Lecture 2

Knowledge Representation

1.1. Knowledge representation

• The objective of knowledge representation is to express the knowledge about the world in a computer-tractable form. Many Knowledge Base systems rely on some variant of logic.

1.2. What are Knowledge Representation Schemes?

In Al, there are four basic categories of representational schemes: logical, procedural, network and structured representation schemes.

1) **Logical representation** uses expressions in formal logic to represent its knowledge base. **Predicate Calculus** is the most widely used representation scheme.

مخطط التمثيل المنطقي:- هذا المخطط يستخدم التعابير في المنطق لتمثيل قاعدة المعرفة. **Predicate Calculus** هو نظام التمثيل الأكثر استخدامًا.

2) **Procedural representation** represents knowledge as a set of instructions for solving a problem. These are usually **if-then rules** we use in rule-based systems.

مخطط التمثيل الاجرائي:- يمثل المعرفة كمجموعة تعليمات لحل المشاكل. النظم الخبيرة التي تعتمد على قانون if-then هي مثال على هذا التمثيل.

3) **Network representation** captures knowledge as a graph in which the nodes represent objects or concepts in the problem domain and the arcs represent relations or associations between them. **Semantic networks and conceptual graph** are example of this scheme

مخطط التمثيل الشبكة:- يظهر المعرفة كمخطط تكون فيه العقد تمثل الأشياء او المفاهيم في مجال المشكلة والأسهم تمثل العلاقة بينهم

4) **Structured representation** extends network representation schemes by allowing each node to have complex data structures named slots with attached values, theses values may be simple numeric or symbolic data, pointer to other frames or even procedures, **scripts and frames** are examples of this scheme. We will focus on logic representation schemes in this chapter.

مخطط التمثيل الهيكلي:- يعمل على توسيع مخططات تمثيل الشبكة من خلال السماح لكل عقدة بان يكون لها هياكل بيانات معقدة تسمى slots يتكون من مجال الاسم بالإضافة الى قيم مرفقة. هذه القيم يمكن ان تكون رقمية بسيطة او بيانات رمزية او تكون مؤشرات الى إطارات frames أخرى او حتى اجراءات.

1.3. Logical representation

Logic : A formal language for expressing knowledge and ways of reasoning.

Logic is defined by:

• A set of sentences – A sentence is constructed from a set of primitives according to syntax rules.

• A set of interpretations – An interpretation gives a semantic to primitives. It associates primitives with values.

• The valuation (meaning) function V – Assigns a value (typically the truth value) to a given sentence under some interpretation V : sentence × interpretation → {True , False }

Example of logic Language of numerical constraints:

• A sentence: $x + 3 \leq z$

x , z - variable symbols (primitives in the language)

• An interpretation: $x = 5, z = 2$ Variables mapped to specific real numbers.

• Valuation (meaning) function

V: $V (x + 3 \leq z, I)$ is False for I: $x = 5, z = 2$ is True for I: $x = 5, z = 10$

1.4. Types of logic

• Different types of logics: 1- Propositional logic 2- predicate logic

1.4.1. Propositional Logic (Propositional Calculus)

Propositional logic defines a language for symbolic reasoning.

• Proposition: a statement that is either true or false.

Formally propositional logic P: – Is defined by

Syntax + interpretation + semantics of P

Syntax:

❖ The symbols of propositional calculus are: {P, Q, R, S, …}

❖ Truth symbols: **{True, false}**

❖ Connectives: {$\wedge$,$\vee$, $\neg$, $\rightarrow$, $\equiv$}

$\wedge$ and [conjunction]

$\vee$ or [disjunction]

$\neg$ not [negation]

$\rightarrow$ implies [implication / conditional]

$\equiv$ or $\leftrightarrow$ is equivalent [biconditional]

The meaning (truth value) of and, or, if…then and not sentences.

- Legal sentences are also called well formed formula (WFF).
- **Semantics of propositional calculus:-** It is concerned with the "meaning" of statements, which is required for reasoning in AI.

❖ Examples:- P means "It is raining"
Or Q mean "I live in a brown house"

- Any proposition might be true "T" or false "F", this is called **interpretation**.
- **An interpretation** of a set of proposition is assignment of truth value, either T or F to each of the proposition symbols.
- The truth assignments are represented in truth tables

And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

If . . . then

p	q	$p \supset q$
T	T	$\rightarrow$
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T

Example: Use a truth table to list all possible truth value assignments to the propositions of the expression (**P** ∧ **Q**) ∨ (¬ **Q** ∨ **P**).

Answer:

P	Q	P∧Q	¬Q	¬Q∨P	(P∧Q)∨(¬Q∨P)
T	T	T	F	T	T
T	F	F	T	T	T
F	T	F	F	F	F
F	F	F	T	T	T

¬A∨B); in other word prove A→B ≡ ¬A∨B.

