Artificial Intelligence

CS 165A

Mar 5, 2020

Instructor: Prof. Yu-Xiang Wang

Today

→ Proposition Logic

→ First order logic

Today

- Propositional Logic
- Inference Rules of Propositional Logic
- First order logic

(some slides borrowed from Pat Virtue and Stephanie Rosenthal)

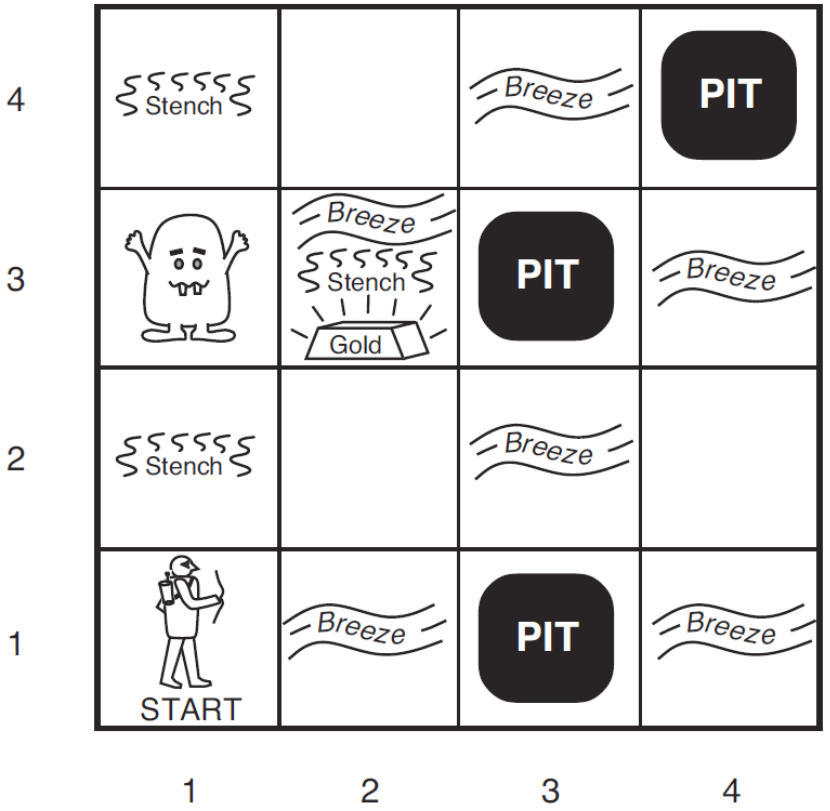
Recap: Last lecture

- RL algorithms
  - Q-learning with linear function approximation
  - Policy class. Learning with policy gradient updates

- Logic
  - Wumpus world
  - Knowledge-based agents
  - Knowledge base: TELL, ASK operations

