AI Notes

Viv Sedov — viv.sedov@hotmail.com

April 15, 2022

Contents

1 Propositional_logic

1.1 Simple Operations

When covering this : there are simple operations that you should know about :

Simple Operation Not:

p	$\neg p$
1	0
0	1

Such that in this case it is the opposite given value $\neg p$ would be the direct opposite of the given Value of P .

For example say that you have the following

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

In this example, we are stating the following command : $\neg(14 > 6)$ is false , we create this truth table to prove if that is true or not .

$P \wedge Q$ is true $\iff$ p and q are true

Simple $\vee$ - OR

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

In this example above , for this to hold true, atleast one of the values would have to have a one it to hold true, this is known as an *Inclusive* or as in you can say the following and it would make sense: **I will go to the shops $\vee$ i will go to the coast**

Simple Operation $\implies$ This is where if something is true the other must be true , or where you given an equivalent pointer to if p then q

p	q	$p \implies q$
1	1	1
1	0	0
0	1	1
1	1	1

With the above example this is not cause and effect , there is a reason for why this occours , and that is that there is a pointer such that it acts like an if statment, with the following code shown below :

foo = True if foo: return 1 else: return 0

The code above is rather simple , but shows that if something is true , then you would have some sort of value expression , or some pointer that would return if it is correct or not .

In the weird scenario of f and t , where if f is false it implies that q is true , that is because the q value is true , meaning that it would hold , a little trick for this one is that for what ever the second value is , if it is true , it will hold , if both are false , then it will true , though if one is true and the other is false , it will not hold .

Logically equivalent $\iff$ this can also be seen as if and only if or iff
Here is an example of the logical truth table behind this

p	q	$p \iff q$
1	1	1
1	0	0
0	1	0
0	0	1

With this , Where if something is true then the other must be true or if its false then the other would have to be false , in this case you are seeing if these two values are the same .

Example Exersises of how this would all work :
Given that :

$$p = Logicisfunforjaneq = daviddoesnotlikecabbager = davideyesareblue$$

We can then further express all of this in a notational form .

- $\neg p \wedge q$ This would imply the following truth table

$\neg p$	p	q	$\neg p \wedge q$
0	1	1	0
1	0	1	1
0	1	0	0
1	0	0	0

- $\neg R \wedge \neg P \implies$ Both are not goint to be true such that you would get only one possible answer for this .

Multiple truth table example : show the following in a truth table :

$$p \wedge (q \implies r)$$

p	q	r	$q \implies r$	$p \wedge (q \implies r)$
1	1	1	1	1
1	1	0	0	0
1	0	1	1	1
0	1	0	0	0
0	1	1	1	0
0	0	0	1	0

Here is another example to understand how these tables would all work together

$$(p \implies q) \wedge (q \implies p)$$

p	q	$q \implies p$	$p \implies q$	$(p \implies q) \wedge (q \implies p)$
1	1	1	1	1
0	1	0	1	0
1	0	1	0	1
0	0	1	1	1

Definition 1.1 Two propositions are equal if they have the same truth values , they are known as $P \implies Q$ this is known as logically equivalent

Definition 1.2 A proposition is tautology if it is always true an example of this is $P \vee \neg P$ this is always true

Definition 1.3 A proposition is a contradiction if it is always false for example $P \wedge \neg P$ this is always false

Definition 1.4 A proposition is **Contingent** if it is neither Always true or false

2 Disjunctive Normal Form

A given formula is said to be a *Disjunctive normal form* when it is an Or $\implies \vee$ this is known as **DNF** a function is conjunctive when it has an $\wedge$ form , within their proposition logic .

$$(p \wedge \neg q \wedge r) \vee (\neg q \wedge \neg r) \vee q$$

Everytime a given formula is built , we would folow the rules of propositional calculus , and how for each conjunctive formula there should be a disjunctive formula as well .

2.1 DNF of mini terms for truth tables

- For each row whose truth value is true , write doen for each of the prposition variables , of p_i in the formula of it self , either P_i is true in row or $\neg P_i$ if false.
- Repeate the first pointer , for the truth table where the formula is true and write down the dijunction of the conjuctions .

What you will rsee is that those two values will equal up such that the result of the formula in DNF is the equivalent to the original formula .

3 Conjunctive Normal Form

A formula is said to be **Conjunctive Normal Form** when its conjunction is $\wedge$ of disjunctive of $\vee$ an example of this is shown below :

$$(\neg P \vee Q \vee R \vee \neg S) \wedge (P \vee Q) \wedge \neg S \wedge (Q \vee \neg R \vee S)$$

Every expression built up according to the rules of calc , and suchj that for each conjunctive formula there is a similar or an equivelent formual that can be written in disjunctive form .

