COMP4418 Assignment 2

Question 1

a)

instance	v(1...6). e(1,2). e(2,1). e(1,3). e(3,1). e(1,4). e(4,1). e(2,4). e(4,2). e(2,5). e(5,2). e(2,6). e(6,2). e(3,4). e(4,3). e(3,5). e(5,3). e(3,6). e(6,3). e(4,5). e(5,4). e(5,6). e(6,5).
Generator rule	k{c(X): v(X)}k.
Test rule	:-not e(X,Y), c(X),c(Y),X!=Y.
	#show c/1.

```
v(1..6).
e(1,2). e(2,1).
e(1,3). e(3,1).
e(1,4). e(4,1).
e(2,4). e(4,2).
e(2,5). e(5,2).
e(2,6). e(6,2).
e(3,4). e(4,3).
e(3,5). e(5,3).
e(3,6). e(6,3).
e(4,5). e(5,4).
e(5,6). e(6,5).
k{c(X): v(X)}k.
:-not e(X,Y), c(X),c(Y),X!=Y.
#show c/1.
```

b)

K=	Result
3	d:\4418>clingo --const k=3 -n 0 D:\4418\clique.lp clingo version 5.3.0 Reading from D:\4418\clique.lp Solving... Answer: 1 c(5) c(2) c(4) Answer: 2 c(5) c(3) c(4) Answer: 3 c(5) c(6) c(2) Answer: 4 c(5) c(6) c(3) Answer: 5 c(1) c(2) c(4) Answer: 6 c(1) c(3) c(4) SATISFIABLE Models : 6 Calls : 1 Time : 0.071s (Solving: 0.06s 1st Model: 0.00s Unsat: 0.00s) CPU Time : 0.000s
4	d:\4418>clingo --const k=4 -n 0 D:\4418\clique.lp clingo version 5.3.0 Reading from D:\4418\clique.lp Solving... UNSATISFIABLE Models : 0 Calls : 1 Time : 0.011s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s) CPU Time : 0.016s
5	d:\4418>clingo --const k=5 -n 0 D:\4418\clique.lp clingo version 5.3.0 Reading from D:\4418\clique.lp Solving... UNSATISFIABLE Models : 0 Calls : 1 Time : 0.006s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s) CPU Time : 0.000s
6	d:\4418>clingo --const k=6 -n 0 D:\4418\clique.lp clingo version 5.3.0 Reading from D:\4418\clique.lp Solving... UNSATISFIABLE Models : 0 Calls : 1 Time : 0.014s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s) CPU Time : 0.000s

Question 2

S	Reduct PS	Stable model?
{a,b,c,d}	d←a. d←b. d←c.	×
{a,b,c}	d←a. d←b. d←c.	×
{a,b,d}	d←a. d←b. d←c.	×

{a,c,d}	d←a. d←b. d←c.	×
{b,c,d}	d←a. d←b. d←c.	×
{a,b}	d←a. d←b. d←c.	×
{a,c}	d←a. d←b. d←c.	×
{a,d}	d←a. d←b. d←c. a.	√
{b,c}	d←a. d←b. d←c.	×
{b,d}	d←a. d←b. d←c. b.	√
{c,d}	d←a. d←b. d←c. c.	√
{a}	d←a. d←b. d←c. a.	×
{b}	d←a. d←b. d←c. b.	×
{c}	d←a. d←b. d←c. c.	×
{d}	d←a. d←b. d←c. a. b. c.	×
{}	d←a. d←b. d←c. a. b. c.	×

Question 3

a)

Sat-naïve	More than 5 minutes.
Sat-up	0.001688 s
Sat-cdcl	0.001155 s

b) Briefly explain why the run times differ:

SAT-naïve is a way to gain a topology tree whose node are the different conditions of every variable to find a satisfiable solution of CNF problems and SAP-up add unit propagation in to the process of SAT-naïve which can earlier to cut the subtree when it would cause some conflicts. SAT-cdcl is an approach which add conflict driven clause learning into SAT-up, which can earlier than SAT-up to cut off the conflict subtree.

Question 4

a)

False. Because from the Tseitin transformation algorithm, we can know that the time complexity should be polynomial not exponential.

b)

True. Because SAT can only solve discrete problems with discrete input and output. IF the problem is continuous, SAT cannot solve it very well. And there are still some undecidable decision problem, for example: the Halting Problem. But all problems in NP can be reduced to SAT efficiently.

c)

True. Because the algorithm of UP is :

- Let $I^0 = I$
- Repeat for $j > 0$ until $I^j = I^{j+1}$:
  - If there is a $(x_1 \vee \ldots \vee x_k) \in \phi$ with $\bar{x}_1, \ldots, \bar{x}_k \in I^j$: Return conflict $(x_1 \vee \ldots \vee x_k)$
  - If there is a $(x_1 \vee \ldots \vee x_{k+1}) \in \phi$ with $\bar{x}_1, \ldots, \bar{x}_k \in I^j$: Let $I^{j+1} = I^j \cup \{x_{k+1}\}$
- Return I^j

Hence if I close to φ and I U {x} close to φ, it can be inferred that $\bar{x}$ in φ.

