Unit - 5

Reasoning

5.1 Inference in First-Order Logic

Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.

Substitution:

Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL. If we write F[a/x], so it refers to substitute a constant "a" in place of variable "x".

Note: First-order logic is capable of expressing facts about some or all objects in the universe.

Equality:

First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, which is equality in FOL. For this, we can use equality symbols which specify that the two terms refer to the same object.

Example: Brother (John) = Smith.

As in the above example, the object referred by the Brother (John) is similar to the object referred by Smith. The equality symbol can also be used with negation to represent that two terms are not the same objects.

Example: ¬(x=y) which is equivalent to x ≠y.

FOL inference rules for quantifier:

As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:

- Universal Generalization
- Universal Instantiation
- Existential Instantiation
- Existential introduction

1. Universal Generalization:

- Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).
- It can be represented as: $\frac{P(c)}{\forall x P(x)}$
- This rule can be used if we want to show that every element has a similar property.
- In this rule, x must not appear as a free variable.

Example: Let's represent, P(c): "A byte contains 8 bits", so for ∀ x P(x) "All bytes contain 8 bits.", it will also be true.

2. Universal Instantiation:

- Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

- The new KB is logically equivalent to the previous KB.

- As per UI, we can infer any sentence obtained by substituting a ground term for the variable.

- The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ x P(x) for any object in the universe of discourse.

- It can be represented as: $\frac{\forall x P(x)}{P(c)}$.

Example:1.

IF "Every person like ice-cream"=> ∀x P(x) so we can infer that "John likes ice-cream" => P(c)

Example: 2.

Let's take a famous example,

"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:

∀x king(x) ∧ greedy (x) → Evil (x),

So from this information, we can infer any of the following statements using

Universal Instantiation:

- King(John) ∧ Greedy (John) → Evil (John),
- King(Richard) ∧ Greedy (Richard) → Evil (Richard),
- King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)),

3. Existential Instantiation:

- Existential instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic.
- It can be applied only once to replace the existential sentence.
- The new KB is not logically equivalent to old KB, but it will be satisfiable if old KB was satisfiable.
- This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c.
- The restriction with this rule is that c used in the rule must be a new term for which P(c) is true.
- It can be represented as: $\frac{\exists x P(x)}{P(c)}$

Example:

From the given sentence: ∃x Crown(x) ∧ OnHead(x, John),

So we can infer: Crown(K) ∧ OnHead(K, John), as long as K does not appear in the knowledge base.

- The above used K is a constant symbol, which is called Skolem constant.
- The Existential instantiation is a special case of Skolemization process.

4. Existential introduction

- An existential introduction is also known as an existential generalization, which is a valid inference rule in first-order logic.
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.
- It can be represented as: $\frac{P(c)}{\exists x P(x)}$
- Example: Let's say that,

"Priyanka got good marks in English."

"Therefore, someone got good marks in English."

Generalized Modus Ponens Rule:

For the inference process in FOL, we have a single inference rule which is called Generalized Modus Ponens. It is lifted version of Modus ponens.

Generalized Modus Ponens can be summarized as, " P implies Q and P is asserted to be true, therefore Q must be True."

According to Modus Ponens, for atomic sentences pi, pi', q. Where there is a substitution θ such that SUBST (θ, pi',) = SUBST(θ, pi), it can be represented as:

$$\frac{p1', p2', \ldots, pn, , (p1 \wedge p2 \wedge \ldots \wedge pn \Rightarrow q)}{SUBST(\theta, q)}$$

Example:

We will use this rule for Kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.

1. Here let say, p1' is king(John) p1 is king(x)
2. p2' is Greedy(y) p2 is Greedy(x)
3. θ is {x/John, y/John} q is evil(x)

SUBST(θ,q).

Key takeaway

Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.

5.2 Propositional vs. First-Order Inference

Propositional vs. First-Order Inference

Previously, inference in first order logic was checked via propositionalization, which is the act of turning the Knowledge Base included in first order logic into propositional logic and then utilizing any of the propositional logic inference mechanisms to check inference.

Inference rules for quantifiers:

To get sentences without quantifiers, several inference techniques can be applied to sentences containing quantifiers. We'll be able to make the conversion if we follow these requirements.

Universal Instantiation (UI):

The rule states that by substituting a ground word (a term without variables) for the variable, we can deduce any sentence. Let SUBST () stand for the outcome of the substitution on the sentence a. The rule is then written.

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

For any v and g ground term combinations. For example, in the knowledge base, there is a statement that states that all greedy rulers are Evil.

$\forall$ x King(x)A Greedy (x)$\Rightarrow$Evil(x)

For the variable x, with the substitutions like {x/John}, {x/Richard} the following sentences can be inferred

King(John)A Greedy (John)$\Rightarrow$Evil(John)
King(Richard)A Greedy (Richard)$\Rightarrow$Evil(Richard)

As a result, the set of all potential instantiations can be used to substitute a globally quantified phrase.

Existential Instantiation (EI):

The existential statement states that there is some object that fulfills a requirement, and that the instantiation process is simply giving that object a name that does not already belong to another object. A Skolem constant is the moniker given to this new name. Existential Instantiation is a subset of a broader process known as "skolemization."

If the statement a, variable v, and constant symbol k do not occur anywhere else in the knowledge base,

$$\frac{\exists v_{\alpha}}{\mathrm{SUBST}(\{v/k\}_{\alpha})}$$

For example, from the sentence

$3\ x\ \text{Crown}(x)\text{A On Head}(x, \text{John})$

So we can infer the sentence

$\text{Crown}\ (C_1)\text{A On Head}\ (C_1,\text{John})$

As long as C1 does not appear elsewhere in the knowledge base. Thus, an existentially quantified sentence can be replaced by one instantiation

Elimination of Universal and Existential quantifiers should give new knowledge base which can be shown to be inferentially equivalent to old in the sense that it is satisfiable exactly when the original knowledge base is satisfiable.

5.3 Unification

Unification is the process of finding a substitute that makes two separate logical atomic expressions identical. The substitution process is necessary for unification.

- It accepts two literals as input and uses substitution to make them identical.
- Let Ψ_1 and Ψ_2 be two atomic sentences and σ be a unifier such that, $\Psi_1\sigma = \Psi_2\sigma$, then it can be expressed as UNIFY (Ψ_1, Ψ_2).
- Example: Find the MGU for Unify {King(x), King (John)}

Let Ψ_1 = King(x), Ψ_2 = King (John),

Substitution θ = {John/x} is a unifier for these atoms, and both phrases will be identical after this replacement.

- For unification, the UNIFY algorithm is employed, which takes two atomic statements and returns a unifier for each of them (If any exist).
- All first-order inference techniques rely heavily on unification.
- If the expressions do not match, the result is failure.

- The replacement variables are referred to as MGU (Most General Unifier).

Example: Let's say there are two different expressions, **P (x, y), and P (a, f(z))**.

In this example, we need to make both above statements identical to each other. For this, we will perform the substitution.

P (x, y)..... (i)

P (a, f(z))..... (ii)

- Substitute x with a, and y with f(z) in the first expression, and it will be represented as **a/x** and f(z)/y.
- With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: **[a/x, f(z)/y]**.

Conditions for Unification:

The following are some fundamental requirements for unification:

- Atoms or expressions with various predicate symbols can never be united since they have different predicate symbols.
- Both phrases must have the same number of arguments.
- If two comparable variables appear in the same expression, unification will fail.

Generalized Modus Ponens:

For atomic sentences pi, pi ', and q, where there is a substitution θ such that SU θ, pi) = SUBST (θ, pi '), for all i

$$\frac{p1',p2',...,pn, ,(p1 \wedge p2 \wedge ... \wedge pn \Rightarrow q)}{SUBST(\theta,q)}$$

N + 1 premises = N atomic sentences + one implication. Applying SUBST (θ, q) yields the conclusion we seek.

p1' = King (John) p2' = Greedy(y)

p1 = King(x) p2 = Greedy(x)

θ = {x / John, y/ John} q = Evil(x)

SUBST (θ, q) is Evil (John)

Generalized Modus Ponens = lifted Modus Ponens

Lifted - transformed from:

Propositional Logic → First-order Logic

Unification Algorithms

Algorithm: Unify (L1, L2)

1. If L1 or L2 are both variables or constants, then:

1. If L1 and L2 are identical, then return NIL.

2. Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).

3. Also, Else if L2 is a variable, then if L2 occurs in L1 then return {FAIL}, else return (L1/L2). d. Else return {FAIL}.

2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.

3. If LI and L2 have a different number of arguments, then return {FAIL}.

4. Set SUBST to NIL. (At the end of this procedure, SUBST will contain all the substitutions used to unify L1 and L2.)

5. For I ← 1 to the number of arguments in L1:

1. Call Unify with the ith argument of L1 and the ith argument of L2, putting the result in S.

2. If S contains FAIL, then return {FAIL}.

3. If S is not equal to NIL, then:

2. Apply S to the remainder of both L1 and L2.

3. SUBST: = APPEND (S, SUBST).

6. Return SUBST.

Key takeaway

Unification is a process of making two different logical atomic expressions identical by finding a substitution.

Unification depends on the substitution process.

5.4 First-Order Inference

Inference in first order logic was checked via propositionalization, which is the act of turning the Knowledge Base included in first order logic into propositional logic and then utilizing any of the propositional logic inference mechanisms to check inference.

Inference rules for quantifiers:

To get sentences without quantifiers, several inference techniques can be applied to sentences containing quantifiers. We'll be able to make the conversion if we follow these requirements.

Universal Instantiation (UI):

The rule states that by substituting a ground word (a term without variables) for the variable, we can deduce any sentence. Let SUBST () stand for the outcome of the substitution on the sentence a. The rule is then written.

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

For any v and g ground term combinations. For example, in the knowledge base, there is a statement that states that all greedy rulers are Evil.

$\forall x\ King(x) A\ Greedy\ (x) \Rightarrow Evil(x)$

For the variable x, with the substitutions like {x/John},{x/Richard}the following sentences can be inferred

King(John)A Greedy (John)⇒Evil(John)
King(Richard)A Greedy (Richard)⇒Evil(Richard)

As a result, the set of all potential instantiations can be used to substitute a globally quantified phrase.

Existential Instantiation (EI):

The existential statement states that there is some object that fulfills a requirement, and that the instantiation process is simply giving that object a name that does not already belong to another object. A Skolem constant is the moniker given to this new name. Existential Instantiation is a subset of a broader process known as "skolemization."

If the statement a, variable v, and constant symbol k do not occur anywhere else in the knowledge base,

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

For example, from the sentence

$3x\ \mathrm{Crowm}(x)\ \mathrm{A}\ \mathrm{OnHead}(x,\mathrm{John})$

So we can infer the sentence

$\mathrm{Crowm}(C_1)\ \mathrm{A}\ \mathrm{OnHead}(C_1,\mathrm{John})$

As long as C1 does not appear elsewhere in the knowledge base. Thus an existentially quantified sentence can be replaced by one instantiation

Elimination of Universal and Existential quantifiers should give new knowledge base which can be shown to be inferentially equivalent to old in the sense that it is satisfiable exactly when the original knowledge base is satisfiable.

Key takeaway

1. In First-Order Logic, inference is used to derive new facts or sentences from existing ones.
2. In First-Order Logic, atomic sentences are formed not only via the use of predicate and words, but also via the application of equality.
3. There are some Inference rules that can be applied to sentences with quantifiers to obtain sentences without quantifiers.

5.5 Forward Chaining, Backward Chaining

Forward chaining is also known as forward deduction or forward reasoning when using an inference engine. Forward chaining is a method of reasoning that begins with atomic sentences in a knowledge base and uses inference rules to extract more information in the forward direction until a goal is reached.

Beginning with known facts, the Forward-chaining algorithm activates all rules with satisfied premises and adds their conclusion to the known facts. This procedure is repeated until the problem is resolved.

It's a method of inference that starts with sentences in the knowledge base and generates new conclusions that may then be utilized to construct subsequent inferences. It's a plan based on data. We start with known facts and organize or

chain them to get to the query. The ponens modus operandi is used. It begins with the data that is already accessible and then uses inference rules to extract new data until the goal is achieved.

Example: To determine the color of Fritz, a pet who croaks and eats flies, for example. The Knowledge Base contains the following rules:

If X croaks and eats flies — → Then X is a frog.

If X sings — → Then X is a bird.

If X is a frog — → Then X is green.

If x is a bird — -> Then X is blue.

In the KB, croaks and eating insects are found. As a result, we arrive at rule 1, which has an antecedent that matches our facts. The KB is updated to reflect the outcome of rule 1, which is that X is a frog. The KB is reviewed again, and rule 3 is chosen because its antecedent (X is a frog) corresponds to our newly confirmed evidence. The new result is then recorded in the knowledge base (i.e., X is green). As a result, we were able to determine the pet's color based on the fact that it croaks and eats flies.

Properties of forward chaining

- It is a down-up approach since it moves from bottom to top.
- It is a process for reaching a conclusion based on existing facts or data by beginning at the beginning and working one's way to the end.
- Because it helps us to achieve our goal by exploiting current data, forward-chaining is also known as data-driven.
- The forward-chaining methodology is often used in expert systems, such as CLIPS, business, and production rule systems.

Backward chaining

Backward-chaining is also known as backward deduction or backward reasoning when using an inference engine. A backward chaining algorithm is a style of reasoning that starts with the goal and works backwards through rules to find facts that support it.

It's an inference approach that starts with the goal. We look for implication phrases that will help us reach a conclusion and justify its premises. It is an approach for proving theorems that is goal-oriented.

Example: Fritz, a pet that croaks and eats flies, needs his color determined. The Knowledge Base contains the following rules:

If X croaks and eats flies — → Then X is a frog.

If X sings — → Then X is a bird.

If X is a frog — → Then X is green.

If x is a bird — -> Then X is blue.

The third and fourth rules were chosen because they best fit our goal of determining the color of the pet. For example, X may be green or blue. Both the antecedents of the rules, X is a frog and X is a bird, are added to the target list. The first two rules were chosen because their implications correlate to the newly added goals, namely, X is a frog or X is a bird.

Because the antecedent (If X croaks and eats flies) is true/given, we can derive that Fritz is a frog. If it's a frog, Fritz is green, and if it's a bird, Fritz is blue, completing the goal of determining the pet's color.

Properties of backward chaining

- A top-down strategy is what it's called.
- Backward-chaining employs the modus ponens inference rule.
- In backward chaining, the goal is broken down into sub-goals or sub-goals to ensure that the facts are true.

- Because a list of objectives decides which rules are chosen and implemented, it's known as a goal-driven strategy.
- The backward-chaining approach is utilized in game theory, automated theorem proving tools, inference engines, proof assistants, and other AI applications.
- The backward-chaining method relied heavily on a depth-first search strategy for proof.

Key takeaway

1. Forward chaining is a type of reasoning that starts with atomic sentences in a knowledge base and uses inference rules to extract more material in the forward direction until a goal is attained.
2. The Forward-chaining algorithm begins with known facts, then activates all rules with satisfied premises and adds their conclusion to the known facts.
3. A backward chaining algorithm is a type of reasoning that begins with the objective and goes backward, chaining via rules to discover known facts that support it.

5.6 Resolution

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.

Clause: Disjunction of literals (an atomic sentence) is called a clause. It is also known as a unit clause.

Conjunctive Normal Form: A sentence represented as a conjunction of clauses is said to be conjunctive normal form or CNF.

