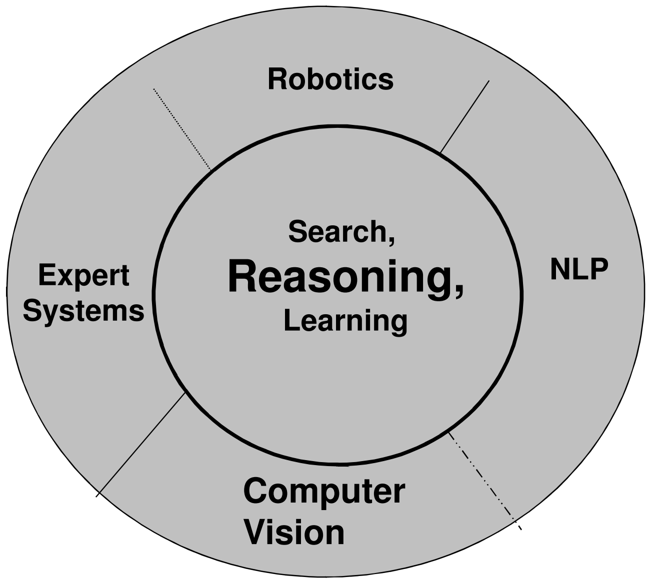
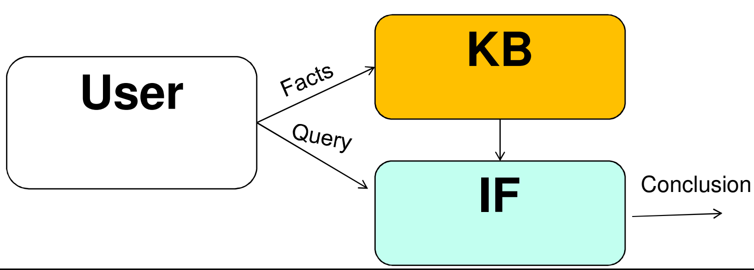
Reasoning



First-order logic Chapter 8-Russel
Representation and Reasoning

- In order to determine appropriate actions to take, an intelligent system needs to represent information about the world and draw conclusions based on general world knowledge and specific facts.
- Knowledge is represented by sentences in some language stored in a **knowledge base (KB).**
- A system draws conclusions from the KB to answer questions, take actions using **Inference Engine (IF)**.



Knowledge Representation

- Logics are formal languages for representing information such that conclusions can be drawn
- **Syntax:** defines the sentences in the language
- **Semantics:** define the "meaning" of sentences: i.e., define truth of a sentence in a world
- E.g., the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x2+y > \{\}$ is not a sentence syntax
  - $x+2 \geq y$ is true in a world where $x = 7, y = 1$
  - $x+2 \geq y$ is false in a world where $x = 0, y = 6$

semantics

Inference

- Logical Inference (deduction) derives new sentences in the language from existing ones,.

  Socrates is a man.

  All men are mortal.

  Socrates is mortal.
- Proper inference should only derive sound conclusions

Examples of Types of Logics

Language	What exist	Degree of belief of an Agent
Propositional Logic	Facts	{o,1} T or F
First Order Logic	Facts, Objects, Relations	{o,1} T or F
Temporal Logic	Facts, Objects, Relations, Time	{o,1} T or F
Probability Theory	Facts	Chances of belief [0,1]
Fuzzy Logic	Degree of truth about Facts	Degree of belief [0,1]

Propositional calculus & First-order logic

- Propositional logic assumes world contains facts.
- First-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, …
  - Relations: red, round, prime,…
  - Functions: fatherof, friend, in,…
- Propositional calculus
  $A \wedge B \Rightarrow C$
- First-order predicate calculus
  $(\forall x)(\exists y)$ Mother(y,x)

Syntax of PC Chapter 7-Russel

- Connectives: ¬, ∧, ∨, ⇒
- Propositional symbols, e.g., P, Q, R, …
  - *True*, *False*
  - **Syntax of PC**
- sentence → atomic sentence | complex sentence
- atomic sentence → Propositional symbol, *True*, *False*
- Complex sentence → ¬sentence
  | (sentence ∧ sentence)
  | (sentence ∨ sentence)
  | (sentence ⇒ sentence)
  - **Rules of Inference:**
  - Ex: Modus ponens

