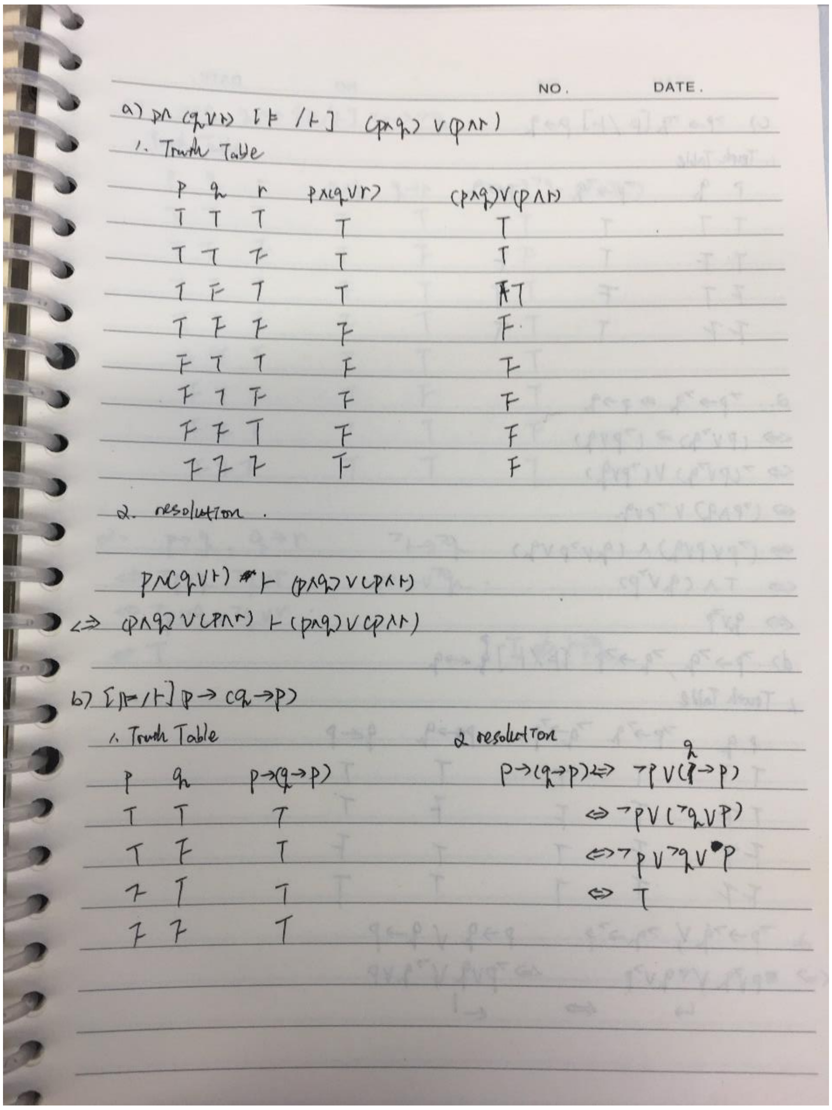
# COMP4418 Assignment 1

Question 1.



NO. DATE.

a) $p \wedge (q \vee r)$ [$\models$ / $\vdash$] $(p \wedge q) \vee (p \wedge r)$

1. Truth Table

| p | q | r | $p\wedge(q\vee r)$ | $(p\wedge q)\vee(p\wedge r)$ |
|---|---|---|---|---|
| T | T | T | T | T |
| T | T | F | T | T |
| T | F | T | T | ~~F~~T |
| T | F | F | F | F |
| F | T | T | F | F |
| F | T | F | F | F |
| F | F | T | F | F |
| F | F | F | F | F |

2. resolution.

