

# Artificial Intelligence

CS 165A

June 6, 2023

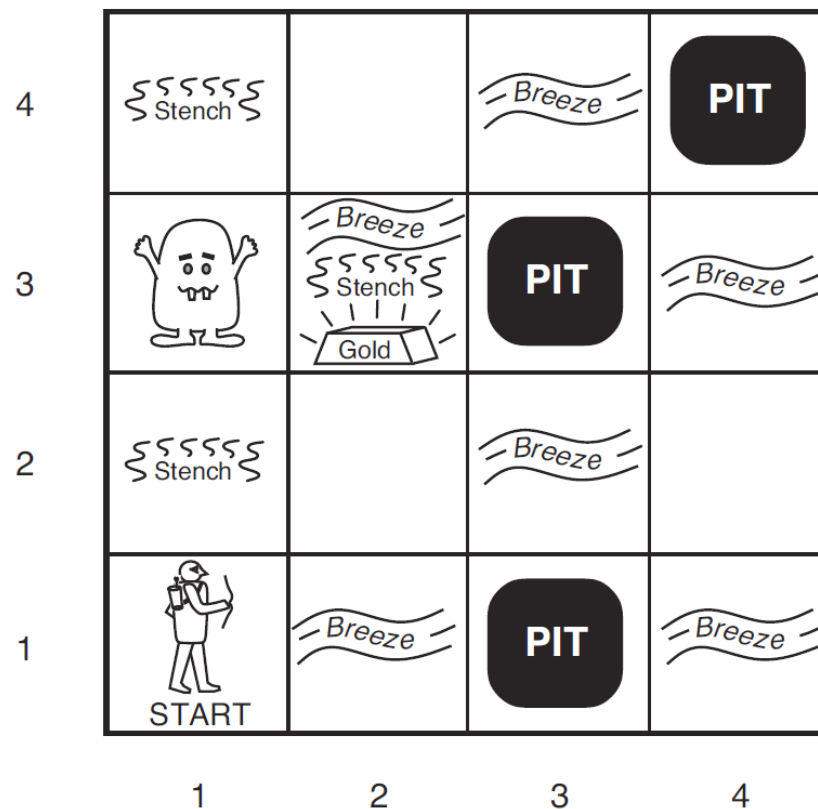
Instructor: Prof. Yu-Xiang Wang

Today

→ First Order Logic

# Recap: Wumpus World

- Logical Reasoning as a CSP
  - $B_{ij}$  = breeze felt
  - $S_{ij}$  = stench smelt
  - $P_{ij}$  = pit here
  - $W_{ij}$  = wumpus here
  - $G$  = gold

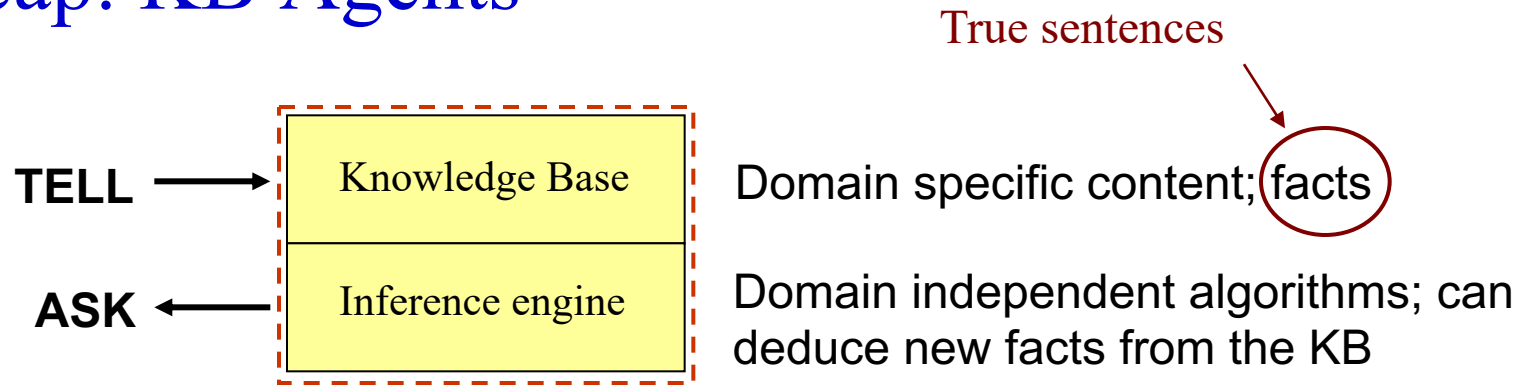


<http://thiagodnf.github.io/wumpus-world-simulator/>

\*The agent can only observe blocks that she has visited.

\*Cannot observe the state directly. So cannot solve offline with search.

