Propositional Logic

- تذكرة:

-- تحدثنا في المحاضرة السابقة عن طرق حل الفرضيات، وشملنا طريقتي الـResolution وResolution Refutation

✓ مثال: (من السلايد رقم 84 في القسم PL)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

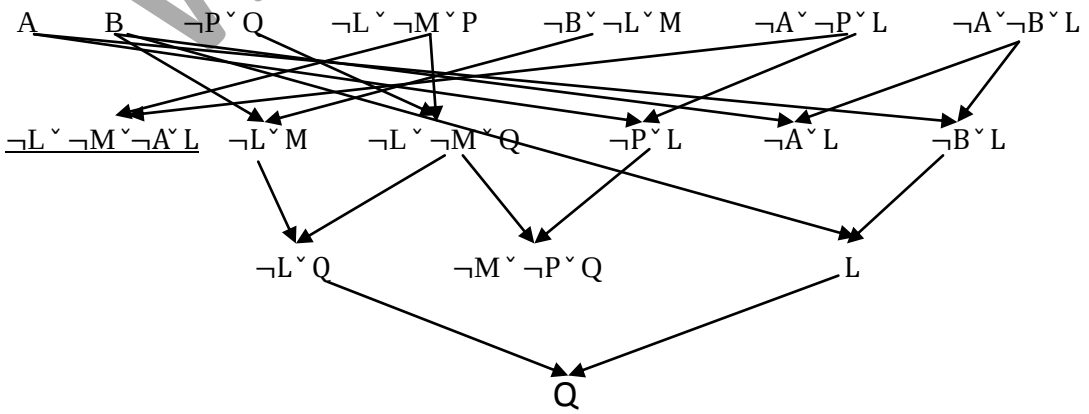
A

B

أثبت تحقق الهدف Q؟

يمكن الحل كما ذكرنا في المحاضرة الماضية باستخدام جدول الحقيقة لتحقيق الـEntailment ولكن هذا الأمر مكلف للوقت والجهد، لذلك سنستخدم خوارزميتي الـinference اللتان درسناهما:

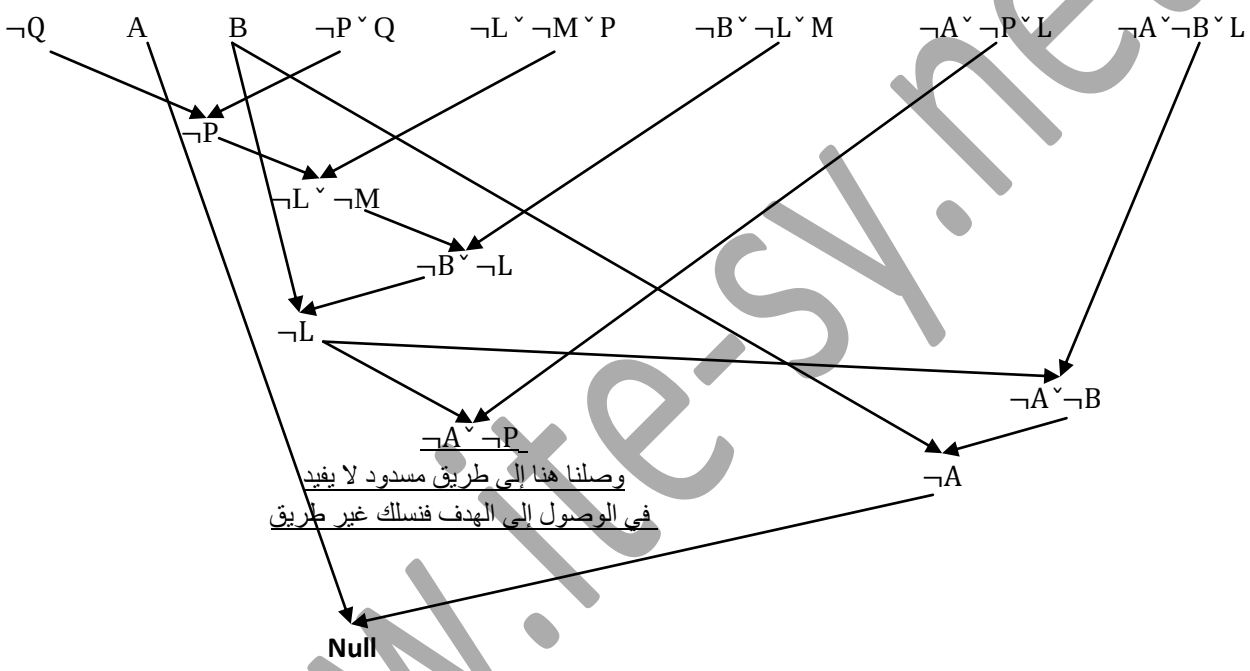
١ -طريقة الـ**Resolution** (سنستخدم طريقة الـBreadth First Search والتي نقوم فيها بتوليد جميع الاحتمالات الممكنة من كل مستوى ومن ثم نستخدم الذرّة التي تفيدنا للوصول إلى الهدف):



