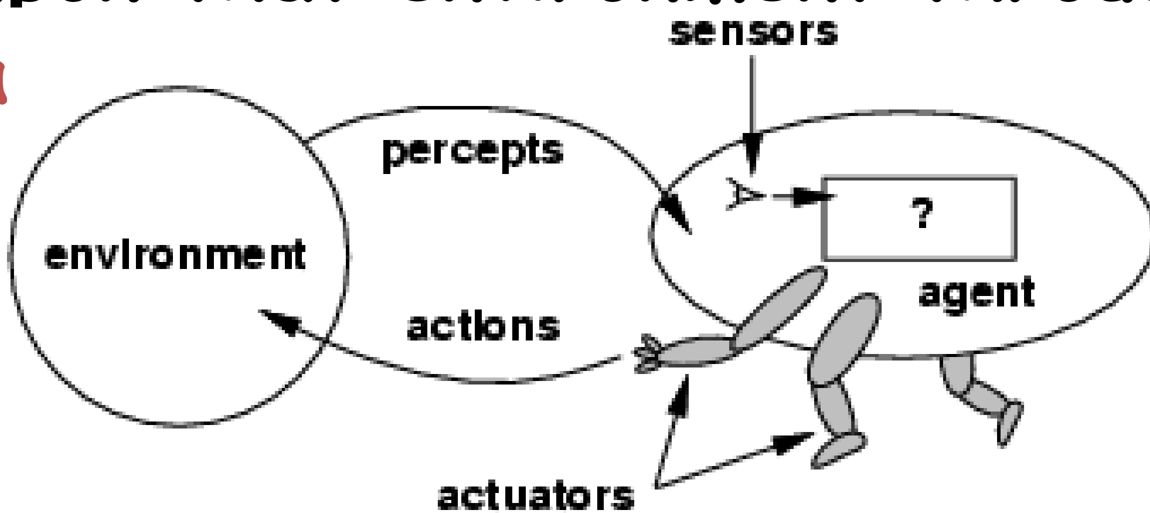
CS 235: Logical Agent

Propositional Logic

Dr. Moumita Roy
CSE Dept., IIITG

Agents

- Definition: An agent perceives its environment via sensors and acts upon that environment through its a



Examples of Agent

- *A* **software agent** has Keystrokes, file contents, received network packages which act as sensors and displays on the screen, files, sent network packets acting as actuators

- *A* **Human agent** has eyes, ears, and other organs which act as sensors and hands, legs, mouth, and other body parts acting as actuators

- *A* **Robotic agent** has Cameras and infrared range finders which act as sensors and various motors acting as actuators

Rational agents

- An agent should strive to "do the right thing", based on what:
  - it can perceive and
  - the actions it can perform.
  - Right action: Select the action which can maximize the performance

Performance measure: ***An objective criterion for success of an agent's behavior.***

Performance measures of a vacuum-cleaner agent: amount of dirt cleaned up, amount of time taken, amount of electricity consumed, level of noise generated, etc.

Performance measures self-driving car: time to reach destination (minimize), safety, predictability of behavior for other agents, reliability, etc.

Performance measure of game-playing agent: win/loss percentage (maximize), robustness, unpredictability (to "confuse" opponent), etc.

Types of AI agents

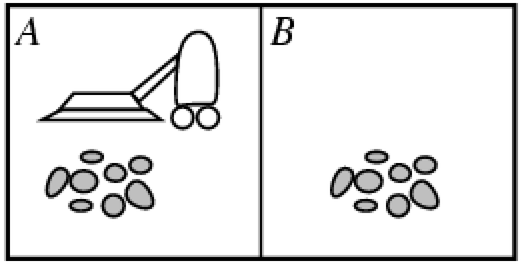
Table-lookup driven agents

- Uses a percept sequence / action table in memory to
- find the next action. Implemented as a (large) lookup table.

Drawbacks:

- Huge table (often simply too large)
- Takes a long time to build/learn the table

Toy example:
Vacuum world.



Percepts: robot senses it's location and "cleanliness."

So, location and contents, e.g., [A, Dirty], [B, Clean].