# Recap: KB Agents

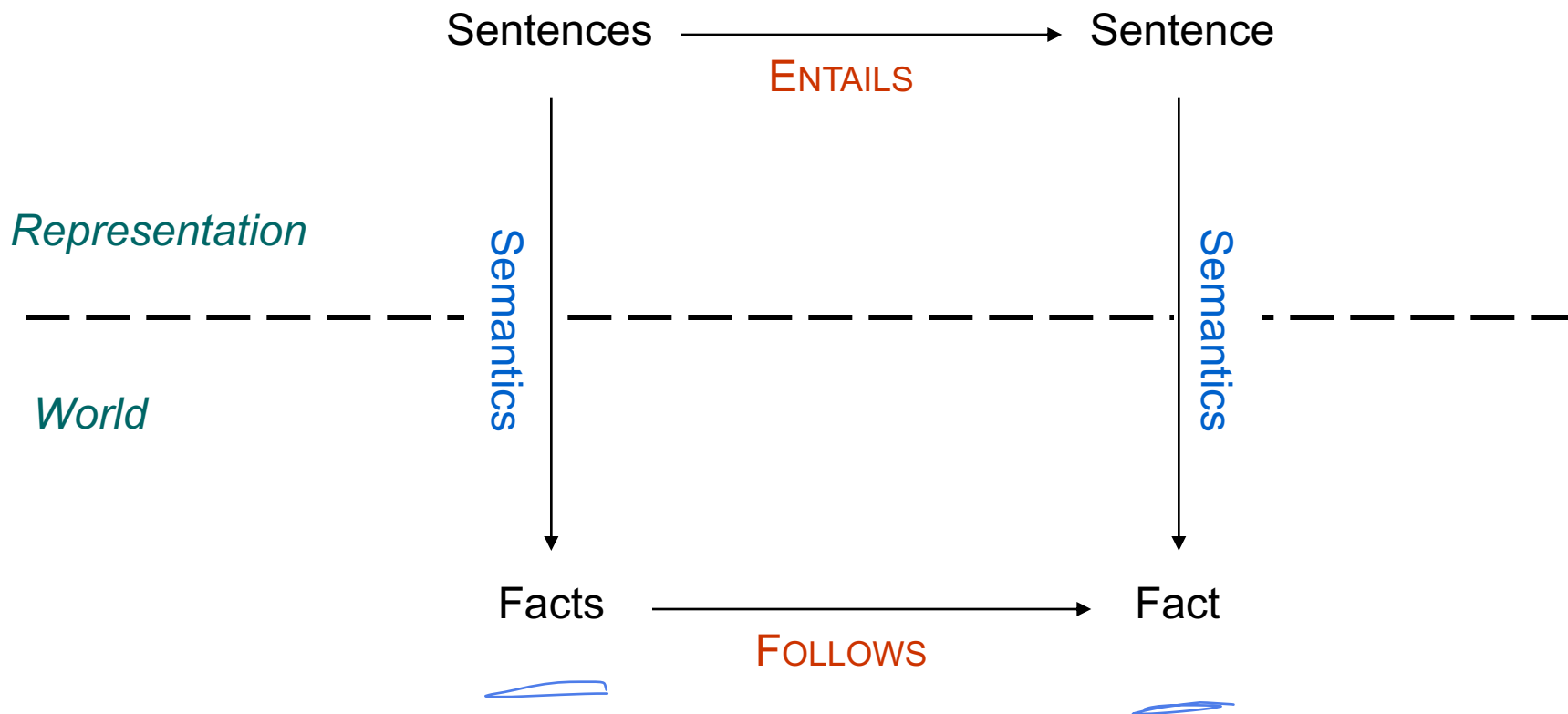


- Need a formal logic system to work
- Need a data structure to represent known facts
- Need an algorithm to answer ASK questions

# Recap: syntax and semantics

- Two components of a logic system
- Syntax --- How to construct sentences
  - The symbols
  - The operators that connect symbols together
  - A precedence ordering
- Semantics --- Rules the assignment of sentences to truth
  - For every possible worlds (or “models” in logic jargon)
  - The truth table is a semantics

# Recap: Entailment



A is entailed by B, if A is true in all possible worlds consistent with B under the semantics.

# Recap: Inference procedure

$$\frac{\alpha \quad \alpha \Rightarrow \beta}{\beta}$$


$$\alpha \models \beta \iff \underline{M(\alpha)} \subseteq \underline{M(\beta)}$$

- Inference procedure
  - Rules (algorithms) that we apply (often recursively) to derive truth from other truth.
  - Could be specific to a particular set of semantics, a particular realization of the world.
- Soundness and completeness of an inference procedure
  - Soundness: All truth discovered are valid.
  - Completeness: All truth that are entailed can be discovered.

# Recap: Propositional Logic

- **Syntax:**
  - *True, false*, propositional symbols
  - $( )$ ,  $\neg$  (not),  $\wedge$  (and),  $\vee$  (or),  $\Rightarrow$  (implies),  $\Leftrightarrow$  (equivalent)
- **Semantics:**
  - Assigning values to the variables. Each row is a “model”.

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>



# Recap: Logical Inference in Propositional Logic

- A simple algorithm for checking: KB entails  $\alpha$ 
  - Enumerate  $M(\text{KB})$
  - Check that it is contained in  $M(\alpha)$
- This inference algorithm is **sound** and **complete**.
- Are there other ways to do logical inference?
- Are they sound / complete?



# Recap: Using propositional logic: rules of inference

- Inference rules capture patterns of sound inference
  - Once established, don't need to show the truth table every time
  - E.g., we can define an inference rule:  $((P \vee H) \wedge \neg H) \vdash P$  for variables  $P$  and  $H$
- Alternate notation for inference rule  $\alpha \vdash \beta$  :

$$\frac{\alpha}{\beta}$$

“If we know  $\alpha$ , then we can conclude  $\beta$ ”

(where  $\alpha$  and  $\beta$  are propositional logic sentences)

# Recap: Inference

- We're particularly interested in

$$\frac{\mathbf{KB}}{\beta} \quad \text{or} \quad \frac{\alpha_1, \alpha_2, \dots}{\beta}$$

- Inference steps

$$\frac{\mathbf{KB}}{\beta_1} \rightarrow \frac{\mathbf{KB}, \beta_1}{\beta_2} \rightarrow \frac{\mathbf{KB}, \beta_1, \beta_2}{\beta_3} \rightarrow \dots$$

**So we need a mechanism to do this!**

**Inference rules that can be applied to sentences in our KB**

# Recap: Resolution Rule: one rule for all inferences

$$\frac{p \vee q, \wedge \neg q \vee r}{p \vee r}$$

Propositional calculus resolution

Remember:  $p \Rightarrow q \Leftrightarrow \neg p \vee q$ , so let's rewrite it as:

$$\frac{\neg p \Rightarrow q, \quad q \Rightarrow r}{\neg p \Rightarrow r}$$

or

$$\frac{a \Rightarrow b, \quad b \Rightarrow c}{a \Rightarrow c}$$

Resolution is really the “chaining” of implications.