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

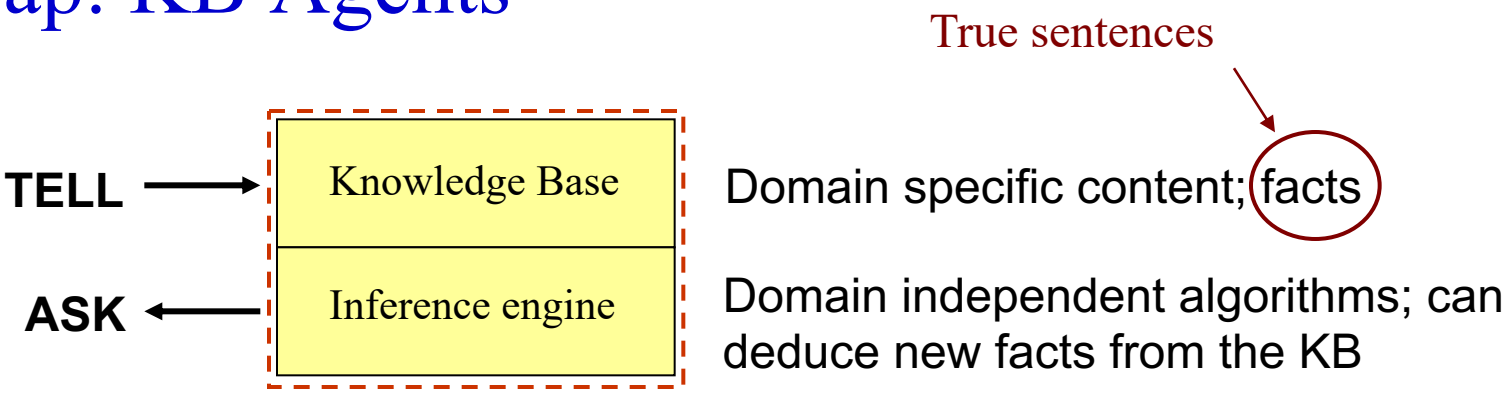


http://thiagodnf.github.io/wumpus-world-simulator/

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

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

Recap: KB Agents



function KB-AGENT(*percept*) **returns** an *action*
  static: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

Recap: Fundamental Concepts of Logical Language Representation and Concepts

- **Syntax**
  - Grammar / rules to follow for form a well-defined sentence
  - $x + y = 4$ is a valid sentence in "arithmetics", x4y+= is not.
- **Semantics**
  - The meaning of sentences. Truth of each sentence w.r.t. each possible world.
  - Possible World 1: x=3, y=1. Possible World 2: x=1, y=1.
- **Model** (Possible world, a.k.a. "interpretations" in some text)
  - Each model is an assignment of values to variables.
  - Each model fixes the truth value of all sentences.
  - If sentence α is true in Model m, we say: Model m satisfies sentence α, or m is a model of α, or $m \in M(\alpha)$,

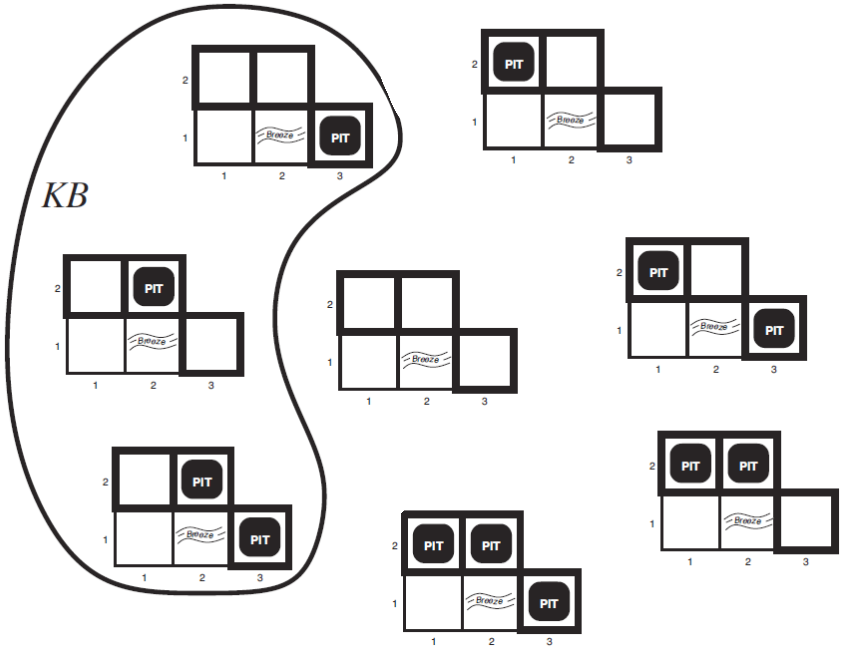
Fundamental Concepts of Logical Language Representation and Concepts

- **Entailment**
  - Sentence β logically follows from Sentence α
  - Denoted by $\alpha \models \beta$
  - α entails β *if an only if* $M(\alpha) \subseteq M(\beta)$
  - If all models of α are also models of β

- **Logical Inference**
  - The procedure of checking whether a sentence is entailed by a given a knowledge base
  - Simplest algorithm for logical inference: **Simple Model checking**
  - Enumerate all models in $\mathrm{M}(\alpha)$, check whether they are in $\mathrm{M}(\beta)$.
  - We will come back to logical inference!