With 2 locations, we get **4 different possible sensor inputs.**

Actions: *Left*, *Right*, *Suck*, *NoOp*

Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Simple reflex agents

- The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history

- These agents only succeed in the fully observable environment

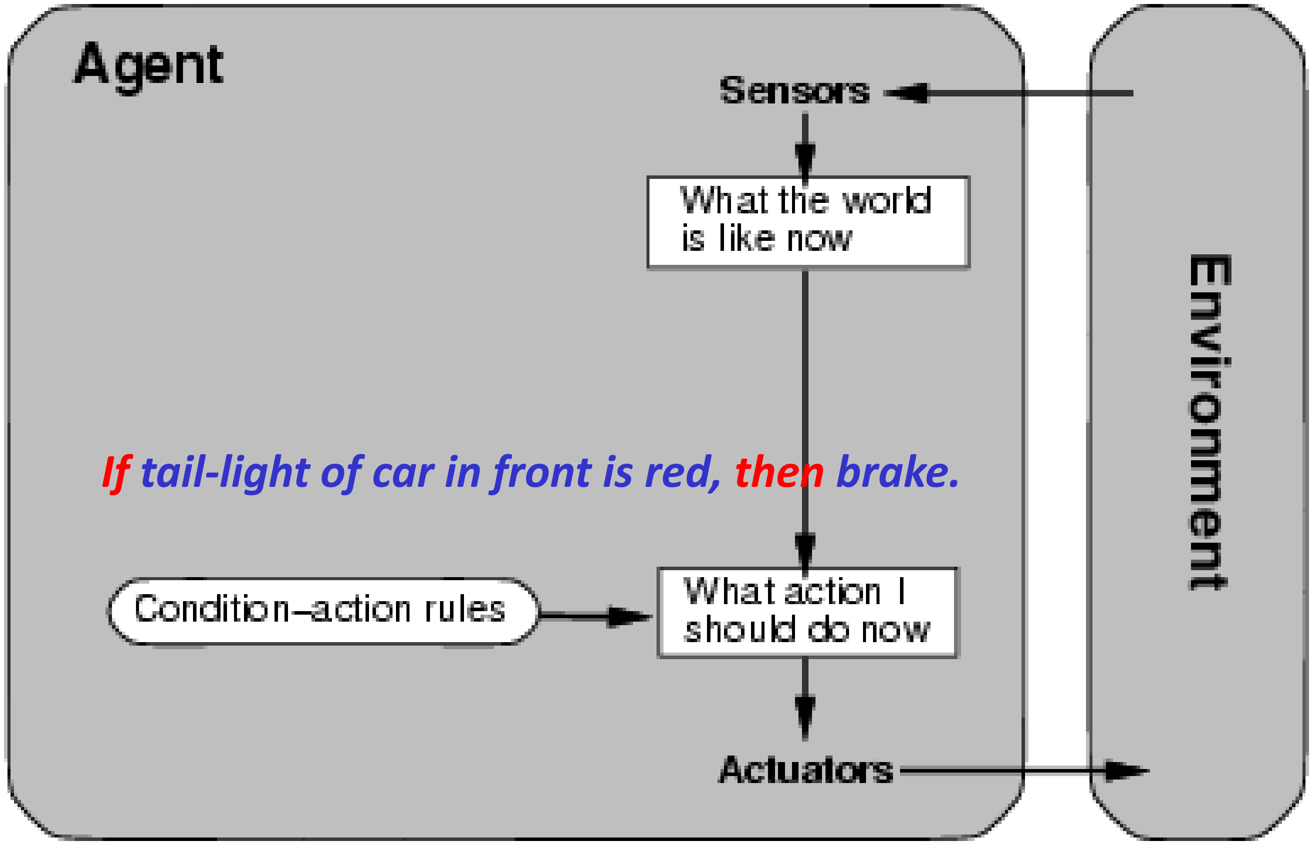
- The Simple reflex agent does not consider any part of percepts history during their decision and action process

- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action

- Problems for the simple reflex agent design approach:

➢ They have very limited intelligence
➢ They do not have knowledge of non-perceptual parts of the current state
➢ Mostly too big to generate and to store.
➢ Not adaptive to changes in the environment.