The resolution inference rule:

The resolution rule for first-order logic is simply a lifted version of the propositional rule. Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.

$$\frac{l_1 V \ldots V l_k,\ m_1 V \ldots . V m_n}{\text{SUBST}\ (\theta, l_1 V \ldots V l_{i-1} V l_{i+1} V \ldots . V l_k V m_{j-1} V \ldots V m_n}$$

Where li and mj are complementary literals.

This rule is also called the binary resolution rule because it only resolves exactly two literals.

Example:

We can resolve two clauses which are given below:

[Animal (g(x) V Loves (f(x), x)] and [¬ Loves(a, b) V ¬Kills(a, b)]

Where two complimentary literals are: Loves (f(x), x) and ¬ Loves (a, b)

These literals can be unified with unifier θ= [a/f(x), and b/x], and it will generate a resolvent clause:

[Animal (g(x) V ¬ Kills(f(x), x)].

Steps for Resolution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

To better understand all the above steps, we will take an example in which we will apply resolution.

Example:

John likes all kind of food.

Apple and vegetable are food

Anything anyone eats and not killed is food.

Anil eats peanuts and still alive

1. Harry eats everything that Anil eats.

Prove by resolution that:

John likes peanuts.

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.

a. $\forall x: food(x) \rightarrow likes\ (John, x)$

b. $food(Apple) \wedge food(vegetables)$

c. $\forall x \forall y: eats(x,y) \wedge \neg killed(x) \rightarrow food(y)$

d. $eats\ (Anil, Peanuts) \wedge alive(Anil)$

e. $\forall x: eats(Anil, x) \rightarrow eats(Harry,x)$

f. $\forall x: \neg killed(x) \rightarrow alive(x)$

g. $\forall x: alive\ (x) \rightarrow \neg killed(x)$

h. $likes(John, Peanuts)$

Here f and g are added predicates.

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

Eliminate all implication (→) and rewrite

1. ∀x ¬ food(x) V likes(John, x)
2. Food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ [eats(x, y) Λ¬ killed(x)] V food(y)
4. Eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. ∀x¬ [¬ killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. Likes(John, Peanuts).

Move negation (¬)inwards and rewrite

9. ∀x ¬ food(x) V likes(John, x)

10. Food(Apple) Λ food(vegetables)

11. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)

12. Eats (Anil, Peanuts) Λ alive(Anil)

13. ∀x ¬ eats(Anil, x) V eats(Harry, x)

14. ∀x ¬killed(x)] V alive(x)

15. ∀x ¬ alive(x) V ¬ killed(x)

16. Likes(John, Peanuts).

Rename variables or standardize variables

17. ∀x ¬ food(x) V likes(John, x)

18. Food(Apple) Λ food(vegetables)

19. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)

20. Eats (Anil, Peanuts) Λ alive(Anil)

21. ∀w¬ eats(Anil, w) V eats(Harry, w)

22. ∀g ¬killed(g)] V alive(g)

23. ∀k ¬ alive(k) V ¬ killed(k)

24. Likes(John, Peanuts).

- Eliminate existential instantiation quantifier by elimination.

In this step, we will eliminate existential quantifier ∃, and this process is known as Skolemization. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

- Drop Universal quantifiers.

In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.

1. ¬ food(x) V likes(John, x)
2. Food(Apple)
3. Food(vegetables)
4. ¬ eats(y, z) V killed(y) V food(z)
5. Eats (Anil, Peanuts)
6. Alive(Anil)
7. ¬ eats(Anil, w) V eats(Harry, w)
8. Killed(g) V alive(g)
9. ¬ alive(k) V ¬ killed(k)
10. Likes(John, Peanuts).

Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.

- Distribute conjunction ∧ over disjunction ¬.

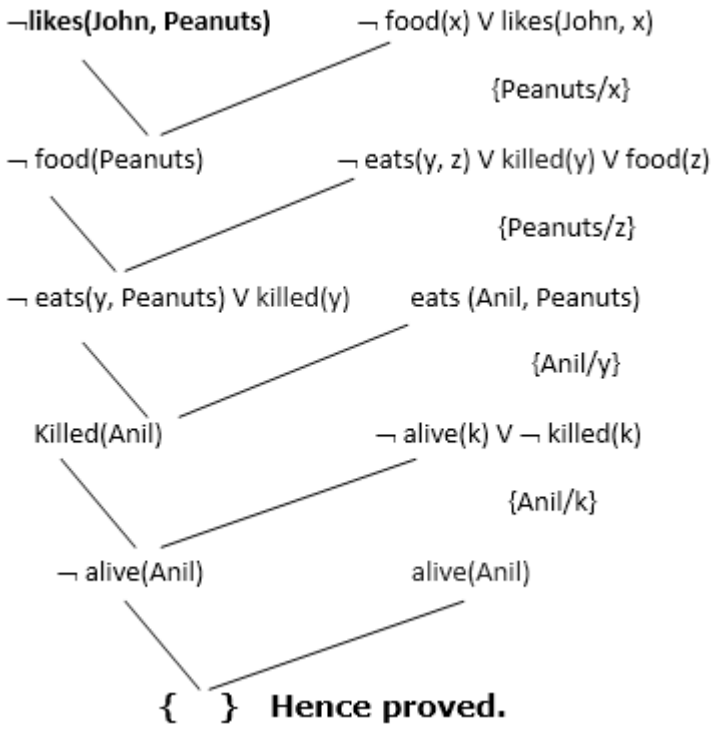
This step will not make any change in this problem.

Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph:

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:



we are left with ¬ food(Peanuts)

Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

Explanation of Resolution graph:

- In the first step of resolution graph, ¬likes(John, Peanuts) , and likes(John, x) get resolved(cancelled) by substitution of {Peanuts/x}, and

- In the second step of the resolution graph, ¬ food(Peanuts) , and food(z) get resolved (cancelled) by substitution of { Peanuts/z}, and we are left with ¬ eats(y, Peanuts) V killed(y) .

- In the third step of the resolution graph, ¬ eats(y, Peanuts) and eats (Anil, Peanuts) get resolved by substitution {Anil/y}, and we are left with Killed(Anil) .

- In the fourth step of the resolution graph, Killed(Anil) and ¬ killed(k) get resolve by substitution {Anil/k}, and we are left with ¬ alive(Anil) .

- In the last step of the resolution graph ¬ alive(Anil) and alive(Anil) get resolved.

Key takeaway

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.

5.7 Knowledge Representation

Humans are best at understanding, reasoning, and interpreting knowledge. Humans know things, which is that of the knowledge and as per that of their knowledge they perform several of the actions in that of the real world. **But how machines do all of these things come under the knowledge representation and the reasoning**. Hence we can describe that of the Knowledge representation as follows they are:

- Knowledge representation and the reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behaviour of agents.

- It is responsible for representing information about the real world so that a computer can understand and then can utilize that of the knowledge to solve that

of the complex real world problems for instance diagnosis a medical condition or communicating with humans in natural language.

- It is also a way which describes that how we can represent that of the knowledge in that of the artificial intelligence. Knowledge representation is not just that of the storing data into some of the database, but it also enables of an intelligent machine to learn from that of the knowledge and experiences so that of the it can behave intelligently like that of a human.

What to Represent:

Following are the kind of knowledge which needs to be represented in that of the AI systems:

- **Object:** All the facts about that of the objects in our world domain. Example Guitars contains strings, trumpets are brass instruments.

- **Events:** Events are the actions which occur in our world.

- **Performance:** It describes that of the behaviour which involves the knowledge about how to do things.

- **Meta-knowledge:** It is knowledge about what we know.

- **Facts:** Facts are the truths about the real world and what we represent.

- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Knowledge: Knowledge is the awareness or familiarity gained by that of the experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

Types of knowledge

Following are the various types of knowledge:

1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes the concepts, the facts, and the objects.
- It is also called descriptive knowledge and expressed in declarative sentences.
- It is simpler than procedural language.

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

3. Meta-knowledge:

- Knowledge about the other types of that of the knowledge is known as Meta-knowledge.

4. Heuristic knowledge:

- Heuristic knowledge is representing knowledge of some experts in a filed or subject.

- Heuristic knowledge is the rules of the thumb based on the previous experiences, awareness of the approaches, and which are good to that of the work but not guaranteed.

5. Structural knowledge:

- Structural knowledge is basic knowledge to problem-solving.

- It describes relationships between various concepts such as kind of, part of, and grouping of something.

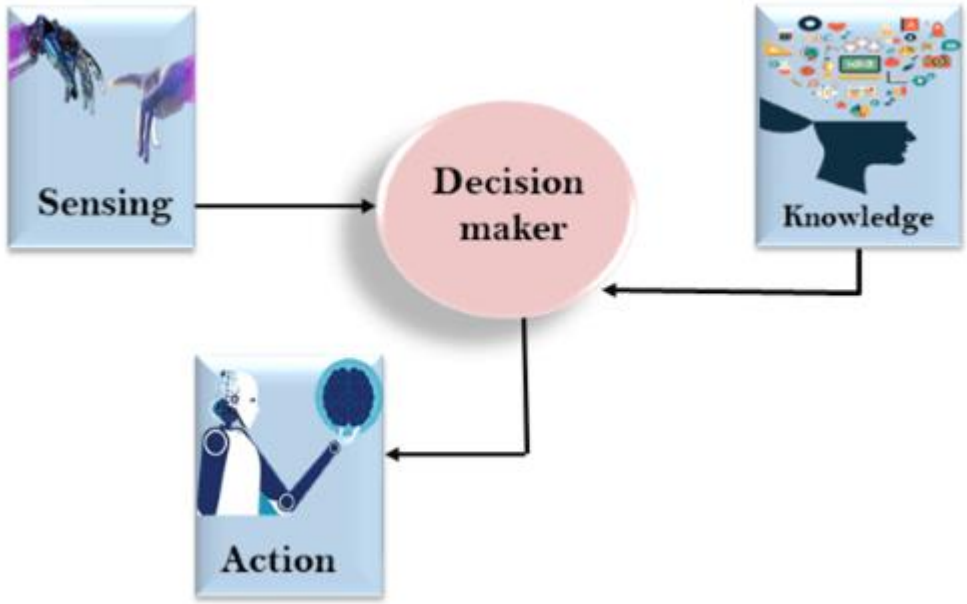
- It describes the relationship that exists between concepts or objects.

The relation between knowledge and intelligence they are as follows:

Knowledge of real-worlds plays a very important role in the intelligence and the same for creating the artificial intelligence. Knowledge plays an important role in that of the demonstrating intelligent behaviour in the AI agents. An agent is the only able to accurately act on some of the input when he has some of the knowledge or the experience about that of the input.

Let's suppose if you met some of the person who is speaking in that of a language which you don't know, then how you will able to act on that. The same thing applies to that of the intelligent behaviour of the agents.

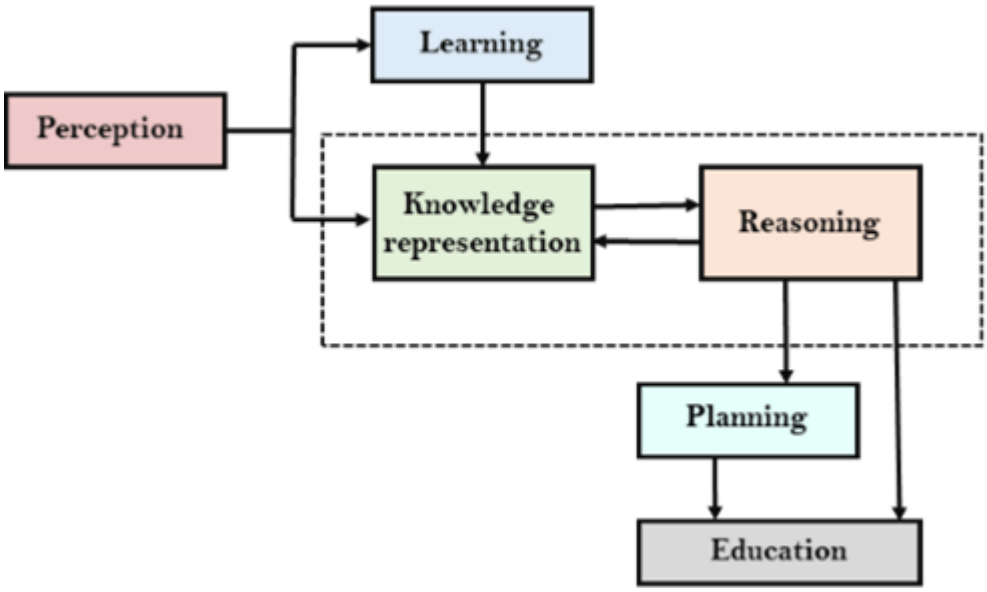
As we can see in below diagram, there is one decision maker which act by sensing that of the environment and using the knowledge. But if the knowledge part will not present then, it cannot display intelligent behaviour.



AI knowledge cycle:

An Artificial intelligence system has that of the following components for displaying intelligent behaviour they are as follows:

- Perception
- Learning
- Knowledge Representation and Reasoning
- Planning
- Execution



The above diagram is showing how that of an AI system can interacts with that of the real world and what the components help it to show the intelligence. AI system has the Perception component by which it retrieves information from that of its environment. It can be visual, audio or another form of the sensory input. The learning component is that the responsible for learning from data captured by the Perception comportment.

In the complete cycle, the main components are the knowledge representation and the Reasoning. These two components are involved in showing that of the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of that of the Knowledge representation and the reasoning.

Approaches to knowledge representation:

There are mainly four approaches to knowledge representation, which are given below:

1. Simple relational knowledge:

- It is the simplest way of that of the storing facts which uses that of the relational method and each fact about that of a set of the object is set out systematically in that of the columns.

- This approach of the knowledge representation is famous in the database systems where the relationship between that of the different entities is represented.

- This approach has little opportunity for that of the inference.

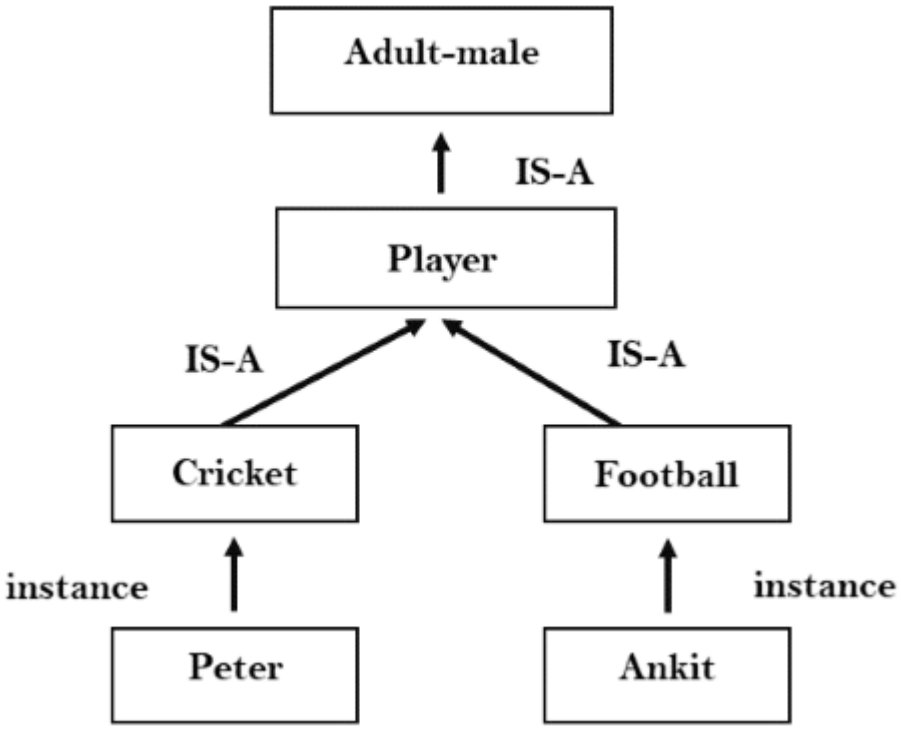
Example: The following is the simple relational knowledge representation.