Conjunctive form , or in brackets , and on the outside
Disjunctive form, And in the brackets and or on the outside

4 Logical Equivalences

We can use this to obtain normal form , when we use the implication law to eliminate subproccess - when ever you have a doubel negation and demorgans to bring a $\neg$ you what this value to be on the outside : here are the sub process of how this can be done :

$$\neg\neg P \iff P$$

This rule is the double negation Law

$$\begin{aligned} (P \vee Q) &\iff (Q \vee P) \\ (P \wedge Q) &\iff (Q \wedge P) \\ (P \iff Q) &\iff (Q \iff P) \end{aligned}$$

Commutaive laws where both values would have to equal towards each other .

$$\begin{aligned} ((P \vee Q) \vee R) &\iff (P \vee (Q \vee R)) \\ ((P \wedge Q) \wedge R) &\iff (P \wedge (Q \wedge R)) \end{aligned}$$

This is the associative laws , where it is very similar to how they work in matricies in which they can equate towards each other .

$$\begin{aligned} ((P \vee Q) \wedge R) &\iff (P \vee (Q \wedge R)) \\ ((P \wedge Q) \vee R) &\iff (P \wedge (Q \vee R)) \end{aligned}$$

This lase is the distributive law , in which the given values would be changed within a DNF and CNF representation

$$\begin{aligned} (P \vee P) &\iff P \\ (P \wedge P) &\iff P \end{aligned}$$

Idempotent laws where the values of it self would always equal to it self no matter what .

Demorgans Law :

$$\begin{aligned} \neg(P \vee Q) &\iff (\neg P \wedge \neg Q) \\ \neg(P \wedge Q) &\iff (\neg P \vee \neg Q) \\ (P \wedge Q) &\iff \neg(\neg P \vee \neg Q) \\ (P \vee Q) &\iff \neg(\neg P \wedge \neg Q) \end{aligned}$$

Most times when you lookat demorgans law , you will notice that its very similar to the laws that have been stated above, but the thing that you want to note is that you will see that they are equal in some sense , where an or , is a direct link with Not and And it self.

§ Contrapositive Laws

$$(P \implies Q) \iff (\neg Q \implies \neg P)$$

implication that imply towards each other are contrapositive and hence you can switch out the given details of that infomation .

where If Q is an active reciever then P must be an active pointer is the same as stating if not p equates to Not q, in some sense you should understand how that would work .

§ Implication

$$(P \implies Q) \iff (\neg P \vee Q)$$
$$(P \implies Q) \iff \neg(P \wedge \neg Q)$$

§ Furhter implication

$$(P \vee Q) \iff (\neg P \implies Q)$$
$$(P \wedge Q) \iff \neg(\neg p \implies \neg Q)$$

Thi sone is rather annyoing , but the principle of how this works is very intriguing , if you do prove this via proof table you will see that they are truly equivalent: $P \vee Q$

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

this is the same as : $(\neg P \implies Q)$

P	Q	$p \implies Q$	$(\neg P \implies Q)$
1	1	1	1
0	1	1	1
1	0	0	1
0	0	1	0

If you look at the given tables above you will notice that indeed they are the same, a truth tabel may be long but they are very good at breaking down the given data that you have into something more readble .

Further Implies and equivalences

$$((P \implies R) \wedge (Q \implies R)) \iff ((P \vee Q) \implies R)$$
$$((P \implies Q) \wedge (P \implies R)) \iff ((P \implies (Q \wedge R))$$

With this law you are using the given equivelences that are shown above with the disjunctive and conjunctive views ,but within an equivelence ratio

§ Exportion Law

$$((P \wedge Q) \implies R) \iff (P \implies (Q \implies R))$$

This one is a good one , Mainly because if anything that does imply to another pointer , you can show that they are all equal towards each other .

§ Side Notes Within the compound proposition $\neg(P \vee Q)\&(\neg P \wedge \neg Q)$ they are the same , hence why when you look at the proof that is shown above you will see that they are the same .

When ever you look at equivelences you will notice that connectives $\vee\wedge$ will allways suggest that $P \vee Q \implies Q \vee P$

5 Formal definition

Definition 5.1,–,Valid An argument would be considered valid if and only if it takes a form that makes it impossible fro the premise to be true, in the sense that the conclusion is never going to be false. it is not possible to show it to be false in some sense. A formula is valid if and only if it is true under every understanding of its given argument, or its given schema, we can say it is valid if true holds for everything..