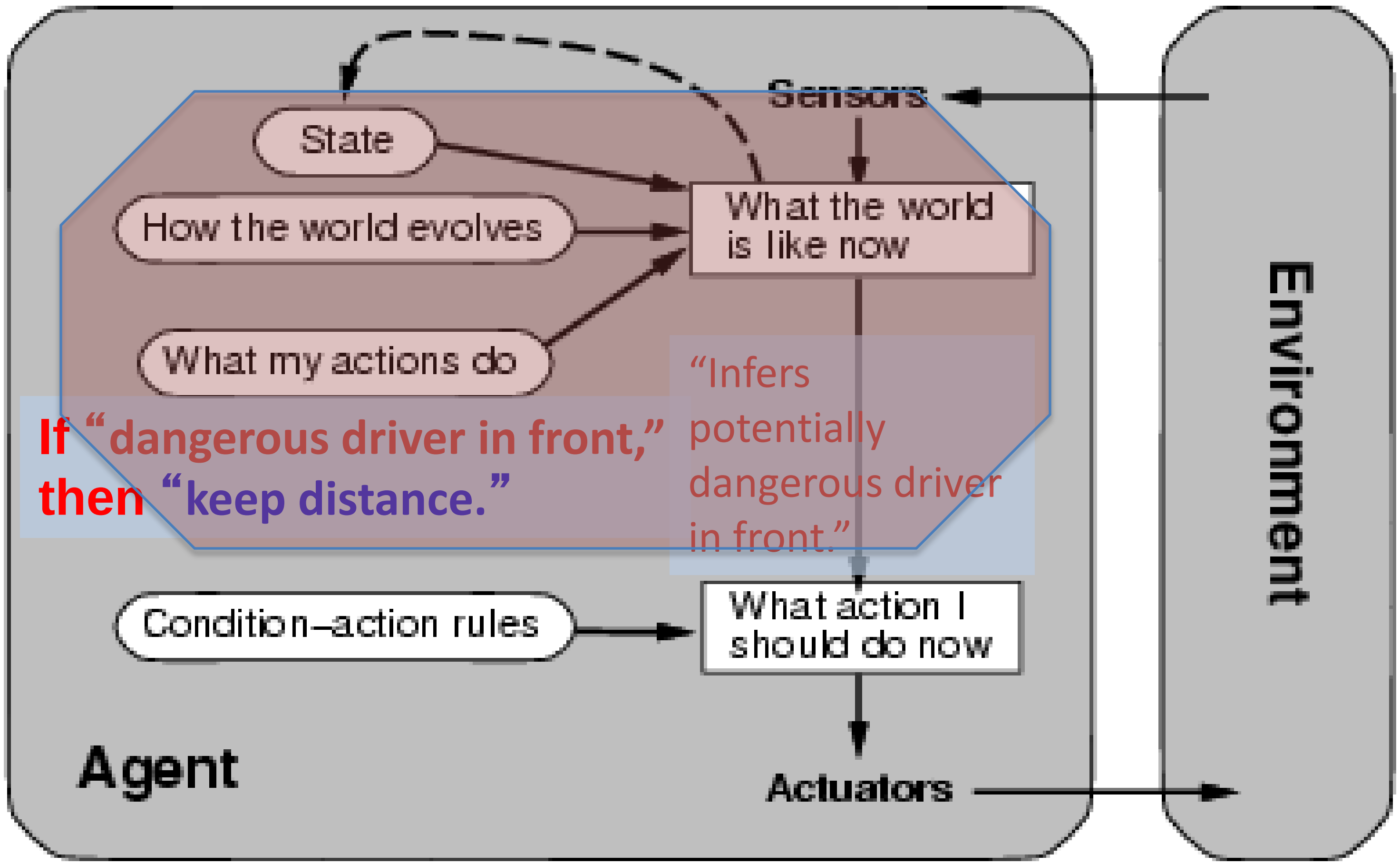
Agent selects actions on the basis of *current* percept only.