# Recap: Conversion to Conjunctive Normal Form: CNF

- Resolution rule is stated for conjunctions of disjunctions
- Question:
  - Can every statement in PL be represented this way?
- Answer: Yes
  - Can show every sentence in propositional logic is equivalent to conjunction of disjunctions
    - Conjunctive normal form (CNF)
- Procedure for obtaining CNF
  - Replace  $(P \Leftrightarrow Q)$  with  $(P \Rightarrow Q)$  and  $(Q \Rightarrow P)$
  - Eliminate implications: Replace  $(P \Rightarrow Q)$  with  $(\neg P \vee Q)$
  - Move  $\neg$  inwards:  $\neg\neg$ ,  $\neg(P \vee Q)$ ,  $\neg(P \wedge Q)$
  - Distribute  $\wedge$  over  $\vee$ , e.g.:  $(P \wedge Q) \vee R$  becomes  $(P \vee R) \wedge (Q \vee R)$   
[What about  $(P \vee Q) \wedge R$  ?]
  - Flatten nesting:  $(P \wedge Q) \wedge R$  becomes  $P \wedge Q \wedge R$

$KB \vdash \alpha$   
 $\neg \alpha \wedge KB \rightarrow \vdash \text{False}$

# Recap: Complexity of reasoning

- Validity
  - NP-complete
- Satisfiability
  - NP-complete
- $\alpha$  is valid iff  $\neg \alpha$  is unsatisfiable
- Efficient decidability test for validity iff efficient decidability test for satisfiability.
- To check if  $KB \models \alpha$ , test if  $(KB \wedge \neg \alpha)$  is unsatisfiable.
- For a restricted set of formulas (Horn clauses), this check can be made in linear time.
  - Forward chaining
  - Backward chaining

# Recap: Propositional logic is quite limited

- Propositional logic has simple syntax and semantics, and limited expressiveness
  - Though it is handy to illustrate the process of inference
- However, it only has one representational device, the proposition, and cannot generalize
  - Input: facts; Output: facts
  - Result: Many, many rules are necessary to represent any non-trivial world
  - It is impractical for even very small worlds
- The solution?
  - **First-order logic**, which can represent propositions, objects, and relations between objects
  - Worlds can be modeled with many fewer rules

# This lecture: First order logic

- More expressive language
  - Relations and functions of objects.
  - Quantifiers such as, All, Exists.
- Easier to construct a KB.
  - Need much smaller number of sentences to capture a domain.
- Inference algorithms for First order logic

# Propositional logic

- “All men are mortal”
- “Tom is a man”
- What can you infer?
  - Men  $\Rightarrow$  Mortal? ✓
  - Tom  $\Rightarrow$  Man? ✓
  - Tom  $\Rightarrow$  Mortal? ✓?



# Propositional logic vs. FOL

- Propositional logic:

- **P** stands for “All men are mortal”
- **Q** stands for “Tom is a man”
- What can you infer from P and Q?
  - Nothing!

R: Tom is mortal

$$\begin{array}{l} P \wedge Q \models P \\ \hline P \wedge Q \models Q \end{array}$$

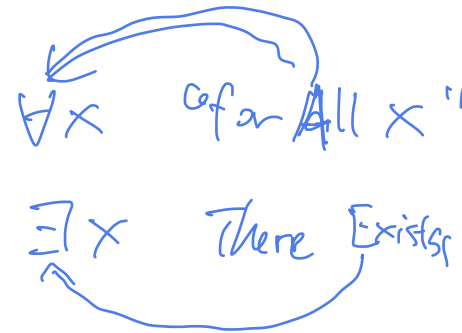
- First-order logic:

- $\forall x \text{Man}(x) \Rightarrow \text{Mortal}(x)$
- $\text{Man}(\text{Tom})$
- What can you infer from these?
  - Can infer  $\text{Mortal}(\text{Tom})$

A method of analysis or calculation using a special symbolic notation

# First-Order Logic (FOL)

- Also known as *First-Order Predicate Calculus*
  - Propositional logic is also known as *Propositional Calculus*
- An extension to propositional logic in which quantifiers can bind variables in sentences
  - Universal quantifier (  $\forall$  )
  - Existential quantifier (  $\exists$  )
  - Variables:  $x, y, z, a, joe, table...$



- Examples

- $\forall x$  Beautiful(x)
- $\exists x$  Beautiful(x)

# First-Order Logic (cont.)

- It is by far the most studied and best understood logic in use
- It does have limits, however
  - Quantifiers ( $\forall$  and  $\exists$ ) can only be applied to objects, not to functions or predicates
    - Cannot write  $\forall P \ P(\text{mom}) = \text{good}$
    - This is why it's called *first-order*
  - This limits its expressiveness
- Let's look at the syntax of first-order logic
  - I.e., what logical expressions can you legally construct?

# FOL Syntax

- Symbols
  - Object symbols (constants):  $P$ ,  $Q$ ,  $Fred$ ,  $Desk$ ,  $True$ ,  $False$ , ...
    - These refer to *things*
  - **Predicate** symbols:  $Heavy$ ,  $Smart$ ,  $Mother$ , ...
    - These are *true or false statements* about objects:  $Smart(rock)$
  - **Function** symbols:  $Cosine$ ,  $IQ$ ,  $MotherOf$ , ...
    - These return objects, exposing *relations*:  $IQ(rock)$
  - Variables:  $x$ ,  $y$ ,  $\lambda$ , ...
    - These represent unspecified objects
  - Logical connectives to construct complex sentences:  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ ,  $\Leftrightarrow$
  - Quantifiers:  $\forall$  (universal),  $\exists$  (existential)
  - Equality:  $=$
- Usually variables will be lower-case, other symbols capitalized

*Function on (x)*

# FOL Syntax (cont.)

- Terms
  - Logical expressions that refer to objects (evaluates to an object)
  - Can be constants, variables, functions
- Examples
  - $P$
  - 2001
  - *Richard*
  - $x$
  - $y$
  - $\text{BrotherOf}(\textit{Richard})$
  - $\text{Age}(\text{NephewOf}(x))$  [Why not  $\text{AgeOf}()$  ? (No reason...!)]