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

2.Inheritable knowledge:

- In the inheritable knowledge approach, all the data must be stored into that of a hierarchy of the classes.
- All classes should be arranged in that of a generalized form or a hierarchal manner.
- In this approach, we apply the inheritance property.
- Elements inherit values from the other members of a class.
- This approach contains the inheritable knowledge which shows that of a relation between instance and the class, and it is known as instance relation.
- Every individual frame can represent that of the collection of attributes and its value.
- In this approach, objects and the values are represented in the Boxed nodes.
- We use Arrows which point from that of the objects to their values.

Example:



3. Inferential knowledge:

- Inferential knowledge approach represents the knowledge in the form of the formal logics.
- This approach can be used to derive more that of the facts.

- It guaranteed the correctness.
- **Example:** Let's suppose there are two statements:

1. Marcus is a man
2. All men are mortal

Then it can represent as;

Man(Marcus)

∀x = man (x) ----------> mortal (x)s

4. Procedural knowledge:

- Procedural knowledge approach uses that of the small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is **If-Then rule**.

- In this knowledge, we can use various coding languages such as **LISP language** and **Prolog language**.

- We can easily represent heuristic or domain-specific knowledge using this approach.

- But it is not necessary that we can represent all cases in this approach.

Requirements for knowledge Representation system:

A good knowledge representation system must possess the following properties.

1. **Representational Accuracy:**

KR system should have that of the ability to represent that the all kind of the required knowledge.

2. **Inferential Adequacy:**

KR system should have ability to manipulate that of the representational structures to produce that of the new knowledge corresponding to the existing structure.

3. **Inferential Efficiency:**

The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.

4. **Acquisitioned efficiency-** The ability to acquire the new knowledge easily using automatic methods.

Key takeaway

Humans are best at understanding, reasoning, and interpreting knowledge. Humans know things, which is that of the knowledge and as per that of their knowledge they perform several of the actions in that of the real world.

5.8 Ontological Engineering

This field of engineering describes;

How to make representations that are more broad and adaptable

• Actions, time, physical objects, and beliefs are examples of concepts.

• Works on a far larger scale than K.E.

Define general framework of concepts

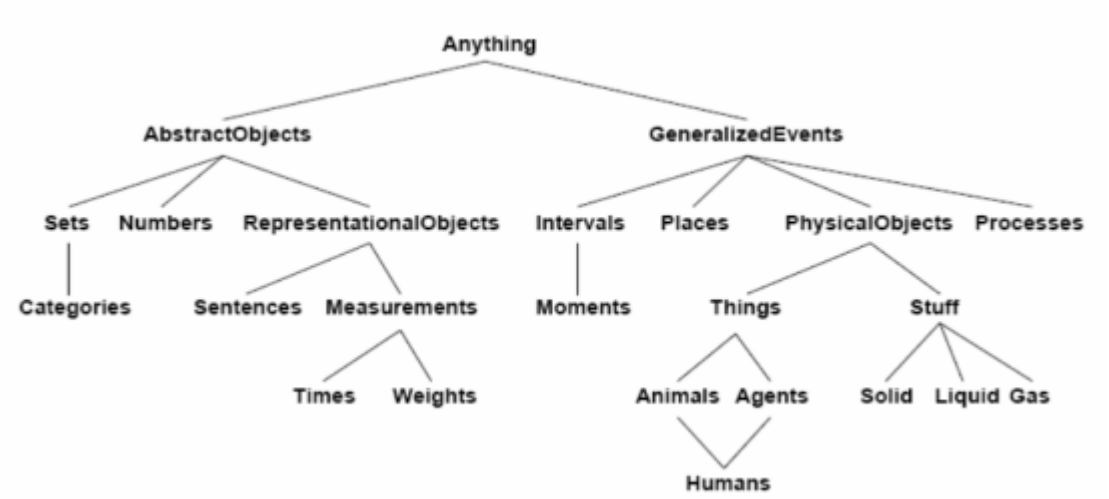
• Upper ontology

Limitations of logic representation

• Red, green and yellow tomatoes: exceptions and uncertainty.

Ontological engineering is a term used to describe the process of representing abstract concepts.

The upper ontology of the world



Difference with special-purpose ontologies

- A general-purpose ontology should work in pretty much any special-purpose domain.
  - Axioms that are domain-specific should be added.
- Different areas of knowledge must be unified in any sufficiently demanding domain.
  - Multiple areas of reasoning and problem solving may be involved at the same time.
- What do we have to say?
  - Physical Objects, Substances, Mental Objects, Beliefs, Categories, Measures, Composite Objects, Time, Space, Change, Events, Processes, Physical Objects, Substances, Mental Objects

5.9 Categories and Objects

- Sorting the items into categories.
- At the level of categories, some reasoning takes happen.
- "I'd like to eat an apple," says the narrator.
- Member(x, Apple) and Subset(x, Apple) are Apple(x) and Member(x, Apple) respectively (Apple, Food).
- The categories create a hierarchy, or simply a network, in which each class inherits the properties of the parent (apples are both food and fruit).
- A taxonomy is made up of the categories of a class.

The organisation of items into categories is required by KR.

- Interaction at the object level;
- Reasoning at the category level

Categories play a role in predictions about objects

- Based on perceived properties

FOL can represent categories in two ways (First Order Logic)

- Predicates: apple(x)
- Reification of categories into objects: apples

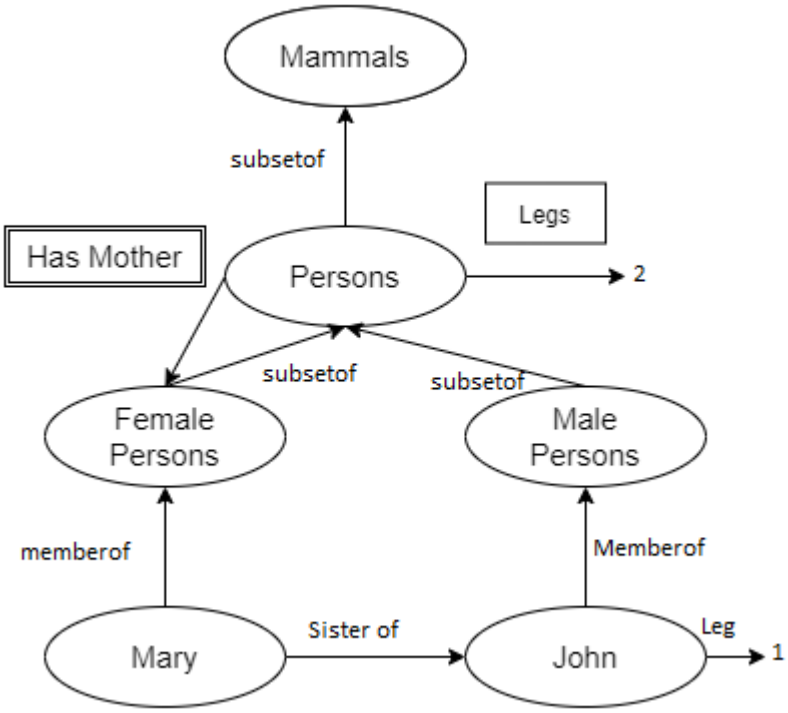
Category = set of its members.

Category organization

Relation = inheritance:

- Food is edible in all forms; fruit is a subclass of food, and apples are a subclass of fruit, hence an apple is edible.

Defines a taxonomy



FOL and categories

- An object is a member of a category
  - MemberOf(BB12,Basketballs)
- A category is a subclass of another category
  - SubsetOf(Basketballs,Balls)
- All members of a category have some properties
  - ∀x (MemberOf(x,Basketballs) ⇒Round(x))
- All members of a category can be recognized by some properties
  - ∀x (Orange(x) ∧Round(x) ∧Diameter(x) = 9.5in

∧MemberOf(x,Balls)⇒MemberOf(x,BasketBalls))

- A category as a whole has some properties
  - MemberOf(Dogs,DomesticatedSpecies)

Relations between categories

- If two or more categories have no members in common, they are disjoint:

- Disjoint(s)⇔(∀c1,c2 c1 ∈s ∧c2 ∈s ∧c1 ≠ c2 ⇒Intersection(c1,c2) ={})

- Example: Disjoint({animals, vegetables})

- If all members of the set are covered by categories in s, a set of categories s represents an exhaustive decomposition of a category c:

- E.D.(s,c) ⇔(∀i i∈c ⇒∃c2 c2∈s ∧i ∈c2)

- Example: ExhaustiveDecomposition({Americans, Canadian, Mexicans},

NorthAmericans).

- A partition is an exhaustive disjoint decomposition:

- Partition(s,c) ⇔Disjoint(s) ∧E.D.(s,c)

- Example: Partition({Males,Females},Persons).

- Is ({Americans,Canadian, Mexicans},NorthAmericans) a partition?

- No! There might be dual citizenships.

- By defining membership criteria that are both necessary and sufficient, categories can be created.

Substances and Objects

The real universe is made up of primitive items (such as atomic particles) and composite objects made up of them.

We can avoid the complexity of dealing with large numbers of primitive things separately by reasoning at the level of large objects like apples and cars.

However, there is a large part of reality that appears to defy any clear individuation—division into different objects.

b∈Butter ∧PartOf(p, b) ⇒p ∈Butter .

We may now say that butter melts at a temperature of roughly 30 degrees Celsius:

b∈Butter ⇒MeltingPoint(b,Centigrade(30))

What's going on is that some properties are intrinsic, meaning they pertain to the thing's very substance rather than the object as a whole.

When you cut a piece of stuffin in half, the fundamental properties—such as density, boiling temperature, flavour, colour, ownership, and so on—remain intact.

Extrinsic qualities like as weight, length, and shape, on the other hand, are lost during subdivision.

Stuff is the most broad substance type, with no inherent qualities specified.

Thing is the most broad discrete object category, with no extrinsic attributes specified.

5.10 Events

The usefulness of situation calculus is limited: it was created to represent a world in which activities are discrete, instantaneous, and occur one at a time.

Consider filling a bathtub, which is a continual action. Situation calculus can claim that the tub is empty before the action and filled after the action, but it can't describe what happens in the middle of the activity. It also can't depict two simultaneous actions, as cleaning one's teeth while waiting for the tub to fill. We present an alternative formalism known as event calculus to address such circumstances, which is based on points of time rather than situations.

Instances of event categories are used to describe events.4 Shankar's flight from San Francisco to Washington, D.C. Is described as follows:

E1 ∈ Flyings ∧ Flyer (E1, Shankar) ∧ Origin(E1, SF) ∧ Destination(E1,DC) .

Then we use Happens(E1, I to indicate that the event E1 occurred at the time interval i. A (start, finish) pair of times is used to express time intervals.

For one form of the event calculus, the entire collection of predicates is

T(f, t) Fluent f is true at time t

Happens(e, i) Event e happens over the time interval i

Initiates(e, f, t) Event e causes fluent f to start to hold at time t

Terminates(e, f, t) Event e causes fluent f to cease to hold at time t

Clipped(f, i) Fluent f ceases to be true at some point during time interval i

Restored (f, i) Fluent f becomes true sometime during time interval i

T is defined as a fluent that holds at a given point in time if it was started by an event in the past and was not made false (clipped) by an intervening event.

If an event terminates a fluent and it is not made true (restored) by another event, the fluent is no longer valid. The axioms are as follows:

$$\text{Happens}(e, (t1, t2)) \land \text{Initiates}(e, f, t1) \land \neg\text{Clipped}(f, (t1, t)) \land t1 < t \Rightarrow T(f, t)$$

$$\text{Happens}(e, (t1, t2)) \land \text{Terminates}(e, f, t1) \land \neg\text{Restored}(f, (t1, t)) \land t1 < t \Rightarrow \neg T(f, t)$$

Where Clipped and Restored are terms used to describe as:

$$\text{Clipped}(f, (t1, t2)) \Leftrightarrow \exists\, e, t, t3\ \text{Happens}(e, (t, t3)) \land t1 \leq t < t2 \land \text{Terminates}(e, f, t)$$

$$\text{Restored}(f, (t1, t2)) \Leftrightarrow \exists e, t, t3\ \text{Happens}(e, (t, t3)) \land t1 \leq t < t2 \land \text{Initiates}(e, f, t)$$

T can be extended to function over intervals as well as time points; for example, a fluent holds across an interval if it holds at every point within the interval:

$$T(f, (t1, t2)) \Leftrightarrow [\forall\, t\ (t1 \leq t < t2) \Rightarrow T(f, t)]$$

5.11 Mental Objects and Modal Logic

So far, the agents we've created have beliefs and can infer new ones. Despite this, none of them has any understanding of beliefs or deduction. Controlling inference requires understanding of one's own knowledge and reasoning processes.

Propositional attitudes - the way a person feels about mental objects (e.g. Believes, Knows, Wants, Intends, and Informs). These propositional attitudes do not behave in the same way that ordinary predicates do.

To say, for example, that Lois is aware of Superman's ability to fly:

Knows(Lois, CanFly(Superman))

The inferential rules conclude that "Lois knows Clark can fly" if "Superman is Clark Kent" is true (this is an example of referential transparency). Lois, on the other hand, has no idea that Clark can fly.

Superman = Clark

(Superman = Clark) and (Knows(Lois, CanFly(Superman)) ⊨ Knows(Lois, CanFly(Clark))

Referential transparency

- In whatever situation, an expression always yields the same outcome.
- For example, if an agent understands that 2+2=4 and 4<5, it should also know that 2+25.
- Integrated into inferential rules in most formal logic languages.

Referential opacity

- Transparent but not referential
- For propositional attitudes, we want referential opacity because terms matter, and not all agents are aware of which terms are co-referential.
- In most formal logic languages, this is not directly achievable (except Modal Logic).

Modal Logic

Created to allow for referential opacity in the knowledge base

Regular logic is concerned with a single modality (the truth modality), allowing us to say "P is true."

Modal logic comprises modal operators that use sentences as arguments rather than terms (e.g. "A knows P" is represented with notation KAP where K is the modal operator for knowledge, A is the agent, and P is a sentence).

Modal logic has the same syntax as first-order logic, with the exception that modal operators can also be used to build sentences.

Modal logic's semantics are more sophisticated. A model in first-order logic has a set of objects as well as an interpretation that translates each name to the correct item, relation, or function. We want to be able to consider both the possibility that Superman's secret identity is Clark and the chance that it isn't in modal logic. As a result, a model in modal logic is a collection of possible universes (instead of 1 true world). Accessibility relations connect the worlds in a graph (one relation for each modal operator). If everything in w1 is consistent with what A knows in w0, we say that world w1 is accessible from world w0 with respect to the modal operator KA, and we express this as Acc (KA, w0, w1).

As an arrow between possible universes, the figure depicts accessibility.

