	curry
josie	samosas
josie	curry
rajiv	burgers
rajiv	dal



# Simple constraints in Prolog

Here's a program defining the “eats” constraint:



- `eats(sam, dal).`                      `eats(josie, samosas).`
- `eats(sam, curry).`                      `eats(josie, curry).`
- `eats(rajiv, burgers).`                      `eats(rajiv, dal).` ...
- Now at the Prolog prompt you can type
  - `eats(Person1, Food1).`    % constraint over two variables
  - `eats(Person2, Food2).`    % constraint over two **other** variables

To say that Person1 and Person2 must eat a common food, conjoin two constraints with a **comma**:

- `eats(Person1, Food), eats(Person2, Food).`
- Prolog gives you possible solutions:
  - `Person1=sam, Person2=sam, Food=dal,`
  - `Person1=sam, Person2=sam, Food=curry`
  - `Person1=sam, Person2=josie, Food=curry` ...

# Queries in Prolog

The things you type at the prompt are called “queries.”

- Prolog answers a query as “Yes” or “No” according to whether it can find a satisfying assignment.
  - If it finds an assignment, it prints the first one before printing “Yes.”
  - You can press Enter to accept it, in which case you’re done, or “;” to reject it, causing Prolog to backtrack and look for another.
    - `eats(Person1, Food1).` % constraint over two variables
    - `eats(Person2, Food2).` % constraint over two **other** variables
- 
- `eats(Person1, Food), eats(Person2, Food).`
  - Prolog gives you possible solutions:
    - `Person1=sam, Person2=josie, Food=curry` [ press “;” ]
    - `Person1=josie, Person2=sam, Food=curry` ...
- 

# Constants vs. Variables

Here's a program defining the “eats” constraint:

- `eats(sam, dal).`                      `eats(josie, samosas).`
- `eats(sam, curry).`                      `eats(josie, curry).`
- `eats(rajiv, burgers).`    ...
- Now at the Prolog prompt you can type
  - `eats(Person1, Food1).`    % constraint over two variables
  - `eats(Person2, Food2).`    % constraint over two **other** variables

Nothing stops you from putting constants into constraints:

- `eats(josie, Food).` % what Food does Josie eat? (2 answers)
- `eats(Person, curry).`        % what Person eats curry? (2 answers)
- `eats(josie, Food), eats(Person, Food).` % who'll share what with Josie?
  - `Food=curry, Person=sam`

# Constants vs. Variables

- Variables start with A,B,...Z or underscore:
  - Food, Person, Person2, \_G123
- Constant “atoms” start with a,b,...z or appear in single quotes:
  - josie, curry, 'CS325'
  - Other kinds of constants besides atoms:
    - Integers -7, real numbers 3.14159, the empty list []
    - eats(josie,curry) is technically a constant structure

Nothing stops you from putting constants into constraints:

- `eats(josie, Food).` % what Food does Josie eat? (2 answers)
- `eats(Person, curry).` % what Person eats curry? (2 answers)
- `eats(josie, Food), eats(Person, Food).` % who'll share what with Josie?
  - `Food=curry, Person=sam`

# Rules in Prolog

Let's augment our program with a new constraint:

`eats(sam, dal).`

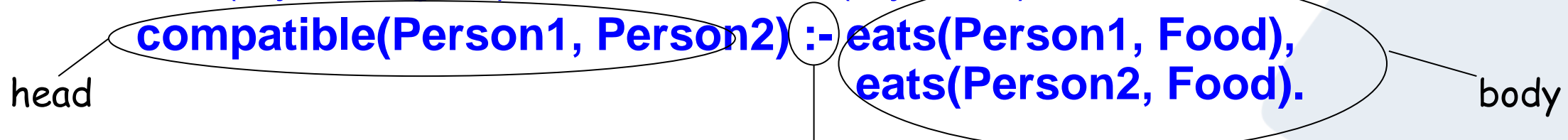
`eats(josie, samosas).`

`eats(sam, curry).`

`eats(josie, curry).`

`eats(rajiv, burgers).`

`eats(rajiv, dal).`



means "if" - it's supposed to look like "←"

"Person1 and Person2 are compatible if there exists some Food that they both eat."