**Remember – syntax and semantics are different, and separate!!**

# FOL Syntax

- Note on predicates and functions: **typical** usage

- Beautiful( $y$ )  $\rightarrow$  “ $y$  is beautiful”

- is Mother( $x$ )  $\rightarrow$  “ $x$  is a Mother”

- is BrotherOf( $x, y$ )  $\rightarrow$  “ $x$  is a brother of  $y$ ”

- is NextTo( $x, y$ )  $\rightarrow$  “ $x$  is next to  $y$ ”

} Predicates

- BrotherOf( $x$ )  $\rightarrow$  “the brother of  $x$ ”

- NextTo( $y$ )  $\rightarrow$  “the thing next to  $y$ ”

- SquareRoot( $x$ )  $\rightarrow$  “the square root of  $x$ ”

} Functions


MotherOf( $x$ )

# FOL Sentences

- **Sentences** state facts
  - Just like in propositional logic...
- 3 types of sentences:
  - Atomic sentences (atoms)
  - Logical (complex) sentences
  - Quantified sentences –  $\forall$  (universal),  $\exists$  (existential)

# Sentences

## 1. Atomic sentence

- A predicate applied to some **terms** 

Constant, variable, or function – evaluates to an object

  - Brothers(Bill, FatherOf(John))
  - LessThan(3, 5)
- Equality – states that two terms refer to the same object
  - $x = \text{MotherOf}(y)$
  - Instructor(cs165a) = Wang
  - This is equivalent to the predicate:  $\text{Equal}(\text{Instructor}(\text{cs165a}), \text{Wang})$

## 2. Logical (complex) sentence – logical combination of other sentences

- $\neg \text{Brothers}(\text{Bill}, \text{HusbandOf}(\text{Sue}))$
- $\text{Above}(\text{Sky}, \text{Ground}) \Rightarrow \text{Below}(\text{Ground}, \text{Sky})$
- $\text{Brothers}(\text{Bill}, \text{John}) \Leftrightarrow \text{Brothers}(\text{John}, \text{Bill})$

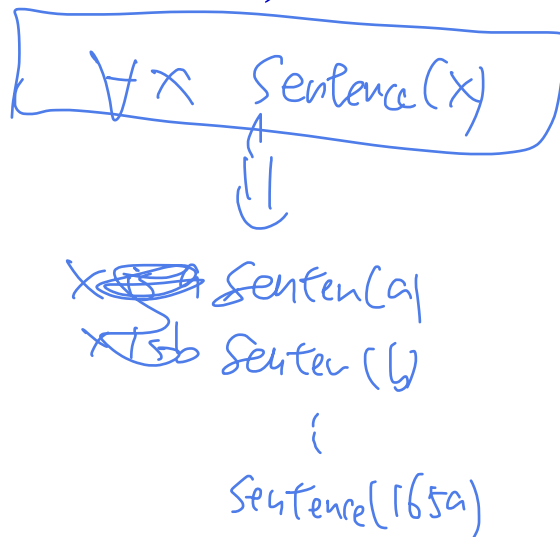
## 3. Quantified sentence – sentences with quantified variables

- $\forall x, y \text{ ParentOf}(x, y) \Rightarrow \text{ChildOf}(y, x)$
- $\exists x \text{ US-President}(x)$



# Universal Quantifier (“For all...”)

- $\forall$  <variables> <sentence>
  - $\forall x$  – “For all  $x$ ...”
  - $\forall x, y$  – “For all  $x$  and  $y$ ...”
- Examples
  - “Everything is beautiful”
    - $\forall x$  Beautiful( $x$ )
    - Equivalent to:  $\prod_i$  Beautiful( $x_i$ )
      - Beautiful(Joe)  $\wedge$  Beautiful(Mary)  $\wedge$  Beautiful(apple)  $\wedge$  Beautiful(dirt)  $\wedge$  Beautiful(death)  $\wedge$  ...
  - “All men are mortal”
    - $\forall x$  Man( $x$ )  $\Rightarrow$  Mortal( $x$ )
  - “Everyone in the class is smart”
    - $\forall x$  Enrolled( $x$ , cs165a)  $\Rightarrow$  Smart( $x$ )
  - What does this mean:
    - $\forall x$  Enrolled( $x$ , cs165a)  $\wedge$  Smart( $x$ )



# Expansion of universal quantifier

- $\forall x \text{ Enrolled}(x, \text{cs165a}) \Rightarrow \text{Smart}(x)$
- This is equivalent to
  - $\text{Enrolled}(\text{Tom}, \text{cs165a}) \Rightarrow \text{Smart}(\text{Tom}) \wedge$   
 $\text{Enrolled}(\text{Mary}, \text{cs165a}) \Rightarrow \text{Smart}(\text{Mary}) \wedge$   
 $\text{Enrolled}(\text{Chris}, \text{cs165a}) \Rightarrow \text{Smart}(\text{Chris}) \wedge$   
 $\text{Enrolled}(\text{chair}, \text{cs165a}) \Rightarrow \text{Smart}(\text{chair}) \wedge$   
 $\text{Enrolled}(\text{dirt}, \text{cs165a}) \Rightarrow \text{Smart}(\text{dirt}) \wedge$   
 $\text{Enrolled}(\text{surfboard}, \text{cs165a}) \Rightarrow \text{Smart}(\text{surfboard}) \wedge$   
 $\text{Enrolled}(\text{tooth}, \text{cs165a}) \Rightarrow \text{Smart}(\text{tooth}) \wedge$   
 $\text{Enrolled}(\text{Mars}, \text{cs165a}) \Rightarrow \text{Smart}(\text{Mars}) \wedge \dots$
  - Everything!
- So,  $\forall x \text{ Enrolled}(x, \text{cs165a}) \wedge \text{Smart}(x)$  is equivalent to
  - $\text{Enrolled}(\text{Tom}, \text{cs165a}) \wedge \text{Smart}(\text{Tom}) \wedge$   
 $\text{Enrolled}(\text{chair}, \text{cs165a}) \wedge \text{Smart}(\text{chair}) \wedge \dots$

# Existential Quantifier (“There exists...”)

- $\exists$  <variables> <sentence>
  - $\exists x$  – “There exists an  $x$  such that...”
  - $\exists x, y$  – “There exist  $x$  and  $y$  such that...”
- Examples
  - “Somebody likes me”
    - $\exists x$  Likes( $x, \text{Me}$ ) ??? ←
    - Equivalent to:  $\sum_i \text{Likes}(x_i, \text{Me})$ 
      - $\text{Likes}(\text{Joe}, \text{Me}) \vee \text{Likes}(\text{Mary}, \text{Me}) \vee \text{Likes}(\text{apple}, \text{Me}) \vee$   
 $\text{Likes}(\text{dirt}, \text{Me}) \vee \text{Likes}(\text{death}, \text{Me}) \vee \dots$
    - Really “Something likes me”
  - $\exists x$  Person( $x$ )  $\wedge$  Likes( $x, \text{Me}$ ) ←
  - $\exists x$  Enrolled( $x, \text{cs165a}$ )  $\wedge$  WillReceiveAnA+( $x$ )