Model-based reflex agents

- The Model-based agent can work in a partially observable environment, and track the situation
- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent
  - **Internal State:** It is a representation of the current state based on percept history
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions
- Updating the agent state requires information about:
  - How the world evolves
  - How the agent's action affects the world

Model-based reflex agents



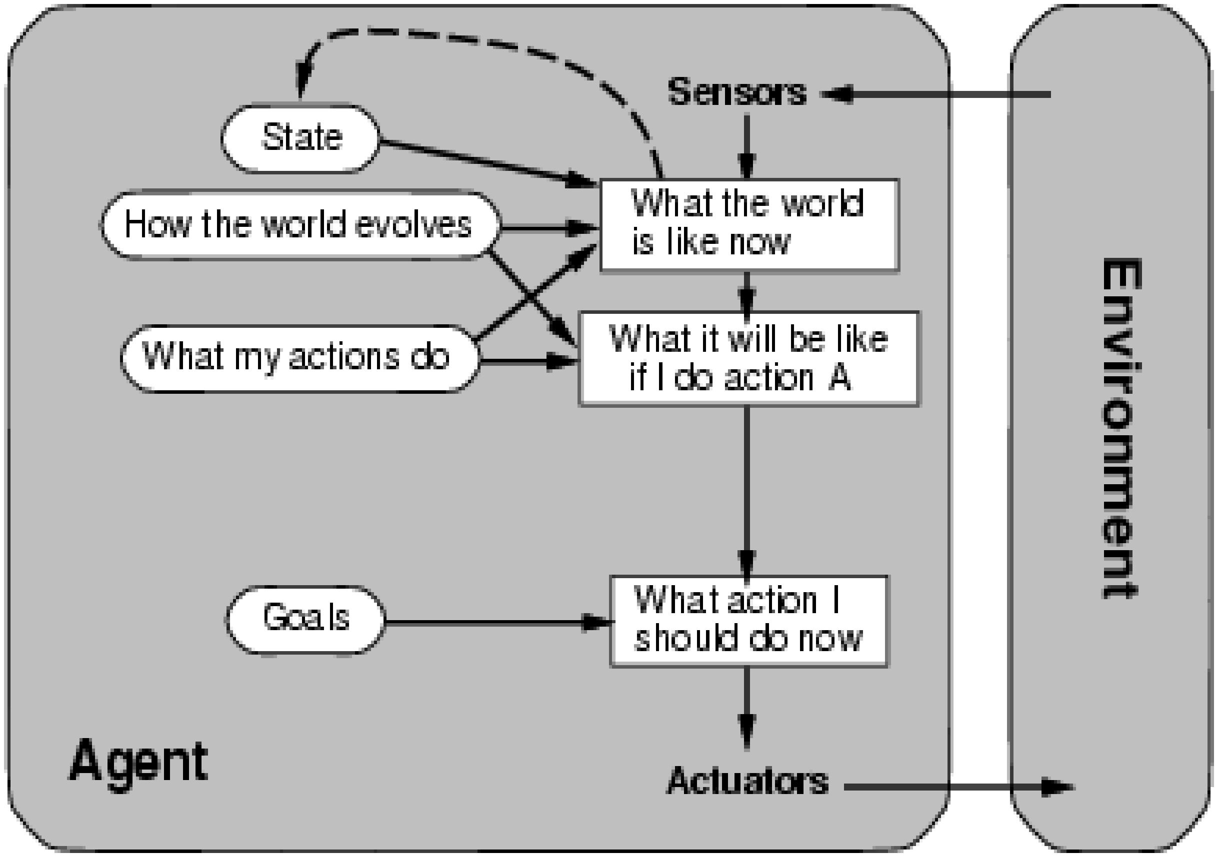
Goal-based agents

- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do
- The agent needs to know its goal which describes desirable situations
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information
- They choose an action, so that they can achieve the goal
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not
- Such considerations of different scenario are called **searching and planning**, which makes an agent proactive