Answer:

A	B	A→B	¬A	¬A∨B
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

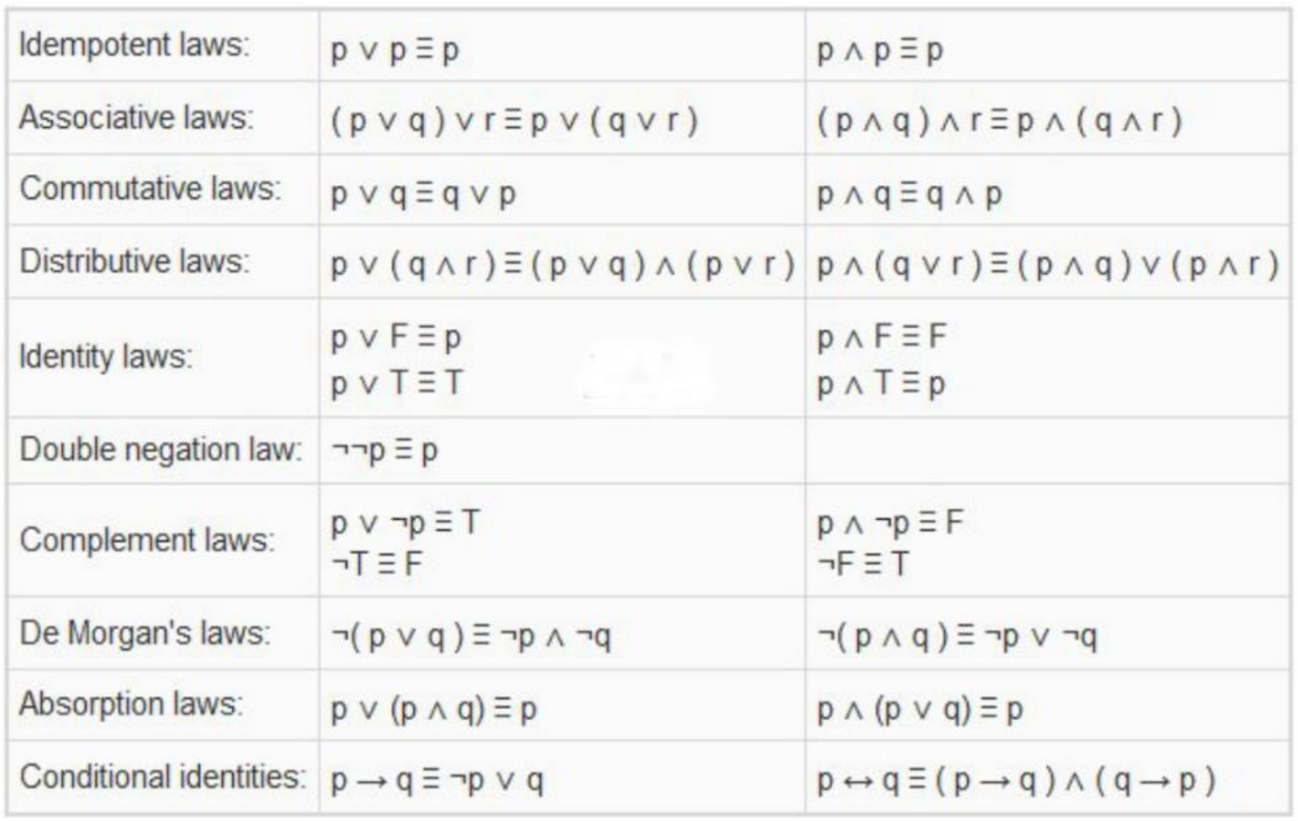
≡

H.W1\ Prove that the two expressions are equivalence $P \rightarrow Q \equiv \neg P \vee Q$

P	Q	$\neg P$	$\neg P \vee Q$	$P \rightarrow Q$	$(P \rightarrow Q) \equiv \neg P \vee Q$
T	T				
T	F				
F	T				
F	F				

H.W2\ Prove that the two expressions are equivalence $P \vee Q \equiv \neg P \rightarrow Q$

Table 2.4.4: Laws of propositional logic.



Idempotent laws:	$p \vee p \equiv p$	$p \wedge p \equiv p$
Associative laws:	$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Commutative laws:	$p \vee q \equiv q \vee p$	$p \wedge q \equiv q \wedge p$
Distributive laws:	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Identity laws:	$p \vee F \equiv p$ $p \vee T \equiv T$	$p \wedge F \equiv F$ $p \wedge T \equiv p$
Double negation law:	$\neg\neg p \equiv p$	
Complement laws:	$p \vee \neg p \equiv T$ $\neg T \equiv F$	$p \wedge \neg p \equiv F$ $\neg F \equiv T$
De Morgan's laws:	$\neg(p \vee q) \equiv \neg p \wedge \neg q$	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
Absorption laws:	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
Conditional identities:	$p \rightarrow q \equiv \neg p \vee q$	$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Contrapositive law	$(P \rightarrow Q) \equiv (\neg Q \rightarrow \neg P)$

And there are more laws you can check it.