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

$\Leftrightarrow (p\wedge q)\vee(p\wedge r) \vdash (p\wedge q)\vee(p\wedge r)$

b) [$\models$ / $\vdash$] $p \to (q \to p)$

1. Truth Table

| p | q | $p\to(q\to p)$ |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | T |

2. resolution

$p\to(q\to p) \Leftrightarrow \neg p \vee (q \to p)$

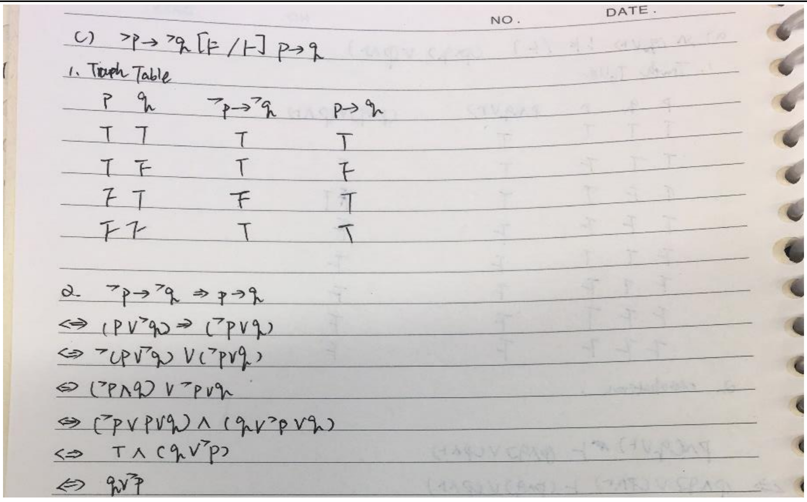
$\Leftrightarrow \neg p \vee (\neg q \vee p)$

$\Leftrightarrow \neg p \vee \neg q \vee p$

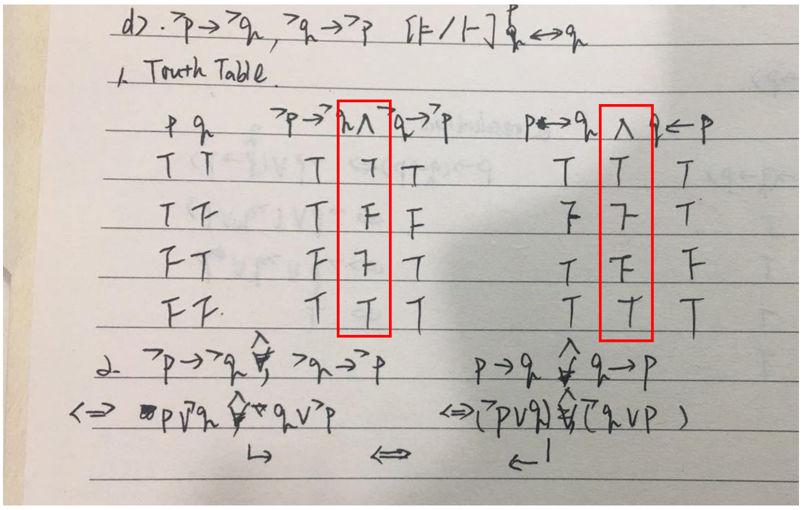
$\Leftrightarrow T$

a) From the truth table, we can see the left is always the same as the right. So this sentence is proved.

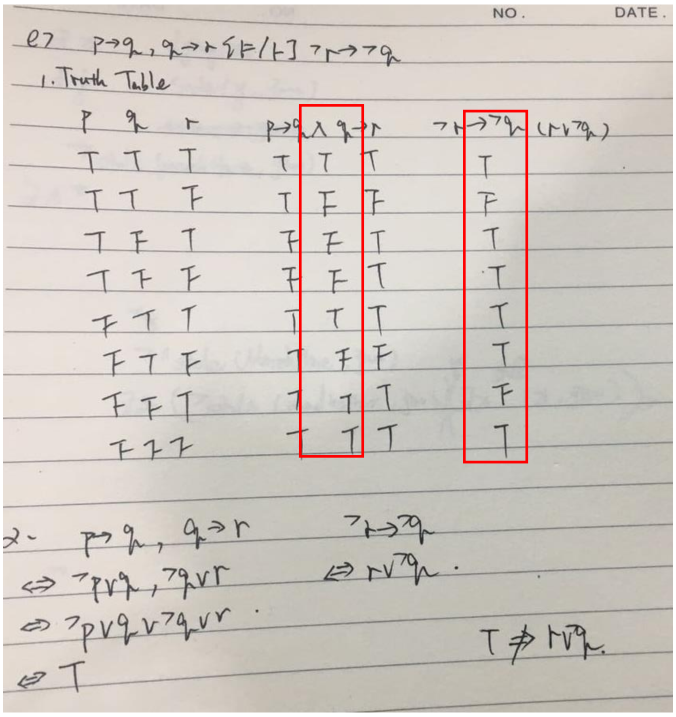
b) Because there's nothing in the left, so only if the right be always true the sentence is proved. And the right is all true.
So it is Proved.