→ ***problem solving and search!***

Module:
Problem Solving

Goal-based agents



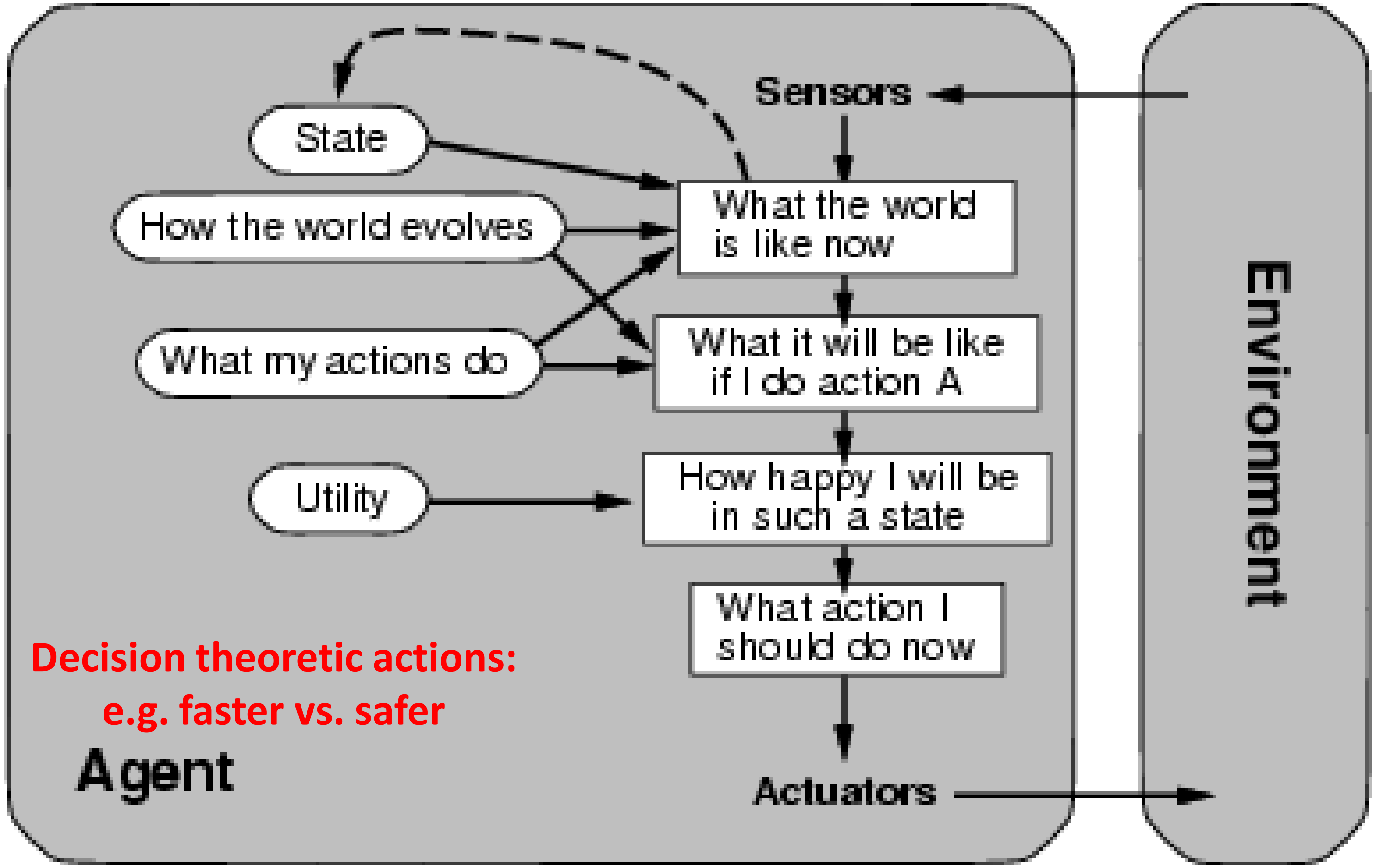
Agent keeps track of the world state as well as set of goals it's trying to achieve: chooses actions that will (eventually) lead to the goal(s).
More flexible than reflex agents → may involve search and planning

Utility-based agents

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state
- Utility-based agent act based not only goals but also the best way to achieve the goal
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action
- The utility function maps each state to a real number to check how efficiently each action achieves the goals

Module:
Decision Making

Utility-based agents



Decision theoretic actions:
e.g. faster vs. safer

Learning Agent

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities

- It starts to act with basic knowledge and then able to act and adapt automatically through learning.

- Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance.