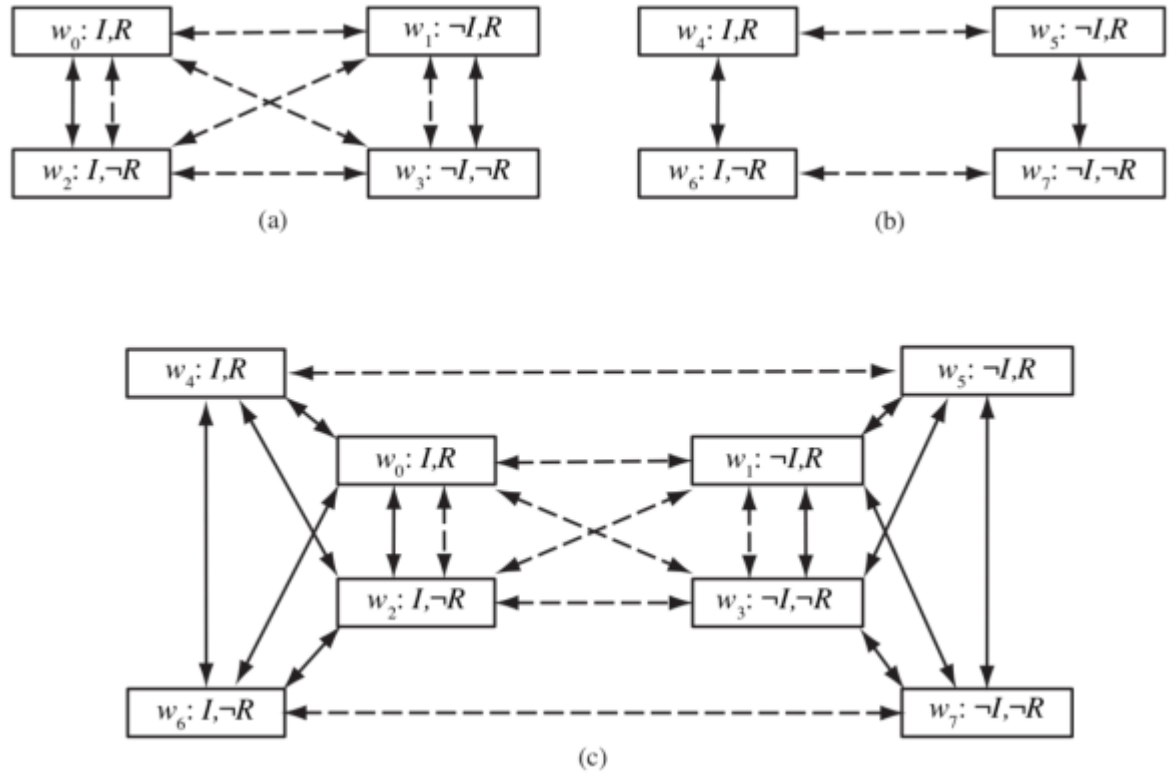


Fig: Possible worlds with accessibility relations $K_{Superman}$ and K_{Lois} .

An atom of knowledge If and only if P is true in every world accessible from w, KAP is true in world w. Recursive application of this rule with the standard laws of first-order logic yields the truth of more complex sentences. That is, modal logic can be used to reason about nested knowledge sentences, or what one agent knows about the knowledge of another agent.

We can claim, for example, that while Lois does not know if Superman's secret identity is Clark Kent, she does know that Clark does:

$\mathbf{K}_{Lois}$ [$\mathbf{K}_{Clark}$ Identity(Superman, Clark) ∨ $\mathbf{K}_{Clark}$ ¬Identity(Superman, Clark)]

The situation depicted in the TOP-LEFT of the picture is one in which Superman knows his own identity and neither he nor Lois has received the weather report. So, in w0, Superman has access to the worlds w0 and w2; rain may or may not be forecast.

For Lois, all four universes are accessible from one another; she has no idea what the report is about or whether Clark is Superman. But she is aware that Superman is aware of whether he is Clark or not, because in every reality accessible to Lois,

either Superman is aware of I or he is aware of ¬I. Lois isn't sure which is the case, but she knows Superman is aware of it.

The situation depicted in the top-right corner of the picture is one in which it is widely assumed that Lois has seen the weather report. So she knows it's going to rain in w4 and it's not going to rain in w6. Superman is unaware of the report, but he is aware that Lois is aware of it, because in every reality to which he has access, she either knows R or knows¬ R.

The situation depicted in the BOTTOM of the figure is one in which Superman's identity is widely known, and Lois may or may not have seen the weather report. We show this by combining the two top situations and adding arrows to show that Superman has no idea which one is correct. We don't need to draw any arrows for Lois because she already knows. In w0, Superman still recognises I but not R, and he has no idea whether Lois recognises R. According to what Superman knows, he might be in w0 or w2, in which case Lois is unsure whether R is true, w4, in which case she is aware of R, or w6, in which case she is aware of ¬R.

Modal Logic's Tricky Interplay of Quantifiers and Knowledge

The English line "Bond knows that someone is a spy," for example, is ambiguous.

"There is a certain someone that Bond knows is a spy," says the first reading:

$\exists x \, \mathbf{K}_{\text{Bond}} \text{Spy}(x)$

In modal logic, this indicates that there is an x that Bonds knows to be a spy in all accessible worlds.

The second interpretation is that "Bond just knows that at least one spy exists":

$\mathbf{K}_{\text{Bond}} \exists x \, \text{Spy}(x)$

In modal logic, this indicates that there is an x who is a spy in each accessible world, but it does not have to be the same x in each world.

Modal Logic - Writing Axioms With Modal Operators

State that agents can deduce; "if an agent knows P and recognises that P implies Q, then the agent knows Q."

$(\mathbf{K}_a P \wedge \mathbf{K}_a(P \Rightarrow Q)) \Rightarrow \mathbf{K}_a Q$

Declare that every agent is aware of the truth or falsity of every proposition P:

$\mathbf{K}_a(P \vee \neg P)$

True belief is justified knowledge:

$\mathbf{K}_a P \Rightarrow P$

If an agent is aware of something, they are aware that they are aware of it:

$\mathbf{K}_a P \Rightarrow \mathbf{K}_a(\mathbf{K}_a P)$

Similar axioms can be defined for belief (commonly abbreviated as B) and other modal operators.

One issue with modal logic is that it presupposes agents have logical omniscience (that is, if an agent knows a set of axioms, then it knows all consequences of those axioms).

5.12 Reasoning Systems for Categories

The main building blocks of large-scale knowledge representation schemes are categories.

There are two families of systems that are closely related:

Description logics provide a formal language for constructing and combining category definitions, as well as efficient algorithms for deciding subset and superset relationships between categories. Semantic networks provide graphical aids for visualising a knowledge base and efficient algorithms for inferring properties of an object based on its category membership; and semantic networks provide graphical aids for visualising a knowledge base and efficient algorithms for inferring properties of an object based on its category membership.

Semantic networks:

In 1909, Charles S. Peirce created existential graphs, a graphical notation of nodes and edges that he dubbed "the logic of the future."

- Semantic networks come in a variety of shapes and sizes, but they may all represent individual objects, categories of objects, and relationships between things.

- The names of objects or categories are displayed in ovals or boxes, and they are connected with labelled connections in a standard graphical format.

For example, Figure below has a Member Of link between Mary and Female Persons, corresponding to the logical assertion Mary ∈Female Persons; similarly, the Sister Of link between Mary and John corresponds to the assertion Sister Of (Mary, John). We can connect categories using Subset Of links, and so on.

It's easy to get carried away when painting bubbles and arrows.

Can we draw a Has Mother link from Persons to Female Persons, for example, because we know that persons have female persons as mothers?

Has Mother is a relationship between a person and his or her mother, and categories do not have mothers, hence the answer is no.

As a result, in Figure below, we've employed a specific notation—the double-boxed link.

According to this link,

$\forall x\ x \in \text{Persons} \Rightarrow [\forall\ y\ \text{HasMother}(x, y) \Rightarrow y \in \text{FemalePersons}]$.

We might also say that people have two legs—that is, that they have two legs.

$\forall x\ x \in \text{Persons} \Rightarrow \text{Legs}(x, 2)$.

The single-boxed link in Figure below is used to assert properties for each category member.

One of the key attractions of semantic networks has been the simplicity and effectiveness of this inference process when compared to logical theorem proving.

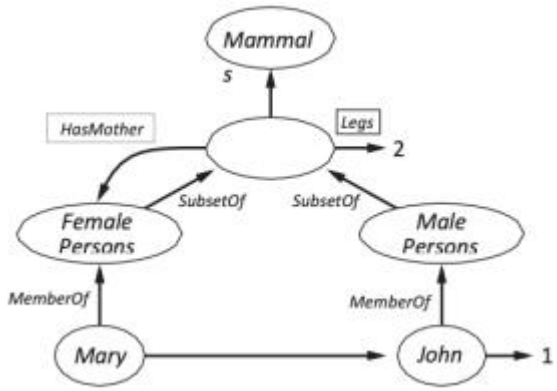


Fig: Semantic network with four objects and four categories. Relations are denoted by labeled links

Inverse linkages are another prominent method of inference. Has Sister, for example, is the inverse of Sister Of, implying that

When compared to first-order logic, the reader may have recognised an evident disadvantage of semantic network notation: the fact that linkages between bubbles only reflect binary relations.

In a semantic network, for example, the sentence Fly(Shankar, New York, New Delhi,Yesterday) cannot be affirmed explicitly.

Fig: Fragment of a semantic network showing the representation of the logical assertion Fly(Shankar, New York, New Delhi, Yesterday)

The capacity of semantic networks to represent default values for categories is one of its most essential features.

If you look closely at Figure, you'll find that John only has one leg, despite the fact that he's a person, and all people have two legs.

The default value is said to be overridden by the more particular value. We might alternatively override the default number of legs by adding a One Legged Persons category, which is a subset of Persons that includes John.

If we say that the Legs assertion for Persons includes an exception for John, we can keep the network's semantics purely logical:

∀x x∈ Persons ∧ x _= John ⇒ Legs(x, 2) .

This is semantically appropriate for a fixed network.

Description logics

First-order logic's syntax is intended to make it simple to declare things about objects.

Description logics are notations that make describing the definitions and properties of categories easier.

Subsumption (testing if one category is a subset of another by comparing their definitions) and classification are the two main inference tasks for description logics (checking whether an object belongs to a category)..

The consistency of a category definition—whether the membership criteria are logically satisfiable—is also considered in some systems.

$$
\begin{aligned}
Concept &\rightarrow \textbf{Thing} \mid ConceptName \\
&\mid \textbf{And}(Concept, \ldots) \\
&\mid \textbf{All}(RoleName, Concept) \\
&\mid \textbf{AtLeast}(Integer, RoleName) \\
&\mid \textbf{AtMost}(Integer, RoleName) \\
&\mid \textbf{Fills}(RoleName, IndividualName, \ldots) \\
&\mid \textbf{SameAs}(Path, Path) \\
&\mid \textbf{OneOf}(IndividualName, \ldots) \\
Path &\rightarrow [RoleName, \ldots]
\end{aligned}
$$

Fig: Syntax of description in sunset of CLASSIC language

A typical description logic is the CLASSIC language (Borgida et al., 1989). Figure depicts the syntax of CLASSIC descriptions.

To say that bachelors are unmarried adult males, for example, we would write

Bachelor = And(Unmarried, Adult ,Male) .

In first-order logic, the equivalent would be

Bachelor (x) ⇔ Unmarried(x) ∧ Adult(x) ∧ Male(x).

Any CLASSIC description can be translated into a first-order equivalent statement, however some CLASSIC descriptions are more clear.

For example, to characterise a group of males who have at least three unemployed sons who are all married to doctors, and at most two daughters who are all professors in physics or math departments, we would use

And(Man, AtLeast(3, Son), AtMost(2, Daughter),

All(Son, And(Unemployed,Married, All(Spouse, Doctor))),

All(Daughter , And(Professor , Fills(Department , Physics,Math)))) .

Translating this into first-order logic is left as an exercise.

Negation and disjunction, for example, are generally absent from description logics.

In the Fills and OneOf constructions, CLASSIC only provides for a restricted type of disjunction.

5.13 Reasoning with Default Information

Circumscription and default logic

When one sees a car parked on the street, for example, one is usually inclined to think it has four wheels despite the fact that only three are visible.

While probability theory can surely lead to the conclusion that the fourth wheel exists with a high probability, most people do not consider the idea that the car does not have four wheels unless new data is presented.

In the absence of any reason to doubt it, it appears that the four-wheel conclusion is reached by default. The conclusion can be retracted if fresh evidence is discovered—for example, if the owner is seen carrying a wheel and the car is jacked up.

Because the collection of beliefs does not develop monotonically over time when new evidence emerges, this type of reasoning is known as non monotonicity.

In order to capture such behaviour, non monotonic logics have been constructed with changed ideas of truth and entailment.

Circumscription and default logic are two such logics that have been extensively investigated.

The closed world assumption can be considered as a more powerful and exact variant of circumscription.

The goal is to define specific predicates that are supposed to be "as false as feasible," that is, false for all objects except those for which they are known to be true.

Let's say we wish to establish the default rule that birds fly.

We'd create a predicate, say Abnormal 1(x), and write it down.

Bird(x) ∧¬Abnormal 1(x) ⇒ Flies(x) .

If we say that Abnormal 1 is to be circumscribed, a circumscriptive reasoner is entitled to assume ¬Abnormal 1(x) unless Abnormal 1(x) is known to be true. This allows the conclusion Flies(Tweety) to be drawn from the premise Bird(Tweety).

Default logic is a formalism that allows for the creation of contingent, non-monotonic conclusions using default principles. The following is an example of a default rule:

Bird(x) : Flies(x)/Flies(x)

If Bird(x) is true and Flies(x) is compatible with the knowledge base, Flies(x) can be concluded by default. In most cases, a default rule looks like this:

P : J1, . . . , Jn/C

P stands for the precondition, C for the conclusion, and Ji for the justifications; if any of them is demonstrated to be untrue, the conclusion cannot be formed.

5.14 Case Study: The Amazing Ways How Wikipedia Uses Artificial Intelligence

Wikipedia is a free online encyclopaedia that was created by a group of volunteers known as Wikipedians. An article can be submitted for publication by anybody who has registered on the site; however, amending articles does not require registration. The website's name comes from wiki, a server tool that allows anyone to edit website content via their browser.

Wikipedia has opted to use artificial intelligence to learn more about the problem it is facing and to evaluate potential solutions.

Here are a few examples of how Wikipedia makes use of AI:

1. Wikipedia is a free, open-source online encyclopaedia that anyone may edit. By crowdsourcing the creation of an encyclopaedia, the non-profit website forever changed the way we get information. It is one of the top ten most popular websites on the internet. It is not, however, without flaws. Anyone with the ability to edit Wikipedia has the potential to add erroneous information unknowingly.

According to a Wired piece, computer scientist Aaron Halfaker recently began applying an AI system he built to detect vandalism and false updates on articles using machine learning — it can identify typical patterns in vandalous edits, such as a predisposition for poor letter spacing.

On the one hand, this implies less work for volunteers who monitor for suspicious changes, but Wikipedia anticipates a surge in new editors as a result of the shift.

It's all about lowering the barrier to entrance. To prevent vandalism, Wikipedia must develop strict criteria for who can make changes to essential documents because it relies on crowdsourcing for its content. On the other hand, it deters many people with good intentions and accurate information.

