# Foundations of Artificial Intelligence

## Knowledge-Based Systems

**CS472 – Fall 2007**
**Thorsten Joachims**

## History of AI
## 1943 – 1969 The Beginnings

1943 **McCulloch and Pitts show networks of neurons can compute and learn any function**

1950 **Shannon and Turing wrote chess programs**

1951 **Minsky and Edmonds build the first neural network computer (SNARC)**

1956 **Dartmouth Conference – Newell and Simon brought a reasoning program "The Logic Theorist" which proved theorems.**

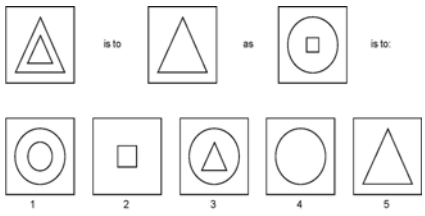
1952 **Samuel's checkers player**

1958 **McCarthy designed** LISP**, helped invent time-sharing and created Advice Taker (a domain independent reasoning system)**

1960's **Microworlds – solving limited problems: SAINT (1963), ANALOGY (1968), STUDENT (1967), blocksworld invented.**

1962 **Perceptron Convergence Theorem is proved.**

## Example ANALOGY Problem




## History of AI
## 1966 – 1974 Recognizing Lack of Knowledge

- **Herb Simon (1957): Computer chess program will be world chess champion within 10 years.**
- **Intractable problems, lack of computing power (Lighthill Report, 1973)**
- **Machine translation**
- **Limitations in knowledge representation (Minsky and Papert, 1969)**

➔ **Knowledge-poor programs**

## Knowledge Representation

- **Human intelligence relies on a lot of background knowledge**
  - the more you know, the easier many tasks become
  - "knowledge is power"
  - E.g. SEND + MORE = MONEY puzzle.
- **Natural language understanding**
  - Time flies like an arrow.
  - Fruit flies like a banana.
  - The spirit is willing but the flesh is weak. (English)
  - The vodka is good but the meat is rotten. (Russian)
- **Or: Plan a trip to L.A.**

## Knowledge-Based Systems/Agents

- **Key components:**
  - Knowledge base: a set of sentences expressed in some knowledge representation language
  - Inference/reasoning mechanisms to query what is known and to derive new information or make decisions.
- **Natural candidate:**
  - logical language (propositional/first-order)
  - combined with a logical inference mechanism
- **How close to human thought?**
  - In any case, appears reasonable strategy for machines.

## Example: Autonomous Car

**State: k-tuple**

(PersonInFrontOfCar, Policeman, Policecar, Slippery, YellowLight, RedLight)

**Actions:**

Brake, Accelerate, TurnLeft, etc.

**Knowledge-base describing when the car should brake?**

( PersonInFrontOfCar ⇒ Brake )
((( YellowLight ∧ Policeman ) ∧ ( ¬Slippery )) ⇒ Brake )
( Policecar ⇒ Policeman )
( Snow ⇒ Slippery )
( Slippery ⇒ ¬Dry )
( RedLight ⇒ Brake )

## Logic as a Knowledge Representation

- **Components of a Formal Logic:**
  - syntax
  - semantics (link to the world)
  - logical reasoning
    - entailment: $\alpha \models \beta$
      if, in every model in which α is true, β is also true.
  - inference algorithm
    - $KB \vdash \alpha$, i.e., α is derived from KB
    - should be sound and complete

## Soundness and Completeness

**Soundness:**

An inference algorithm that derives only entailed sentences is called *sound* or *truth-preserving*.

***KB* ⊢ α implies *KB* ⊨ α**

**Completeness:**

An inference algorithm is *complete* if it can derive any sentence that is entailed.

***KB* ⊨ α implies *KB* ⊢ α**

Why soundness and completeness important?

➔ Allow computer to ignore semantics and "just push symbols"!

## Propositional Logic: Syntax

- **Propositional Symbols**
  - A, B, C, …
- **Connectives**
  - ∧, ∨ , ¬, ⇒, ⇔
- **Sentences**
  - Atomic Sentence: True, False, Propositional Symbol
  - Complex Sentence:
    - ( ¬ Sentence )
    - ( Sentence ∨ Sentence )
    - ( Sentence ∧ Sentence )
    - ( Sentence ⇒ Sentence )
    - ( Sentence ⇔ Sentence )

## Example: Autonomous Car

**Knowledge-base describing when the car should brake?**

( PersonInFrontOfCar ⇒ Brake )
∧ ((( YellowLight ∧ Policeman ) ∧ ( ¬Slippery )) ⇒ Brake )
∧ ( Policecar ⇒ Policeman )
∧ ( Snow ⇒ Slippery )
∧ ( Slippery ⇒ ¬Dry )
∧ ( RedLight ⇒ Brake )

**Observation from sensors:**

YellowLight
∧ ¬RedLight
∧ ¬Snow
∧ Dry
∧ Policecar
∧ ¬PersonInFrontOfCar

## Propositional Logic: Semantics

| $P$ | $Q$ | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \Rightarrow Q$ | $P \Leftrightarrow Q$ |
|---|---|---|---|---|---|---|
| *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* |

- **Model (i.e. possible world):**
  - Assignment of truth values to symbols
  - Example: m={P=True , Q=False}
    - Note: Often called "assignment" instead of "model", and "model" is used for an assignment that evaluates to true.
- **Validity:**
  - A sentence α is valid, if it is true in every model.
- **Satisfiability:**
  - A sentence α is satisfiable, if it is true in at least one model.
- **Entailment:**
  - $\alpha \models \beta$ if and only if, in every model in which α is true, β is also true.