Definition 5.2,–,Sound implies Valid Valid allows us to imply Soundness

We can say that if you have a valid argument, then you also have a sound statement. am argument would be considered sound if it is valid and all the premise is true.

Definition 5.3 Sound, in logic a premise or conclusion is said to be valid if it is true under every possible understanding of its given argument, or its given schema.

1	2	1
24	5	1
7	8	1

The example their would be both valid and sound $\forall axiom \exists axiom \implies \vdash A$ What this just means is that A consist of either an axiom or can get derived from an axiom set using only the rules of inference

To Dumb this down even more, if you have statement x and you want to prove statement y, you can only do so by breaking x down into smaller statements to see if you can prove and show that y exists

$\forall x \exists y \implies \vdash y$

$\forall x \exists y \implies \vdash x \land y$

$\forall x \exists y \implies \vdash x \lor y$

$\forall x \exists y \implies \vdash x \implies y$

$\forall x \exists y \implies \vdash x \iff y$

$\forall x \exists y \implies \vdash x \leftarrow y$

$\forall x \exists y \implies \vdash x \rightarrow y$

$\forall x \exists y \implies \vdash x \leftrightarrow y$

Thing to note is that it does *Not* mean that **A is satisfied** this is a deduction but if you want to show satisfaction you would have to show

$A \ \ indicates \ \ \forall Axiom A$

$\forall AA(\models A) \implies True$

Theorem 5.1,–,Validity of statement Validity says nothing about whether or not any statement of the premises is true or not, it only says that the conclusion is true under every possible understanding of the premises. the key work their is Understanding of its own premises .

Such that validity states that its more about the form of an argument than it being true of it self.

So we can say a argument is valid if it has the proper form . An argument can have the right form, but be false.

```
Daffy Duck is a duck
All ducks are insects
their for daffy duck is a insect
```

Notice how these argument contain a form for *if x is Y* but then you see that they are not true. Notice However that if the premises **Were** True then the conclusion would also have to be true - this is a valid proof for validity. A valid argument needs not have true premises or a true conclusion.

Theorem 5.2,–,Sound requires a true premises Sound logically implies that a statement is true, this is due to the fact that, when a statement is sound, it means that it has a true premises and a true conclusion, we can formally derive x from y using this factor.

Soundness Is an argument or a factor if it means the following arguments

- It is valid
- it has a true premises

1. Sound requires both valid and to have a true premisies
2. for all valid arguments if the premises are true then the conlusion must also be true

Example :

1. All rabits are mammals
2. bugs bunny is a rabbit
3. Therfore, Bugs bunny is a mammal

In this argument we state that all of the premises are true, then the conclusion is true, so it is valid, and the premises are true, all rabbitss in fact are mammals and bugs bunny is a rabitt - so our conclusion makes sense.

Definition 5.4 Completeness

$$\alpha \models \beta \implies \alpha \vdash \beta$$

i.e if we can show something is true, then we can say that it is provable - we want to be able to prove all true statements, but you can also get false statements - such that you can prove both false and true Statements, such that if you end up proving false then your statement is no longer sound.

Definition 5.5 Soundness

$$\alpha \vdash \beta \implies \alpha \models \beta$$

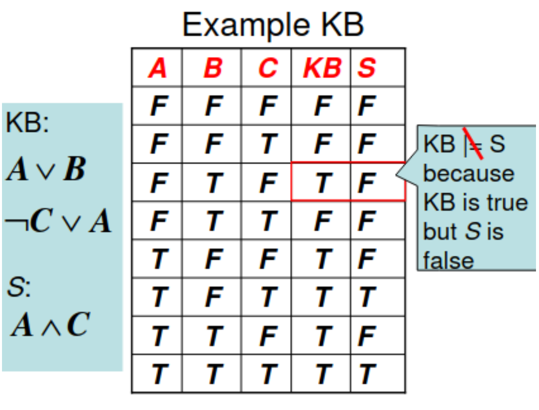
If we have a formulation i.e xy = 10, then we want to be able to show that fact. We do not want a system where we start out with something true and dedice something to be false, if we know something we should prove with our current knoweldge of breaking something down, that given statement would hold, through infierence rules.