By having smarter AI identify defective things and alterations, it is hoped that more people would be committed to legitimate material. Wikipedia's rules for newcomers could be loosened a little once the police machines are smart enough. Some people, on the other hand, will be anxious about whether machines will ever be sophisticated enough to replace a large number of human editors.

2. To tackle the trolls, the Wikimedia Foundation collaborated on a research project called Detox with Jigsaw (the digital incubator formerly known as Google Ideas), which utilises machine learning to identify comments that may be personal attacks. Jigsaw's open-source AI effort to combat abuse on social networking sites and web forums includes this project.

3. A Google Brain team, according to Forbes, taught software how to summarise material from online pages and create Wikipedia-style articles. Text summarization out to be more difficult than most of us anticipated. Google Brain's efforts to get a computer to summarise text are marginally better than previous attempts, but there's still a long way to go before a machine can write with the same cadence and flair as humans. It turns out that we aren't quite ready to have a machine automatically generate Wikipedia entries, but progress is being made.

Because of its toxicity, even the Wikipedia community, a free encyclopaedia built on a paradigm of freely modifiable information, is well-known. The problem was so bad that over an eight-year period, the number of active editors or contributors (those who made you edit each month) plummeted from 40% to 20%. Despite the fact that there is no one solution to this problem, Wikimedia Foundation, the organisation that runs Wikipedia, elected to use artificial intelligence to learn more about it and examine possible solutions.

Collaboration with Wikimedia Foundation and Jigsaw to Stop Abusive Comments

Wikimedia Foundation teamed with Jigsaw (the technology incubator formerly known as Google Ideas) on a study project called Detox, which used machine learning to flag comments that could be personal assaults in an attempt to stop the

trolls. This project is part of Jigsaw's effort to build open-source AI systems to aid in the battle against abuse on social media platforms and forums.

The first step in the effort was to train the machine learning algorithms using 100,000 toxic comments from Wikipedia Talk pages, which were identified by a 4,000-person human team, with ten different human reviewers for each remark. This annotated data set was one of the most comprehensive studies of online abuse ever conducted. Not only were direct personal insults included, but so were third-party and indirect personal attacks as well. ("You're the worst.") "Bob is awful." ("Sally stated Bob is awful.") After some training, the robots can determine if a comment is a personal attack with the help of three individual moderators.

The project team then needed the algorithm to examine 63 million English Wikipedia comments submitted over a 14-year period from 2001 to 2015 in order to find similarities in the violent statements. The findings were detailed in the publication Ex Machina: Personal Attacks Discovered at Scale:

Rather than a small handful of trolls, over 9,000 people who made less than five violent remarks in a calendar year were responsible for more than 80% of all abusive comments.

Only 34 users were responsible for nearly 10% of all attacks.

Anonymous users accounted for 34% of all Wikipedia comments.

Although anonymous users are six times more likely to start personal assaults, documented users account for more than half of all personal attacks. (Registered users outnumber anonymous users by a factor of 20.)

Now that the algorithms have revealed who is contributing to the community's toxicity, Wikipedia may be able to figure out the best strategy to combat the hostility. Even though human moderation will almost certainly be required, algorithms can assist in sorting through the opinions and identifying those that require human intervention.

Objective Revision Evaluation Service (ORES System)

The company's extensive structure, as well as its demanding editing procedures, are seen to be another explanation for the dramatic fall of editors entering Wikipedia. It was very uncommon for first-time contributors/editors to have an entire body of work rejected for no apparent reason. The ORES system, a machine that works as an editing system and is powered by an algorithm trained to rate the quality of edits and changes, is one approach they aim to solve this predicament. Wikipedia editors used an online application to categorise instances of previous modifications, which is how the algorithm learned the severity of errors. Individuals can be directed to analyse the most detrimental edit and determine the grade of errors using the ORES system; beginner mistakes are treated more appropriately as innocent.

AI to Write Wikipedia Articles

Well, AI can write Wikipedia articles "OK," but you have to start somewhere, right? A Google Brain team teaches programmes how to summarise information from web pages and generate Wikipedia-style articles. It turns out that text summarization is more difficult than most of us anticipated. Google Brain's efforts to acquire a machine to outline content are a step forward from past attempts, but there is more work to be done before a machine can create with the same cadence and flair that humans can. It turns out that we aren't yet ready to have a machine generate Wikipedia entries, but progress is being made.

Objective Revision Evaluation Service (ORES System)

The company's extensive structure, as well as its demanding editing procedures, are seen to be another explanation for the dramatic fall of editors entering Wikipedia. It was very uncommon for first-time contributors/editors to have an entire body of work rejected for no apparent reason. The ORES system, a machine that works as an editing system and is powered by an algorithm trained to rate the quality of edits and changes, is one approach they aim to solve this predicament. Wikipedia editors used an online application to categorise instances of previous modifications, which is how the algorithm learned the severity of errors. Individuals can be directed to analyse the most detrimental edit and determine the grade of errors using the ORES system; beginner mistakes are treated more appropriately as innocent.

Unit - 6

Planning

6.1 Automated Planning

The field of Artificial Intelligence known as Automated Planning investigates this discussion process computationally. Its goal is to aid planning by reasoning on conceptual models, which are abstract and formal representations of the domain, the effects and combinations of activities, and the requirements to be met and objectives to be met. The planning domain is the conceptual model of the domain in which activities are carried out, plans are combinations of actions, and goals are the needs to be met. Intuitively, a planning problem entails determining a plan that satisfies the goal in a specific domain, given a planning domain and a goal. For Automated Planning, we give a broad formal framework.

Domains, plans, and goals are the three fundamental components of the planning problem that the framework is built upon.

Domain - We allow for nondeterministic domains, which are domains in which actions may have varied effects and it is impossible to predict which of the various possible outcomes will actually occur at the time of planning. We also take partial observability into account. It represents the reality that the state of the domain cannot always be observed completely, and so cannot be determined uniquely in some instances. The special circumstances of full observability, where the state may be totally observed and hence uniquely determined, and null observability, where no observation is ever feasible at run time, are both included in a model with partial observability.

Plans - We define plans in which the action to be performed in a particular state may be influenced by information about prior execution phases. Sequential plans, i.e., plans that are just sequences of activities, conditional plans, i.e., plans that can pick a different action depending on the current circumstance at execution time, and iterative plans that can run actions until a situation happens are all covered by the definition. There are plans that rely on a finite number of execution steps (finite-memory plans) and plans that do not rely on prior execution steps (non-memory plans) (memory-less plans). Plan executions, in general, result in trees (called execution trees) whose nodes correspond to domain states.

Goals - Goals are defined as sets of acceptable trees that correspond to intended planning domain evolutions. They can be used to represent traditional reachability goals, which are requirements expressed on the leaves of execution trees that specify the end states to be achieved once a plan is performed. They can also represent more complicated types of "extended objectives," such as temporally extended goals, that convey conditions across the entire execution tree.

Our framework is broad enough to encompass a wide range of relevant and important planning issues. Deterministic domains, plans that are sequences of actions, and reachability goals can all be used to mimic traditional planning. Furthermore, our framework is ideally adapted for modeling specific types of planning under uncertainty and imperfect knowledge, which has recently been addressed in the scientific literature and is relevant to a number of real-world applications. Nondeterministic domains do, in fact, model uncertainty in action effects, whereas partial observability does so for observations.

Nondeterministic domains, null observability, sequential plans, and reachability goals, for example, can be used to model conformant

planning. Nondeterministic domains, conditional plans, and reachability goals can all be used to model contingent planning.

Nondeterministic domains, history-dependent plans, and objectives that describe desirable domain evolutions can all be used to simulate planning for temporally extended goals.

For practical reasons, the framework cannot be too broad to encompass all of the various planning issues that have been discussed in the literature thus far. For example, we do not represent probabilities of action outcomes in action domains, and goals are represented as utility functions, which is a difference from planning based on Markov Decision Processes (MDP).

Key takeaway

The field of Artificial Intelligence known as Automated Planning investigates this discussion process computationally. Its goal is to aid planning by reasoning on conceptual models, which are abstract and formal representations of the domain, the effects and combinations of activities, and the requirements to be met and objectives to be met.

6.2 Classical Planning

Classical planning is the process of constructing complicated plans of action using the issue structure as a guide.

Invariants are used in optimum planning (graph theory, complexity theory).

Domain-specific solvers are integrated to improve the efficiency of classical planning on certain domains (puzzles, logistics, video games).

Machine learning approaches are being used in the planning and extraction of structural data (machine learning, deep learning, neural networks).

Optimize your strategy.

In traditional planning, the agent performs three tasks:

- planning: After determining the problem, the agent makes a plan.
- Acting: It chooses the course of action to adopt.
- Learning: The agent's actions cause him to learn new things.

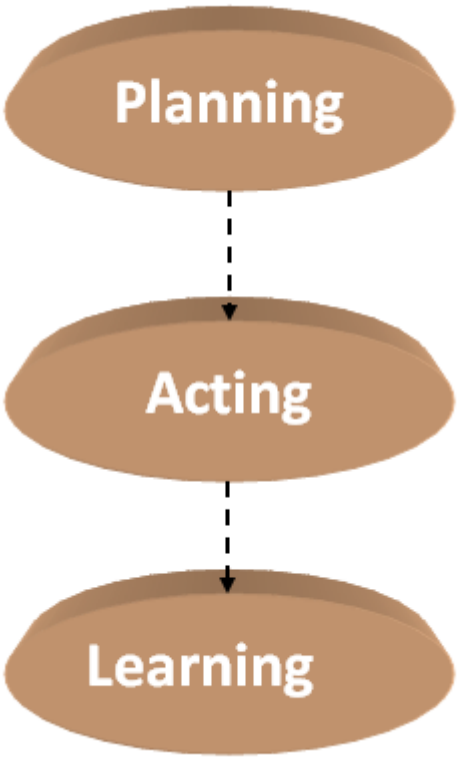


Fig 1: Agent task

PDDL (Planning Domain Definition Language) is a language that is used to represent all actions in a single action schema.

The four basic things required in a search problem are described by PDLL:

- Initial state - Each state is represented as the union of the ground and functionless atoms in its initial state.
- Action - It is defined by a series of action schemas that define the ACTION() and RESULT() functions implicitly.
- Result - The collection of activities taken by the agent yielded this result.

- Goal - The goal is the same as a precondition, which is a literal conjunction (whose value is either positive or negative).

There are various examples which will make PDLL understandable:

- Air cargo transport
- The spare tire problem
- The blocks world and many more.

Advantages of Classical Planning

Classical planning has the following benefits:

- It has enabled the development of precise domain-independent heuristics.
- The systems are simple to use and operate efficiently.

Complexity of the classical planning

There are two decision difficulties that arise in traditional planning:

- PlanSAT: is a query that asks if any plan exists that solves a planning difficulty.
- Bounded PlanSAT: It's a question that asks if there's a solution with a length of k or less.

Key takeaway

Classical planning is the process of constructing complicated plans of action using the issue structure as a guide.

6.3 Algorithms for Classical Planning

Now we'll take a look at planning algorithms. We saw how the description of a planning problem defines a search problem: we can search through the space of states, looking for a goal, starting from the initial state. One of the wonderful features of the declarative representation of action schemas is that we can search for the initial state backwards from the objective.

Forward (progression) state-space search

Now that we've seen how a planning problem maps to a search problem, we may utilize any of the heuristic search algorithms or a local search algorithm to solve planning problems (provided we keep track of the actions used to reach the goal). Forward state-space search was thought to be too inefficient to be feasible from the beginning of planning research (about 1961) until roughly 1998. It's not difficult to think of reasons for this.

For starters, forward search is prone to looking into behaviors that aren't relevant. Take, for example, the heroic undertaking of purchasing a copy of AI: A Modern Approach from an internet bookseller. Assume there is a Buy(isbn) action schema with the effect Own (isbn). Because ISBNs have ten digits, this action schema corresponds to ten billion ground actions. To locate one that leads to the goal, an uninformed forward-search algorithm would have to start enumerating these 10 billion activities.

Second, state spaces in planning issues are frequently huge. Consider a cargo scenario with ten airports, each having five planes and 20 items of goods. The objective is to transport all of the cargo from airport A to airport B. The problem has a straightforward solution: load the cargo into one of the planes at A, fly the plane to B, and discharge the cargo.

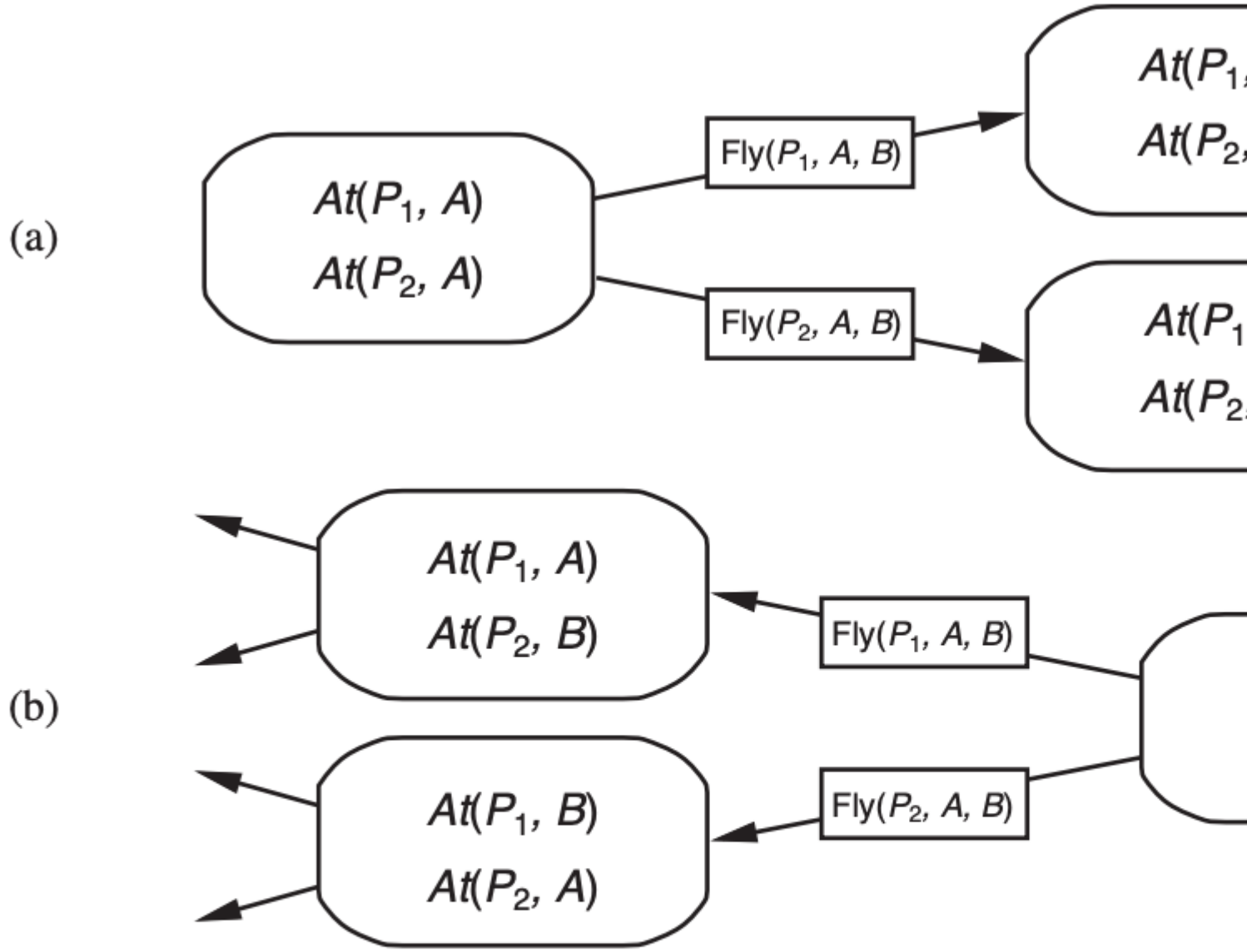


Figure 10.5 Two approaches to searching for a plan. (a) Forward (
through the space of states, starting in the initial state and using the p
search forward for a member of the set of goal states. (b) Backward
through sets of relevant states, starting at the set of states representing th
inverse of the actions to search backward for the initial state.