Example: Convert the following English sentences to propositional calculus sentences:

- ☒ It is hot

P

- ☒ It is not hot

┐P

- ☒ If it is raining, then will not go to mountain

$P \rightarrow \neg Q$

- ☒ The food is good and the service is good

$X \wedge Y$

- ☒ If The food is good and the service is good then the restaurant is good

$X \wedge Y \rightarrow Z$

The prepositional calculus has its limitations that you cannot deal properly with general statements because it represents each statement by using some symbols jointed with connectivity tools.

1.4.2. Predicate Logic (Predicate Calculus)

To solve the limitations in the prepositional calculus, you need to analyze propositions into predicates and arguments, and deal explicitly with quantification. Predicate Logic provides formalism for performing this analysis of prepositions and additional methods for reasoning with quantified expressions.

For example, instead of letting a single propositional symbol, **P**, denote the entire sentence “it rained on Tuesday”, we can create a predicate **weather** that describes a relationship between a date and weather: weather(tuesday, rain)

Syntax of Predicate Calculus

The predicate calculus uses the following types of symbols:

set of letters, set of digits, and the **underscore (_)**. Any predicate calculus symbol should always start with a letter. It may represent a variable, a constant, a function or a predicate.

Constants: A constant symbol denotes a particular entity, object or properities and always begin with a lowercase symbol. E.g. john, muriel, flight_102.

Functions: A function symbol denotes a mapping from a number of entities to a single entities: E.g. fatherof is a function with one argument. plus is a function with two arguments. fatherof(john) is some person. plus(2,7) is some number.

Predicates: A predicate denotes a relation on a number of entities. e.g. married is a predicate with two arguments. odd is a predicate with one argument. married(john, sue) is a sentence that is true if the relation of marriage holds between the people john and sue. odd(plus(2,7)) is a true sentence.

Variables: These represent some undetermined entity. Examples: X, S1, etc.

Boolean operators: ¬, ∨, ∧, ⇒, ⇔.

Quantifiers: The symbols ∀ (for all) and ∃ (there exists).

∀X likes(X, chocolate)^ likes(X, chips) ⇒ ¬ healthy(X), it means that "everybody who likes chocolates and chips are not healthy".

Grouping symbols: The open and close parentheses and the comma.

Example: Convert the following English sentences to predicate calculus sentences:

- If it is raining, tom will not go to mountain

  rain(weather) →┐go(tom, mountain)

- All basketball players are tall

  ∀X play(X, basketball) →tall(X)

- John like anyone who likes books

  ∃X like(X, book)→like(john, X)

- Nobody likes taxes

  ┐∃X likes(X, taxes)

- There is a person who write computer class

  ∃X write(X, computer class)

- All dogs are animals

  ∀X dogs(X)→animals(X)

- John did not study but he is lucky

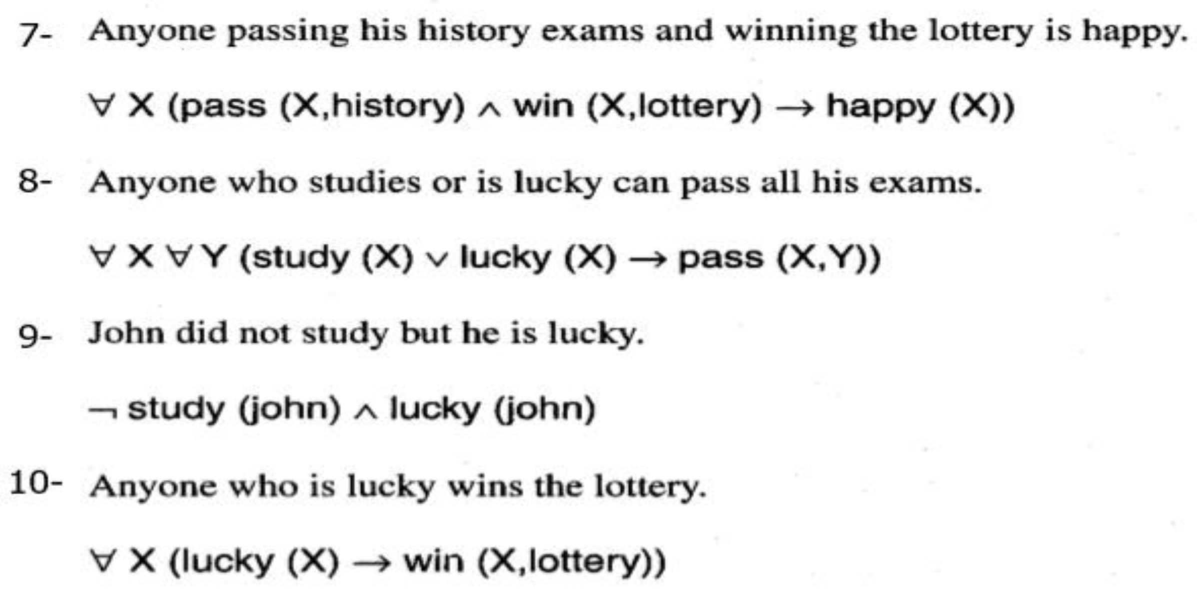
  ┐study(john) ∧ lucky (john)

