Artificial Intelligence

CS 165A

Dec 3, 2020

Instructor: Prof. Yu-Xiang Wang

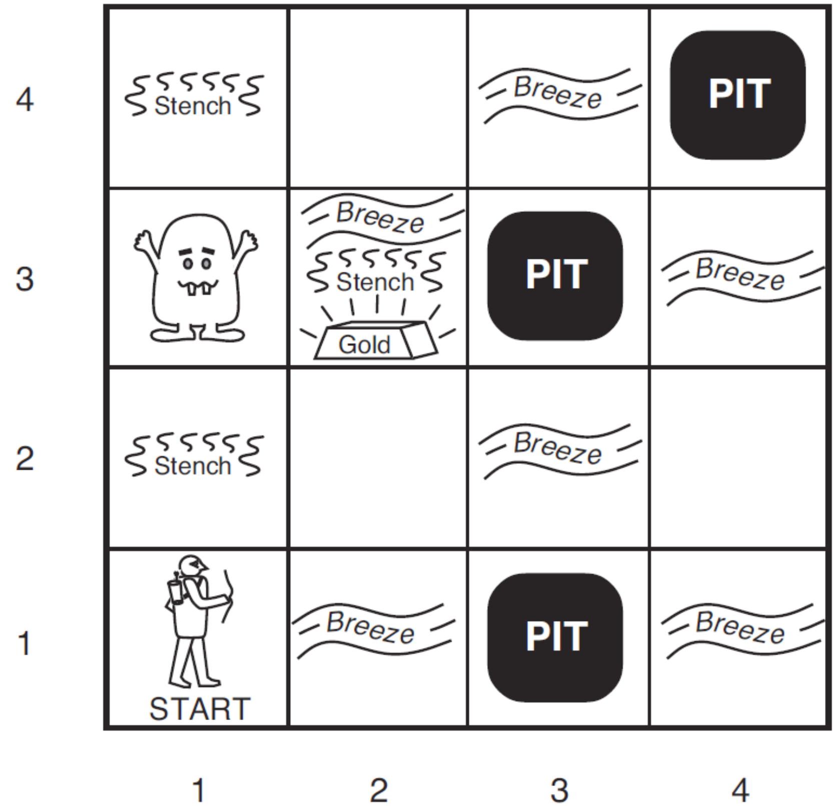
Today

→ Propositional Logic (Inference)