# Scope of Quantifiers

- Scope – the portion of the {program, function, definition, sentence...} in which the object can be referred to by its simple name
- Parentheses can clarify the scope (make it explicit)
  - $\forall x (\exists y \langle sentence \rangle )$
- However, the scope of quantifiers is often implicit
  - $\forall w \forall x \exists y \exists z \langle sentence \rangle$   
is the same as
  - $\forall w (\forall x (\exists y (\exists z \langle sentence \rangle)))$
  - $\forall w \forall x \exists y \exists z \langle term-1 \rangle \wedge \langle term-2 \rangle$   
is the same as
  - $\forall w \forall x \exists y \exists z (\langle term-1 \rangle \wedge \langle term-2 \rangle)$

# Scope of Quantifiers (cont.)

- $\exists x \langle \text{sentence-1} \rangle \wedge \exists x \langle \text{sentence-2} \rangle$ 
  - $\exists x (\langle \text{sentence-1} \rangle) \wedge \exists x (\langle \text{sentence-2} \rangle)$
  - $\exists x (\langle \text{sentence-1} \rangle) \wedge \exists y (\langle \text{sentence-2-subst-y-for-x} \rangle)$
  - $\exists x \text{ Rich}(x) \wedge \text{ Beautiful}(x)$ 
    - “Someone is both rich and beautiful”
  - $\exists x \text{ Rich}(x) \wedge \exists x \text{ Beautiful}(x)$ 
    - “Someone is rich and someone is beautiful”
    - Same as  $\exists x \text{ Rich}(x) \wedge \exists y \text{ Beautiful}(y)$
- How about
  - $\exists x (\text{ Rich}(x) \wedge \exists x (\text{ Beautiful}(x)) )$
  - The same as  $\exists x \text{ Rich}(x) \wedge \exists x \text{ Beautiful}(x)$

Equivalent



Same as in scope of variables in programming (C/C++, Java, etc.)

# Order, nesting of Quantifiers

- Implied nesting:
  - $\forall x \forall y \langle \text{sentence} \rangle$  is the same as  $\forall x (\forall y \langle \text{sentence} \rangle)$
  - $\exists x \forall y \langle \text{sentence} \rangle$  is the same as  $\exists x (\forall y \langle \text{sentence} \rangle)$
- $\forall x \forall y \langle \text{sentence} \rangle$  is the same as  $\forall y \forall x \langle \text{sentence} \rangle$ 
  - Also,  $\forall x, y \langle \text{sentence} \rangle$
- $\exists x \exists y \langle \text{sentence} \rangle$  is the same as  $\exists y \exists x \langle \text{sentence} \rangle$ 
  - Also,  $\exists x, y \langle \text{sentence} \rangle$
- $\exists x \forall y \langle \text{sentence} \rangle$  is **not** the same as  $\forall y \exists x \langle \text{sentence} \rangle$ 
  - Try  $\exists x \forall y \text{ Loves}(x, y)$  and  $\forall y \exists x \text{ Loves}(x, y)$

# Example of quantifier order

- $\exists x \forall y \text{ Loves}(x, y)$ 
  - $\exists x [ \forall y \text{ Loves}(x, y) ]$
  - $\exists x [ \text{Loves}(x, \text{Fred}) \wedge \text{Loves}(x, \text{Mary}) \wedge \text{Loves}(x, \text{Chris}) \wedge \dots ]$
  - “There is at least one person who loves everybody”
    - Assuming the domain consists of only people
- $\forall y \exists x \text{ Loves}(x, y)$ 
  - $\forall y [ \exists x \text{ Loves}(x, y) ]$
  - $\forall y [ \text{Loves}(\text{Joe}, y) \vee \text{Loves}(\text{Sue}, y) \vee \text{Loves}(\text{Kim}, y) \vee \dots ]$
  - “Everybody is loved by at least one person”

# Logical equivalences about $\forall$ and $\exists$

- $\forall$  can be expressed using  $\exists$ 
  - $\forall x$  Statement-about- $x$  ... is equivalent to ...
  - $\neg \exists x \neg$ Statement-about- $x$
  - Example:  $\forall x$  Likes( $x$ , IceCream)
    - $\neg \exists x \neg$ Likes( $x$ , IceCream)
- $\exists$  can be expressed using  $\forall$ 
  - $\exists x$  Statement-about- $x$  ... is equivalent to ...
  - $\neg \forall x \neg$ Statement-about- $x$
  - Example:  $\exists x$  Likes( $x$ , Spinach)
    - $\neg \forall x \neg$  Likes( $x$ , Spinach)



# Examples of FOL

- Brothers are siblings
  - $\forall x, y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y)$
- Sibling is transitive
  - $\forall x, y, z \text{ Sibling}(x, y) \wedge \text{Sibling}(y, z) \Rightarrow \text{Sibling}(x, z)$
- One's mother is one's sibling's mother
  - $\forall x, y, z \text{ Mother}(x, y) \wedge \text{Sibling}(y, z) \Rightarrow \text{Mother}(x, z)$
- A first cousin is a child of a parent's sibling
  - $\forall x, y \text{ FirstCousin}(x, y) \Leftrightarrow$   
 $\exists v, w \text{ Parent}(v, x) \wedge \text{Sibling}(v, w) \wedge \text{Parent}(w, y)$

# Implication and Equivalence

- Note the difference between  $\Rightarrow$  and  $\Leftrightarrow$ 
  - Implication / conditional ( $\Rightarrow$ )
    - $A \Rightarrow B$  : “A implies B”, “If A then B”
  - Equivalence / biconditional ( $\Leftrightarrow$ )
    - $A \Leftrightarrow B$  : “A is equivalent to B”
    - Same as  $(A \Rightarrow B) \wedge (B \Rightarrow A)$  : “A if and only if B”, “A iff B”

- For “Sisters are siblings”, which one?

$$\begin{aligned} \forall x, y \text{ Sister}(x, y) \Leftrightarrow \text{Sibling}(x, y) &\equiv \forall x, y (\text{Sister}(x, y) \Rightarrow \text{Sibh}(x, y)) \\ \forall x, y \text{ Sister}(x, y) \Rightarrow \text{Sibling}(x, y) &\wedge (\text{Sibh}(x, y) \Rightarrow \text{Sister}(x, y)) \end{aligned}$$

# Where we are...

- Basics of logic: Propositional logic
- More general logic representation: First-order logic
- Now, let's see how to use FOL to do logical inference
  - I.e., to reason about the world

# Reminder

- Term
  - Constant, variable, function( )
- Atomic sentence
  - Predicate( ),  $\text{term}_1 = \text{term}_2$
- Literal
  - An atomic sentence or a negated atomic sentence
- Sentence
  - Atomic sentence, sentences with quantifiers and/or connectives

# Simple example of inference in FOL

constants

Bob is a buffalo

Pat is a pig

Buffaloes outrun pigs

Buffalo(Bob)

Pig(Pat)

$\forall x,y$   
 $\underline{\text{Buffalo}}(x) \wedge \underline{\text{Pig}}(y) \Rightarrow \underline{\text{Outrun}}(x,y)$

**KB<sub>0</sub>**

Does Bob outrun Pat?