- All cats and dogs are animals

  $\forall X \forall Y\ cats(X) \land dogs(Y) \rightarrow animals(X) \land animals(Y)$

- If it doesn't rain tomorrow, Tom will go to the mountains.

  $\neg\ rain\ (weather, tomorrow) \rightarrow go(tom, mountains).$



7- Anyone passing his history exams and winning the lottery is happy.

$\forall X\ (pass\ (X,history) \land win\ (X,lottery) \rightarrow happy\ (X))$

8- Anyone who studies or is lucky can pass all his exams.

$\forall X\ \forall Y\ (study\ (X) \lor lucky\ (X) \rightarrow pass\ (X,Y))$

9- John did not study but he is lucky.

$\neg\ study\ (john) \land lucky\ (john)$

10- Anyone who is lucky wins the lottery.

$\forall X\ (lucky\ (X) \rightarrow win\ (X,lottery))$

Example: Convert the following statements to predicate logic.

All people that are not poor and smart are happy. Those people that read are not stupid. John can read and wealthy. Happy people have exciting life. Can anyone found with an exciting life.

Answer:

$\forall X(\neg poor(X) \land smart(X)) \rightarrow happy(X)$

$\forall Y\ (read(Y) \rightarrow \neg\ stupid(Y))$

$read(john) \land\ wealthy(john)$

$\forall Z\ (happy(Z) \rightarrow\ exciting(Z,\ life))$

$\exists W\ exciting(W,\ life)$

Note $\neg$ stupid≡ smart, wealthy≡$\neg$poor

$\forall X(\neg poor(X) \wedge smart(X)) \rightarrow happy(X)$

$\forall Y\ (read(Y) \rightarrow smart\ (Y))$

$read(john) \wedge \neg poor(john)$

$\forall Z\ (happy(Z) \rightarrow exciting(Z, life))$

$\exists W\ exciting(W, life)$

Homework: Convert the following statements to predicate logic. Everyone passing their AI exam and winning the lottery is happy. But everyone who studies or lucky can pass all their exams, John did not study but he is lucky. Everyone who is lucky wins the lottery. Prove that John is happy.

1.4.3. Resolution:

Resolution is a technique for proving theorems in the predicate calculus using the resolution by refutation algorithm. The resolution refutation proof procedure answers a query or deduces a new result by reducing the set of clauses to a contradiction.

The Resolution by Refutation Algorithm includes the following steps:-

a) Convert the statements to predicate calculus (predicate logic).

b) Convert the statements from predicate calculus to **clause forms**.

c) Add the negation of what is to be proved to the clause forms.

d) Resolve the clauses to producing new clauses and producing a contradiction by generating the empty clause.

Clause Forms

The statements that produced from predicate calculus method are nested and very complex to understand, so this will lead to more complexity in resolution stage , therefore the following algorithm is used to convert the predicate calculus to clause forms:-

Suppose the expression we will reduce to clause form is:

$$\boxed{\text{(i) } (\forall X)([a(X) \land b(X)] \Rightarrow [c(X,I) \land (\exists Y)((\exists Z)[c(Y,Z)] \Rightarrow d(X,Y))]) \lor (\forall X)(e(X))}$$

1. Eliminate all ($\rightarrow$) by replacing each instance of the form ($P \rightarrow Q$) by expression ($\neg P \lor Q$)

$$\boxed{\text{(ii) } (\forall X)(\neg[a(X) \land b(X)] \lor [c(X,I) \land (\exists Y)(\neg(\exists Z)[c(Y,Z)] \lor d(X,Y))]) \lor (\forall X)(e(X))}$$

2. Reduce the scope of negation.

$\neg(\neg a) \equiv a$

$\neg(\exists X)\, a(X) \equiv (\forall X)\, \neg a(X)$

$\neg(\forall X)\, b(X) \equiv (\exists X)\, \neg b(X)$

$\neg(a \land b) \equiv \neg a \lor \neg b$

$\neg(a \lor b) \equiv \neg a \land \neg b$

Using the second and fourth equivalences (ii) becomes:

$$\boxed{\text{(iii) } (\forall X)([\neg a(X) \lor \neg b(X)] \lor [c(X,I) \land (\exists Y)((\forall Z)[\neg c(Y,Z)] \lor d(X,Y))]) \lor (\forall X)(e(X))}$$