Fig 2: Two approaches to searching for a plan. (a) forward search (b) backward search

The objective is to transport all of the cargo from airport A to airport B. The problem has a straightforward solution: load the cargo into one of the planes at A, fly the plane to B, and discharge the cargo. The average branching factor is high, therefore finding the answer can be difficult: each of the 50 planes can travel to 9 different airports, and each of the 200 packages can be unloaded (if loaded) or loaded into any plane at its airport (if it is unloaded). So there are a minimum of 450 activities (when all the packages are at airports with no planes) and a maximum of 10,450 actions in any situation (when all packages and planes are at the same airport).

Let's say there are roughly 2000 potential actions per state on average, therefore the search graph has about 200041 nodes up to the depth of the obvious solution.

Backward (regression) relevant-states search

In regression search, we begin at the goal and work backwards until we identify a set of steps that leads to the original state. Because we only evaluate activities that are related to the objective, it's termed relevant-states search (or current state). At each phase, just like in belief-state search, there are a number of relevant states to consider, not just one.

We begin with the goal, which is a set of literals that describes a set of states—for example, the goal PoorFamous describes states in which Poor is false, Famous is true, and any other fluent can have any value. There are 2^n ground states (each fluent might be true or false) but 3^n descriptions of sets of target states in a domain with n ground fluents (each fluent can be positive, negative, or not mentioned).

6.4 Heuristics for Planning

Without a good heuristic function, neither forward nor backward search is efficient. Remember that a heuristic function h(s) measures the distance between a state s and the goal, and that we may utilize A search to identify optimal solutions if we can construct an admissible heuristic for this distance—one that does not overestimate. By establishing a relaxed issue that is easier to solve, an admissible

heuristic can be generated. The heuristic for the original problem is then the exact cost of a solution to this easier problem.

Because there is no mechanism to analyze an atomic state by definition, it takes some ingenuity on the part of a human analyst to come up with suitable domain-specific heuristics for atomic state search problems. For states and action schemas, planning employs a factored representation. This allows for the definition of good domain-independent heuristics and the automatic use of a good domain - independent heuristic for a given problem by programmes.

Consider a search issue as a network with nodes representing states and edges representing operations. The goal is to find a path from the original state to the desired state. We may make this problem easier in two ways: by adding more edges to the graph, making it stricter to identify a path, or by grouping several nodes together, generating an abstraction of the state space with fewer states, making it easier to search.

First, we'll look at heuristics for adding edges to the graph. The disregard preconditions heuristic, for example, removes all preconditions from actions. Every action becomes relevant in every stage, and any single goal can be accomplished in a single step (assuming an applicable action exists—if not, the problem is impossible to solve). This almost indicates that the number of steps required to solve the relaxed problem is equal to the number of unmet goals—almost, but not quite, because some acts may accomplish several goals and others may negate the results of others. Consider and ignore is a good way to get an accurate heuristic for many problems.

First, we loosen up the actions by removing all preconditions and consequences that aren't literals in the goal. Then we count the least number of activities required to achieve the goal when the impacts of those actions are added together. This is a case of the set-cover problem in action. One small annoyance is that the set-cover problem is NP-hard. Thankfully, a basic greedy algorithm will always provide a set covering that is within a factor of log n of the true lowest covering, where n is the number of literals in the target. Unfortunately, the greedy algorithm loses its admissibility guarantee.

It's also feasible to ignore only some action preconditions. Consider the puzzle of sliding blocks (8-puzzle or 15-puzzle). This might be expressed as a tile-based planning problem with a single schema. Slide:

Action(Slide(t,s1,s2),

PRECOND:On(t,s1) ∧ Tile(t) ∧ Blank (s2) ∧ Adjacent(s1,s2)

EFFECT:On(t,s2) ∧ Blank(s1) ∧ ¬On(t,s1) ∧ ¬Blank(s2))

When the preconditions Blank (s2) ^ Adjacent(s1,s2) are removed, every tile can move to any space in one action, yielding the number-of-misplaced-tiles heuristic. The Manhattan-distance heuristic is obtained by removing Blank(s2). These heuristics might easily be deduced automatically from the action schema description.

The factored form of planning issues has a significant advantage over the atomic representation of search problems in terms of ease of manipulation of the schemas.

The ignore remove listings heuristic is another option. Assume that all objectives and preconditions include only positive literals3 for a moment. We want to make a more relaxed version of the original issue that is easier to solve and uses the length of the solution as a heuristic. This can be accomplished by deleting delete lists from all actions (i.e., removing all negative literals from effects). This allows for consistent progress toward the goal, as no action will ever negate progress achieved by another. The optimal solution to this relaxed problem is still NP - hard to discover, but an approximate solution can be achieved in polynomial time through hill-climbing.

Decomposition is an essential concept in defining heuristics: breaking a problem into parts, solving each portion separately, and then merging the parts. The cost of achieving a conjunction of subgoals is approximated by the total of the costs of solving each subgoal independently, according to the subgoal independence assumption. The assumption of subgoal independence might be either optimistic or pessimistic. When there are negative interactions between the sub plans for each subgoal, such as when an action in one sub plan deletes a goal reached by another sub plan, it is optimistic. When sub plans contain redundant actions—for example, two actions that might be replaced by a single action in the merged plan—it is gloomy, and thus inadmissible.

Key takeaway

Without a good heuristic function, neither forward nor backward search is efficient. Remember that a heuristic function h(s) measures the distance between a state s and the goal, and that we may utilize A search to identify optimal solutions if we can construct an admissible heuristic for this distance—one that does not overestimate. By establishing a relaxed issue that is easier to solve, an admissible heuristic can be generated. The heuristic for the original problem is then the exact cost of a solution to this easier problem.

6.5 Hierarchical Planning

When planning, it's common to have hierarchical information on the actions, or a description of how complex actions are broken down. For instance, a complicated activity such as "serving coffee" can be broken down into two complex actions: "making coffee" and "bringing coffee." As a result, there exist planners, such as ABSTRIPS, that take as input the hierarchical description of the activities in addition to the description of the actions. For example, one can start planning at a general level and then drill down into the details if necessary (as does ABSTRIPS for example). A hierarchical task network is then used to express the goal (HTN).

- Hierarchical planning, often known as HTN planning, is a planning system based on the Hierarchical Task Network (HTN).

- It combines Partial Order Planning and HTN Planning concepts.

- HTN planning is frequently organized around a single "top-level" action termed Act, with the purpose of finding an Act execution that meets the goal.

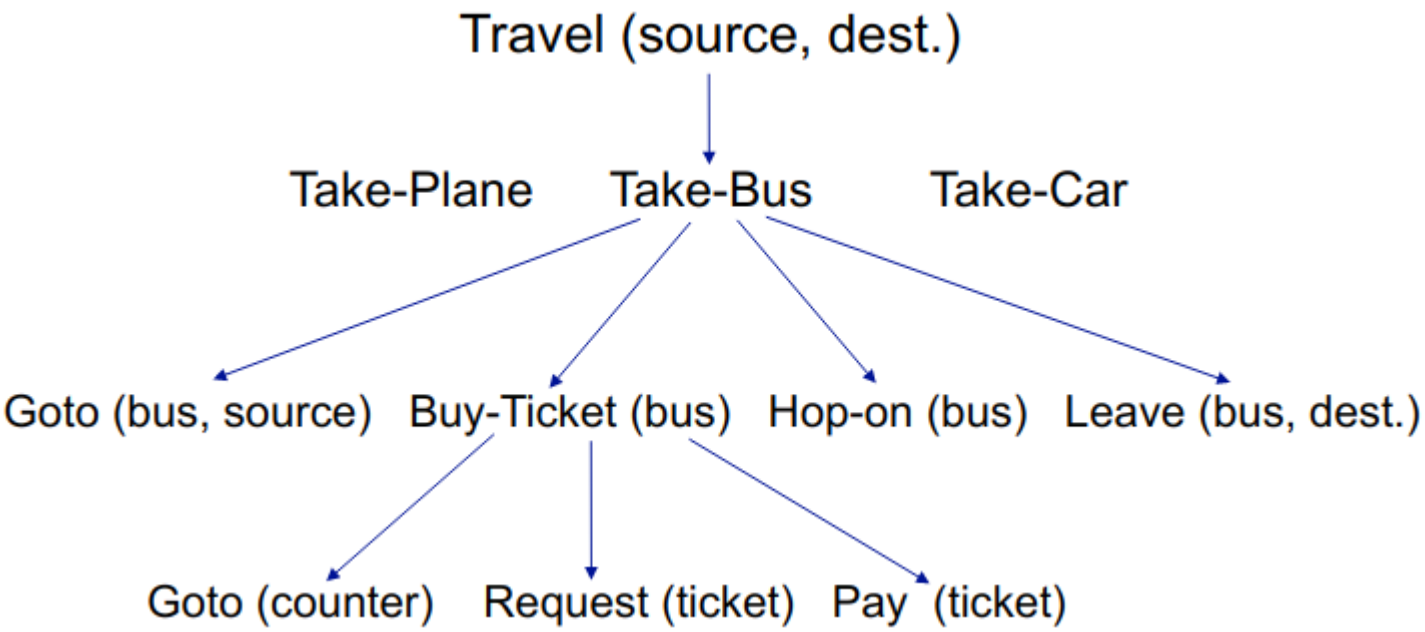
- The first plan is considered as a high-level statement of what needs to be done in HTN planning.

- Decomposition actions are used to refine this strategy.

- Each action decomposition breaks down a higher-level activity into a jumble of lower-level actions.

- This deconstruction continues until the plan is reduced to its most basic actions.

- Consider a hierarchical travel plan from a specific source to a specific destination.



- Assume we're traveling from source "Mumbai" to destination "Goa" in the above hierarchical planner diagram.

- Then you can decide how you want to travel: by plane, bus, or car. Assume you decide to travel by "Bus."

- The "Take-Bus" strategy can then be broken down into a series of tasks, such as going to Mumbai – Bus stop, purchasing a bus ticket, boarding the bus, and departing for Goa.

- The four acts in the previous point can now be split down into their component parts. Take, for example, "By-Ticket for Bus."

- It may be broken down into three steps: go to the bus stop counter, request a ticket, and pay for the ticket.

- As a result, each of these actions can be further deconstructed until we reach the level of actions that can be performed without thought in order to generate the needed motor control sequences.

Advantages of Hierarchical Planning:

- The main advantage of a hierarchical structure is that it reduces the plan at each level of the hierarchy to a small number of activities at the next lower level, lowering the computing cost of determining the best method to organise those activities for the current challenge.

- Many real-world applications demand very big plans, which HTN algorithms may generate.

- In the event that something goes wrong, the hierarchical structure makes it simple to rectify the problem.

- Hierarchical planning is far more efficient than single-level planning for complex issues.

Disadvantages of Hierarchical Planning:

- A Deterministic environment is required by many HTN planners.

- Many HTN planners are unable to deal with unpredictable results of actions.

Key takeaway

When planning, it's common to have hierarchical information on the actions, or a description of how complex actions are broken down.

6.6 Planning and Acting in Nondeterministic Domains, Time, Schedules, and Resources

Domain

We allow for nondeterministic domains, which are domains in which actions may have varied effects and it is impossible to predict which of the various possible

outcomes will actually occur at the time of planning. We also take partial observability into account. It represents the reality that the state of the domain cannot always be observed completely, and so cannot be determined uniquely in some instances. The special circumstances of full observability, where the state may be totally observed and hence uniquely determined, and null observability, where no observation is ever feasible at run time, are both included in a model with partial observability.

Time, Schedule and Resources

The standard planning representation can communicate about what to do and in what order, but it can't talk about how to execute it.

- **Temporal ordering constraint** - When an activity should occur (before and/or after a particular time and/or specific action(s)), there are temporal ordering constraints.

- **Resource constraints** - outlines the resources required to carry out a task.

Representing Temporal Constraints, Resource Constraints, and the Available Resources

Each action is represented by:

- A period of time

- A collection of temporal ordering constraints (actions that must be performed before this action may be carried out)

- A set of limits on resources

Each resource is represented by 3 things:

- The resource's kind (e.g. Bolts, wrenches, or pilots).

- The quantity of that resource at the start.

- Whether or not that resource:
  - Consumable - e.g. The bolts are no longer available for use
  - Reusable - e.g. a pilot is occupied during a flight but is available again when the flight is over
  - Sharable

Actions have the ability to generate resources.

A solution must meet all of the activities' and resources' temporal ordering constraints.

Example of a Scheduling Issue (Assembly of 2 Cars).

The problem of assembling two cars consists of the following steps:

- Resources: There are four different sorts of resources, with a certain number of each type available at the start:

Resource type	Number available	Consumable or reusable
Engine hoist	1	Reusable
Wheel station	1	Reusable
Lug nuts	500	Consumable
Inspector	2	Reusable

- Ordering & resource constraints: 2 jobs, each of form [AddEngine, AddWheels, Inspect] and their individual action durations and its resource constraints

Action	Duration units	Resources constraints
AddEngine1	30	1 engine hoist
AddEngine2	60	1 engine hoist
AddWheels1	30	1 wheel station and 40 lug nuts
AddWheels2	15	1 wheel station and 40 lug nuts
Inspect1	10	1 inspector
Inspect2	10	1 inspector

Key takeaway

We allow for nondeterministic domains, which are domains in which actions may have varied effects and it is impossible to predict which of the various possible outcomes will actually occur at the time of planning. We also take partial observability into account.

6.7 Analysis of Planning Approaches

Planning brings together the two key AI disciplines we've discussed so far: search and logic. A planner can be thought of as either a programme that looks for a solution or one that (constructively) argues that one exists. The cross-fertilization of ideas from the two fields has resulted in performance gains of several orders of magnitude over the last decade, as well as a rise in the usage of planners in industrial applications. Regrettably, we do not yet have a firm grasp of which strategies are most effective in particular situations. It's possible that new procedures will arise that will supplant old ones.

Controlling combinatorial explosion is at the heart of planning. There are 2n states in a domain if there are n propositions. As we've seen, planning is difficult. Identifying independent subproblems can be a powerful tool against such pessimism. We gain an exponential speedup in the best case—full decomposability of the problem. Negative interactions between acts, on the other hand, destroy decomposability. GRAPHPLAN keeps track of mutexes to help you figure out where the tricky interactions occur.

SATPLAN represents mutex relations in a similar way, but instead of employing a specialized data structure, it uses the universal CNF form. Forward search tries to uncover patterns (subsets of propositions) that cover all of the independent subproblems heuristically. Because this is a heuristic approach, it can be used even if the subproblems are not totally independent.