وبذلك نكون قد توصلنا إلى الهدف ولكن نلاحظ وجود بعض الحالات التي لم تفدنا في الوصول إليه مثل تلك التي تحتها خط في الشكل أعلاه (لأنها دوماً تعطي القيمة true حيث يتواجد ¬L وL في نفس الذرة) وهذه هي مشكلة خوارزمية BFS في الحل.

٢ -طريقة الحل بالنقض **Resolution Refutations**:

ننفي الهدف فيصبح ¬Q ويضاف إلى الـKB، وسنحل بخوارزمية Set of support والتي نقوم فيها بالاختصار مع العبارات المشتقة من الآباء ولا نقوم بتوليد جميع الاحتمالات الممكنة من المرحلة الواحدة، وإنما نستخدم تلك التي في المرحلة التي حصلنا عليها مؤخراً:



- وبما أننا قد وصلنا إلى Null نكون قد وصلنا إلى هدفنا المنشود ☺.

- **Forward Chaining and Backward Chaining**:

- تحدثنا في المحاضرة السابقة عن صيغة رياضية تسمى Horn Clause، وذكرنا في الحقيقة أن أغلب المعارف Knowledge Bases تعتمد على هذه العبارات والصيغ، وهي الأكثر انتشاراً فيها، ومن أهم الفوائد التي تقدمها عبارات Horn:
  - يمكن أن تحول إلى عبارات على الشكل القياسي CNF بسهولة.
  - تفيد بأن عملية الاستتباع (Entailment) باستخدامها تتم بتعقيد خطي (linear) وذلك تبعاً لحجم الـKB، وهذه الخاصة من أهم الخصائص التي جعلت هذه العبارات هي الأكثر انتشاراً.
  - تساعد في عملية الحل بخوارزميتي Forward Chaining وBackward Chaining ذلك لأن كل عبارة منها هي على شكل implication:

$$A \text{ some operation } B \underbrace{\Rightarrow}_{implies} C$$

وهذه الخاصية سهلت عملية استنتاج القيم المرتبطة بالرموز (Symbols) وبالتالي يمكن استخدام هذه الرموز المستتبعة (Entailed Symbols) في الـClauses الأخرى في الـKB.

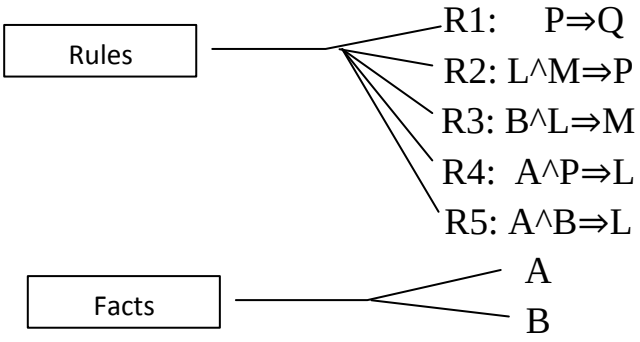
- وسنعرض الآن خوارزمية الحل بكل طريقة من الطريقتين:

١ -Forward Chaining:

في هذه الخوارزمية يُولّد كل شيء قابل للتوليد **دون معرفة مسبقة للهدف.**

← سنطبقها على المثال نفسه في السلايد 84:

أولاً سنرقم هذه الـRules إلى أرقام لنعرف الطريق الذي سيسلكه الحل:



تقوم فكرة الخوارزمية هنا على الانطلاق من البيانات المعروفة لحل الـQueries التي يجب حلها، وعند معرفة قيمة جديدة تستخدم لحل الـRules الاخرى وهكذا..

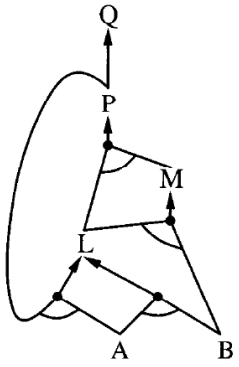
وبالتالي فإن مراحل التنفيذ هي المبينة بالشكل التالي:

Cycles	Rules Executed	Incoming Percepts
C1	R5	A , B , L
C2	R3	A , B , L , M
C3	R2	A , B , L , M , P
C4	R1 , R4	A , B , L , M , P , **Q**

ومن الملاحظ أنه قد قام بتنفيذ جميع الـRules وذلك بحسب المعارف التي لديه.

س:لماذا بدأ بـ R5؟
ج: في البداية لم يكن يعلم إلا A وB لذلك ابتدأ بتنفيذ الـR5 وعندها استطاع معرفة قيمة L والتي استخدمها في حل الـR3 وهكذا.. وصولاً إلى الـR1 التي تعطينا قيمة الهدف Q
من مساوئ هذه الطريقة - بالرغم من أنها توصل إلى الهدف وتعطي نتائج صحيحة- هي تنفيذ الكثير من الـRules حتى غير المفيدة منها في الوصول إلى الهدف.

لذلك توصف بأنها Data-Driven Method أي أنها تنطلق من البيانات التي تعرفها لتحل الـQueries التي تعترضها، وشجرة تنفيذ المثال السابق هي المبينة في الشكل التالي:



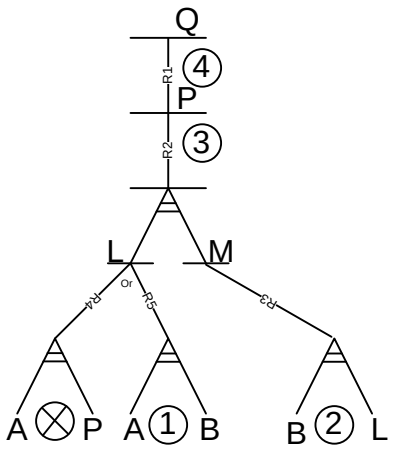
٢- Backward Chaining:

هذه الطريقة هي معاكسة تقريباً للطريقة السابقة، ففي هذه الطريقة ننطلق من الهدف بدلاً من المعارف التي نملكها، ومن الهدف نبدأ بالبحث عن القيم التي نحتاجها للوصول إلى الهدف، لذلك توصف بأنها Goal-Driven Method وليست Data-Driven Method كسابقتها، ولنفهم الخوارزمية بشكل أكبر سنطبقها على المثال نفسه:

R1: P⇒Q
R2: L^M⇒P
R3: B^L⇒M
R4: A^P⇒L
R5: A^B⇒L
A
B

والانطلاق من الهدف يعني أننا سنقوم بتنفيذ الـRules التي تعطينا قيمة الهدف، ولكن في مثالنا نلاحظ أنه لمعرفة قيمة الهدف Q نحتاج لمعرفة الرمز P، والذي نحصل عليه من الـR2 والذي بدوره يحتاج لقيمة كل من M وL لتحديد قيمته، وهكذا يتم التجول في الشجرة بشكل عودي (أو تراجعي بشكل أدق) حتى نصل إلى القيمة الهدف، والشكل التالي يبين طريقة الحل باستخدام هذه الخوارزمية:

الأرقام التي في الدوائر تبين ترتيب تنفيذ التعليمات (Rules) والتعليمة التي عندها X فلم تتنفذ أصلاً



نلاحظ من الشجرة السابقة للبحث أن الحل انطلق من الهدف Q فوجد أنها تتطلب إثبات P التي بدورها، تتطلب إثبات L و M وهكذا... حتى وصلنا إلى Rule جميع قيم طرفها اليساري معروفة والتي هي الـR5 ولذلك تم استنتاج قيمة L منها والتي استفيد منها في برهان M في الـR3، و الآن برهنت P و بالتالي Q.

س: لمَ لم تنفذ الـR4؟
ج: في الحقيقة هذا الشيء حدث كون الحصول على قيمة المتحول L يمكن أن يتم عن طريق تنفيذ R5 أو R4، و عندما بدأ بـ R4 وجد أن إثباتها يتطلب إثبات P و هي موجودة على طريق الحل و بالتالي وصل إلى طريق مسدود وبما أن العلاقة بين R4 و R5 هي علاقة OR، فإن R5 تكفي لبرهان L.
لم يتم تنفيذ R4 كون الـSub Goal عندها كان قيمة L وليس تنفيذ أكبر عدد من التعليمات ☺

- نلاحظ أن خوارزمية Backward Chaining تحقق تعقيداً أقل من تنفيذ الـForward Chaining ولذلك الـBackward Chaining هي المستخدمة في برنامج prolog الذي سنقوم بالتعامل معه من أجل حل الكثير من المسائل المنطقية.

- DPLL (David, Putnam , Logemann , LoveLand) Algorithm:

هي واحدة من الخوارزميات التي تستخدم كذلك في الحل لإيجاد قيم المتحولات والـClauses وهي تحقق فاعلية كبيرة في الحل ذلك لأنها تقوم باختصار الكثير من الحالات والعبارات في طريق البحث عن الحل، وهي في الحقيقة تحسين على طريقة جدول الحقيقة ذلك أنها تختصر الكثير من الحالات التي يحويها.

خطوات الخوارزمية:
لنطبقها مبدئياً على هذا المثال (من السلايد رقم 111):

$$(\neg D \vee \neg B \vee C) \wedge$$
$$(B \vee \neg A \vee \neg C) \wedge$$
$$(\neg C \vee \neg B \vee E) \wedge$$
$$(E \vee \neg D \vee B) \wedge$$
$$(B \vee E \vee \neg C)$$

١ -نقوم بإلغاء الجمل التي تعطي قيمة true إن كان أحد متحولاتها true حتى قبل أن تأخذ المتحولات الأخرى قيماً مثل الجملة التالية:

$$(A \veebar B) \wedge (A \veebar C)$$

تحمل القيمة true بمجرد أن تكون A هي true وقبل أن نعرف قيم كل من B وC!

وكذلك الجمل التي تعطي false إن كان أي من متحولاتها false حتى دون تحديد قيم المتحولات الأخرى مثل الجملة التالية:

$$(A \wedge B) \wedge (A \wedge C)$$

فبمجرد أن يكون المتحول A هو false تصبح كل العبارة false !

وبمثالنا الذي نقوم بحله الآن لا توجد هكذا جمل ☺.

٢ -نقوم بالبحث عن الـPure Symbols في الـKB، حيث أن الـPure Symbol هو متحول يحمل دوماً شكلاً واحداً في جميع عبارات الـKB.