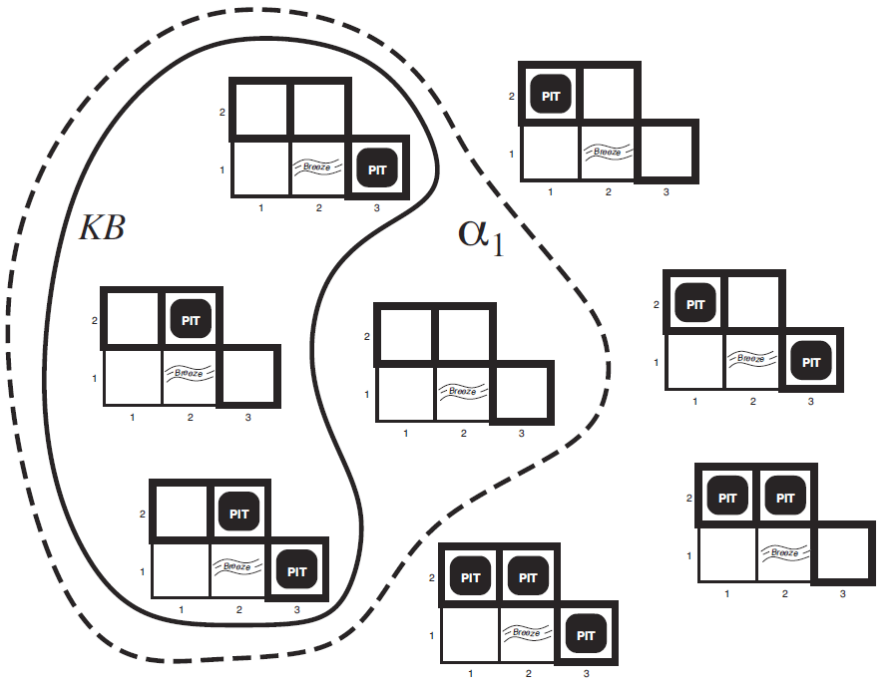
Example: Wumpus World

- Possible Models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]



Example: Wumpus World

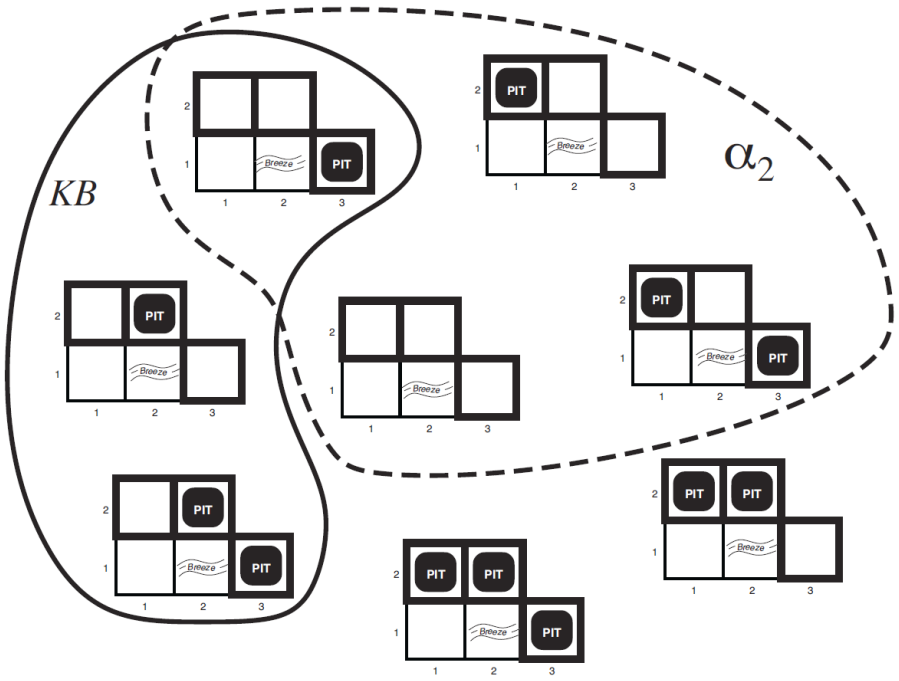
- Possible Models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_1:
  - No pit in [1,2]