Sentence in PC

A **sentence (also called a formula or well-formed formula or wff)** is defined as**:**

- A symbol (S, P, …etc)
- If S is a sentence, then ¬S is a sentence, where "¬" is the "not" logical operator
- If S and T are sentences, then (S v T), (S ^ T), (S => T), and (S <=> T) are sentences, where the four logical connectives correspond to "or," "and," "implies," and "if and only if," respectively

Example

P means "It is hot"
Q means "It is humid"
R means "It is raining“

Examples of PL sentences:
(P ^ Q) => R (here meaning "If it is hot and humid, then it is raining")
Q => P (here meaning "If it is humid, then it is hot")
¬ Q (here meaning "It is not humid.")

Semantics of PC

A	B	¬ A	A ∧ B	A ∨ B	A ⇒ B
True	*True*	*False*	*True*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*

Truth Tables

- **Truth tables can be used to compute the truth value of any wff.**
- Can be used to find the truth of $((P \rightarrow R) \rightarrow Q) \vee \neg S$
- **Given n features there are 2^n different worlds, different interpretations**.
- **Interpretation: any assignment of true and false to atoms**
- **An interpretation satisfies a wff if the wff is assigned true under the interpretation**
- **A model:** An interpretation is a model of a wff if the wff is satisfied in that interpretation.
- **Satisfiability of a wff can be determined by the truth-table**

  $P \wedge (\neg P)$

  $(P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg Q)$
- **Wff is unsatisfiable or inconsistent it has no models**

Semantics of PC

Validity and Inference

• **interpretation of the sentence:** Given the truth values of all of the constituent symbols in a sentence, that sentence can be "evaluated" to determine its truth value (True or False).

• A **model** is an interpretation (i.e., an assignment of truth values to symbols) of a set of sentences such that each sentence is True. A model is just a formal mathematical structure for the world.

• A **valid sentence (also called a tautology)** is a sentence that is True under ***all interpretations. Hence, no matter what the world is actually like or what the semantics is, the sentence is True.***
For example "It's raining or it's not raining."
Remark: Validity can be checked by the truth table

Semantics of PC

Validity and Inference

- An **inconsistent sentence (also called unsatisfiable or a contradiction)** is a sentence that is False under ***all interpretations.***
For example, "It's raining and it's not raining."

- Sentence P **entails sentence Q, written P |= Q,** means that whenever P is True, so is Q. In other words, all models of P are also models of Q

Satisfiability

- A sentence is **satisfiable** if it is true under some interpretation (i.e. it has a model), otherwise the sentence is **unsatisfiable.**
- A sentence is **valid** if and only if its negation is unsatisfiable.
- Therefore, algorithms for either validity or satisfiability checking are useful for logical inference.
- If there are *n propositional symbols in a sentence, then we* must check 2^n *rows for validity*
- **Satisfiability is** NP-complete, i.e. there is no polynomial-time algorithm to solve.
- Yet, many problems can be solved very quickly.

Rules of Inference

- A sequence of inference rule applications that leads to a desired conclusion is called a **logical proof.**
- A |- B , denotes that B can be derived by some inference procedure from the set of sentences A.
- Inference rules can be verified by the truth-table
- The truth table method of inference is **complete for PL**
- Then used to construct **sound proofs.**
- Finding a proof is simply a **search problem** with the inference rules as operators and the conclusion as the goal

Rules of Inference

•**Modus Ponens:** $\{\alpha \Rightarrow \beta, \alpha\} \vdash \beta$

•**And Elimination:** $\{\alpha \wedge \beta\} \vdash \alpha$; $\{\alpha \wedge \beta\} \vdash \beta$

•**Double negation Elimination:** $\{\neg\neg\alpha\} \vdash \alpha$

•**Implication Elimination** $\{\alpha \Rightarrow \beta\} \vdash \neg\alpha \vee \beta$

•**Unit resolution:** $\{\alpha \vee \beta, \neg\beta\} \vdash \alpha$

•**Resolution:** $\{\alpha \vee \beta, \neg\beta \vee \gamma\} \vdash \alpha \vee \gamma$