Model-based reflex agents/Logical Agent

- Human do things not only purely on reflex mechanism; but by the process of reasoning that operate on the internal representation of the knowledge

- To address these issues we will introduce:

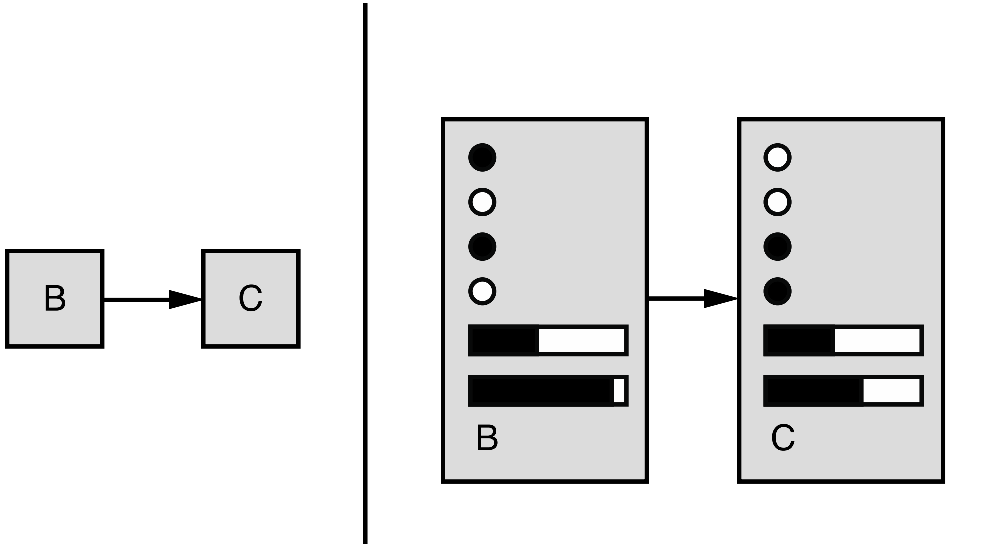
- A knowledge base (KB): a list of facts that are known to the agent; set of sentences in a formal language

- Rules to infer new facts from old facts using rules of inference

- Logic provides the formal language for this

- Declarative approach to building an agent:
- Tell it what it needs to know.
- Ask it what to do -> answers should follow from the KB.

Agent Architecture: Logical Agents



(a) Atomic

(b) Factored

A model is a **structured** representation of the world

- Graph-Based Search: State is **black box**, no internal structure, atomic

- Factored Representation: State is list or vector of facts

- Facts are expressed in **formal logic**.

Knowledge-Based Agents

- KB = knowledge base
  - A set of sentences or facts
  - e.g., a set of statements in a logic language

- Inference
  - Deriving new sentences from old
  - e.g., using a set of logical statements to infer new ones

- A simple model for reasoning
  - Agent is told or perceives new evidence
    - E.g., A is true
  - Agent then infers new facts to add to the KB
    - E.g., KB = { A -> (B OR C) }, then given A and not C we can infer that B is true
    - B is now added to the KB even though it was not explicitly asserted, i.e., the agent inferred B

Logic

- 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

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 students of B.Tech won and the students of M.Tech won" entails "The students of B.Tech won".

To sum up

- A formal language
  - KB = set of sentences
- Syntax
  - what sentences are legal (well-formed)
  - E.g., arithmetic
    - X+2 >= y is a wf sentence, +x2y is not a wf sentence
- Semantics
  - loose meaning: the interpretation of each sentence
  - More precisely:
    - Defines the truth of each sentence wrt to each possible world
  - e.g,
    - X+2 = y is true in a world where x=7 and y =9
    - X+2 = y is false in a world where x=7 and y =1
  - Note: standard logic – each sentence is T of F wrt each world
    - Fuzzy logic – allows for degrees of truth.

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- Atomic sentences = single proposition symbols
  - E.g., P, Q, R (begin with capital letter)
  - Two distinguished atom: True , False
- Complex sentences:
  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic

Consider the propositional logic where
P: It is hot
Q: It is humid
R: It is raining

Formalized the following languages:

1. If it is hot and humid, then it is raining
$(P \wedge Q) \rightarrow R$

2. If it is humid then it is hot
$Q \rightarrow P$

3. It is humid
Q

Propositional logic: Semantics

- The meaning of a sentence determines by its interpretation
- A sentence is interpreted in terms of **models**, or **possible worlds**.

- These are formal structures that specify a truth value for **each sentence** in a consistent manner.

- *m* is a model of a sentence α or m satisfies α if α is true in *m*

- *M*(α) is the set of all models of α

- A model for KB is a possible world- an assignment of truth values to propositional symbols that makes each sentence in KB is true.

- Possible worlds ~ models
  - Possible worlds: potentially real environments
  - Models: mathematical abstractions that establish the truth or falsity of every sentence
- Example:
  - x + y = 4, where x = #men, y = #women

    Possible models = all possible assignments of integers to x and y.