"One way to satisfy the head of this rule is to satisfy the body."

You type the query: `compatible(rajiv, X).` Prolog answers: `X=sam.`

- Prolog doesn't report that `Person1=rajiv`, `Person2=sam`, `Food=dal`. These act like local variables in the rule. It already forgot about them.



# The Prolog Solver (Inference Engine)

Prolog's solver is incredibly simple: depth-first search using unification and backtracking

Query: `eats(sam,X).`

- Iterates in order through the program's "eats" clauses.
- First one to match is `eats(sam,dal).`  
so it returns with `X=dal.`
- If you hit semicolon, it backtracks and continues:  
Next match is `eats(sam,curry).`  
so it returns with `X=curry.`

# Prolog as a Database Language

The various `eats(..., ...)` facts can be regarded as rows in a database (2-column database in this case).

Standard relational database operations:

- `eats(X,dal).` % select
- `edible(Object) :- eats(Someone, Object).` % project
- `parent(X,Y) :- mother(X,Y).` % union  
  `parent(X,Y) :- father(X,Y).`
- `sister_in_law(X,Z) :- sister(X,Y), married(Y,Z).` % join

Why the heck does anyone still use SQL? Beats me.

Warning: Prolog's backtracking strategy can be inefficient.

- But we can keep the little language illustrated above ("Datalog") and instead compile into optimized query plans, just as for SQL.

# Prolog and Horn Clauses



<https://www.youtube.com/watch?v=jr0gRcAyOxk>

# Gödel's Incompleteness Theorem

## Extremely Informal Statement

1. First-order logic is not rich enough to model basic arithmetic
2. For any consistent system of axioms that is rich enough to capture basic arithmetic (in particular, mathematical induction), there exist true sentences that cannot be proved from those axioms
3. This shows that we have not the chance to prove every truth in the “universe”!

Bild: Logicomix





# Gödel's Incompleteness Theorem



<https://www.youtube.com/watch?v=l4pQbo5MQOs>

# Gödel's Incompleteness Theorem

In “Dangerous Knowledge”, David Malone looks at four brilliant mathematicians - Georg Cantor, Ludwig Boltzmann, Kurt Gödel and Alan Turing - whose genius has profoundly affected us ...



<https://www.dailymotion.com/video/xdoj8c>



# A more challenging exercise

## Suppose:

- There are exactly 3 objects in the world,
- If  $x$  is the spouse of  $y$ , then  $y$  is the spouse of  $x$  (spouse is a function, i.e., everything has a spouse)

## Prove:

- Something is its own spouse

there exist  $x, y, z$ :  $(\text{NOT}(x=y) \text{ AND } \text{NOT}(x=z) \text{ AND } \text{NOT}(y=z))$

for all  $w, x, y, z$ :  $(w=x \text{ OR } w=y \text{ OR } w=z \text{ OR } x=y \text{ OR } x=z \text{ OR } y=z)$

for all  $x, y$ :  $((\text{Spouse}(x)=y) \Rightarrow (\text{Spouse}(y)=x))$

for all  $x, y$ :  $((\text{Spouse}(x)=y) \Rightarrow \text{NOT}(x=y))$  *(for the sake of contradiction)*



# Umbrellas in first-order logic

You know the following things:

- You have exactly one other person living in your house, who is wet
- If a person is wet, it is because of the rain, the sprinklers, or both
- If a person is wet because of the sprinklers, the sprinklers must be on
- If a person is wet because of rain, that person must not be carrying any umbrella
- There is an umbrella that “lives in” your house, which is not in its house
- An umbrella that is not in its house must be carried by some person who lives in that house
- You are not carrying any umbrella

Can you conclude that the sprinklers are on?