Famous logical equivalences

- *$(a \vee b) \equiv (b \vee a)$* *commutatitvity*
- *$(a \wedge b) \equiv (b \wedge a)$* *commutatitvity*
- *$((a \wedge b) \wedge c) \equiv (a \wedge (b \wedge c))$* *associativity*
- *$((a \vee b) \vee c) \equiv (a \vee (b \vee c))$* *associativity*
- *$\neg(\neg(a)) \equiv a$* *double-negation elimination*
- *$(a \Rightarrow b) \equiv (\neg(b) \Rightarrow \neg(a))$* *contraposition*
- *$(a \Rightarrow b) \equiv (\neg(a) \vee b)$* *implication elimination*
- *$\neg(a \wedge b) \equiv (\neg(a) \vee \neg(b))$* *De Morgan*
- *$\neg(a \vee b) \equiv (\neg(a) \wedge \neg(b))$* *De Morgan*
- *$(a \wedge (b \vee c)) \equiv ((a \wedge b) \vee (a \wedge c))$* *distributitivity*
- *$(a \vee (b \wedge c)) \equiv ((a \vee b) \wedge (a \vee c))$* *distributitivity*

Rules of Inference

- **Producing an additional wffs from a set of wffs**
- **From alpha infer beta**
  - $\alpha \vdash \beta$
  - $w_1 \wedge w_2 \vdash w_2$
- **Sound inference rule:**
  - The conclusion is true whenever the premises are true.
- **Examples**
  - ***Modus ponens***: { A and A → B |-- B} is sound, resolution is sound.

Pros and cons of propositional logic

- ✓ Propositional logic is declarative:

  pieces of syntax correspond to facts
- ✓ Propositional logic is compositional:

  meaning of A ^ B is derived from meaning of A and B
- ✓ Meaning in propositional logic is context-independent
- (unlike natural language, where meaning depends on context)
- ➢ Propositional logic has very limited expressive power
- (unlike natural language)

Propositional logic is a weak language

- Hard to identify "individuals." Ex. Mary, 3
- Can't directly talk about properties of individuals or relations between individuals. Ex. "Bill is tall"
- Generalizations, patterns, regularities can't easily be represented. Ex. all triangles have 3 sides
- **First-Order Logic** (abbreviated FOL or FOPC) is expressive enough to concisely represent this kind of situation.
  - FOL adds relations, variables, and quantifiers, e.g.,
    - "Every elephant is gray": $\forall$ x (elephant(x) $\rightarrow$ gray(x))
    - "There is a white elephant": $\exists$ x (elephant(x) ^ white(x))

First-order logic

- First-order logic (FOL) models the world in terms of
  - Objects, which are things with individual identities
  - Properties of objects that distinguish them from other objects
  - Relations that hold among sets of objects
  - Functions, which are a subset of relations where there is only one "value" for any given "input"

Ex:Objects: Students, lectures, companies, cars ...

  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

FOL Syntax

- **Variable symbols**
  - E.g., x, y, John
- **Connectives**: $\neg$, $\wedge$, $\vee$, $\Rightarrow$
  - **Quantifiers**
  - Universal $\forall x$
  - Existential $\exists x$

Syntax of First-order logic

Sentence → *Atomicsentence*

| (*Sentence Connective Sentence*)

| *Quantifier Variable,*. . . *Sentence*

| >*Sentence*

AtomicSentence → *Predicate(Term,* . . .)

| (*Term* = *Term*

Term → *Function(Term,* . . .)

I *Constant*

| *Variable*

Connective → ¬, ∧, ∨, ⇒

Quantifier → ∀, ∃

Constant → *A* (XI (*John* 1 . . .

Variable → *a* | *x* | s | . . .

Predicate → *Before...*

Function → *Mother* | *...*

Atomic Sentences

- Propositions are represented by a **predicate applied to a** tuple of terms. A predicate represents a property or relation between terms that can be true or false:
- Brother(John, Fred), Left-of(Square1, Square2), GreaterThan(plus(1,1), plus(0,1))
- Sentences in logic **<u>state facts</u>** that are true or false.
- In FOL properties and n-ary relations do express that:

  LargerThan(2,3) is false. Brother(Mary,Pete) is false.
- Note: **Functions do not state facts** and <u>form no sentence:</u> Brother(Pete) refers to the object John (his brother) and is neither true nor false.
- Brother(Pete,Brother(Pete)) is True.