→ 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

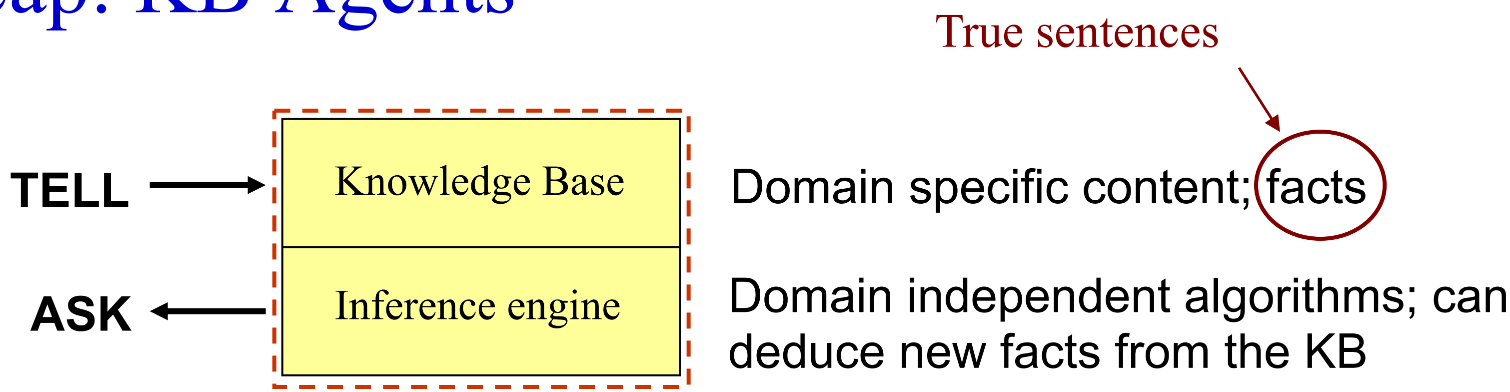


http://thiagodnf.github.io/wumpus-world-simulator/

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

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

Recap: KB Agents

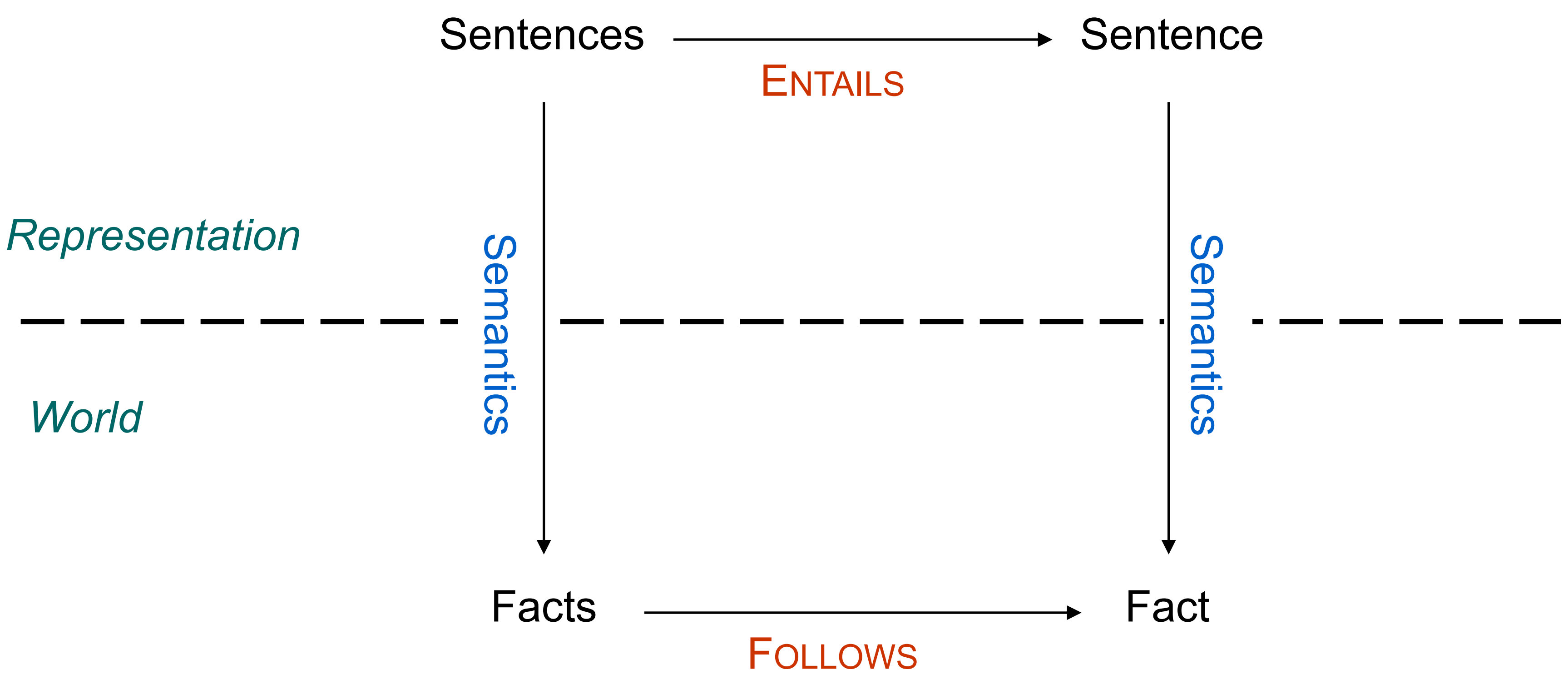


- Need a formal logic system to work
- Need a data structure to represent known facts
- Need an algorithm to answer ASK questions

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

- 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
  - ( ) , ¬ (not), ∧ (and), ∨ (or), ⇒ (implies), ⇔ (equivalent)

- **Semantics**:
  - Assigning values to the variables. Each row is a “model”.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

Recap: Logical Inference in Propositional Logic

- A simple algorithm for checking: KB entails α
  - Enumerate M(KB)
  - Check that it is contained in M(α)

- This inference algorithm is **sound** and **complete**.