في مثالنا السابق A وD وE جاؤوا دوماً بشكل واحد حيث:

A جاءت بالشكل A¬ في جميع العبارات التي تحوي A

D جاءت بالشكل D¬ في جميع العبارات التي تحوي D

E جاءت بالشكل E في جميع العبارات التي تحوي E

وعندها نعطي الشكل الذي جاءت فيه القيمة true دوماً، أي:

¬A=true

¬D=true

E=true

وكذلك هذا الكلام يطبق على الرموز التي تأتي وحيدة في سطرها (فمثلاً أن كان يوجد عبارة في المثال السابق هي A فقط فعندها نعطي A القيمة true ونعوضها في جميع العبارات التي تحويها).

وبعد أن نقوم بهذا الشيء تصبح عملية استنتاج قيم المتحولات الأخرى عملية سهلة.

٣ -في حال لم نجد أي Pure Symbol نقوم بالحل بطريقة مغايرة، وفيها نقوم بتعويض قيمة بأحد المتحولات وعندها نبحث عن قيم المتحولات الأخرى، في المثال الذي في السلايد 112 مثلاً:

$$(\neg A \vee \neg B \vee \neg C) \wedge$$
$$(\neg A \vee B) \wedge$$
$$(\neg A \vee C) \wedge$$
$$(\neg B \vee C) \wedge$$
$$(\neg B \vee A) \wedge$$
$$(\neg C \vee A) \wedge$$
$$(\neg C \vee B)$$

نلاحظ عدم وجود أي متحول Pure لذلك سنقوم باتخاذ الطريقة التالية:

نعطي الرمز A القيمة true بدايةً وعندها نحدد قيم المتحولات الأخرى:

A=true

فتصبح الـKB هي:

¬B ˇ¬C

B

C

¬B ˇC

¬C ˇB

وهنا أصبح لدينا B وC هي متحولات كل منها وحيد في سطره فنعطي كل منها قيمة true تصبح قيم كل من العبارات هي:

¬B ˇ¬C =false

B =true

C =true

¬B ˇC =true

¬C ˇB =true

ولأن العبارة الأولى تحمل القيمة false لذلك فإن هذا الطريق ليس بصحيح لأنه لا يشكل model فيه كل العبارات المحتواة في الـKB صحيحة، لذلك نعطي A القيمة false فتصبح الـKB هي:

¬B ˇC

¬B

¬C

¬C ˇB

وكذلك نجعل كل من ¬B=true و ¬C=true وبالتالي B=C=false وبالتعويض في العبارات:

¬B ˇC=true

¬B=true

¬C=true

¬C ˇB=true

وبالتالي هذا الطريق صحيح وقيم كل من الرموز التي تشكل model هي:

A=B=C=false

وهذه هي خوارزمية الـDPLL والتي تحقق نتائج سريعة وباهرة ويمكن الإطلاع على نص الخوارزمية على شكل Pseudo code في السلايد رقم 110 في الـPL– Chapter2.

ويوجد خوارزميات أخرى ومن بينها خوارزمية WalkSat التي لم يتطرق الدكتور إليها لربما قد تركت للمحاضرة القادمة أو ربما لن يتطرق لها أبداً وتركت لندرسها من السلايدات دون أن يشرحها ..... ☺

✓ تمارين متنوعة من السلايدات 120-121-122 في سلايد الـPL:

١ ـ أثبت أن العبارات التالية تعطي true:

- $P \supset (Q \supset P)$

يمكن استخدام جدول الحقيقة لإثباتها:

P	Q	Q⊃P	P ⊃ (Q⊃P)
0	0	1	1
0	1	0	1
1	0	1	1
1	1	1	1

وبالتالي هي دوماً صحيحة لذلك فهي true.
ويمكن أن نقوم بتحويلها إلى CNF :

$$\neg P \vee (\neg Q \vee P)$$

ومن الشكل يتضح لنا أنها دوماً true (ذلك لوجود P ونقيضه في العبارة).