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Propositional logic: Semantics

Each model/world specifies true or false for each proposition symbol

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$

false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model *m*:

$\neg S$	is true iff	S is false	
$S_1 \wedge S_2$	is true iff	S_1 is true and	S_2 is true
$S_1 \vee S_2$	is true iff	S_1 is true or	S_2 is true
$S_1 \Rightarrow S_2$	is true iff	S_1 is false or	S_2 is true
i.e.,	is false iff	S_1 is true and	S_2 is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true and	$S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \textit{true} \wedge (\textit{true} \vee \textit{false}) = \textit{true} \wedge \textit{true} = \textit{true}$$

Entailment

- One sentence follows logically from another

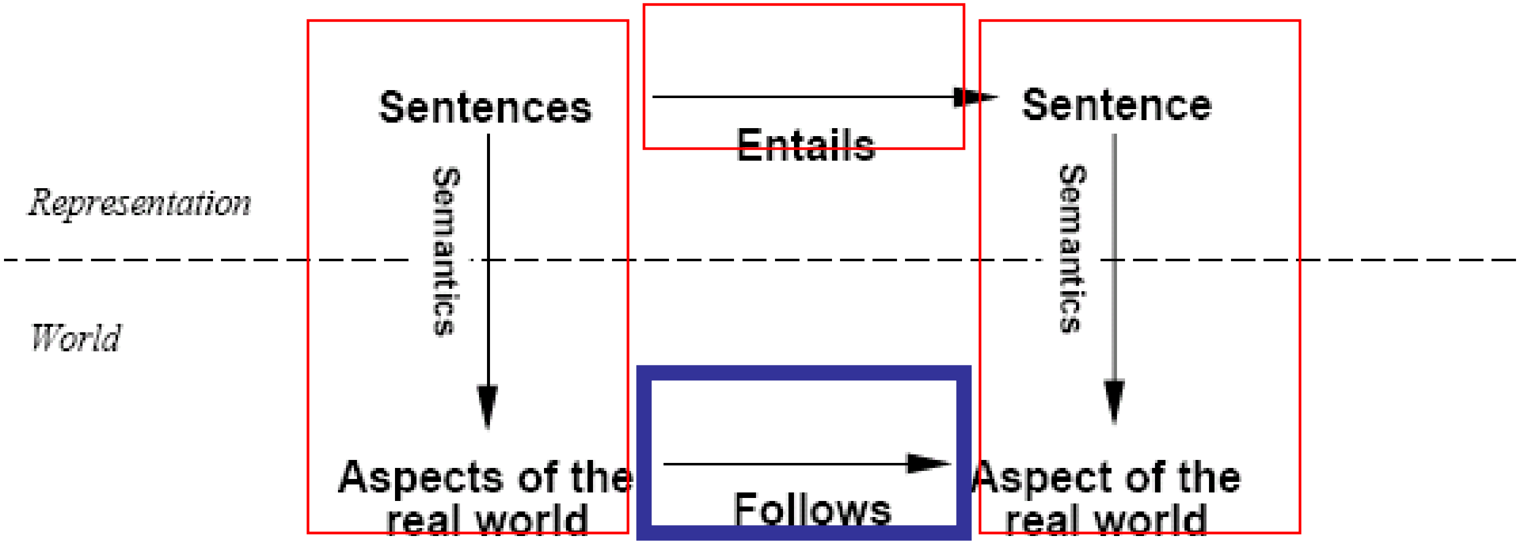
  $\alpha \models \beta$

α entails sentence β *if and only if* β is true in all worlds where α is true.

- Entailment is a relationship between sentences that is based on semantics

- Directly related to logical inference

Schematic perspective



If KB is true in the real world, then any sentence α derived from KB by a sound inference procedure is also true in the real world.

Inference Procedures

- *Logical inference creates new sentences that logically follow from a set of sentences (KB)*
- *KB* $\vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*
- Soundness: *i* is sound if whenever *KB* $\vdash_i \alpha$, it is also true that *KB* $\models \alpha$ *(no wrong inferences, but maybe not all inferences)*
- Completeness: *i* is complete if whenever *KB* $\models \alpha$, it is also true that *KB* $\vdash_i \alpha$ *(all inferences can be made, but maybe some wrong extra ones as well)*

Inference by enumeration

Model checking approach

- We want to see if α is entailed by KB
- Enumeration of all models
- Check that α is true in every model in which KB is true
- For n symbols, time complexity is $O(2^n)$...
- We need a smarter way to do inference!
- In particular, we are going to infer new logical sentences from the data-base and see if they match a query.