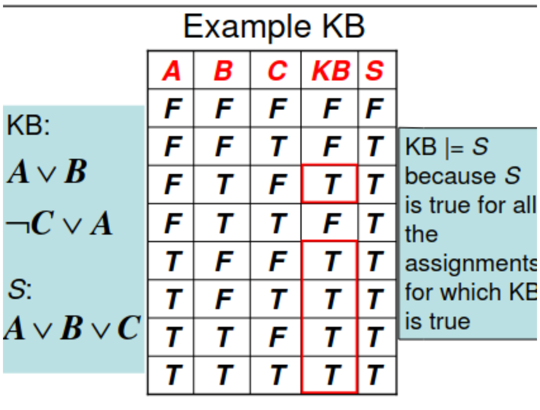
However it is conceivable that even if our system is sound, it maybe incomplete, regarding what it can express hence why it requires to have a competes property to ensure that our given formulation of our proof would hold true.

5.1 Summary without the crap

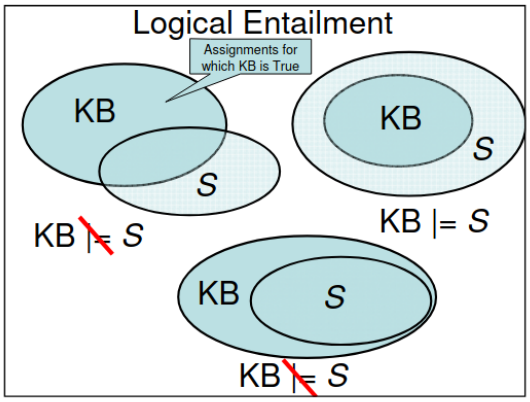
If your KB *Knowledge base* Entails Q then all interpretations (assigning true or false) values to variables that would allow you to evaluate your knowledge to True, also evaluates to Q to true $KB \models Q$

Entailment refers to how premises lead to a conclusion reacall how m(b) is a subset of m(a)





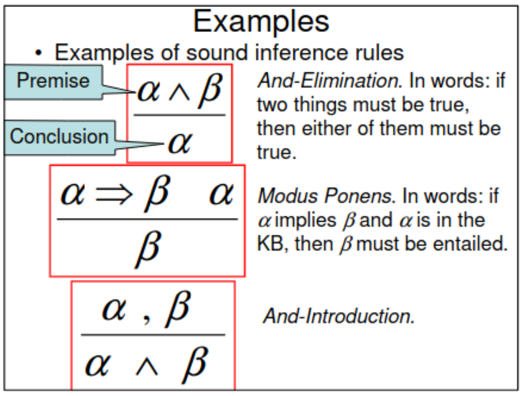
Leads into

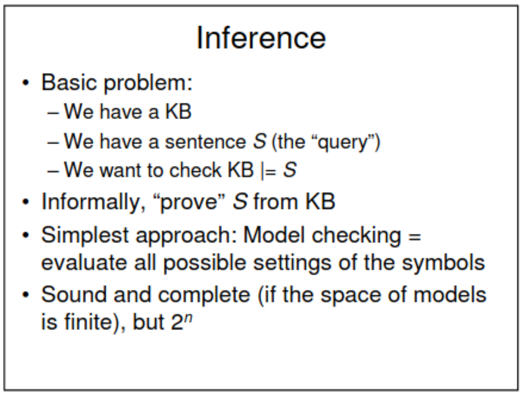


(1)

Inference is a procedure for deriving a new sentence 'p' from 'KB' following some algorithm. $KB \vdash p$ The inference algorithm is sound if it derives only sentences that are entailed by KB. The inference algorithm is complete if whatever can be entailed by KB can also be inferred from KB. Basically, an inference is an informal and less reliable kind of entailment.

- We have a kb
- We have some sentence S - query
- we want to prove S from our KB
- We say it is sound and complete if the space of model is finite within 2 pow n





5.2 Equivalences and normal form

Definition 5.6 A sentance is valid if it is true in all models

$$True, \alpha \vee \neg\alpha, \alpha \implies \alpha, (\alpha \wedge (\alpha \implies \beta)) \implies \beta$$

We can say that Validity is directctly connected through the inference of the deduction theorum

$$KB \models \alpha \iff (KB* \implies \alpha)$$

6 Resolution algorithm

- input Kb and S
- Output true if $KB \models S$ False otherwise
- Initalise a list of clauses CNF(KB and not S)
  - for each pair of clauses C i and c h

  - R implies resolution of i and j
  - new resolution is made

- If clauses are new then return false
- if cluauses unifie each other return true