# Applications

Some serious novel mathematical results proved

Verification of hardware and software

- Prove outputs satisfy required properties for all inputs

Synthesis of hardware and software

- Try to prove that there exists a program satisfying such and such properties,  
**in a constructive way**

# Forward and Backward Chaining

## Forward Chaining

- Bottom-up approach → Starting from the initial state and reaches the goal state.
- Process of making a conclusion based on known facts or data
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.

## Backward Chaining

- Top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- The backward-chaining method mostly used a **depth-first search** strategy for proof.

An example for forward and backward chaining can be found under: <https://www.javatpoint.com/forward-chaining-and-backward-chaining-in-ai>

# Something we cannot do in first-order logic

- We are **not** allowed to reason in general about relations and functions
- The following would correspond to **higher-order logic** (which is more powerful):

“If John is Jack’s roommate, then any property of John is also a property of Jack’s roommate”

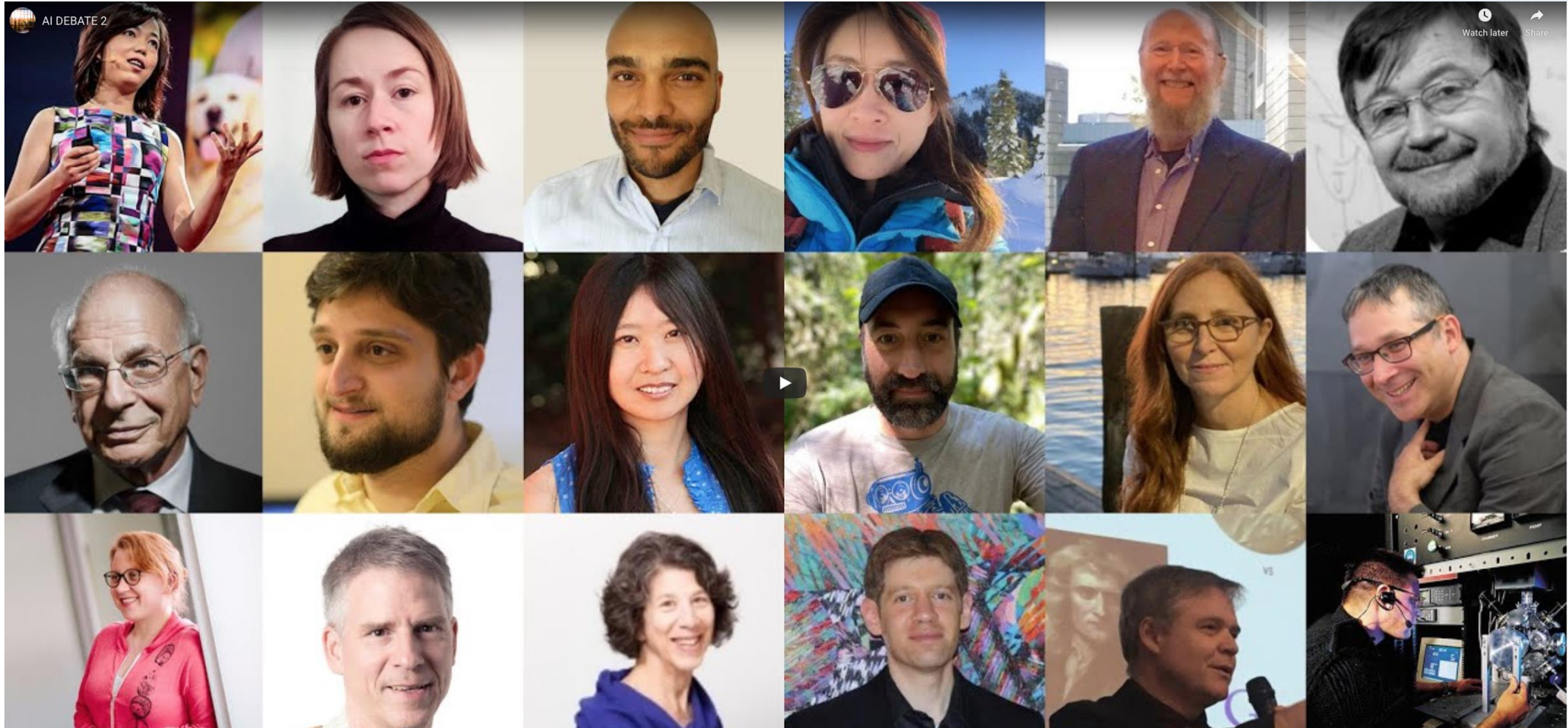
- **$(\text{John}=\text{Roommate}(\text{Jack})) \Rightarrow \text{for all } p: (p(\text{John}) \Rightarrow p(\text{Roommate}(\text{Jack})))$**