3. Standardize variables: rename all variables so that each quantifier has its own unique variable name. For example,

$\forall X\, a(X) \lor \forall X\, b(X) \equiv \forall X\, a(X) \lor \forall Y\, b(Y)$

Because (iii) has two instances of the variable X, we rename:

$$\text{(iv)}\ (\forall X)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\exists Y)((\forall Z)[\neg c(Y,Z)] \vee d(X,Y))]) \vee (\forall W)(e(W))$$

4. Move all quantifiers to the left without changing their order. For example,

$\forall X\ a(X) \vee \forall Y\ b(Y)$

$\forall X\ \forall Y\ a(X) \vee b(Y)$

$$\text{(v)}\ (\forall X)(\exists Y)(\forall Z)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge ([\neg c(Y,Z)] \vee d(X,Y))]) \vee e(W))$$

5. Eliminate existential quantification by using the equivalent function. For example,

$\forall X\ \exists Y\ (mother(X,Y)) \equiv \forall X\ (mother(X,m(X)))$

$\forall X\ \forall Y\ \exists Z\ (p(X,Y,Z) \equiv \forall X\ \forall Y\ (p(X,Y,f(X,Y))$

$$\text{(vi)}\ (\forall X)(\forall Z)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge ([\neg c(f(X),Z)] \vee d(X,f(X)))]) \vee e(W))$$

6. Remove universal quantification symbols. For example,

$\forall X\ \forall Y\ (p(X,Y,\ f(X,Y))) \equiv p(X,Y,\ f(X,Y))$

$$\text{(vii)}\ [\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\neg c(f(X),Z) \vee d(X,f(X)))] \vee e(W)$$

7. Use the associative and distributive properties to get a conjunction of disjunctions called conjunctive normal form. For example,

$a \vee (b \vee c) \equiv (a \vee b) \vee c$

$a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c$

$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$

a∧(b∨c)≡(a∧b)∨(a∧c)

(viii) [¬ a(X) v ¬b(X) v c(X,I) v e(W)] ∧
[¬ a(X) v ¬b(X) v ¬c(f(X),Z) v d(X,f(X)) v e(W)]

8. Split each conjunct into a separate clause. For example,

(˥a(X)∨ ˥ b(X)∨e(W))∧(˥b(X)∨˥d(X,f(X))∨e(W))

˥a(X)∨ ˥ b(X)∨e(W)

˥b(X)∨˥d(X,f(X))∨e(W)

(ixa) [¬ a(X) v ¬b(X) v c(X,I) v e(W)]
(ixb) [¬ a(X) v ¬b(X) v ¬c(f(X),Z) v d(X,f(X)) v e(W)]

9. Standardize variables apart again so that each clause contains variable names that do not occur in any other clause. For example,

(˥a(X)∨ ˥ b(X)∨e(W))∧(˥b(X)∨˥d(X,f(X))∨e(W))

˥a(X)∨ ˥ b(X)∨e(W)

˥b(Y)∨˥d(X,f(X))∨e(V)

(xa) [¬ a(X) v ¬b(X) v c(X,I) v e(W)]
(xb) [¬ a(U) v ¬b(U) v ¬c(f(U),Z) v d(U,f(U)) v e(V)]

Example: Use the Resolution Algorithm for proving that John is happy with regard the following story:

Everyone passing his AI exam and winning the lottery is happy. But everyone who studies or lucky can pass all his exams, John did not study but he is lucky. Everyone who is lucky wins the lottery. Prove that John is happy.

Solution:

a) Convert all statement to predicate calculus.

∀X pass(X,ai_exam)∧win(X,lottery)→happy(X)

∀Y∀E study(Y)∨lucky(E)→pass(Y,E)

¬study(john)∧lucky(john)

∀Z lucky(Z)→win(Z,lottery)

happy(john)?

b) Convert the statements from predicate calculus to clause forms.

1. ∀X ¬(pass(X,ai_exam)∧win(X,lottery))∨ happy(X)

∀Y∀E ¬(study(Y)∨lucky(Y))∨ pass(Y,E)

¬study(john)∧lucky(john)

∀Z ¬(lucky(Z))∨ win(Z,lottery)

happy(john)?

2.

∀X (¬pass(X,ai_exam)∨¬win(X,lottery))∨ happy(X)

∀Y∀E (¬ study(Y)∧¬lucky(Y))∨ pass(Y,E)

¬study(john)∧lucky(john)

∀Z ¬lucky(Z)∨win(Z,lottery)

happy(john)?

3. Nothing to do here.

4. Nothing to do here.

5. Nothing to do here.

6.