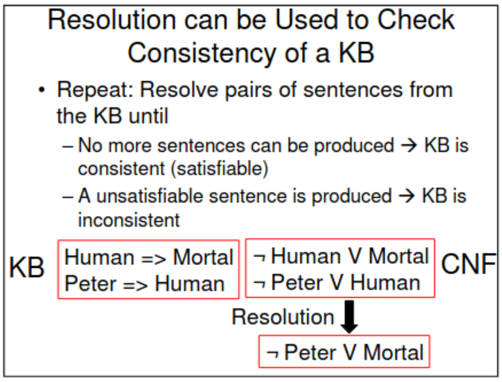
6.1 Resolution properties

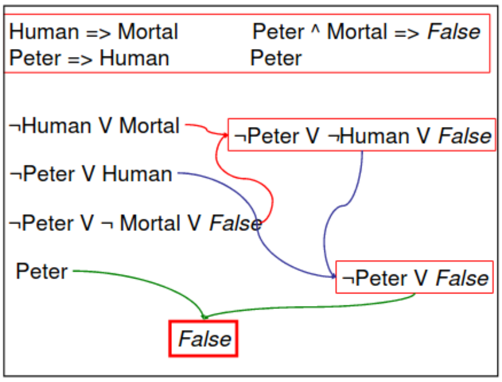
- Resolution is Sound i.e it produces a sentence that are entailed by their orginal owner
- Resolution is complete - it is guarantee to establish entailment of the query for every finite time
- Completeness is based on the key theorem

> **Theorem 6.1** If a set of clauses is unsatisfiable, then the set of all clauses that can be obtained by the resolution contains an empty clause

- So in short, we can say the opposite of a resolute theorem is that if we cannot find the empty clause the query must be satisfiable.

We use resolution to check consistency of how KB holds $Human \implies Mortal$
$Peter \implies Human$ We convert that into CNF $\neg Human \vee Mortal$
$\neg Peter \vee Human$
We can then give our resolution to this being $\neg Peter \vee Mortal$





(2)

Here in this image you can see how we can go from one proof, after another by linking them, to return if a valid argument will be true or false.

6.2 Horn Form KB

Horn Form is a clause or a form in which a logical inference problem is given : Given :

- A KB is a set of information
- A sentance $\alpha \implies theorem$

If a sentence is in KB, are restricted to some special form, some of the sound inference rules might be complete . In short its being able to convert from CNF form into implication rules
$(\alpha \vee \neg\beta) \wedge (\neg\alpha \vee \neg\gamma \vee \delta)$
$\iff$
$(\beta \implies \alpha) \wedge (\alpha \implies \gamma) \implies \delta)$ This is known as horn form normal form.

- Two inference rules that are sound and complete with respect to properistional symbols for kb in the hron normal form
  - resolution on a positive unit
  - Modus pones
- Have to be in conjuctive form $\vee \ldots \vee \ldots \vee \ldots$
- Would Follow the three basic principles
  - fact
  - Goal
  - Rule

  These are defined whith the following statement
- Rule : $ManandGel \implies Tall$ Saying Gel men are tall this can be counter proved with the following statement to prove completness $KB \models \alpha \iff (KB{*}\wedge\neg\alpha)$not provable to be satisfiable i.e unsatisfiab

  You want to end up showing that thing you are trying to prove may or may not have any models

7 Chaining

$\implies$ Just look at the slides

8 Total Summary of propositional logic

Theorem 8.1 Inference: process of deriving sentences entailed by the KB

$$KB \vdash_i \alpha$$

is a sentance where α can be derived from KB in teh process of i

For first-order logic there exists a sound and complete inference procedure - i.e. the procedure will answer any question whose answer follows from what is known by the KB.

8.1 Formal Definitions

Definition 8.1,–,Syntax *Syntax*: formal structure of how a sentence is made

Definition 8.2,–,Semantics *Symatics*: Truth of sentences with respect of the model

Definition 8.3,–,Entailment $KB \models \alpha \iff true \in \mathbf{T}, \forall Models \;\; KB$

Definition 8.4,–,Sound Derivations produced only from entailed sentances

Definition 8.5,–,Completeness Can you prove your derived formulation from - any derivations can produce the same theorm in return

8.2 Syntax

Definition 8.6,–,Atomic $\perp$, $\neg\perp$ Or P $\implies$ Propositional logic for an atom

Definition 8.7,–,Negated Atomic $\neg$ anything with that just means its negated

Definition 8.8,–,Conjuction $\wedge$

Definition 8.9,–,Disjunction $\vee$

8.3 Equivalence Norms

Definition 8.10,–,Equivalence $\equiv$

Definition 8.11,–,Implication $\implies$

Definition 8.12,–,Negated Implication $\neg$

Definition 8.13,–,Biconditional $\iff$

- Conjuctive: Conjuction of disjunctions of literals

Definition 8.14,–,CNF $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Disjunctive: Disjunction of conjuctions of literals

Definition 8.15,–,DNF $(A \wedge B) \vee (A \wedge \neg C) \vee (\neg A \wedge \neg D)$

8.4 Validity and Satisfiability wrt to KB

Theorem 8.2

$$KB \models \alpha \iff true \in \mathbf{T}Kb \models \alpha \iff (KB* \implies \alpha) \in \mathbf{T}$$

Note KB is a set of information in which $KB \implies \alpha$ Is not a wellformed formula - we get this through combining everything in the KB into one big conjuction.