“If a property is inherited by children, then for any thing, if that property is true of it, it must also be true for any child of it”

- **$\text{for all } p: (\text{IsInheritedByChildren}(p) \Rightarrow (\text{for all } x, y: ((\text{IsChildOf}(x,y) \text{ AND } p(y)) \Rightarrow p(x))))$**

# AI Debate 2

How to get logical reasoning and deep learning "under one umbrella"?



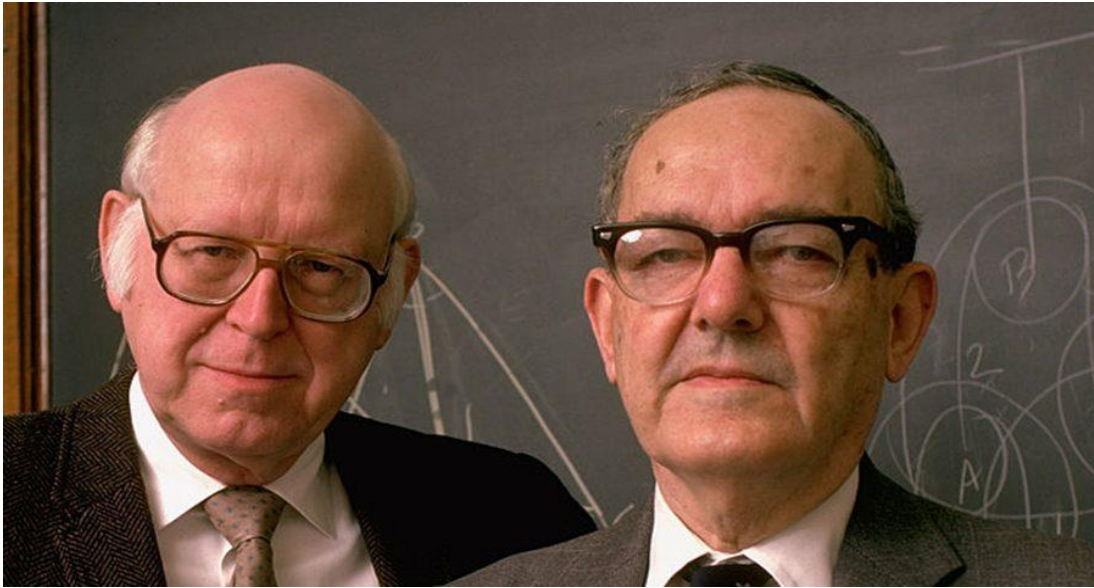
<https://montrealartificialintelligence.com/aidebate2.html>



# Symbols, Neurons and Neuro-symbolic?

## Symbols

“A physical symbol system has the necessary and sufficient means for general intelligent action.”



Allen Newell & Herbert Simon (ACM Turing Awardees)