- $(P \supset (Q \supset R)) \supset ((P \supset Q) \supset (P \supset R))$

لنحولها أيضاً إلى CNF لأنه طريق مختصر أكثر من طريق جدول الحقيقة:

$$\neg (\neg P \vee (\neg Q \vee R)) \vee (\neg (\neg P \vee Q) \vee (\neg P \vee R)) =$$
$$\neg (\neg P \vee \neg Q \vee R) \vee ((P \wedge \neg Q) \vee (\neg P \vee R)) =$$
$$(P \wedge Q \wedge \neg R) \vee (\underbrace{(P \vee \neg P \vee R)}_{\text{true}} \wedge (\neg Q \vee \neg P \vee R)) =$$
$$(P \wedge Q \wedge \neg R) \vee (\neg Q \vee \neg P \vee R) =$$
$$\underbrace{(P \vee \neg Q \vee \neg P \vee R)}_{\text{true}} \wedge \underbrace{(Q \vee \neg Q \vee \neg P \vee R)}_{\text{true}} \wedge \underbrace{(\neg R \vee \neg Q \vee \neg P \vee R)}_{\text{true}}$$

وبالتالي نلاحظ أنها دوماً صحيحة.

٢ ـ حول العبارة التالية إلى CNF:

- $\neg [((P \vee \neg Q) \supset R) \supset (P \wedge R)]$

$$= \neg [\neg (\neg (P \vee \neg Q) \vee R) \vee (P \wedge R)]$$
$$= \neg [\neg ((\neg P \wedge Q) \vee R) \vee (P \wedge R)]$$
$$= \neg [(\neg (\neg P \wedge Q) \wedge \neg R) \vee (P \wedge R)]$$
$$= \neg [((P \vee \neg Q) \wedge \neg R) \vee (P \wedge R)]$$
$$= [\neg ((P \vee \neg Q) \wedge \neg R) \wedge (\neg (P \wedge R))]$$
$$= [((\neg P \wedge Q) \vee R) \wedge (\neg P \vee \neg R)]$$
$$= (\neg P \vee R) \wedge (Q \vee R) \wedge (\neg P \vee \neg R)$$

٣ ـ أثبت أنني أفوز (I win) دوماً في المثال التالي:

Heads, I win; tails, you lose.

لنرمز إلى حدث الـHead بالرمز H ولحدث الـwin بالرمز W ولحدث الـtails بالرمز t ولحدث الـlose بالرمز l.

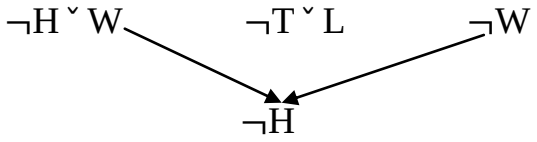
وبالتالي بترجمة العبارة السابقة إلى Rules وFacts نجد أنها تعطينا:

$$H \Rightarrow W$$

$$T \Rightarrow L$$

والهدف هو w .

وبالحل بطريقة النقض:



وبالتالي نجد أن الطريق مسدود ولا يمكن الوصول إلى null مهما حاولنا.

السبب: يوجد معارف ضمنية لم نقم بكتابتها، فمثلاً لم نقم بكتابة أن الحدث الذي يمكن أن يحدث هو إما H