- Are there other ways to do logical inference?

- Are they sound / complete?

This lecture

- “One rule that rules it all” for proposition logic

- First order logic

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: $((P \lor H) \land \neg H) \vdash P$ for variables P and H

- Alternate notation for inference rule $\alpha \vdash \beta$:

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

"If we know α, then we can conclude β"

(where α and β are propositional logic sentences)

Inference

- We're particularly interested in

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

- Inference steps

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

So we need a mechanism to do this!

Inference rules that can be applied to sentences in our KB

Important Inference Rules for Propositional Logic

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

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

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

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

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

$$\frac{\alpha_1, \ \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

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

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

- **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.)

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

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

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

Example (using inference rules)

KB

$Q \rightarrow \neg S$

$P \vee \neg W$

R

P

$P \rightarrow Q$

What can we infer ($\vdash$) if we add this sentence with no inference rules?

$$P \rightarrow Q$$

Nothing

What can we infer ($\vdash$) if we then add this inference procedure:

$$(\alpha \rightarrow \beta) \wedge \alpha \;\vdash\; \beta$$

$$\frac{(\alpha \rightarrow \beta),\ \alpha}{\beta}$$

Q and $\neg S$

Resolution Rule: one rule for all inferences

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

Propositional calculus resolution

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

$$\frac{\neg p \Rightarrow q, \quad q \Rightarrow r}{\neg p \Rightarrow r} \qquad \text{or} \qquad \frac{a \Rightarrow b, \quad b \Rightarrow c}{a \Rightarrow c}$$

Resolution is really the "chaining" of implications.

Soundness:

Show that $(\alpha \vee \beta) \wedge (\neg\beta \vee \gamma) \Rightarrow (\alpha \vee \gamma)$

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha\vee\beta \wedge \neg\beta\vee\gamma$	$\alpha \vee \gamma$
0	0	0	0	1	0	0
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

This is always true for all propositions α, β, and γ, so we can make it an inference rule

Soundness:

Show that $(\neg\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \gamma) \Rightarrow (\neg\alpha \Rightarrow \gamma)$

α	β	γ	$\neg\alpha \Rightarrow \beta$	$\beta \Rightarrow \gamma$	$\neg\alpha \Rightarrow \beta \wedge \beta \Rightarrow \gamma$	$\neg\alpha \Rightarrow \gamma$
0	0	0	0	1	0	0
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

This is always true for all propositions α, β, and γ, so we can make it an inference rule

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 $P \wedge Q \wedge R$

Complexity of reasoning

- Validity
  - NP-complete
- Satisfiability
  - NP-complete
- α 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

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

Remainder of the 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 => Mortal?
  - Tom => Man?
  - Tom => 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!

- First-order logic:
  - $\forall x\ \mathbf{Man}(x) \Rightarrow \mathbf{Mortal}(x)$
  - $\mathbf{Man}(\mathit{Tom})$
  - What can you infer from these?
    - Can infer $\mathbf{Mortal}(\mathit{Tom})$

First-Order Logic (FOL)

A method of analysis or calculation using a special symbolic notation

- 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 (∀ and ∃) can only be applied to objects, not to functions or predicates
    - Cannot write **$\forall P\ P(\mathrm{mom}) = \mathrm{good}$**
    - 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, λ, …
    - 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