*Question: Does KB entails α_1?

Example: Wumpus World

- Possible Models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_2:
  - No pit in [2,2]



*Question: Does KB entails α_2?

Inference and Entailment

- Given a set of (true) sentences, ***logical inference*** generates new sentences
  - Sentence α follows from sentences $\{ \beta_i \}$
  - Sentences $\{ \beta_i \}$ ***entail*** sentence α
  - The classic example is *modus ponens*: $\mathbf{P \Rightarrow Q}$ and **P** entail what?

- A knowledge base (KB) *entails* sentences α

$$KB \models \alpha$$

- An **inference procedure *i*** can derive α from KB

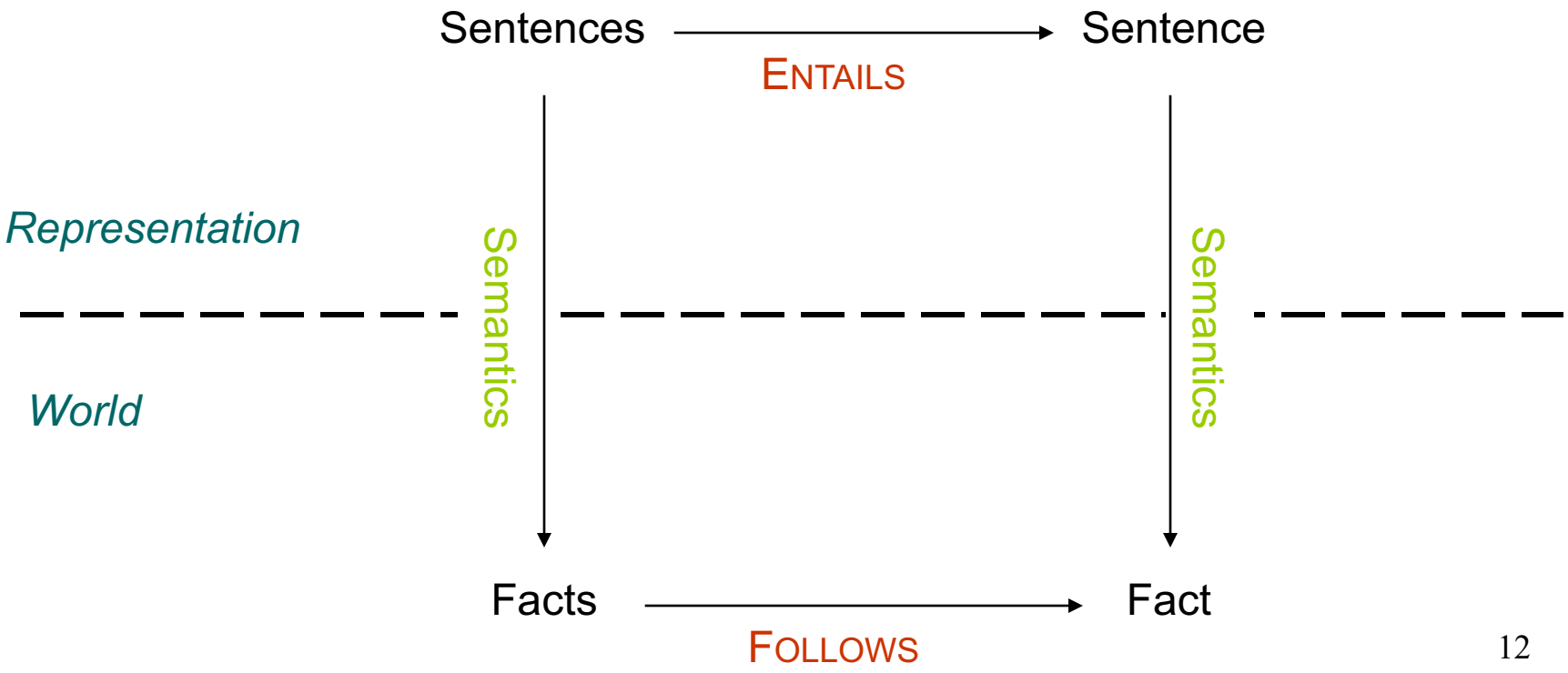
$$KB \vdash_i \alpha$$

Inference and Entailment (cont.)

Inference *(n.):*

a. The act or process of deriving logical conclusions from premises known or assumed to be true.

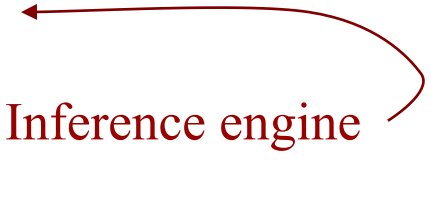
b. The act of reasoning from factual knowledge or evidence.



Inference engine

- An inference engine is a program that applies inference rules to knowledge
  - Goal: To infer new (and useful) knowledge

- Separation of
  - Knowledge
  - Rules
  - Control



Which rules should we apply when?

Inference procedures

- An inference procedure
  - Generates new sentences α that purport to be entailed by the knowledge base

    …or...
  - Reports whether or not a sentence α is entailed by the knowledge base
- Not every inference procedure can derive all sentences that are entailed by the KB
- A *sound* or *truth-preserving* inference procedure generates only entailed sentences
- Inference derives valid conclusions *independent of the semantics* (i.e., independent of the models)

Inference procedures (cont.)

- Soundness of an inference procedure
  - *i* is **sound** if whenever KB $\vdash_i \alpha$, it is also true that KB $\models \alpha$