┐(pass(X,ai_exam)∨┐win(X,lottery))∨ happy(X)

┐(study(Y)∧┐lucky(Y))∨ pass(Y,E)

┐study(john)∧lucky(john)

┐lucky(Z)∨win(Z,lottery)

happy(john)?

7. ┐

pass(X,ai_exam)∨┐win(X,lottery)∨ happy(X)

┐(study(Y)∧┐lucky(Y))∨ pass(Y,E) ≡ (a∧b)∨c ≡ c∨(a∧b)

The second statement become:

pass(Y,E)∨┐study(Y)∧pass(Y,E)∨┐lucky(Y)

┐study(john)∧lucky(john)

┐lucky(Z)∨win(Z,lottery)

happy(john)?

8.

┐pass(X,ai_exam)∨┐win(X,lottery)∨ happy(X)

pass(Y,E)∨┐study(Y)

pass(Y,E)∨┐lucky(Y)

┐study(john)

lucky(john)

┐lucky(Z)∨win(Z,lottery)

happy(john)?

9.

┐pass(X,ai_exam)∨┐win(X,lottery)∨ happy(X)

pass(Y,E)∨┐study(Y)

pass(M,G)∨┐lucky(M)

┐study(john)

lucky(john)

┐lucky(Z)∨win(Z,lottery)

happy(john)?

c) Add the negation of what is to be proved to the clause forms. ┐happy(john).

d) Resolve the clauses to producing new clauses and producing a contradiction by generating the empty clause.

There are two ways to do this, the first is backward resolution and the second is forward resolution.

1) Backward Resolution

The proving for happy(john) using Backward Resolution is shown as follows:

1. ¬pass(X,ai_exam)∨¬win(X,lottery)∨happy(X)
2. pass(Y,E)∨¬study(Y)
3. pass(M,G)∨¬lucky(M)
4. ¬study(john)
5. lucky(john).
6.¬lucky(Z)∨win(Z,lottery).
7.¬happy(john).

7: ¬$\cancel{\text{happy}}$(john) **1**:¬pass(X,ai_exam)∨¬win(X,lottery)∨ $\cancel{\text{happy}}$(X) {X=john}

8: ¬pass(john, ai_exam)∨¬$\cancel{\text{win}}$(john,lottery) **6**: ¬lucky(Z)∨$\cancel{\text{win}}$(Z,lottery){Z=john}

9: ¬pass(john, ai_exam)∨¬$\cancel{\text{lucky}}$(john) **5**: $\cancel{\text{lucky}}$(john)

10: ¬$\cancel{\text{pass}}$(john, ai_exam) **3**: $\cancel{\text{pass}}$(M,G)∨¬lucky(M) {M=john, G=ai_exam}

11: ¬$\cancel{\text{lucky}}$(john) **5**: $\cancel{\text{lucky}}$(john)