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
  - BrotherOf(*Richard*)
  - Age(NephewOf(x))   [Why not 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"
  - Mother(x) $\rightarrow$ "x is a Mother"
  - BrotherOf(x, y) $\rightarrow$ "x is a brother of y"
  - 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

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 $\boxed{\text{terms}}$

     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 = MotherOf(y)
     - Instructor(cs165a) = Wang
     - This is equivalent to the predicate: Equal(Instructor(cs165a), Wang)
2. **Logical (complex) sentence** – logical combination of other sentences
   - ¬Brothers(Bill, HusbandOf(Sue))
   - Above(Sky, Ground) ⇒ Below(Ground, Sky)
   - Brothers(Bill, John) ⇔ Brothers(John, Bill)
3. **Quantified sentence** – sentences with quantified variables
   - ∀ x,y ParentOf(x, y) ⇒ ChildOf(y, x)
   - ∃ x US-President(x)

Universal Quanitifer ("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

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

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, Me) ???
    - Equivalent to: $\sum_i$ Likes(x_i, Me)
      - Likes(Joe, Me) $\vee$ Likes(Mary, Me) $\vee$ Likes(apple, Me) $\vee$ Likes(dirt, Me) $\vee$ Likes(death, Me) $\vee$ …
    - Really “Something likes me”
  - $\exists x$ Person(x) $\wedge$ Likes(x, Me)
  - $\exists x$ Enrolled(x, 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)
  - $\forall\, x\ \ \textbf{(}\, \exists\, y\ \ \langle sentence \rangle\ )$
- However, the scope of quantifiers is often implicit
  - $\forall\, w\, \forall\, x\, \exists\, y\, \exists\, z\ \ \langle sentence \rangle$

    is the same as
  - $\forall\, w\, (\forall\, x\, (\exists\, y\, (\exists\, z\ \ \langle sentence \rangle)\,)\,)$

  - $\forall\, w\, \forall\, x\, \exists\, y\, \exists\, z\ \ \langle term\text{-}1 \rangle \wedge \langle term\text{-}2 \rangle$

    is the same as
  - $\forall\, w\, \forall\, x\, \exists\, y\, \exists\, z\ \ \textbf{(}\langle term\text{-}1 \rangle \wedge \langle term\text{-}2 \rangle\textbf{)}$

Scope of Quantifiers (cont.)

Equivalent

- $\exists$ *x* *<sentence-1>* $\wedge$ $\exists$ *x* *<sentence-2>*
  - $\exists$ *x* **(***<sentence-1>***)** $\wedge$ $\exists$ *x* **(***<sentence-2>***)**
  - $\exists$ *x* **(***<sentence-1>***)** $\wedge$ $\exists$ *y* **(***<sentence-2-subst-y-for-x>***)**
  - $\exists x$ Rich(x) $\wedge$ Beautiful(x)
    - "Someone is both rich and beautiful"
  - $\exists x$ Rich(x) $\wedge$ $\exists x$ Beautiful(x)
    - "Someone is rich and someone is beautiful"
    - Same as $\exists x$ Rich(x) $\wedge$ $\exists y$ Beautiful(y)
- How about
  - $\exists x$ (Rich(x) $\wedge$ $\exists x$ (Beautiful(x)))
  - The same as $\exists x$ Rich(x) $\wedge$ $\exists x$ Beautiful(x)

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

Order, nesting of Quantifiers

- Implied nesting:
  - $\forall x \, \forall y$ <*sentence*> is the same as $\forall x$ ($\forall y$ <*sentence*>)
  - $\exists x \, \forall y$ <*sentence*> is the same as $\exists x$ ($\forall y$ <*sentence*>)

- $\forall x \, \forall y$ <*sentence*> is the same as $\forall y \, \forall x$ <*sentence*>
  - Also, $\forall x, y$ <*sentence*>

- $\exists x \, \exists y$ <*sentence*> is the same as $\exists y \, \exists x$ <*sentence*>
  - Also, $\exists x, y$ <*sentence*>