  - I.e., the procedure only generates entailed sentences
- Completeness of an inference procedure
  - *i* is **complete** if whenever KB $\models \alpha$, it is also true that KB $\vdash_i \alpha$
  - I.e., the procedure can find a proof for any sentence that is entailed
- The derivation of a sentence by a sound inference procedure is called a ***proof***
  - Hence, the ***proof theory*** of a logical language specifies the reasoning steps that are sound

So far, we have defined the jargon and notation of a generic logic language

- Syntax
- Semantics
- Models
- Entailment
- Inference
- Soundness and completeness

- Make sure you know / understand these definitions!

Logics (Specify Syntax, Semantics, Inference procedures and so on…)

- We will soon define a logic which is expressive enough to say most things of interest, and for which there exists a sound and complete inference procedure
  - I.e., the procedure will be able to derive anything that is derivable from the KB
  - This is ***first-order logic***
  - But first, let's review ***propositional logic,*** which you've already learned from CS40

Propositional (Boolean) Logic

- Symbols represent **propositions** (statements of fact, sentences)
  - *P* means "San Francisco is the capital of California"
  - *Q* means "It is raining in Seattle"

- Sentences are generated by combining proposition symbols with Boolean (logical) connectives

Propositional Logic

- Syntax
  - *True, false,* propositional symbols
  - ( ) , ¬ (not), ∧ (and), ∨ (or), ⇒ (implies), ⇔ (equivalent)

- Examples of sentences in propositional logic

P_1, P_2, etc. (propositions)	$S_1 \Rightarrow S_2$
(S_1)	$S_1 \Leftrightarrow S_2$
$\neg S_1$	*true*
$S_1 \wedge S_2$	$P_1 \wedge true \wedge \neg(P_2 \Rightarrow false)$
$S_1 \vee S_2$	$P \wedge Q \Leftrightarrow Q \wedge P$