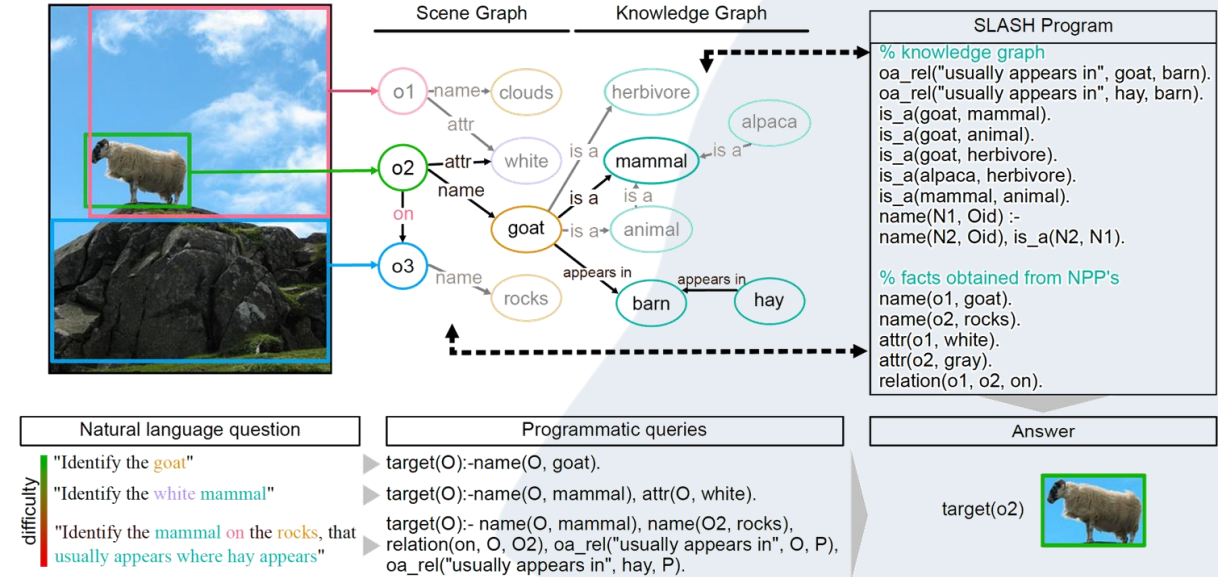
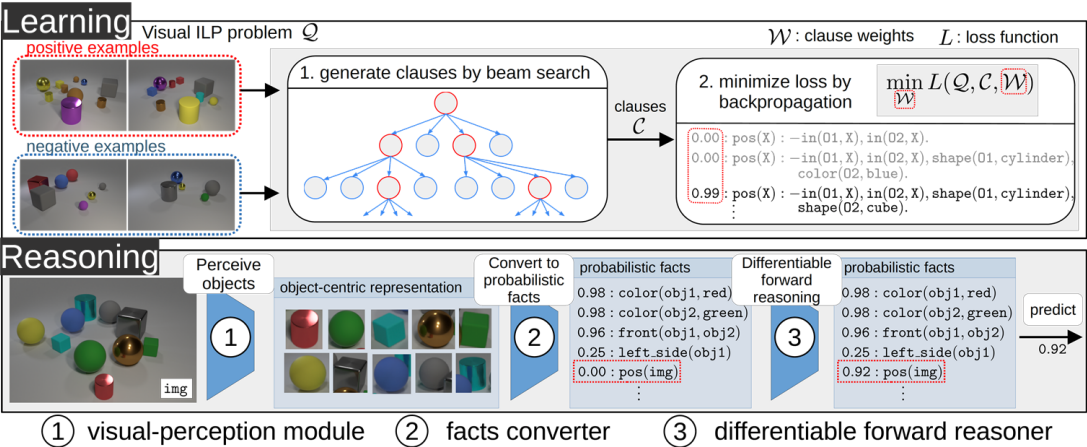
## Neurons

“Symbols are Luminiferous Aether of AI”