- $\exists x \, \forall y$ <*sentence*> is **not** the same as $\forall y \, \exists x$ <*sentence*>
  - Try $\exists x \, \forall y$ Loves(x, y) and $\forall y \, \exists x$ Loves(x, y)

Example of quantifier order

- $\exists x \, \forall y$ Loves(x, y)
  - $\exists x$ [$\forall y$ Loves(x, y)]
  - $\exists x$ [Loves(x, Fred) $\wedge$ Loves(x, Mary) $\wedge$ Loves(x, Chris) $\wedge \ldots$]
  - "There is at least one person who loves everybody"
    - Assuming the domain consists of only people

- $\forall y \, \exists x$ Loves(x, y)
  - $\forall y$ [$\exists x$ Loves(x, y)]
  - $\forall y$ [Loves(Joe, y) $\vee$ Loves(Sue, y) $\vee$ Loves(Kim, y) $\vee \ldots$]
  - "Everybody is loved by at least one person"

Logical equivalences about ∀ and ∃

- ∀ can be expressed using ∃
  - $\forall\, \boldsymbol{x}$ Statement-about-x … is equivalent to …
  - $\neg\exists\, \boldsymbol{x}$ ¬Statement-about-x
  - Example: $\forall\, \boldsymbol{x}$ Likes(x, IceCream)
    - $\neg\exists\, \boldsymbol{x}$ ¬Likes(x, IceCream)

- ∃ can be expressed using ∀
  - $\exists\, \boldsymbol{x}$ Statement-about-x … is equivalent to …
  - $\neg\forall\, \boldsymbol{x}$ ¬Statement-about-x
  - Example: $\exists\, \boldsymbol{x}$ Likes(x, Spinach)
    - $\neg\forall\, \boldsymbol{x}$ ¬ Likes(x, Spinach)

Examples of FOL

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

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?

  $\forall\ x, y\ \ \text{Sister}(x, y) \Leftrightarrow \text{Sibling}(x, y)$

  $\forall\ x, y\ \ \text{Sister}(x, y) \Rightarrow \text{Sibling}(x, y)$

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( ), $term_1 = term_2$
- Literal
  - An atomic sentence or a negated atomic sentence
- Sentence
  - Atomic sentence, sentences with quantifiers and/or connectives

Simple example of inference in FOL

Bob is a buffalo

Pat is a pig

Buffaloes outrun pigs

KB_0

Buffalo(Bob)

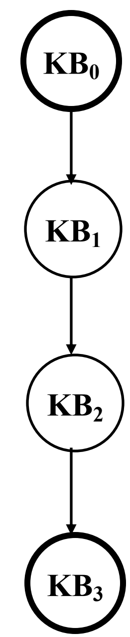
Pig(Pat)

Buffalo(x) $\land$ Pig(y) $\Rightarrow$ Outrun(x,y)

Does Bob outrun Pat?

KB entails $\boxed{\text{Outrun(Bob, Pat)?}}$

S



KB_0 |– Buffalo(Bob) $\land$ Pig(Pat)

(And-Introduction)

KB_1 |– Buffalo(Bob) $\land$ Pig(Pat) $\Rightarrow$ Outrun(Bob, Pat)

(Universal Instantiation) [coming soon]

KB_2 |– 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
    - $\forall\, m, c\ \ \text{Mother}(m, c) \Leftrightarrow \text{Female}(m) \land \text{Parent}(m, c)$
    - $\forall\, m, c\ \ \text{TheMotherOf}(c) = \text{m} \Leftrightarrow \text{Female}(m) \land \text{Parent}(m, c)$
  - One's husband is one's male spouse
    - $\forall\, w, h\ \ \text{Husband}(h, w) \Leftrightarrow \text{Male}(h) \land \text{Spouse}(h, w)$
  - 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
    - $\forall\, g, c\ \ \text{Grandparent}(g, c) \Leftrightarrow \exists\, p\ \ \text{Parent}(g, p) \land \text{Parent}(p, c)$
  - Etc…
- Now we can reason about family relationships. (How?)