c) From the truth table, the left and right are not all the same, so it is not proved.




d) Regard ',' as an 'and', and from the truth table we can see that the left is same as the right, so it is proved.




e) Regard ',' as an 'and', from the truth table we can see the left is not same as right. So it is not proved.

Question 2.

(a)

The main information about the story is as the following table:

| | |
|---|---|
| marchHare, | "I never stole the jam!" pleaded the March Hare. |
| madHatter | One of us stole it, but it wasn't me! |
| doormouse | "At least one of them did," replied the Doormouse |
| Truth | the March Hare and the Doormouse were not both speaking the truth. |
| Jam Lying(x) Stole(x,y). | |

Hence, the First-order-logic facts are:

| | | |
|---|---|---|
| marchHare, | $\exists \mathrm{x}(\neg \mathrm{stole}(x, \mathrm{jam}) \cap (x = MarchHare))$ | (1) |
| madHatter | $\exists \mathrm{x}(\mathrm{stole}(\mathrm{x}, \mathrm{jam}) \cap (x \neq MadHatter))$ | (2) |
| doormouse | $\exists \mathrm{x} \exists \mathrm{y} \neg(\mathrm{Lying(x)} \cap \mathrm{Lying(x)} \cap (\mathrm{x} = \mathrm{MarchHare}) \cap (\mathrm{y} = \mathrm{MadHatter}))$ | (3) |
| Truth | $\exists \mathrm{x} \exists \mathrm{y}(\mathrm{Lying(x)} \cup \mathrm{Lying(y)} \cap (\mathrm{x} = \mathrm{MarchHare}) \cap (\mathrm{y} = \mathrm{Doormouse}))$ | (4) |

(b)

No.

To be semantical, we assume that the jam is stolen by MarchHare, MadHatter and DoorMouse respectively.

| Assume stolen by | Result |
|---|---|
| MarchHare | So (1) is False and (2) is true because Madhatter didn't steal the jam and there really exist one thief in Doormouse and MarchHare and so (3) is proved to be True. So (4) is satisfied. |
| MadHatter | So (2) is False, and (1) is True and (3) is True but (4) is conflicted. |
| DoorMouse | So (1) (2) are both True and (3) is True so (4) is conflicted. |
| There are still other conditions such as there are two thieves among the 3 people. | |
| MarchHare, MadHatter | So both (1)(2) are False, and (3) is False, either. (4) is satisfied. |
| MarchHare, DoorMouse | So (1) is False and (2) is False because the number of thieves is not one. Then (3) is False, (4) is conflicted. |
| MadHatter, DoorMouse | So (2) is False and (1) is True and (3) is True, (4) is conflicted. |
| If all of them stole | So (1) (2) Both False and (3) is False either. (4) is satisfied. |
| If none of them stole | So (1) is True, and (2) is False, (3) is True, (4) is conflicted. |

Above all, we can learn from the table that these information cannot prove who really stole the jam. There are still 3 possible conditions.

(c)

From the interpretation of the question (b) we can know that there are still 3 possible condition, to make the condition to be the only solution I think we should add a clause: There can only exist one thief. In FOL should be like:

$$\exists \mathrm{x}, \mathrm{stole}(\mathrm{x}, \mathrm{jam}) \cap ((\mathrm{x} = \mathrm{MarchHare}) \cup (\mathrm{x} = \mathrm{MadHatter}) \cup (\mathrm{x} = \mathrm{Doormouse}))$$

(d)