KB entails **Outrun(Bob, Pat)?**

**S**

KB<sub>0</sub> |— Buffalo(Bob) ∧ Pig(Pat)

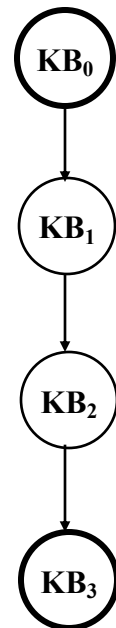
(And-Introduction)

KB<sub>1</sub> |— Buffalo(Bob) ∧ Pig(Pat) ⇒ Outrun(Bob, Pat)

(Universal Instantiation) [coming soon]

KB<sub>2</sub> |— Outrun(Bob, Pat)

(Modus Ponens)



# Using FOL to express knowledge

- One can express the knowledge of a particular domain in first-order logic
- Example: The “kinship domain”
  - **Objects:** people
  - **Properties:** gender, family relationships
  - **Unary predicates:** Male, Female
  - **Binary predicates:** Parent, Sibling, Brother, Sister, Son, Daughter, Father, Mother, Uncle, Aunt, Grandparent, Grandfather, Grandmother, Husband, Wife, Spouse, Brother-in-law, Stepmother, etc....
  - **Functions:** MotherOf, FatherOf...
- Note: There is usually (always?) more than one way to specify knowledge

# Kinship domain

- Write down what we know (what we want to be in the KB)
  - One's mother is one's female parent
    - $\forall m, c \text{ Mother}(m, c) \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c)$
    - $\forall m, c \text{ TheMotherOf}(c) = m \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c)$
  - One's husband is one's male spouse
    - $\forall w, h \text{ Husband}(h, w) \Leftrightarrow \text{Male}(h) \wedge \text{Spouse}(h, w)$
  - One is either male or female
    - $\forall x \text{ Male}(x) \Leftrightarrow \neg \text{Female}(x)$
  - Parent-child relationship
    - $\forall p, c \text{ Parent}(p, c) \Leftrightarrow \text{Child}(c, p)$
  - Grandparent-grandchild relationship
    - $\forall g, c \text{ Grandparent}(g, c) \Leftrightarrow \exists p \text{ Parent}(g, p) \wedge \text{Parent}(p, c)$
  - Etc...
- Now we can reason about family relationships. (How?)

# Kinship domain (cont.)

## Assertions (“Add this sentence to the KB”)

TELL( KB,  $\forall m, c \text{ Mother}(c) = m \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c)$  )

TELL( KB,  $\forall w, h \text{ Husband}(h, w) \Leftrightarrow \text{Male}(h) \wedge \text{Spouse}(h, w)$  )

TELL( KB,  $\forall x \text{ Male}(x) \Leftrightarrow \neg \text{Female}(x)$  )

TELL( KB,  $\text{Female}(\text{Mary}) \wedge \text{Parent}(\text{Mary}, \text{Frank}) \wedge \text{Parent}(\text{Frank}, \text{Ann})$  )

- Note:  $\text{TELL}(\text{KB}, S1 \wedge S2) \equiv \text{TELL}(\text{KB}, S1) \text{ and } \text{TELL}(\text{KB}, S2)$   
(because of and-elimination and and-introduction)

## Queries (“Does the KB entail this sentence?”)

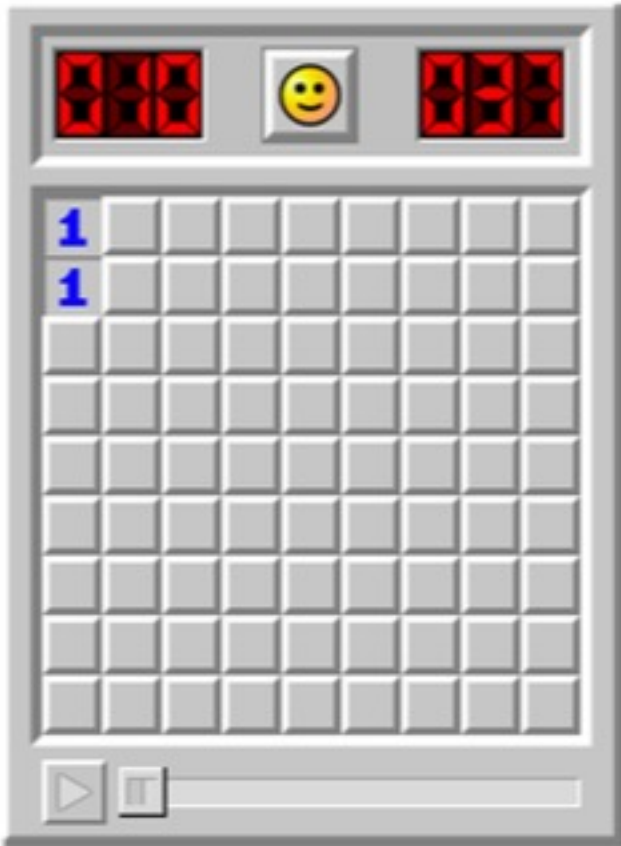
ASK( KB,  $\text{Grandparent}(\text{Mary}, \text{Ann})$  )  $\rightarrow$  True

ASK( KB,  $\exists x \text{ Child}(x, \text{Frank})$  )  $\rightarrow$  True

- But a better answer would be  $\rightarrow \{ x / \text{Ann} \}$
- This returns a **substitution** or **binding**