8.4.1 CNTD Valid and Satisfiable

Definition 8.16 A sentence is satisfiable if it is true in some model

$$\alpha \vee \beta, \gamma$$

we can prove this by proving that α and β are true in some model and γ is true in some model we can derive values using modus pones

Definition 8.17,–,Unsatisfiable A sentence is unsatisfiable if it is false in some model

$$\alpha \wedge \neg\beta, \gamma$$

- Satisfiability is met only for the knolwedge base under the pretence that $KB \models \alpha \iff (Kb * \wedge \neg\alpha)$ which proving this will allow us to say this has no models and therefore is unsatisfiable

8.5 Modus Ponens

Definition 8.18,–,Modus Ponens $\forall \alpha \in KB \quad \alpha \models \beta \quad \beta \models \gamma \quad \gamma \models \delta \quad \delta \models \alpha$ Where
$\beta \in KB \quad \gamma \in KB \quad \delta \in KB \quad \alpha \models \gamma \quad \gamma \models \delta \quad \delta \models \alpha$

8.6 Chaining

8.6.1 Forward Chaining

Forward chaining is the process of deriving a new sentence from a set of sentences that are already known.

Theorem 8.3

$$\left.\begin{array}{l}\mathbf{KB} \vdash_i \alpha \\ \mathbf{KB} \vdash_i \beta \\ \mathbf{KB} \vdash_i \gamma \\ \mathbf{KB} \vdash_i \delta \\ \mathbf{KB} \vdash_i \epsilon\end{array}\right. \Rightarrow \alpha \models \beta \models \gamma \models \delta \models \epsilon$$

- Forward chaining is a automatic process that can be used to derive new sentences from a set of known sentences.
- May do allot of dead work

8.6.2 Backward Chaining

Backward chaining is the process of deriving a new sentence from a set of sentences that are already known. the principle is teh same as forward chaining but we start from the end of the sentence and work our way back to the beginning.

8.7 Summary

Propositional Logic does not have enough power... We have this big kb and we have to go through each one to represent it. it is impracticle, forward and backward chaining are linear time, and are only *Complete* if you are working for horn clauses , else it is rather hard to convey.

9 First Order Logic

Note : Just use implication over and when you have $\forall x \implies Y$ this is sound and works but if you have or use and with for all, then you are stating that everything and everything would be x and y

$\forall x \land y(x)$ see how that does not hold

$\exists xp \implies y$ this does not pair well

$\exists xP \land Y$ this is better

- And with Exist
- forall with Implies

9.1 Examples

- Which sentance best represents someones's mother is someone's female parent
  $\exists x, \exists y(Mother(x, y) \land (Female(x) \land Parent(x, y)))$
  Or
  $\forall x \forall y(Mother(x, y) \iff (Female(x) \land Parent(x, y)))$
  Or
  $\exists x \forall y(Mother(x, y) \implies (Female(x) \land Parent(x, y)))$ This can be quote opiniated, wrt containement, but lets read them through *Its the first and second one*
  - there exist some x and some y where there is a mother x ,y and that x is a female and that x y is a parent what that kinda means is saying that there can be a parent that is not a mother, in short you can say that y and x can be a parent but does not implcity imply that it can be a mother its a bit confusing but just go with that logic

9.1.1 Quantifiers order

The order between all given Quantifiers are not the same and can be swapped between the first order logic points

- $\forall x \forall y$ is the same as $\forall x \forall y \iff \forall y \forall x$
- $\exists x \exists y$ is the same as $\exists x \exists y \iff \exists y \exists x$
- $\forall x \exists y$ is *Not* the same as $\forall y \exists x \not\Longleftrightarrow \forall x \exists y$

- $\forall x \exists y Loves(x,y)$ Every x loves some y so *Everyone in the world is loved by atleast one person*
- $\exists x \forall y Loves(x,y)$Some x loves every Y is not the same as every x loves some y this is *There exist a person who loves everyone in the world* the reason for this is because when you have two $\forall A \forall B$ this is a pair, that you can sort through