## Model Checking

- **Idea:**
  - To test whether $\alpha \models \beta$, enumerate all models and check truth of $\alpha$ and $\beta$.
  - $\alpha$ entails $\beta$ if no model exists in which $\alpha$ is true and $\beta$ is false (i.e. $(\alpha \wedge \neg\beta)$ is unsatisfiable)
- **Proof by Contradiction:**
  $\alpha \models \beta$ if and only if the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable.
- **Model Checking:**
  - Variables: One for each propositional symbol
  - Domains: {true, false}
  - Objective Function: $(\alpha \wedge \neg\beta)$
  - Which search algorithm works best?

## Propositional Logic: Some Inference Rules

### Modus Ponens:

| | | |
|---|---|---|
| Know: | $\alpha \Rightarrow \beta$ | If raining, then soggy courts. |
| and | $\alpha$ | It is raining. |
| Then: | $\beta$ | Soggy Courts. |

### Modus Tollens:

| | | |
|---|---|---|
| Know: | $\alpha \Rightarrow \beta$ | If raining, then soggy courts. |
| And | $\neg\beta$ | No soggy courts. |
| Then: | $\neg\alpha$ | It is not raining. |

### And-Elimination:

| | | |
|---|---|---|
| Know: | $\alpha \wedge \beta$ | It is raining and soggy courts. |
| Then: | $\alpha$ | It is raining. |

## Example: Forward Chaining

**Knowledge-base describing when the car should brake?**
( PersonInFrontOfCar ⇒ Brake )
∧ ((( YellowLight ∧ Policeman ) ∧ ( ¬Slippery )) ⇒ Brake )
∧ ( Policecar ⇒ Policeman )
∧ ( Snow ⇒ Slippery )
∧ ( Slippery ⇒ ¬Dry )
∧ ( RedLight ⇒ Brake )
∧ ( Winter ⇒ Snow )
**Observation from sensors:**
YellowLight ∧ ¬RedLight ∧ ¬Snow ∧ Dry ∧ Policecar ∧ ¬PersonInFrontOfCar
**What can we infer?**
- And-elimination: Policecar ∧ ( Policecar ⇒ Policeman )
- Modus Ponens: Policeman
- And-elimination: Dry ∧ ( Slippery ⇒ ¬Dry )
- Modus Tollens: ¬Slippery
- And-elimination: YellowLight ∧ Policeman ∧ ¬Slippery
  ∧ ((( YellowLight ∧ Policeman ) ∧ ( ¬Slippery )) ⇒ Brake )
- Modus Ponens: Brake
- And-elimination: ¬Snow
- Modus Tollens: ¬Winter ∧ ( Winter ⇒ Snow )

## Inference Strategy: Forward Chaining

**Idea:**

- Infer everything (?) that can be inferred.
- Notation: In implication $\alpha \Rightarrow \beta$, $\alpha$ (or its compontents) are called premises, $\beta$ is called consequent/conclusion.

**Forward Chaining:**

Given a fact *p* to be added to the KB,

1. Find all implications *I* that have *p* as a premise
2. For each *i* in *I*, if the other premises in *i* are already known to hold
   a) Add the consequent in *i* to the KB

Continue until no more facts can be inferred.

## Inference Strategy: Backward Chaining

**Idea:**

- Check whether a particular fact *q* is true.

**Backward Chaining:**

Given a fact *q* to be "proven",

1. See if *q* is already in the KB. If so, return TRUE.
2. Find all implications, *I*, whose conclusion "matches" *q*.
3. Recursively establish the premises of all *i* in *I* via backward chaining.

➔ **Avoids inferring unrelated facts.**

## Example: Backward Chaining

**Knowledge-base describing when the car should brake?**
( PersonInFrontOfCar ⇒ Brake )
∧ ((( YellowLight ∧ Policeman ) ∧ ( ¬Slippery )) ⇒ Brake )
∧ ( Policecar ⇒ Policeman )
∧ ( Snow ⇒ Slippery )
∧ ( Slippery ⇒ ¬Dry)
∧ ( RedLight ⇒ Brake )
∧ ( Winter ⇒ Snow )
**Observation from sensors:**
YellowLight ∧ ¬RedLight ∧ ¬Snow ∧ Dry ∧ Policecar ∧ ¬PersonInFrontOfCar
**Should the agent brake (i.e. can "brake" be inferred)?**
- Goal: Brake
  - Modus Ponens (brake): PersonInFrontOfCar
    - Failure: PersonInFrontOfCar ➔ Backtracking
- Goal: Brake
  - Modus Ponens (brake): YellowLight ∧ Policeman ∧ ¬Slippery
  - Known (YellowLight): Policeman ∧ ¬Slippery
  - Modus Ponens (Policeman): Policecar ∧ ¬Slippery
  - Known (Policecar): ¬Slippery
  - Modus Tollens (¬Slippery): Dry
  - Known (Dry)