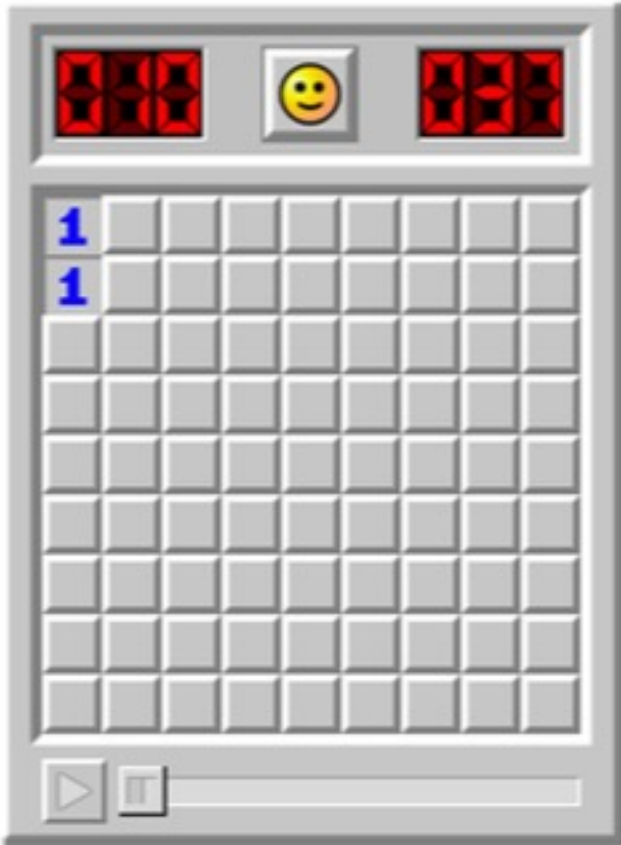
# (3 min discussion) Using first order logic to represent Minesweeper



## Rules of the game:

- In Minesweeper, mines (that resemble naval mines in the classic theme) are scattered throughout a board, which is divided into cells. Cells have two states: unopened, opened. An unopened cell is blank and clickable, while an opened cell is exposed.
- A player left-clicks a cell to open it. **If a player opens a mined cell, the game ends**, as there is only one life per game. Otherwise, **the opened cell displays either a number, indicating the number of mines diagonally and/or adjacent to it**, or a blank tile (or "0"), and all adjacent non-mined cells will automatically be opened.
- **The player wins when all non-mine cells are opened.**

# (3 min discussion) Using first order logic to represent Minesweeper



- If a player opens a mined cell, the game ends.

$$\forall c, g \quad \underbrace{\text{isCell}(c)} \wedge \underbrace{\text{isMine}(c)} \wedge \underbrace{\text{isPlayer}(g)} \wedge \underbrace{\text{isOpen}(c)} \Rightarrow \text{GameOver}(g)$$

- The opened cell displays either a number, indicating the number of mines diagonally and / or adjacent to it, or a blank.

$$\forall c, g \quad \text{isOpen}(c) \wedge \neg \text{GameOver}(g)$$

$$\Rightarrow \left( \begin{array}{l} \text{isBlank}(c) \\ \vee \end{array} \right) \left( \text{isNumber}(c) \right)$$

$$\text{display}(c) =$$

$$\text{Sum}(\text{neighbors}(x) \mid \text{hasMine}(c))$$

$$\neg \text{NotZero}(c)$$

$$\text{NotZero}(c)$$

$$\text{Neighbors}(c) = \left( \begin{array}{l} \text{isf}(N(c), W(c), E(c), S(c)) \\ \text{isf}(NW(c), NN(c), SE(c), SW(c)) \end{array} \right)$$

*get number(c, neighbors(c))*

# Recap: Implementing ASK: Inference

- We want a sound and complete inference algorithm so that we can produce (or confirm) *entailed* sentences from the KB

$$\text{KB} \models \alpha \qquad \text{KB} \vdash \alpha$$

- The **resolution** rule, along with a complete search algorithm, provides a complete inference algorithm to confirm or refute a sentence  $\alpha$  in propositional logic (Sec. 7.5)
  - Based on *proof by contradiction* (refutation)
- Refutation: To prove that the KB entails P, assume  $\neg P$  and show a contradiction:

$$(\text{KB} \wedge \neg P \Rightarrow \text{False}) \equiv (\text{KB} \Rightarrow P)$$

Prove this!

# Inference in First-Order Logic

- Inference rules for propositional logic:
  - Modus ponens, and-elimination, and-introduction, or-introduction, resolution, etc.
  - These are valid for FOL also
- But since these don't deal with quantifiers and variables, we need new rules, especially those that allow for substitution (binding) of variables to objects
  - These are called *lifted* inference rules

# Substitution and variable binding

- Notation for substitution:
  - SUBST( **Binding list**, **Sentence** )
    - Binding list:  $\{ var / \text{ground term}, var / \text{ground term}, \dots \}$
    - “ground term” = term with no variables
  - SUBST(  $\{var/gterm\}$ , Func( $var$ ) ) = Func(gterm)
    - SUBST( $\theta$ , p)
  - Examples:
    - SUBST(  $\{x/Mary\}$ , FatherOf( $x$ ) ) = FatherOf(Mary)
    - SUBST(  $\{x/Joe, y/Lisa\}$ , Siblings( $x,y$ ) ) = Siblings(Joe, Lisa)

# Three new inference rules using $SUBST(\theta, p)$

- Universal Instantiation

$$\frac{\forall v \quad \alpha}{SUBST(\{v / g\}, \alpha)}$$

$g$  – ground term

- Existential Instantiation

$$\frac{\exists v \quad \alpha}{SUBST(\{v / k\}, \alpha)}$$

$k$  – constant that does not appear elsewhere in the knowledge base

- Existential Introduction

$$\frac{\alpha}{\exists v \quad SUBST(\{g / v\}, \alpha)}$$

$v$  – variable not in  $\alpha$   
 $g$  – ground term in  $\alpha$

## To Add to These Rules

- ◇ **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- ◇ **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

- ◇ **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

- ◇ **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$$

- ◇ **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

$$\frac{\neg\neg\alpha}{\alpha}$$

- ◇ **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

$$\frac{\alpha \vee \beta, \quad \neg\beta}{\alpha}$$

- ◇ **Resolution**: (This is the most difficult. Because  $\beta$  cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