9.1.2 Quantifier duality

Duality is important as these quantifiers are reversable So you can say something like this

$\forall x \ likes(x, icecream) \equiv \neg\exists x \ \neg likes(x, icecram) \qquad \exists x \ Likes(c, brocolai) \equiv \neg\forall x \ \neg likes(x, brocolai)$

9.2 More Examples to work on

- $\forall x, y(brother(x, y) \implies Sibling(x, y))$
  This is saying, for every x, if brother(x,y) then that logiclly implies every brother is a sibling
  $\forall xy(Sibling(x, y) \iff Sibling(y, x))$
  Every x there is a y where x and y and y and x will always be a sibling
- A first cousin, is a child of a parents sibling
  $\forall x, y(FirstCoursin(x, y) \implies \exists p, p' Parent(p, x) \wedge Sibling(P', p) \wedge Parent(P', x))$
  for every x y , you have a first cousin under the condition that x and y is a firstcoursin for x to y and that there exist some p and p prime where parent of x is P and sibling of P prime is p and parent of y is P prime

9.2.1 Wumpus world example

- Squares are breezy near a it : *Diagnostic rule*, this is where infierence would come into play, where you infer *Cause from effect* $\forall y Breeezy(y) \implies \exists x \ \text{Pit}(x) \wedge Adjacent(x, y)$
- Casual Rule : infern effect from cause $\forall x, y \ \text{Pit} \ (x) \wedge Adjacent(x, y) \implies Breezy(y)$
- None of this is complete, the point is we done imply each other , though this can be built to become complete $\forall y Breezy(y) \iff [\exists x \text{Pit}(x) \wedge Adjacent(x, y)]$

10 Usefull notes from the quiz

- Information about the knowledge base
  - If a model is true in the *Real world* then any sentance derived from that given model is sound through inference, procedure, and hence is entailed within the real world
- How do you tell if an agent is true in the real world ?
  - If the agent has sensors, that allow you to create the connection with the sentance and that the agents knoledge base is sound then it would be true within the real world
  - The ai agents learning ability generates general rules from expierence that the ai belives to be true, this is can be faliable, as this depends on how good the agent is at expressing its own information
- A satisfiable sentance is a sentance that is true in all models : *True*
- Following statements are unstatisfiable $(\alpha \wedge \beta) \wedge (\neg\alpha \wedge (\neg\alpha \implies \neg\beta))$ this will always resolve to zero
  $\Delta \wedge \neg\Delta$ This will also result in zero, anything with contains a negative will never be satisifable
- Horn Clause examples

- $(\neg B_1 \lor \neg B_2 \ldots \neg B_n \lor C)$Horn clause rule, everything is negative but one
- B This is true
- $\neg(D_1 \lor d_2 \lor d_3)$Everything is a negative -¿ Goal state
- Proof facts
  - to prove $KB \models \alpha$ is equivalent to showing that $Kb * \land \neg\alpha$ has no models
  - the expression $Kb* \implies \alpha$ and $KB \models \alpha$ are connected through deduction theorem

- ***Problems with Propositional***$_{l}ogic \implies$ ***does not support variables*** Propositinal logic has an issue where it does not support variables, due to this, your statement would have to be very large

  this is true, propositional logic works through the factor that you work with values, and cant assign anything directicly as it propositional logic follows two main principles

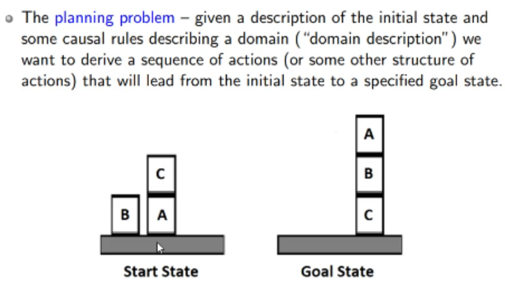
  - Declarative, pieces of syntax respond to facts
  - Compositional meaning that $p_{12} \land B_{11}$ is derived from me meaning of p over 1 2 and b over 1 1

11 Temporal Reasoning

11.1 Classification of Ai temporal reasoning problems

- Prediction problem : i.e project problem, given an inital state and casual rule, describing a domain, we want to derive the state of the world resulting from some given sequence of actions Think of how a chess game would work, thiunk of a stack, when you are going through data .

- Planning problem : Planning problem given a description of the inital state, how are you going to change to rules to get to your domain state, through what actions will allow you to get to that point ?