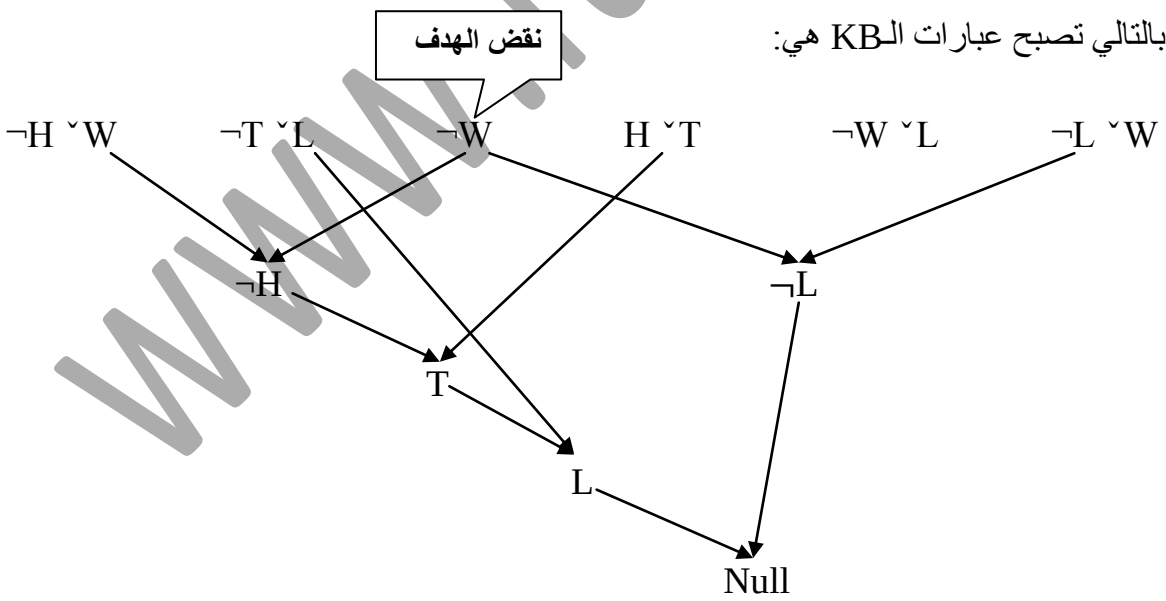
أو T أي :

$$H \veebar T$$

ولم نقم بكتابة أن حدث I Win يقتضي You lose وكذلك حدث You lose يقتضي I Win، أي:

$$W \Leftrightarrow L$$

وبالتالي تصبح عبارات الـKB هي:



وبذلك نكون قد أثبتنا تحقق الهدف W دوماً بطريقة الحل بالنقض لتوصلنا إلى Null.

First Order Logic

- قمنا بعدها ببدء الحل لبعض الأمثلة والمسائل التي تعتبر ضمن الـFirst Order Logic وليست ضمن الـPropositional Logic، ولكن السؤال المهم هو ما الفرق بينهما؟

١- Propositional Logic: هذا المنطق بسيط جداً ولا يمكن أن نعتمده في تحقيق جميع معارفنا ولحل جميع مسائلنا الواقعية، ذلك أنه يقوم بترجمة كل شيء إلى عبارات ويقوم بدراسة صحتها. حيث يعتبر أن كل شيء في الوجود عبارة عن حقيقة (Fact) أو عبارة عن Rule or query يريد أن يسأل عن صحته، أي جميع العبارات فيه ترد إما true أو false، لذلك هو عاجز عن التعبير بشكل دقيق عن جميع مسائلنا الواقعية، ففي بعض المسائل نحتاج إلى أكثر من ذلك حتى نستطيع نمذجة مسألة معينة وحلها.

٢- First Order Logic (FOL): هذا المنطق نستطيع القول عنه أنه قريب جداً إلى طريقة تفكيرنا كبشر، ويمكن اعتبار أن اللغات الطبيعية تخضع لهذا المنطق، ففيه نستطيع أن نعبر عن كل شيء يحيط بنا ونستطيع نمذجة العديد من القضايا والمسائل الواقعية التي تواجهنا في حياتنا، فهو يفترض أن العالم يتكون من:

١) Objects: مثل الناس والبيوت والأرقام والنظريات والمطاعم والألوان ووو.. إلخ
٢) Relations: وهي علاقات بين الـObjects وربما للـObject الواحد مثل علاقة أكبر من، أصغر من، أخ، أحمر (هي علاقة أحادية لها طرف واحد)، ابتدائي (مثل سابقتها)، .. إلخ
٣) Functions: مثل أب لـ ، أفضل صديق لـ ، أكبر بواحد من ... إلخ

- وبعد هذه المقارنة البسيطة بين هذين المنطقين، يمكن أن نعرض أهم عناصر الـSyntax المتضمنة في الـFOL:

- Constants: John, Tom, 2, 5 ,…. (والتي تحمل قيمة وحيدة دوماً)
- Predicates: Brother (…), skier(…), pilot(…),… (والتي يمكن أن نمرر لها العديد من القيم ولكنها لا ترد لنا إلا قيمة منطقية بوليانية)
- Functions: Sqrt (..), color (..), …. (ليس بالضرورة ترد لنا قيمة بوليانية ويمكن أن ترد قيم أخرى مختلفة مثل ألوان أو أرقام)
- Variables: x, y, z, …. (يمكن أن تحمل قيماً متعددة وليست ثابتة)
- Connectives: ¬ (not) , ⇒ (implies), ^ (and), ˇ(or), ⇔ (is equivalent)
- Equality: =
- Quantifiers: ∃ (يوجد), ∀ (مهما يكن)

✓ مثال 1: (من سلايد رقم 75 في الـChapter5)

- **Problem Statement:** Tony, Shi-Kuo and Ellen belong to the Hoofers Club. Every member of the Hoofers Club is either a skier and mountain climber or both. No mountain climber likes rain, and all skiers like snow. Ellen dislikes whatever Tony likes and likes whatever Tony dislikes. Tony likes rain and snow.
- **Query:** Is there a member of the Hoofers Club who is a mountain climber but not a skier?

- وبالتالي فإن المسألة تقسم إلى Constants وPredicates وغيرها.. والتي هي:

- Constants: Tony, Shi-Kuo, Ellen
- Predicates: Skier (x) , Mountain Climber (x) , Like (x, y)
- Variables: x : is a person, y: is a thing
- Sentences:

$\forall x$ Skier(x) ˇ Mountain climber(x)

$\neg \exists x$ Mountain climber(x) ^ likes (x, Rain)

$\forall x$ Skier(x) $\Rightarrow$ Likes(x, Snow)

دائماً مع $\forall$ نستخدم $\Rightarrow$

$\forall y$ Likes(Tony, y) $\Rightarrow \neg$Likes(Ellen, y)

$\forall y$ $\neg$ Likes(Tony, y) $\Rightarrow$ Likes(Ellen, y)

Likes(Tony, Snow)

Likes(Tony, Rain)

- Goal:

$\exists x$ Mountain Climber(x) ^ $\neg$ Skier(x) ??

دائماً مع $\exists$ نستخدم ^

وسأترك الإجابة لكم ☺

✓ مثال 2:

Mary تحب لون واحدة من كرافات Tony.

والحل هو:

- Constants: T, M
- Variables: y: is a person , x: is a thing
- Predicates: Likes(y, x), Tie(x), Owner(x, y)
- Functions: Color(x)
- Sentences:

∃x Likes(Mary, Color(x)) ^ Tie(x) ^ Owner(x, Tony) ?

- ✓ مثال 3: من السلايد رقم 36 في الـ4 Chapter:

All Packets in room 27 are smaller than some packet in room 28

في الحقيقة هذا المثال قمنا بحله لنعرف أن العبارات نفسها التي نقرأها يمكن أن تفهم أو تفسر بأكثر من طريقة، فمثلاً في هذا المثال يمكن أن نترجم العبارة السابقة بطريقتين متناقضتين:

١ -جميع الطرود التي في الغرفة 27 هي أصغر من الطرود التي في الغرفة رقم 28، وعندها العبارة المقابلة هي:

(∀x)(∃y) {[Package(x) ^ Package(y) ^ In room(x,27) ^ In room(y,28)]⇒Smaller(x,y)}

٢ -يوجد طرد في الغرفة 28 أكبر من جميع الطرود الموجودة في الغرفة 27، وعندها المقابلة هي:

(∃y)(∀x) {[Package(x) ^ Package(y) ^ In room(x,27) ^ In room(y,28)]⇒Smaller(x,y)}

ويمكن ملاحظة الفرق في ترتيب الـ∃ والـ∀.

- وسنكمل ان شاء الله في المحاضرة القادمة في الـFOL.

انتهت المحاضرة