# Universal Instantiation – examples

$$\frac{\forall v \quad \alpha}{\text{SUBST}(\{v / g\}, \alpha)} \quad g - \text{ground term}$$

- $\forall x \text{ Sleepy}(x)$ 
  - $\text{SUBST}(\{x/\text{Joe}\}, \alpha)$ 
    - $\text{Sleepy}(\text{Joe})$
- $\forall x \text{ Mother}(x) \Rightarrow \text{Female}(x)$ 
  - $\text{SUBST}(\{x/\text{Mary}\}, \alpha)$ 
    - $\text{Mother}(\text{Mary}) \Rightarrow \text{Female}(\text{Mary})$
  - $\text{SUBST}(\{x/\text{Dad}\}, \alpha)$ 
    - $\text{Mother}(\text{Dad}) \Rightarrow \text{Female}(\text{Dad})$
- $\forall x, y \text{ Buffalo}(x) \wedge \text{Pig}(y) \Rightarrow \text{Outrun}(x, y)$ 
  - $\text{SUBST}(\{x/\text{Bob}\}, \alpha)$ 
    - $\forall y \text{ Buffalo}(\text{Bob}) \wedge \text{Pig}(y) \Rightarrow \text{Outrun}(\text{Bob}, y)$



# Existential Instantiation – examples

$$\frac{\exists v \quad \alpha}{\text{SUBST}(\{v/k\}, \alpha)}$$

$k$  – constant that does not appear elsewhere in the knowledge base

- $\exists x \text{ BestAction}(x)$ 
  - $\text{SUBST}(\{x/B\_A\}, \alpha)$ 
    - $\text{BestAction}(B\_A)$ 
      - “ $B\_A$ ” is a constant; it is not in our universe of actions
- $\exists y \text{ Likes}(y, \text{Broccoli})$ 
  - $\text{SUBST}(\{y/Bush\}, \alpha)$ 
    - $\text{Likes}(Bush, \text{Broccoli})$ 
      - “ $Bush$ ” is a constant; it is not in our universe of people

# Existential Introduction – examples

$$\frac{\alpha}{\exists v \text{ SUBST}(\{g / v\}, \alpha)}$$

$v$  – variable not in  $\alpha$   
 $g$  – ground term in  $\alpha$

- Likes(Jim, Broccoli)
  - SUBST({Jim/ $\underline{x}$ },  $\alpha$ )
    - $\exists x$  Likes( $x$ , Broccoli)
- $\forall x$  Likes( $x$ , Broccoli)  $\Rightarrow$  Healthy( $x$ )
  - SUBST({Broccoli/ $y$ },  $\alpha$ )
    - $\exists y \forall x$  Likes( $x$ ,  $y$ )  $\Rightarrow$  Healthy( $x$ )

# We can formulate the logical inference problem as a search problem.

- Formulate a **search process**:
  - Initial state
    - KB
  - Operators
    - Inference rules
  - Goal test
    - KB contains S
- What is a node?
  - KB + new sentences (generated by applying the inference rules)
  - In other words, the new state of the KB
- What kind of search to use?
  - I.e., which node to expand next?
- How to apply inference rules?  $\alpha \Rightarrow \beta$ 
  - Need to match the premise pattern  $\alpha$

**Question: What's our goal here?**

# Historical AI figure in Logical Reasoning

- Built a calculating machine that could add and subtract (which Pascal's couldn't)
- But his dream was much grander – to reduce human reasoning to a kind of calculation and to ultimately build a machine capable of carrying out such calculations
- Co-inventor of the calculus



Gottfried Leibniz (1646-1716)

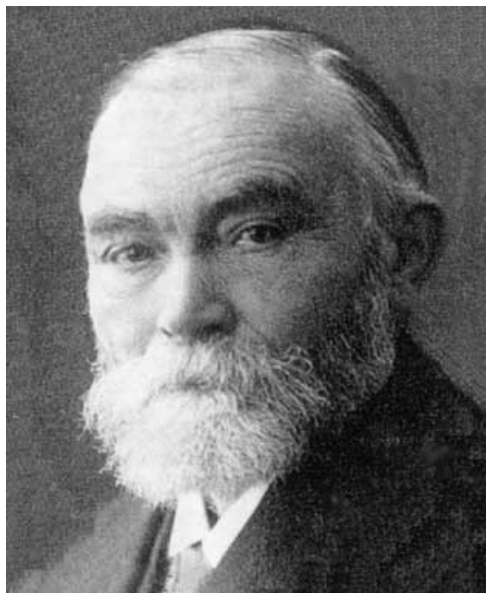
*“For it is unworthy of excellent men to lose hours like slaves in the labor of calculation which could safely be relegated to anyone else if the machine were used.”*

# George Boole (1815-1864) British



- More than 100 years later, he didn't know about Leibniz, but proceeded to bring to life part of Leibniz' dream
  - His insight: Logical relationships are expressible as a kind of algebra
    - Letters represent classes (rather than numbers)
    - So logic can be viewed as a form of mathematics
  - Published *The Laws of Thought*
- 
- He extended Aristotle's simple syllogisms to a broader range of reasoning
    - Syllogism: Premise\_1, Premise\_2  $\rightarrow$  Conclusion
    - His logic: Propositional logic

# Gottlob Frege (1848-1925) German



- He provided the first fully developed system of logic that encompassed all of the deductive reasoning in ordinary mathematics.
- He intended for logic to be the *foundation* of mathematics – all of mathematics could be based on, and derived from, logic
- In 1879 he published *Begriffsschrift*, subtitled “A formula language, modeled upon that of arithmetic, for pure thought”
  - This can be considered the ancestor of all current computer programming languages
  - Made the distinction between *syntax* and *semantics* critical
- He invented what we today call predicate calculus (or first-order logic)

# Inference algorithms in first order logic will not be covered in the final. (FOL will be!)

- However, it is a powerful tool.
  - Expert systems (since 1970s)
  - Large scale industry deployment.
- It is however fragile and rely on the correct / error-free representation of the world in black and white
  - This limits its use in cases when the evidence is collected stochastically and imprecisely by people's opinions in large scale.
- Somewhat superseded by machine learning on many problems, but:
  - Research on logic agent is coming back.
  - Add knowledge and reasoning to ML-based solution
  - After all, ML are just reflex agents usually.