It is sometimes possible to efficiently address an issue by understanding that unfavorable interactions can be ruled out. If there is a SERIALIZABLE SUBGOAL an order of subgoals such that the planner can achieve them in that order without having to reverse any of the previously achieved subgoals, we say the problem has serializable subgoals. If the goal is to build a tower (e.g., A on B, which is on C, which is on Table), then the subgoals are serializable bottom to top: if we first achieve C on Table, we will never have to undo it while achieving the other subgoals. A planner who employs the bottom-to-top strategy can solve any problem in the world of blocks without having to go backwards (although it might not always find the shortest plan).

For a more complex example, it was determined that the propositions involved in commanding NASA's Deep Space One spacecraft are serializable for the Remote Agent planner that commanded NASA's Deep Space One spacecraft. This is perhaps unsurprising, given that engineers construct spaceships to be as simple to manage as possible (subject to other constraints). The Remote Agent planner

was able to eliminate the majority of the search by taking use of the serialized ordering of goals. This meant it could control the spacecraft in real time, which had previously been thought impossible.

GRAPHPLAN, SATPLAN, and FF are examples of planners who have advanced the science of planning by improving the performance of planning systems, explaining the representational and combinatorial challenges involved, and developing useful heuristics. However, it is unclear how far these procedures can be scaled. Further progress on larger issues is likely to require some type of synthesis of first-order and hierarchical representations with the efficient techniques already in use, rather than relying solely on factored and propositional representations.

Key takeaway

Planning brings together the two key AI disciplines we've discussed so far: search and logic. A planner can be thought of as either a programme that looks for a solution or one that (constructively) argues that one exists.

6.8 Limits of AI

The availability of data is one of the most significant obstacles to AI implementation. Data is frequently fragmented, inconsistent, and of low quality, posing hurdles for firms attempting to leverage AI at scale. To avoid this, you should have a defined approach for obtaining the data that your AI will require from the start.

Another major impediment to AI adoption is a skills scarcity and a lack of technical professionals with the appropriate knowledge and training to deploy and run AI technologies. According to research, experienced data scientists, as well as other specialized data professionals competent in machine learning, training effective models, and so on, are in short supply.

Another important factor to consider when purchasing AI technology is the price. Businesses that lack in-house capabilities or are unfamiliar with AI are frequently forced to outsource, posing cost and maintenance issues. Smart technologies can

be costly due to their complexity, and you may pay additional fees for repair and continuous maintenance. Computational costs for training data models, for example, can be an added cost.

Software programmes must be updated on a regular basis to keep up with the changing business environment, and if they fail, there is a risk of losing code or essential data. It can take a long time and be expensive to restore this. This danger, however, is no higher with AI than with any other software development. These dangers can be reduced if the system is well-designed and people purchasing AI are aware of their needs and options.

Cost and maintenance - Purchase costs can be high, and there may be a need for ongoing maintenance and repair, as with any new technology. In order to react to the ever-changing business environment, your AI software will need to be updated on a regular basis. Before you go forward and adopt any AI system, your organization should thoroughly analyze the return on investment.

Lack of creativity - A good marketing effort requires a high level of creativity. Machines just do not possess the ability to think creatively. Humans, unlike robots, have the ability to think and feel, which influences their decision-making when it comes to creativity. Yes, AI can help estimate what type of picture, for example, a consumer is likely to click on - everything from colour preferences to style and price. A machine, on the other hand, cannot compete with the human brain when it comes to originality and creative thinking. Both humans and machines are still required.

Other AI limitations include:

- Implementation times, which can vary based on what you're trying to accomplish.
- Integration issues and a lack of knowledge about cutting-edge technology.
- Interoperability with various systems and platforms, as well as usability.

Key takeaway

The availability of data is one of the most significant obstacles to AI implementation. Data is frequently fragmented, inconsistent, and of low quality, posing hurdles for firms attempting to leverage AI at scale.

6.9 Ethics of AI

Artificial Intelligence ethics, often known as AI ethics, is a set of beliefs, ideas, and methodologies that use commonly accepted criteria of right and wrong to govern moral behavior in the development and deployment of AI systems.

In recent years, the ethics of AI and robotics have gotten a lot of press attention, which helps support related research but also risks undermining it: the press frequently talks as if the issues under discussion are just predictions of what future technology will bring, and as if we already know what would be most ethical and how to get there.

The ethics of AI and robotics are frequently centered on various "concerns," which is a common reaction to new technologies. Many of these concerns turn out to be rather quaint (trains are too fast for souls); some are predictably wrong when they claim that technology will fundamentally change humans (telephones will destroy personal communication, writing will destroy memory, video cassettes will render going out obsolete); some are broadly correct but moderately relevant (digital technology will destroy industries that make photographic film, cassette tapes, or vinyl records); and some are egregiously wrong when they claim that technology will fundamentally change humans (telephones will destroy personal (cars will kill children and fundamentally change the landscape). The purpose of a piece like this is to dissect the concerns and deflate the non-issues.

Some technologies, such as nuclear power, automobiles, and plastics, have sparked ethical and political debate as well as considerable regulatory initiatives to limit their trajectory, usually after some damage has been done. New technologies, in addition to "ethical issues," challenge present norms and conceptual frameworks, which is of particular interest to philosophy. Finally, once we've grasped the context of a technology, we must shape our societal reaction, which includes regulation and law. All of these characteristics are present in

modern AI and robotics technologies, as well as the more fundamental worry that they will usher in the end of the period of human control on Earth.

Though there is a promising outline (European Group on Ethics in Science and New Technologies 2018) and there are beginnings on societal impact (Floridi et al. 2018; Taddeo and Floridi 2018; S. Taylor et al. 2018; Walsh 2018; Bryson 2019; Gibert 2019; Whittlestone et al. 2019), and policy recommendations, the ethics of AI and robotics is a very young field within applied ethics, with significant dynamics but few well-established issues (AI HLEG 2019 [OIR]; IEEE 2019). As a result, this page must not only repeat what the community has already accomplished, but also offer an order where none exists.

A variety of ethical challenges have arisen as a result of AI's rapid growth. These are some of them:

- The potential for employment loss as a result of automation technologies
- Employees must be redeployed or retrained in order to keep their positions.
- Machine-created wealth should be distributed fairly.
- The impact of human-machine interaction on human attention and behaviour
- The necessity to address algorithmic bias in data resulting from human bias
- The safety of AI systems that have the capacity to cause harm (e.g. Autonomous weapons).
- Because smart machines are considered to learn and develop independently, there is a need to guard against unforeseen outcomes.

6.10 Future of AI

Artificial Intelligence (AI) is a breakthrough discipline of computer science that is poised to become a key component of several future technologies such as big data, robotics, and the Internet of Things (IoT). In the coming years, it will continue to be a technical trailblazer. AI has gone from science fiction to reality in a few of years. Machines that assist people with intelligence can be found in the real world as well

as in science fiction films. We now live in a world of Artificial Intelligence, which was only a story a few years ago.

We use AI technology in our daily lives, whether consciously or unconsciously, and it has become a part of our lives. Everyone uses AI in their day-to-day lives, from Alexa/Siri to Chatbots. The advancement and evolution of this technology is occurring at a breakneck speed. It was not, however, as smooth and simple as it appeared to us. It has taken several years, as well as a lot of hard effort and contributions from a variety of people, to get AI to this point.

Because AI is such a revolutionary technology, there are numerous questions concerning its future and impact on humans. It's risky, but it's also a fantastic opportunity. Artificial intelligence will be used to improve both defensive and offensive cyber operations. In addition, new cyber-attack methods will be developed to exploit AI technology's specific vulnerabilities.

Future impact of AI in different sectors

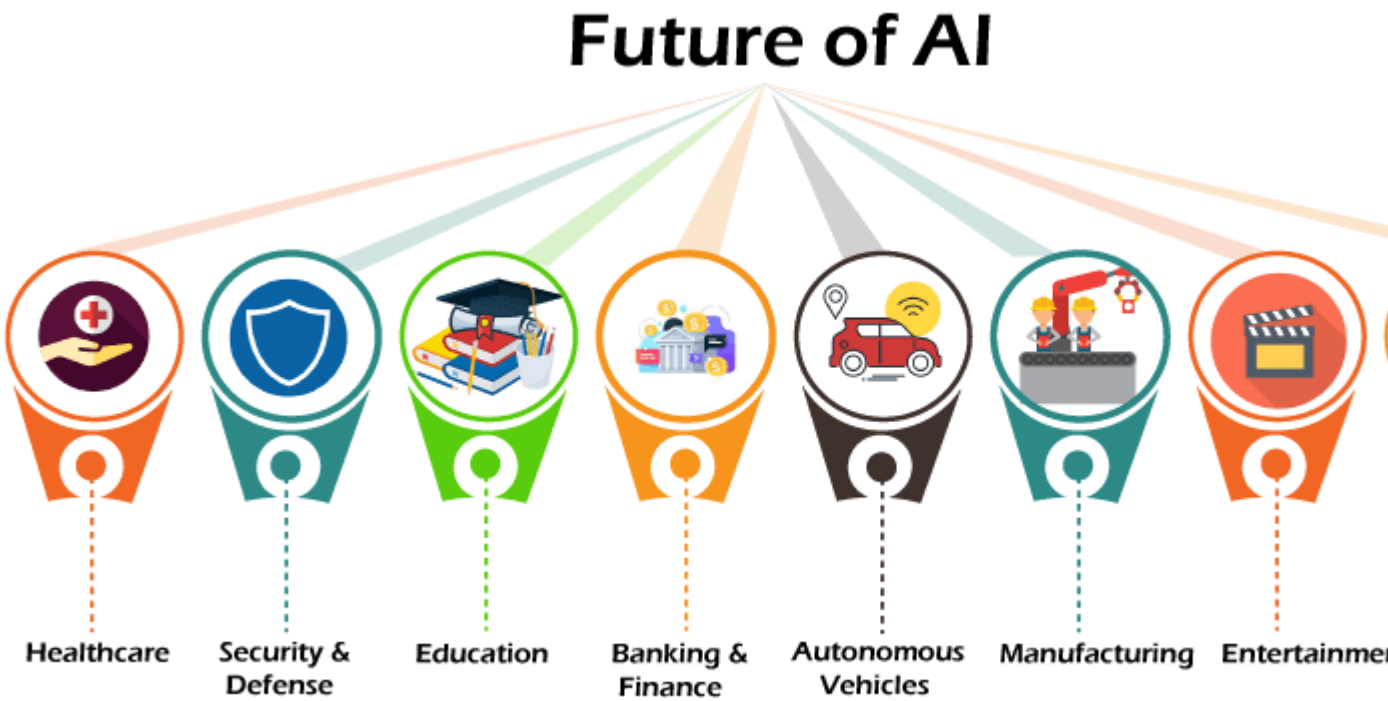


Fig 3: Future of AI

Healthcare: In the healthcare industry, AI will be critical in promptly and effectively diagnosing ailments. With the help of AI, new drug discovery will be speedier and more cost-effective. It will also improve patient engagement in their care by making appointment booking and bill payment easier and with fewer errors. Apart from these helpful applications, one of AI's greatest challenges in healthcare is ensuring its acceptance in daily clinical operations.

Cyber security: Without a question, ensuring data security is a top responsibility for every company. The following are some predictions about how cyber security will alter as a result of AI:

- Security incidents will be tracked using AI techniques.
- NLP is used to identify the source of cyber-attacks.
- RPA bots are used to automate rule-based operations and procedures.

As a brilliant technology, however, it can also be used by attackers as a threat. They can employ AI in an unethical manner by automating attacks that are difficult to defend.

Transportation: In the transportation industry, a fully autonomous vehicle has not yet been produced, but researchers are working on it. In the cockpit, artificial intelligence and machine learning are being used to help minimise workload, manage pilot stress and fatigue, and increase on-time performance. There are a number of barriers to AI adoption in transportation, particularly in public transit. There is a significant risk of becoming overly reliant on automated and autonomous technologies.

E-commerce: In the near future, Artificial Intelligence will play a critical role in the e-commerce industry. It will have a favorable impact on every part of the e-commerce industry, from user experience to product marketing and delivery. In the future, we can expect e-commerce to include automated warehouse and inventory management, shopper personalisation, and the employment of chatbots.

Employment: Because of the usage of Artificial Intelligence, finding work has become easier for both job seekers and companies. In the job market, AI is already being employed with stringent regulations and algorithms that automatically reject an employee's résumé if it does not meet the company's requirements. Most AI-enabled applications, spanning from marking written interviews to telephonic rounds, are expected to drive the job process in the future.

Various AI programmes, such as Rezi, Jobseeker, and others, are assisting job searchers in creating great resumes and finding the perfect position for their skills.

Apart from the aforementioned industries, AI has a bright future in manufacturing, finance and banking, entertainment, and other fields.

Key takeaway

Artificial Intelligence (AI) is a breakthrough discipline of computer science that is poised to become a key component of several future technologies such as big data, robotics, and the Internet of Things (IoT). In the coming years, it will continue to be a technical trailblazer.

6.11 AI Components

Learning, thinking, problem solving, perception, and language use are all aspects of intelligence.

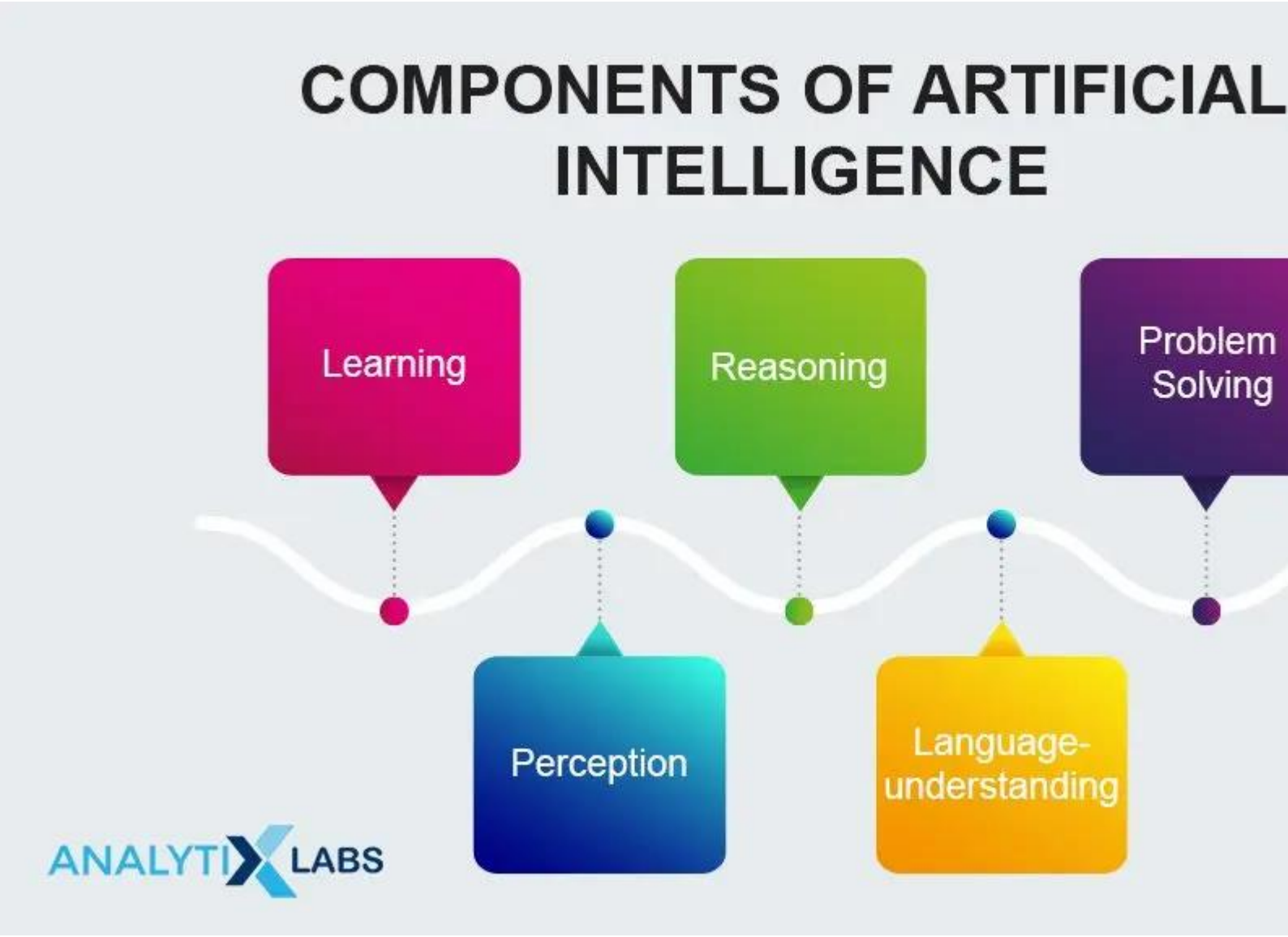


Fig 4: Components of AI

Learning

Computer programmes, like humans, learn in a variety of ways. When it comes to AI, this platform's learning is divided into a variety of categories. Learning for AI comprises the trial-and-error approach, which is one of the most important aspects of AI. The solution continues to solve problems until it obtains the desired results. As a consequence, the programme keeps track of all the movements that resulted in positive outcomes and stores them in its database for use the next time the computer is presented with the same problem.