Entailment and equivalence

- What is the meaning of α entails β, or $\alpha \models \beta$
- $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
- Examples of logical equivalences
  - Commutativity of $\wedge$, $\vee$
  - Associativity of $\wedge$, $\vee$
  - Distributive laws
    - A and (B or C)=(A and B) or (A and C)
  - De Morgan's laws
    - NOT (P OR Q) = (NOT P) AND (NOT Q)
    - NOT (P AND Q) = (NOT P) OR (NOT Q)
  - $\boldsymbol{P \Rightarrow Q} \equiv \boldsymbol{\neg P \vee Q}$

Precedence of operators (logical connectives)

- Levels of precedence, evaluating left to right
  1. $\neg$ (NOT)
  2. $\wedge$ (AND, conjunction)
  3. $\vee$ (OR, disjunction)
  4. $\Rightarrow$ (implies, conditional)
  5. $\Leftrightarrow$ (equivalence, biconditional)

- $P \wedge \neg Q \Rightarrow R$
  - $(P \wedge (\neg Q)) \Rightarrow R$
- $P \vee Q \wedge R$
  - $P \vee (Q \wedge R)$
- $P \Leftrightarrow Q \wedge R \Rightarrow S$
  - $P \Leftrightarrow ((Q \wedge R) \Rightarrow S)$

Satisfiability and Validity

- Is this true: $(P \wedge Q)$?
  - It depends on the values of P and Q
  - This is a **satisfiable** sentence – there are some **models** for which it is true and others for which it is false
- Is this true: $(P \wedge \neg P)$?
  - No, it is never true
  - This is an **unsatisfiable** sentence (self-contradictory) – there is no **models** for which it is true
- Is this true: $(((P \vee Q) \wedge \neg Q) \Rightarrow P)$?
  - Yes, independent of the values of P and Q
  - This is a **valid** sentence – it is true under all possible **models** (a.k.a. a **tautology**)

Things to know!

- What is a sound inference procedure?
  - The procedure only generates entailed sentences
- What is a complete inference procedure?
  - The procedure can find a proof for any sentence that is entailed
- What is a satisfiable sentence?
  - There are some models for which it is true
- What is an unsatisfiable sentence?
  - There is no model for which it is true
- What is a valid sentence?
  - It is true under all possible models

Propositional (Boolean) Logic (cont.)

- Semantics
  - Defined by clearly interpreted symbols and straightforward application of truth tables
  - Rules for evaluating truth: Boolean algebra
  - Simple method: truth tables

Propositions / Variables

Sentences

Models

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*

2^N rows (models) for N propositions

Knowledge are constraints that eliminate rows

- Adding a sentence to our knowledge base constrains the
- number of possible models:

- KB: Nothing

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Knowledge are constraints that eliminates rows

- Adding a sentence to our knowledge base constrains the
- number of possible models:

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

- KB: Nothing
- KB: $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$

Knowledge are constraints that eliminates rows

- Adding a sentence to our knowledge base constrains the
- number of possible models:

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

- KB: Nothing
- KB: $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$
- KB: R, $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$

Sherlock Entailment

- "When you have eliminated the impossible, whatever remains, however improbable, must be the truth" – *Sherlock Holmes via Sir Arthur Conan Doyle*

- Knowledge base and inference allow us to remove impossible models, helping us to see what is true in all of the remaining models

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?

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 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.)

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

- **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.)

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

- **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.)