Kinship domain (cont.)

Assertions ("Add this sentence to the KB")

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

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

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

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

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

Queries ("Does the KB entail this sentence?")

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

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

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

Implementing ASK: Inference

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

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

- The **resolution** rule, along with a complete search algorithm, provides a <u>complete inference algorithm</u> to confirm or refute a sentence α in propositional logic (Sec. 7.5)
  - Based on *proof by contradiction* (refutation)

- Refutation: To prove that the KB entails P, assume ¬P and show a contradiction:

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

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* / ground term, *var* / ground term, … }
    - "ground term" = term with no variables

  - SUBST({*var*/gterm}, Func(*var*)) = Func(gterm)
    - SUBST(θ, 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($\boldsymbol{\theta}$, $\boldsymbol{p}$)

- 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

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

v – variable not in α

g – ground term in α

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.)

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

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

$$\frac{\alpha_1, \ \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

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

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

◇ **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 β 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

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

g – ground term

- $\forall x$ Sleepy(x)
  - SUBST({x/Joe}, α)
    - Sleepy(Joe)
- $\forall x$ Mother(x) ⇒ Female(x)
  - SUBST({x/Mary}, α)
    - Mother(Mary) ⇒ Female(Mary)
  - SUBST({x/Dad}, α)
    - Mother(Dad) ⇒ Female(Dad)
- $\forall x, y$ Buffalo(x) ∧ Pig(y) ⇒ Outrun(x,y)
  - SUBST({x/Bob}, α)
    - $\forall y$ Buffalo(Bob) ∧ Pig(y) ⇒ Outrun(Bob,y)

Existential Instantiation – examples

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

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

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

Existential Introduction – examples

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

v – variable not in α
g – ground term in α

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

Question: What's our goal here?

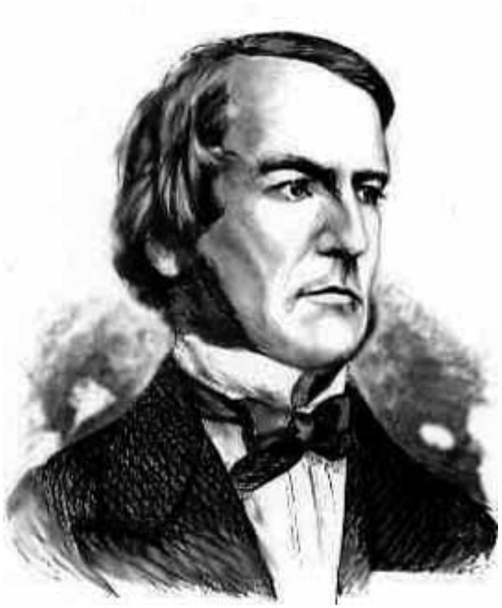
Historical AI figure in Logical Reasoning

- 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."

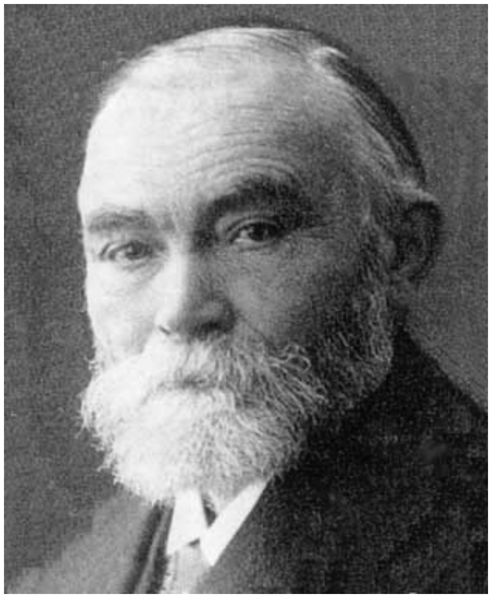
George Boole (1815-1864) British



- 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 → 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.