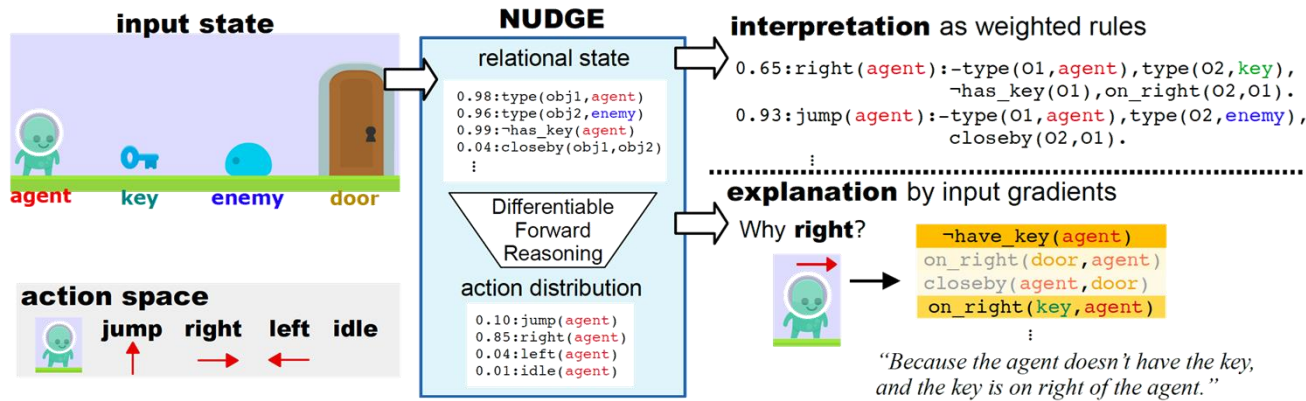


Geoff Hinton (ACM Turing Awardee)

# Symbols, Neurons and Neuro-symbolic?



Visual Question Answering using Neural-Probabilistic Answer-Set Programming (Skryagin, Ochs, Dahmi, Kersting JAIR 2023)



Explainable AND interpretable neuro-symbolic reinforcement learning agents using differentiable logics (Delfosse, Shindo, Dhmi, Kersting NeurIPS 2023)



# Let us focus on the computational and algorithmic level not just the implementational one

Three levels of description (*David Marr, 1982*)

## Computational

Why do things work the way they do?  
What is the goal of the computation?  
What are the unifying principles?

## Algorithmic

What representations can implement  
such computations?  
How does the choice of representations  
determine the algorithm?

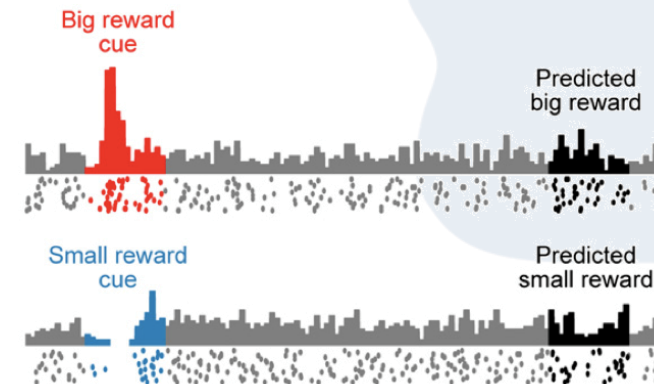
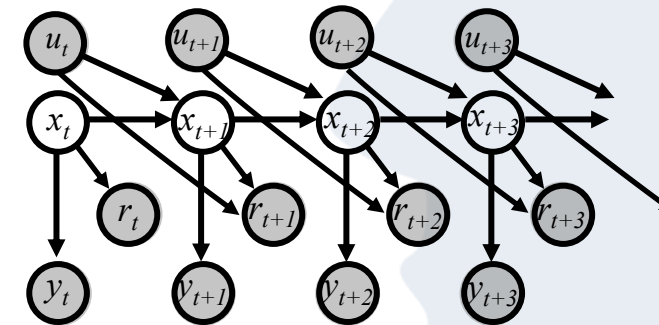
## Implementational

How can such a system be built in  
hardware?  
How can neurons carry out the  
computations?



maximize:

$$R_t = r_{t+1} + r_{t+2} + \dots + r_T$$



# Summary

## First-Order Logic

- Syntax & Semantics
- Variables, relations, and functions
- Skolemization
- FOL Clausal Normal Form
- Resolution
- Incompleteness
- Limits of FOL
- Symbolic / Hybrid AI

### **You should be able to:**

Translate English to FOL and vv

Skolemize sentences

Prove FOL queries using resolution

Argue about incompleteness of FOL

Next Week: Uncertainty