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

Example

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

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

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	α ∨ β	¬β ∨ γ	α∨β ∧ ¬β∨γ	α ∨ γ
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

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 (∀)
  - Existential quantifier (∃)
  - Variables: *x, y, z, a, joe, table*…

- Examples
  - **∀*x* Beautiful(*x*)**
  - **∃*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 **$\forall P \;\; P(\text{mom}) = \text{good}$**
    - 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, λ, …*
    - These represent unspecified objects
  - Logical connectives to construct complex sentences: ¬, ∧, ∨, ⇒, ⇔
  - Quantifiers: ∀ (universal), ∃ (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) → "y is beautiful"
  - Mother(x) → "x is a Mother"
  - BrotherOf(x, y) → "x is a brother of y"
  - NextTo(x, y) → "x is next to y"

  } Predicates

  - BrotherOf(x) → "the brother of x"
  - NextTo(y) → "the thing next to y"
  - SquareRoot(x) → "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 – ∀ (universal), ∃ (existential)

Sentences

1. **Atomic sentence**
   - A predicate applied to some terms
     - 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)

Constant, variable, or function – evaluates to an object

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
   - $\forall x,y$ ParentOf(x, y) ⇒ ChildOf(y, x)
   - $\exists x$ US-President(x)

Universal Quanitifer ("For all…")

- ∀ **<variables> <sentence>**
  - ∀ *x* – *"For all x…"*
  - ∀ *x, y* – *"For all x and y…"*
- Examples
  - "Everything is beautiful"
    - $\forall x$ Beautiful(x)
    - Equivalent to: $\prod_i$ Beautiful(x_i)
      - Beautiful(Joe) ∧ Beautiful(Mary) ∧ Beautiful(apple) ∧ Beautiful(dirt) ∧ Beautiful(death) ∧ **…**
  - "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…")

- **∃ <variables> <sentence>**
  - ∃ *x* – *"There exists an x such that…"*
  - ∃ *x, y* – *"There exist x and y such that…"*
- Examples
  - "Somebody likes me"
    - ∃ *x* Likes(*x*, Me) ???
    - Equivalent to: $\sum_i$ Likes(x_i, Me)
      - Likes(Joe, Me) ∨ Likes(Mary, Me) ∨ Likes(apple, Me) ∨ Likes(dirt, Me) ∨ Likes(death, Me) ∨ **…**
    - Really "Something likes me"
  - ∃ *x* Person(*x*) ∧ Likes(*x*, Me)
  - ∃ *x* Enrolled(*x*, cs165a) ∧ 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)
  - $\forall x$ **(** $\exists y$ <*sentence*>)
- However, the scope of quantifiers is often implicit
  - $\forall w \forall x \exists y \exists z$ <*sentence*>

    is the same as
  - $\forall w (\forall x (\exists y (\exists z$ <*sentence*>)))

  - $\forall w \forall x \exists y \exists z$ <*term-1*> $\wedge$ <*term-2*>

    is the same as
  - $\forall w \forall x \exists y \exists z$ **(**<*term-1*> $\wedge$ <*term-2*>**)**

Scope of Quantifiers (cont.)

- $\exists x$ <*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)

Equivalent

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) $\wedge$ …]
  - "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) $\vee$ …]
  - "Everybody is loved by at least one person"

Logical equivalences about ∀ and ∃

- ∀ can be expressed using ∃
  - ∀ ***x*** Statement-about-x … is equivalent to …
  - ¬∃ ***x*** ¬Statement-about-x
  - Example: ∀ ***x*** Likes(*x*, IceCream)
    - ¬∃ ***x*** ¬Likes(*x*, IceCream)

- ∃ can be expressed using ∀
  - ∃ ***x*** Statement-about-x … is equivalent to …
  - ¬∀ ***x*** ¬Statement-about-x
  - Example: ∃ ***x*** Likes(*x*, Spinach)
    - ¬∀ ***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
  - $\forall x, y, z \ \text{Sibling}(x, y) \wedge \text{Sibling}(y, z) \Rightarrow \text{Sibling}(x, z)$
- 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?

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

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

Where we are…

- Basics of logic: Propositional logic
- More general logic representation: First-order logic
- Next lecture:
  - More on FOL
  - FOL inference: i.e., to reason about the world