Inference by enumeration
Example

KB: $P \wedge Q \rightarrow R$ (R1)

$Q \rightarrow P$ (R2)

5 models

R4: $\neg Q \vee R$

Check R4 is true for all the model

P	Q	R	R1	R2	KB	R4
F	F	F	T	T	T ✓	T
F	F	T	T	T	T ✓	T
F	T	F	T	F	NO	
F	T	T	T	F	NO	
T	F	F	T	T	T ✓	T
T	F	T	T	T	T ✓	T
T	T	F	F	T	NO	
T	T	T	T	T	T ✓	T

New KB:

$P \wedge Q \rightarrow R$

$Q \rightarrow P$

$\neg Q \vee R$

Logical equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are logically equivalent iff they are true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Validity and satisfiability

A sentence is valid or **tautology** if it is true in all models,

e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is false in all models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

(there is no model for which KB=true and is α false)

Normal Form

We like to prove: $\boxed{\begin{array}{l} KB \models \alpha \\ \textit{equivalent to}: KB \wedge \neg\alpha \textit{ unsatisfiable} \end{array}}$

We first rewrite $KB \wedge \neg\alpha$ into conjunctive normal form (CNF).

A "conjunction of disjunctions"

literals

$$\underbrace{(A \vee \neg B)}_{\text{Clause}} \wedge \underbrace{(B \vee \neg C \vee \neg D)}_{\text{Clause}}$$

- Any KB can be converted into CNF

Example: Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation: $\neg(\alpha \vee \beta) = \neg\alpha \wedge \neg\beta$
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributive law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$
   A conjunction of three clauses

Sound rules of inference

- Here are some examples of sound rules of inference
- Each can be shown to be sound using a truth table

RULE	PREMISE	CONCLUSION
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$\mathbf{A \vee B, \neg B \vee C}$	$\mathbf{A \vee C}$

Resolution

- Resolution: inference rule for CNF: sound and complete!

$(A \vee B \vee C)$

$(\neg A)$ “If A or B or C is true, but not A, then B or C must be true.”

$\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee C)$

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

$\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee C \vee D \vee E)$

“If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true.”

$(A \vee B)$

$(\neg A \vee B)$

$\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$ Simplification

$\therefore (B \vee B) \equiv B$

Resolution

- **Resolution** is a valid inference rule producing a new clause implied by two clauses containing *complementary literals*
  - A literal is an atomic symbol or its negation, i.e., P, ~P

Resolution

- A KB is actually a set of sentences all of which are true, i.e., a conjunction of sentences.
- To use resolution, put KB into *conjunctive normal form* (CNF), where each sentence written as a disjunction of (one or more) literals

Example

- KB: [$P \rightarrow Q$, $Q \rightarrow R \wedge S$]
- KB in CNF: [$\sim P \vee Q$, $\sim Q \vee R$, $\sim Q \vee S$]
- Resolve KB(1) and KB(2) producing: $\sim P \vee R$ *(i.e., $P \rightarrow R$)*
- Resolve KB(1) and KB(3) producing: $\sim P \vee S$ *(i.e., $P \rightarrow S$)*
- New KB: [$\sim P \vee Q$, $\sim Q \vee \sim R \vee \sim S$, $\sim P \vee R$, $\sim P \vee S$]

Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- $KB \wedge \neg\alpha$ is converted to CNF
- The resolution rule is applied to the resulting clauses.
- Each pair that contains complementary literals is resolved to produce the new clause, which is added if it is not already present.

- Continue until one of two things happens:

1. There is no new clause that can be added, in which case KB dose not entail α (no contradiction; there is a model that satisfies the sentence $KB \wedge \neg\alpha$ (non-trivial)

2. Two clauses resolve to yield the empty clause, , in which case KB entails α. The empty clause represents a contradiction is to observe that it arises only from resolving two complementary unit clauses

Resolution example

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$