(3)

- Explanation problem - given some casual rules describing a domain, we want to discover facts about our given state *You wake up in the morning and you head downstairs in the kitchen theres a plate on the table and a bowl with a little milk left in, you can then say by assumption that your housemake was awoke before you and already had their breakfast*

11.2 Terminology

> **Definition 11.1,–,Fluents** Where the Truth varies over a period of time, think of them as a boolean state variable, Like when a person x lies, but sometimes they can state the truth, it would follow that logic
>
> we can also call this : *Temporal Propositional statements*

> **Definition 11.2,–,Time instances and time periods** In short, year month day, or what happend on that date, we state to be time instances for example Last year my university decided to do in person exams to F every CS student over this allows us to give an initiation of our time which leads into what occoured

We generate new fluences when a state is changed, for example in the image above, when you have your start state, b [c,[a]] in case you see that every variable there is in flux, whihch would imply that we would have to move c from a, and then move b to a etc ... to get to our goal state

11.3 Frame problem

The frame problem is about finding a great way to handle non change .

- Peristence or Intertia - the assumption that the facts are not affect by its actions - such that they would hold the same truth value after the action, as they had before hadn
- The root of the problem lies in the fact of what the domain would represent
- For smaller domains, with just a small number of fluents, and action types, we can just write it out properly.

To most AI researchers, the frame problem is the challenge of representing the effects of action in logic without having to represent explicitly a large number of intuitively obvious non-effects
types of problems

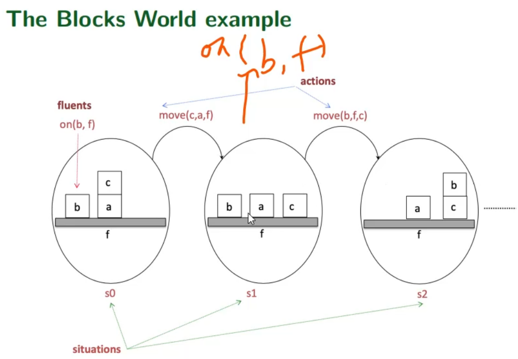
1. Qualification problem - exhuastivly specifying preconditions of an action
2. Computational fram to verify what was inferred
3. Ramification problem - problem of exhastively specifying the effects of an action - think of chess

TODO: Write more about Frame problem, as im not 100 % sure how this works

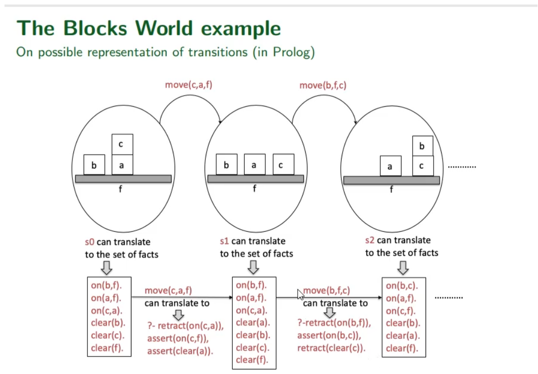
11.4 Situational Calculus

The situation calculus is a logic formalism designed for representing and reasoning about dynamical domains. It was first introduced by John McCarthy in 1963. The main version of the situational calculus that is presented in this article is based on that introduced by Ray Reiter in 1991. Wikipedia *Domain - tend to be in first order logic :*

- Fluents bassicly the item you are trying to change
- Actions the function you are pushing on those fluents
- Situations like an instance of the current state



(4)



(5)

- Remover and append operatiions within a fleucen
- Representation of actions as it stands does not check whether it is possible for a block to be moved from one place to another $\implies$ They dont check if something has been moved
- How can you improve cause and effect

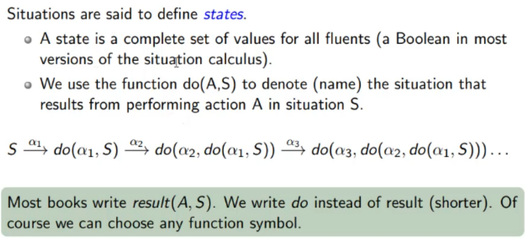
Situation Calculus is trying to deal with the reasoning of the present

We can use situational calculus to represent the current condition of a state.

11.4.1 Situational Calc state Representation

Situational Calc is defined through states

if you look at the image above we have multiple situations, from s1 to s3, with different fluents and actions that have been made upon some given variable



(6)

11.4.2 Language

11.4.3 Axioms

11.4.4 Examples

11.5 Summary