1.$\exists \mathrm{x} \exists \mathrm{y}(\mathrm{Lying(x)} \cup \mathrm{Lying(y)} \cap (\mathrm{x} = \mathrm{MarchHare}) \cap (\mathrm{y} = \mathrm{Doormouse}))$ [Premise]

2.$\exists \mathrm{x}, \mathrm{stole}(\mathrm{x}, \mathrm{jam}) \cap ((\mathrm{x} = \mathrm{MarchHare}) \cup (\mathrm{x} = \mathrm{MadHatter}) \cup (\mathrm{x} = \mathrm{Doormouse}))$ [Premise]

3.$\exists \mathrm{x} \neg \mathrm{Lying(x)} \cap \mathrm{x} = \mathrm{MadHatter}$ [Conclusion from 1,2]

4.$\exists \mathrm{x}, \mathrm{stole}(\mathrm{x}, \mathrm{jam}) \cap \mathrm{x} = \mathrm{MarchHare}$ [Conclusion from 1,3]

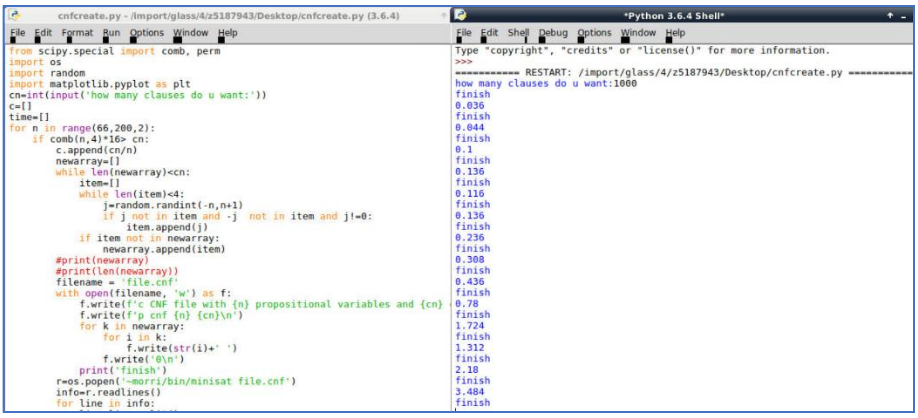
So now, it can be proved that the real thief is MarchHare.

Question 3

I wrote a simple programming in Python 3, and the codes is attached.

The main components of this program include the CNF generator, using OS module to execute the Minisat tool and collecting the CPU time of every case. At last I plot all the data in a graph by matplotlib module.

To make CNFs, I use random module to generate numbers from minus n to plus n (except zero). All the clauses should have 4 different numbers (different absolute numbers), and every clause should be different.