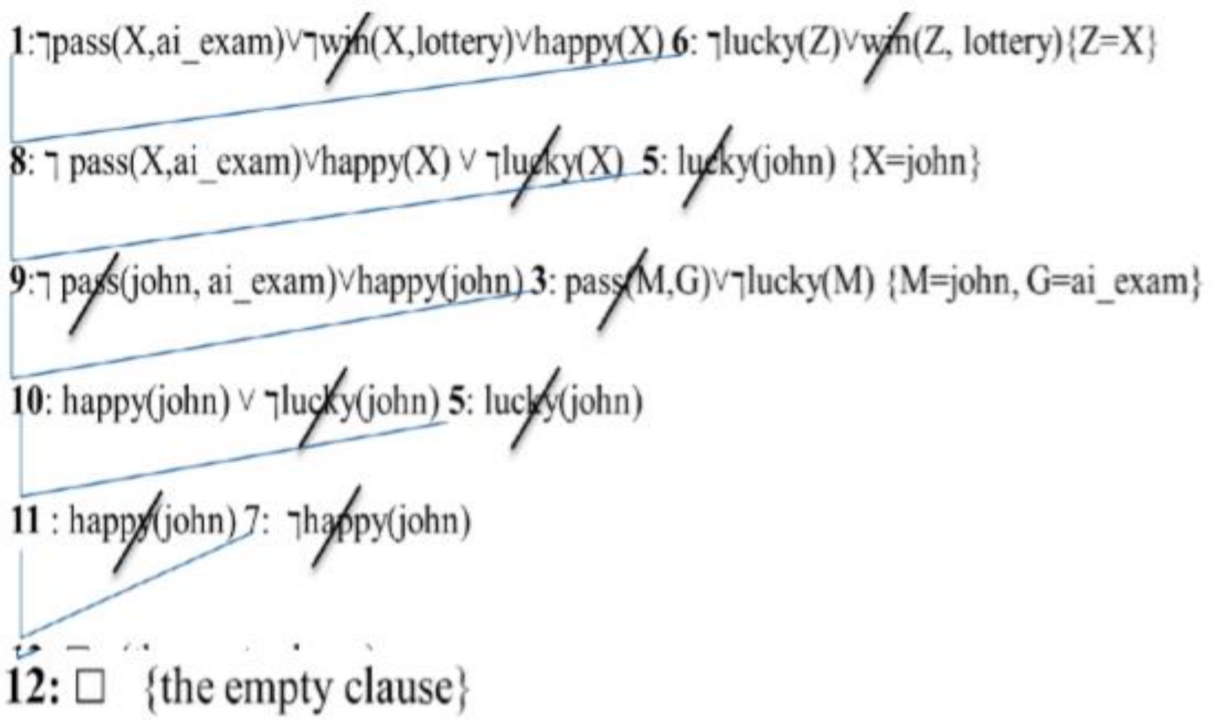
12: □ {the empty clause}

∴ **John is happy**

2) Forward Resolution

The proving for ***happy(john)*** using **Backward Resolution** is shown as follows:

1. ¬pass(X,ai_exam)∨¬win(X,lottery)∨happy(X)
2. pass(Y,E)∨¬study(Y)
3. pass(M,G)∨¬lucky(M)
4. ¬study(john)
5. lucky(john).
6. ¬lucky(Z)∨win(Z,lottery).
7. ¬happy(john).



1:¬pass(X,ai_exam)∨¬$\cancel{\text{win}}$(X,lottery)∨happy(X) **6**: ¬lucky(Z)∨$\cancel{\text{win}}$(Z, lottery){Z=X}

8: ¬ pass(X,ai_exam)∨happy(X) ∨ ¬$\cancel{\text{lucky}}$(X) **5**: $\cancel{\text{lucky}}$(john) {X=john}

9:¬ $\cancel{\text{pass}}$(john, ai_exam)∨happy(john) **3**: $\cancel{\text{pass}}$(M,G)∨¬lucky(M) {M=john, G=ai_exam}

10: happy(john) ∨ ¬$\cancel{\text{lucky}}$(john) **5**: $\cancel{\text{lucky}}$(john)

11 : $\cancel{\text{happy}}$(john) 7: ¬$\cancel{\text{happy}}$(john)

12: □ {the empty clause}

∴ **John is happy**

Ex(2): We now present an example of a resolution refutation for the predicate calculus. We wish to prove that "Fido will die" from the statements that "Fido is a dog. All dogs are animals. All animals will die."

Fido is a dog: dog (fido).

All dogs are animals: $\forall$(X) (dog (X) $\rightarrow$ animal (X)).

All animals will die: $\forall$(Y) (animal (Y) $\rightarrow$ die (Y)).

Converts these predicates to clause form:

PREDICATE FORM	CLAUSE FORM
1. $\forall$(X) (dog) (X) $\rightarrow$ animal (X))	$\neg$ dog (X) $\vee$ animal (X)
2. dog (fido)	dog (fido)
3. $\forall$(Y) (animal (Y) $\rightarrow$ die (Y))	$\neg$ animal (Y) $\vee$ die (Y)

Negate the conclusion that Fido will die:

4. $\neg$ die (fido)	$\neg$ die (fido)

Resolve clauses having opposite literals, producing new clauses by resolution as in Figure (3-2).