↑ Binary relation ↑ Function

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies:

constant symbols → objects
predicate symbols → relations
function symbols → functional relations

- An atomic sentence *predicate*($term_1$,...,$term_n$) is true iff the objects referred to by $term_1$,...,$term_n$ are in the relation referred to by *predicate*

Entailment

- Entailment means that one thing follows from another:

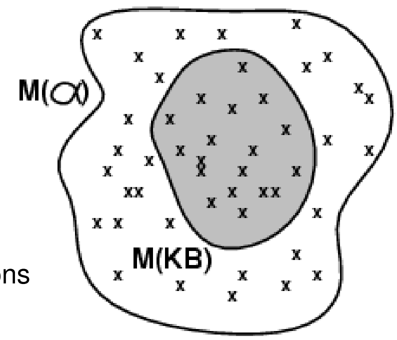
$$KB \models \alpha$$

Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

- E.g., the KB containing "the Greens won" and "the Reds won" entails "Either the Greens or the reds won"
- E.g., x+y = 4 entails 4 = x+y
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
- entailment: necessary truth of one sentence given another

Models

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say *m* is a model of a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- Then KB ⊨ α iff *M(KB)* ⊆ *M(*α)
  - E.g. *KB* = Greens won and Reds won
    α = Greens won



- Think of KB and α as collections of constraints and of models m as possible states. M(KB) are the solutions to KB and M(α) the solutions to α. Then, KB ⊨ α when all solutions to KB are also solutions to α.

Nested Quantifiers

- Combinations of universal and existential quantification are possible:

$$\forall x \forall y\, Father(x,y) \equiv \forall y \forall x\, Father(x,y)$$
$$\exists x \exists y\, Father(x,y) \equiv \exists y \exists x\, Father(x,y)$$
$$\forall x \exists y\, Father(x,y) \neq \exists y \forall x\, Father(x,y)$$
$$\exists x \forall y\, Father(x,y) \neq \forall y \exists x\, Father(x,y)$$
$$x, y \in \{All\ \ people\}$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
- Ex:

$\forall x\ At(x,CU) \wedge Smart(x)$

means "Everyone is at CU and everyone is smart"

Yet to say Everyone at CU is smart

$\forall x\ At(x,CU) \Rightarrow Smart(x)$

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists \boldsymbol{x} At(x,CU) \Rightarrow Smart(x)$$

is true if there is anyone who is smart not at CU.

Yet to say: there exists someone in CU that is smart

$$\exists \boldsymbol{x} At(x,CU) \wedge Smart(x)$$

Properties of quantifiers

∀x ∀y is the same as ∀y ∀x
∃x ∃y is the same as ∃y ∃x

∃x ∀y is not the same as ∀y ∃x
∃x ∀y Loves(x,y)
- "There is a person who loves everyone in the world"

∀y ∃x Loves(x,y)
- "Everyone in the world is loved by at least one person"

- Quantifier duality: each can be expressed using the other

Exp.	Negation
∀x Likes(x,IceCream)	∃x ¬Likes(x,IceCream)
∃x Likes(x,Broccoli)	∀x ¬Likes(x,Broccoli)

Equality

Equality:

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

FOPC can include equality as a primitive predicate or require it to be as identity relation

Equal(x,y) or x=y

Examples:

to say "that Mary is taking two courses", you need to insure that x,y are different

∃x ∃y (takes(Mary,x) ^ takes (Mary,y) ^ ~ (x=y))

To say "Everyone has exactly one father"

∀x ∃y father(y,x) ^ ∀z father(z,x) ➔ y=z

Higher Order Logic

- FOPC is called first order because it allows quantifiers to rang only over objects (terms).

  $\forall$x, $\forall$y [*x=y or x>y or y>x*]

- **Second-Order Logic** allows quantifiers to range over predicates and functions as well

  $\forall$f, $\forall$g [*f=g* <=> ($\forall$x f(x)=g(x))]

- **Third-Order Logic** allows quantifiers to range over predicates of predicates,
- .. etc

Examples of FOPC

- Brothers are siblings

  $\forall$x, $\forall$y ***Brother(x,y)*** => ***Sibling(x,y)***

- One's mother is one's female parent

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

- "Sibling" is symmetric

  $\forall$x, $\forall$y ***Sibling(x,y)*** $\Leftrightarrow$ ***Sibling(y,x)***