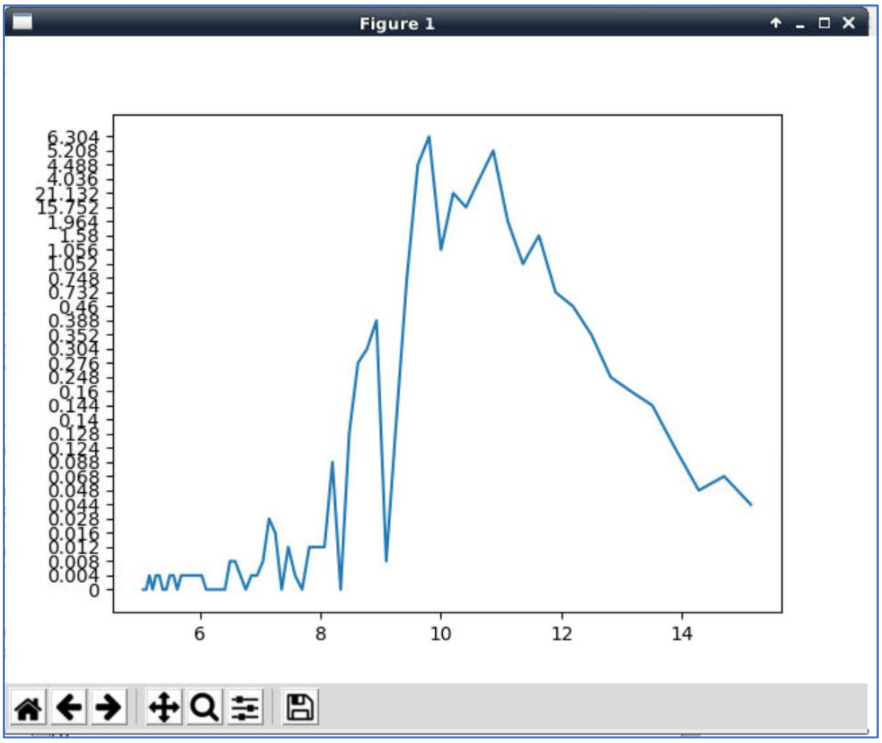



I use comb(m,n) to keep the number of clauses can be available with different number of variables.

$$C = \frac{Cn}{n}$$

Cn=the number of clauses n=the number of variables

I use 1000 clauses and (66,200) with 2-step a time collection which can control the C between (5,15). The graph of all the data is as follows:




From the graph, we can apparently see that the peak of the trend should between 9 and 11. So for 4-SAT questions the most difficult C should in (9,11),   C $\approx$ 10. And the graph also tells that it is looks like an easy-difficult-easy question, because it looks like a hill with an increasing trend at first and a decreasing trend at last.

Question 4

The Introduction of Description logic language (DL)

| | |
|---|---|
| Logical symbols | |
| Punctuation | [ ] ( ) |
| Positive integers | 1,2,3··· |
| Concept-forming operators | ALL, EXIST, FILLS, AND |
| Connectives | ⊑,≐,→ |
| Nonlogical symbols | |
| Atomic concepts | written in capitalized mixed case, e.g., Person, WhiteWine, FatherOfOnlyGirls;<br>DL also has a special atomic concept: Thing |
| Roles | written like atomic concepts, but preceded by “: ,”<br>e.g., :Child, :Height, :Employer, :Arm; |
| Constants | written in uncapitalized mixed case, e.g., desk13, maryAnnJones. |
| There are four types of legal syntactic expressions in DL: constants, roles (both seen earlier), concepts, and sentences. We use c and r to range over constants and roles, respectively, d and e to range over concepts, and a to range over atomic concepts. The set of concepts of DL is the least set satisfying the following:<br>every atomic concept is a concept; | |

- if r is a role and d is a concept, then [ALL r d] is a concept;
- if r is a role and n is a positive integer, then [EXISTS n r] is a concept;
- if r is a role and c is a constant, then [FILLS r c] is a concept;
- if d1 …dn are concepts, then [AND d1 …d n] is a concept.

Finally, there are three types of sentences in DL:]

- if d1 and d2 are concepts, then (d1⊑d2) is a sentence;
- if d1 and d2 are concepts, then (d1≐d2) is a sentence; .
- if c is a constant and d is a concept, then (c → d) is a sentence.

Examples:

| | |
|---|---|
| [**AND** Wine [**FILLS** :Color red] [**EXISTS** 2 :GrapeType]] | Represent the category of a blended red wine (literally, a wine, one of whose colors is red, and which has at least two types of grape in it) |

```
(ProgressiveCompany ≐ [AND Company
                           [EXISTS 7 :Director]
                           [ALL :Manager [AND Woman
                                              [FILLS :Degree phD]]]
                           [FILLS :MinSalary $24.00/hour]])
```

The concept on the right-hand side represents the notion of a company with at least seven directors, and all of whose managers are women with Ph.D.s and whose minimum salary is $24.00/hour. The sentence as a whole says that ProgressiveCompany, as a concept, is equivalent to the one on the right. If this sentence is in a KB, we consider ProgressiveCompany to be fully defined in the KB, that is, we have a set of necessary and sufficient conditions for being a ProgressiveCompany, exactly expressed by the right-hand side. If we used the connective ⊑ instead, the sentence would say only that ProgressiveCompany as a concept was subsumed by the one on the right. Without a ≐ sentence in the KB defining it, we consider ProgressiveCompany to be a primitive concept in that we only have necessary conditions it must satisfy.
As a result, although we could draw conclusions about an individual ProgressiveCompany once we were told it was one, we would not have a way to recognize an individual definitively as a ProgressiveCompany.