Individual things, such as multiple solutions to problems, vocabulary, foreign languages, and so on, are memorized as part of AI's learning component, often known as rote learning. The generalization method is then used to put this learning process into practice.

When it comes to artificial intelligence, there are several different types of learning. The most basic method is to learn through trial and error. A rudimentary computer programme for solving mate-in-one chess situations, for example, might try moves at random until it finds mate. The programme could then save the solution along with the position so that it could be recalled the next time the computer came across the same situation. On a computer, rote learning—the simple memorizing of individual items and procedures—is relatively straightforward to accomplish. The difficulty of implementing what is known as generalization is more difficult. Generalization is the process of adapting previous experience to similar new situations.

Reasoning

Until about five decades ago, the art of reasoning was solely available to humans. Reasoning is one of the most important components of artificial intelligence because of its ability to differentiate. To reason is to allow the platform to make inferences that are appropriate for the situation at hand. In addition, these inferences are classified as inductive or deductive. The difference is that in an inferential scenario, a problem's solution assures a conclusion. In the inductive example, however, the mishap is always caused by instrument failure.

Programming computers have had a lot of success with deductive interferences. Reasoning, on the other hand, always entails drawing meaningful inferences from the circumstance at hand.

To reason is to make inferences that are appropriate for the circumstances. Deductive and inductive inferences are the two types of inferences. "Fred must be in either the museum or the café," for example, is an example of the former. "Previous accidents of this nature were caused by instrument failure; thus, this accident was caused by instrument failure," and "Previous accidents of this nature were caused by instrument failure; therefore, this accident was caused by instrument failure." The most important distinction between these two types of reasoning is that in deductive reasoning, the validity of the premises assures the

truth of the conclusion, whereas inductive reasoning offers support to the conclusion without providing total assurance.

Problem-solving

In its most basic form, AI's problem-solving ability consists of data, with the answer requiring the discovery of x. The AI platform sees a wide range of problems being handled. The various 'Problem-solving' approaches are artificial intelligence components that split questions into special and general categories.

A solution to a given problem is tailor-made in the case of a special-purpose method, which often takes advantage of some of the unique features supplied in the instance where a suggested problem is incorporated. A general-purpose approach, on the other hand, entails a wide range of interesting challenges. In addition, AI's problem-solving component allows programmes to include a step-by-step reduction of the distance between any target state and the current state.

Problem solving, especially in artificial intelligence, can be defined as a methodical search through a set of options in order to arrive at a predetermined goal or solution. There are two types of problem-solving methods: special purpose and general purpose. A special-purpose approach is designed specifically for a problem and typically takes advantage of extremely specific aspects of the scenario in which the problem exists. A general-purpose method, on the other hand, can be used to solve a wide range of issues.

Perception

When the 'perception' component of Artificial Intelligence is used, it scans any given environment utilizing various artificial or actual sense organs. Furthermore, the processes are maintained internally, allowing the perceiver to examine different scenes in suggested objects and comprehend their relationships and characteristics. This analysis is frequently problematic as a result of the fact that similar products might have a wide range of looks depending on the viewpoint of the indicated angle.

Perception is one of the components of artificial intelligence that, in its current condition, can propel self-driving cars at low speeds. FREDDY was one of the first robots that use perception to detect distinct things and put together various artifacts.

Perception involves scanning the surroundings with numerous sense organs, both real and artificial, and decomposing the view into discrete objects in various spatial relationships. The fact that an object can seem differently depending on the angle from which it is viewed, the direction and intensity of illumination in the scene, and how much the object contrasts with the surrounding field, makes analysis more difficult.

Artificial perception has progressed to the point that optical sensors can identify humans, autonomous vehicles can drive at highway speeds, and robots can roam through buildings collecting empty drink cans.

Language - understanding

Language can be defined as a collection of diverse system indications that use convention to justify their means. Language understanding, which is one of the most extensively used artificial intelligence components, employs distinct types of language across various forms of natural meaning, as shown by overstatements.

Human English is one of the most important qualities of languages since it allows us to distinguish between distinct items. Similarly, AI is designed in such a way that it can grasp English, the most widely spoken human language. In this way, the platform enables computers to quickly comprehend the many computer programmes that are run on them.

A language is a set of signs with predetermined meaning. Language does not have to be limited to the spoken word in this sense. For example, traffic signs form a mini language, with "danger ahead" being a matter of convention in some nations. The fact that linguistic units have meaning by convention distinguishes languages from other languages, and linguistic meaning differs significantly from natural meaning, as seen by expressions like "Those clouds signify rain" and "The drop in pressure shows the valve is malfunctioning."

Key takeaway

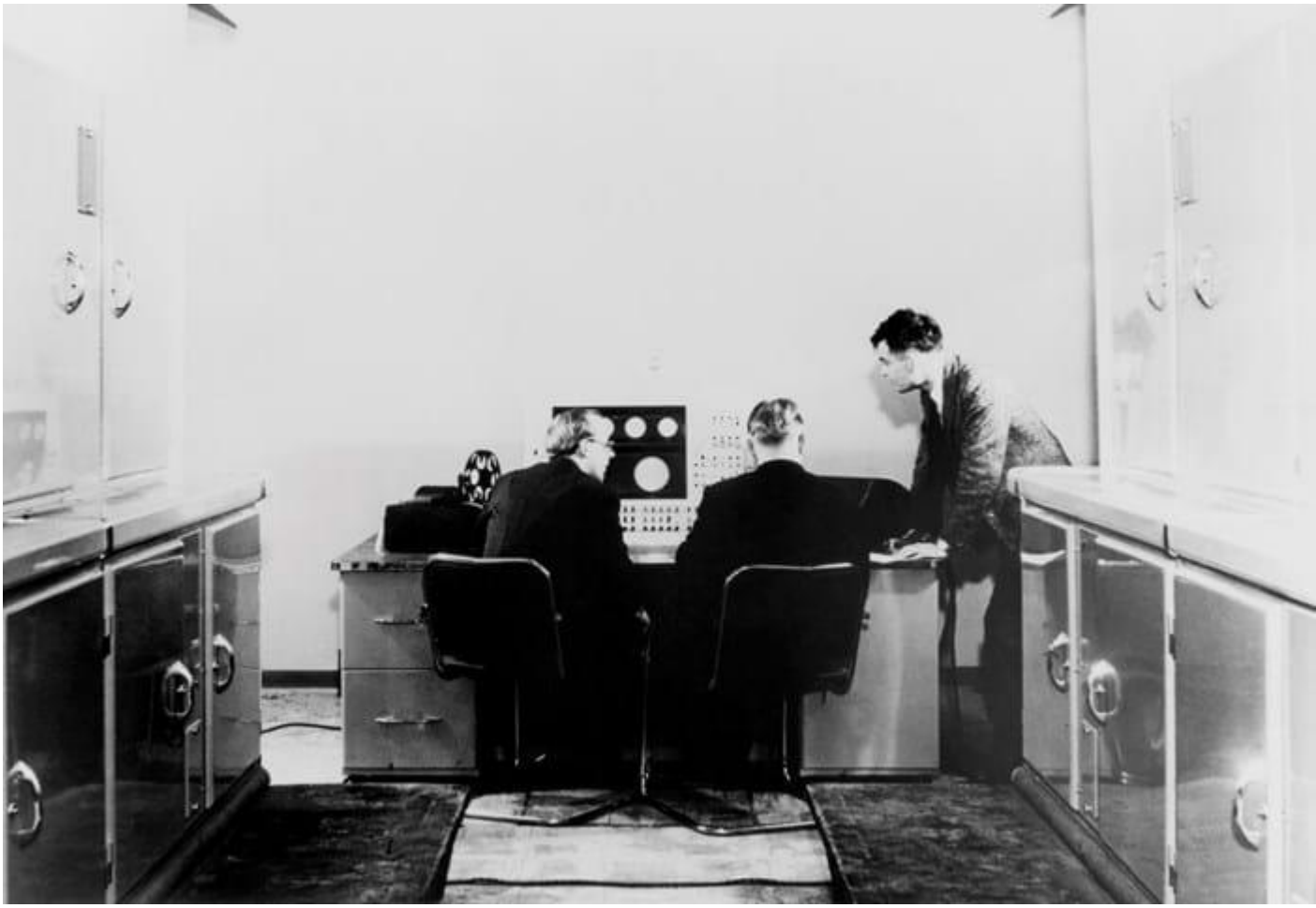
Learning, thinking, problem solving, perception, and language use are all aspects of intelligence.

6.12 AI Architectures

Today's architecture is a multi-skilled profession that draws on a variety of disciplines, including structural and environmental engineering, as well as social and material sciences. We have a long history of studying, adapting, and leveraging available technology to broaden and strengthen our design capabilities at Foster + Partners. The Applied Research and Development (ARD) team is at the forefront of this, having recently investigated the possibilities of employing artificial intelligence (AI) in the creative process.

AI was created as an area of academic research in the 1950s, and it can be roughly characterized as the development of machines that exhibit human intelligence and behaviors. It has been a hot issue of discussion since its inception, frequently making headlines – both good and negative – and being ingrained in popular culture and science fiction. Thankfully, AI has progressed with primarily benevolent innovations: it can assist us in understanding and responding to human speech, screening for breast cancer, developing robots and self-driving cars, or simply recommending films to watch.

Many systems and processes must be perfected in order to accurately mimic human intelligence; one example is the AI subset known as machine learning, which is defined as "the study of computer algorithms that allow computer programmes to automatically improve through experience" by computer scientist Tom Mitchell.



English mathematician and pioneer of computer science, Alan Turing, standing at the Ferranti Mark 1 computer at the University of Manchester in 1951. The earliest successful AI program was written by Christopher Strachey to play checkers, or draughts, and ran on the Ferranti Mark I

Machine learning has the ability to analyze our designs more rapidly and at a lower cost in the context of architecture and design. The ARD team has been experimenting with two methods for incorporating it into the design process, as well as some potentially innovative applications.

The first is surrogate modeling, which is a direct alternative for time-consuming analytical engineering simulations (such as structural deformation, solar radiation, and pedestrian movement) that might take hours or even days to complete.

Many architectural projects today are of immense scale and complexity, characterized by a diverse array of overlapping talent and technologies. Machine learning may be able to assist us in this diverse subject by allowing us to solve

problems and find patterns that were previously dependant on complex analytical simulations and programmes. Due to the time and effort required, design optimization based on their findings has been almost difficult, weakening the value of these tools.

We need to give designers with results in near real-time to tackle this problem. As a result, the ARD team looked into surrogate models, in which a computationally 'cheaper' predictive model is built based on a set of carefully selected simulation results. The surrogate model's approximation of a simulation in real-time can then provide the designer with results that are good enough for them to make a swift decision.

We call the second – and more cutting-edge – field of ARD's machine learning research 'design aid' modeling, and these systems have the ability to function alongside designers' intuition in the creative process.

This is referred to as design-assistance modeling because it aids architectural processes for which we do not always have an analytical solution – one that can be determined from simulations. This could include optimizing the spatial layout of furniture within a space or providing designers with real-time document control support while they work.

6.13 Case Study: The Amazing Ways Samsung Is Using Big Data, Artificial Intelligence And Robots To Drive Performance

Until recently, Samsung was considered to be lagging behind its competitors in terms of artificial intelligence (AI) research and development, but the company's current strategy implies that it is devoted to narrowing the gap and even contending for first place. Samsung is the world's biggest provider of data storage products, with its equipment producing and storing 70 percent of the world's data. Samsung is the world's largest consumer electronics manufacturer by revenue, having surpassed Apple and selling 500 million connected devices per year. Samsung appears to have gone all out in preparing for the fourth industrial revolution, from industry gatherings to setting goals with AI at the forefront to updating products to use artificial intelligence.

Bringing innovators together

Samsung began 2018 with the goal of becoming a leader in artificial intelligence by hosting the Artificial Intelligence (AI) Summit, which brought together 300 university students, technical experts, and leading academics to discuss ways to accelerate AI research and develop the best commercial AI applications.

On Samsung's AI research team is Dr. Larry Heck, a world-renowned AI and voice recognition expert. Dr. Heck stressed the importance of collaboration within the AI sector at the summit in order for customers to have more confidence and adoption of AI and for AI to develop. Samsung announced intentions to hold additional AI-related events as well as the establishment of a new AI Research Center dedicated to AI development and research. Samsung's artificial intelligence expertise will be bolstered by the research facility.

Bixby: Samsung’s AI Assistant

With the Samsung Galaxy S8, Bixby, Samsung's artificial intelligence system designed to make device interaction easier, made its debut. 2.0 is a "major leap forward for digital assistants," according to the current edition. Bixby 2.0 expands the AI system's reach to include TVs, refrigerators, washers, smartphones, and other connected devices. It's also open to developers, increasing the chances of it integrating with other products and services.

Bixby is aware of its surroundings and understands spoken language in order to assist users in interacting with increasingly complicated equipment. To compete with Google Home and Amazon Alexa, Samsung plans to release a Bixby speaker.

Samsung to add AI to all devices by 2020

Samsung said at CES 2018 that artificial intelligence capabilities would be included in every product company produces by 2020. It merged all smart programmes into a new Smart Things app as part of this plan, making it easy to connect and control all devices. By 2020, all Samsung gadgets will not only be Internet of Things ready, but they will also feature AI. "Samsung's primary strategic initiatives are AI and

machine learning," said Young Sohn, President and Chief Strategy Officer at Samsung Electronics.

AI technology based on machine learning to upscale images

Samsung Electronics was the first company to show off 8K AI technology for televisions. The system analyses content and upscales low-resolution photos to 8K visual quality automatically. This innovation addresses the current issue of high-resolution content being available for usage on screens with super-high resolutions. All images can now be converted to 8K, which is the greatest resolution currently available in digital television.