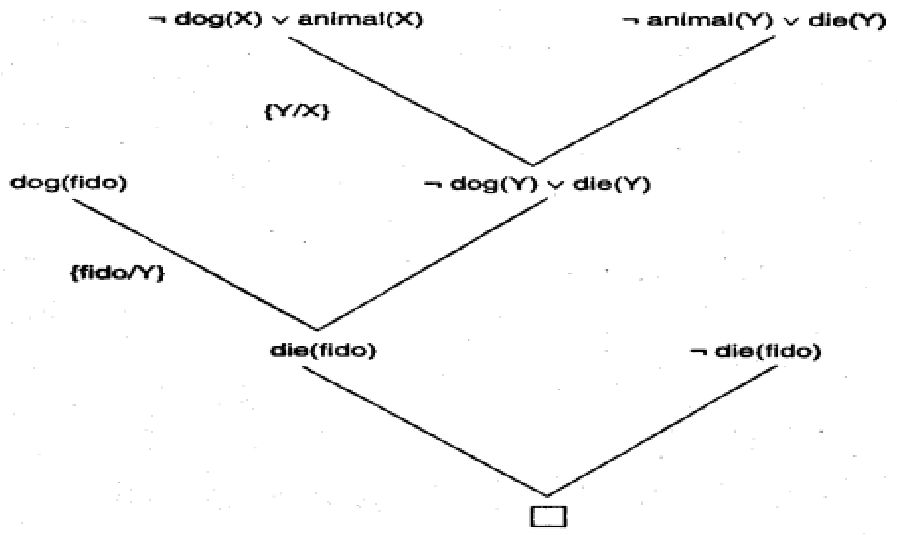


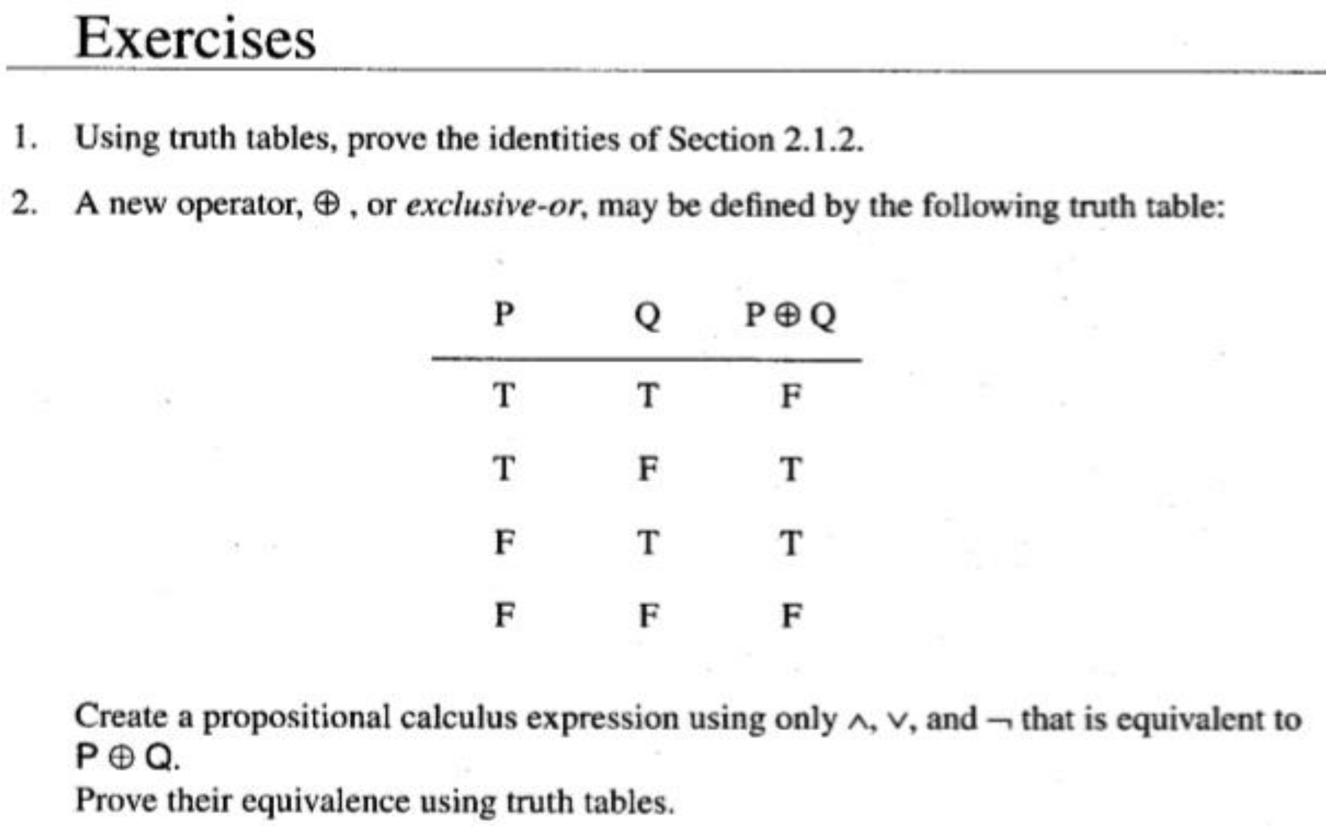
Figure (3-2): Resolution proof for the "dead dog" problem.

Ex: "All people who are not poor and are smart are happy. Those people who read are not stupid. John can read and is wealthy. Happy people have exciting lives. Can anyone be found with an exciting life?

Resources:-

1) https://www.javatpoint.com/types-of-artificial-intelligence
2) https://uomustansiriyah.edu.iq/media/lectures/6/6_2023_02_19!04_40_19_AM.pdf
3) **https://ccms.tu.edu.iq/csd/electronic-lectures/387--محاضرات-مادة-الذكاء-الاصطناعي-المرحلة-الثالثة-٢٠٢١-٢٠٢٢.html**

homework\



Exercises

1. Using truth tables, prove the identities of Section 2.1.2.
2. A new operator, ⊕ , or *exclusive-or*, may be defined by the following truth table:

P	Q	P ⊕ Q
T	T	F
T	F	T
F	T	T
F	F	F

Create a propositional calculus expression using only $\wedge$, $\vee$, and $\neg$ that is equivalent to $P \oplus Q$.
Prove their equivalence using truth tables.