Inference rules:

Simple definitions:

for the distinguished concept Thing, $\mathcal{I}[\text{Thing}] = \mathcal{D}$;

$\mathcal{I}[[\textbf{ALL}\ r\ d]] = \{x \in \mathcal{D} \mid \text{ for any } y, \text{ if } \langle x,y\rangle \in \mathcal{I}[r], \text{ then } y \in \mathcal{I}[d]\}$;

$\mathcal{I}[[\textbf{EXISTS}\ n\ r]] =$
$\{x \in \mathcal{D} \mid \text{there are at least } n \text{ distinct } y \text{ such that } \langle x,y\rangle \in \mathcal{I}[r]\}$;

$\mathcal{I}[[\textbf{FILLS}\ r\ c]] = \{x \in \mathcal{D} \mid \langle x, \mathcal{I}[c]\rangle \in \mathcal{I}[r]\}$;

$\mathcal{I}[[\textbf{AND}\ d_1 \ldots d_n]] = \mathcal{I}[d_1] \cap \ldots \cap \mathcal{I}[d_n]$.

| | |
|---|---|
| Truth in Interpretation | $\Im \models (c \rightarrow d)$ iff $\mathcal{I}[c] \in \mathcal{I}[d]$;<br>$\Im \models (d \sqsubseteq d')$ iff $\mathcal{I}[d] \subseteq \mathcal{I}[d']$;<br>$\Im \models (d \doteq d')$ iff $\mathcal{I}[d] = \mathcal{I}[d']$. |
| Entailment | (Surgeon ⊑ Doctor) |

<table>
<tr><td></td><td>KB $\models$ ([AND Surgeon Female] $\sqsubseteq$ Doctor).<br>(Surgeon $\doteq$ [AND Doctor [FILLS :Specialty surgery]]).<br>$(c \rightarrow d)$, where $c$ is a constant and $d$ is a concept;<br>$(d \sqsubseteq e)$, where $d$ and $e$ are both concepts.[5]</td></tr>
<tr><td>Normalization</td><td>1. expand definitions<br>2. flatten the AND operators<br>3. combine the ALL operators<br>4. combine the EXISTS operators<br>5. deal with<br>6. remove redundant expressions</td></tr>
<tr><td>The result of normalizaition</td><td>$$\begin{array}{l}[\textbf{AND}\ a_1\ \ldots\ a_m \\ \quad [\textbf{FILLS}\ r_1\ c_1]\ \ldots\ [\textbf{FILLS}\ r_{m'}\ c_{m'}] \\ \quad [\textbf{EXISTS}\ n_1\ s_1]\ \ldots\ [\textbf{EXISTS}\ n_{m''}\ s_{m''}] \\ \quad [\textbf{ALL}\ t_1\ e_1]\ \ldots\ [\textbf{ALL}\ t_{m'''}\ e_{m'''}]\,]\end{array}$$</td></tr>
<tr><td colspan="2">Applicaions of DL</td></tr>
<tr><td colspan="2">• Assertion and Query<br>• Contradiction Detection in Configuration<br>• Classification and Contradiction Detection in Knowledge Acquisition<br>• Assertion and Classification in Monitoring Scenarios<br>• Working Memory for a Production System<br>• Using Concepts as Queries and Access to Databases</td></tr>
<tr><td colspan="2">The advantages of DL are:<br>Firstly, it can represent state, action and planning in a uniform way, which is different from scenario calculus;<br>Next, it can perform efficient reasoning, the satisfiability problem under the framework and the inclusion of detection problems are polynomial time;<br>In addition, it has clear semantics;<br>At last, it can automatically perform planning and identification.</td></tr>
<tr><td colspan="2">The disadvantages of DL are:<br>However, the knowledge represented by the description logic is difficult to parse and transmit compared with XML-based files. If there is no specific inference engine that cannot parse the description logic language, there are many restrictions on its use. The knowledge base represented by the description logic is difficult to reuse and share, and there are not many platforms that support the description logic. If the concept of description logic can be combined with the ontology representation based on XML description, the above problem can be solved. Implementation reuse and sharing.</td></tr>
</table>