Question 5

I	clauses and watched literals		
	pVqVrVs	$p \vee \bar{q} \vee \bar{t}$	pVt
	p,q	p, $\bar{q}$	p,t
$\bar{p}$	q,r	$\bar{q}, \bar{t}$	t
t		$\bar{q}, \bar{t}$	
$\bar{q}$	r,s		

Question 6

a) K Happy∧K ¬Happy

Suppose e, w is an interpretation.

$e, w \vDash$ K Happy∧K ¬Happy iff $e, w \vDash$ K Happy and $e, w \vDash$ K ¬Happy

$e, w \vDash$ K Happy iff for all $w' \in e$: $e, w' \vDash$ Happy (1)

$e, w \vDash$ K ¬Happy iff for all $w' \in e$: $e, w' \vDash$ ¬Happy (2)

(1) and (2) are conflict, there's no e, w' can satisfy (1), (2) at the same time, so K Happy ∩ K ¬Happy is unsatisfiable.

b) K(HappyVsad) → ¬K Happy

Suppose e, w is an interpretation.

$e, w \vDash$ K(HappyVsad); we need to show $e, w \vDash$ ¬K Happy.

$e, w \vDash$ K(HappyVsad) $iff\ for\ all\ w'$ ∈ e: $e. w' \vDash$ HappyVsad.

$e, w' \vDash$ HappyVsad iff $e. w' \vDash$ Happy or $e. w' \vDash$ Sad

$e, w \vDash$ K(HappyVsad) iff for all $w' \in$ e: $e. w' \vDash$ Happy or $e. w' \vDash$ Sad (1)

The right side of the clause means:

$e, w \vDash$ ¬K Happy iff e, w does not satisfy K Happy

e, w does not satisfy K Happy iff for not all $w' \in e$: $e. w' \vDash$ Happy (2)

There are 3 conditions of clause (1):

$iff\ all\ w'$ ∈ e: $e. w' \vDash$ Happy (3)

$iff\ some\ w'$ ∈ e: $e. w' \vDash$ Happy and some $w'' \in e$: $e. w'' \vDash$ Sad (4)

$iff\ all\ w'$ ∈ e: $e. w' \vDash$ Sad (5)

If e, w is condition (3), $means\ that\ for\ all\ w'$ ∈ e: $e. w' \vDash$ Happy, falsifies $e, w \vDash$ ¬K Happy.

If e, w is condition (4) or (5), $means\ that$ for not all $w' \in e$: $e. w' \vDash$ Happy which satisfies $e, w \vDash$ ¬K Happy.

Hence K(HappyVsad) → ¬K Happy is not valid but satisfiable.

Question 7

a) Positive effect axiom: □∀a∀x(a = pickUp(x)) → [a]Holding(x))

Negative effect axiom:

$$\square\forall a\forall x\forall y(a = putOn(y)) \vee (y = T \bigwedge a = putOnTable) \wedge Holding(x) \rightarrow [a]\neg Holding(x))$$

SSA for Holding(x):

$\square\forall a\forall x\forall y([a]\mathrm{Holding}(x) \leftrightarrow (a = \mathrm{pickUp}(x))\vee(\mathrm{Holding}(x)\wedge(a \neq \mathrm{putOn}(y)\vee(y = T\wedge a \neq putOnTable)))$

b) Regression:

$$\square\forall a\forall x\forall y\Big([a]\mathrm{On}(x,y) \leftrightarrow \big(\mathrm{Holding}(x) \wedge (a = \mathrm{putOn}(y) \vee (y = \mathrm{T} \wedge a = \mathrm{putOnTable}))\big) \vee \big(\mathrm{On}(x,y) \wedge a \neq \mathrm{pickUp}(x)\big)\Big).$$

We want to prove the following clause which ϕ is nothing and Σ is the SSA of On(x, y).

$$\phi\wedge\Sigma \vDash [\mathrm{pickUp}(B)][[\mathrm{putOn}(c)]\mathrm{On}(B,C)$$

Iff $\phi \vDash R([\mathrm{pickUp}(B)][[\mathrm{putOn}(c)]\mathrm{On}(B,C))$

Using the ASS of On(x, y):

Iff $\phi \vDash R([\mathrm{pickUp}(B)]\gamma_{On}{}^{a}_{putOn(c)}{}^{x}_{b}{}^{y}_{c})$

Because y=C a=putOn(C) doesn't satisfy $(y = \mathrm{T} \wedge a = \mathrm{putOnTable})$ so we drop this part.

Iff $\phi \vDash R([\mathrm{pickUp}(B)]\big(Holding(B)\wedge\big(putOn(C) = putOn(C)\big) \vee \ldots\big)$

Iff $\phi \vDash R[\mathrm{pickUp}(B)](Holding(B)\wedge(putOn(C) = putOn(C)) \vee \ldots)$

Using the ASS of Holding(x, y):

Iff $\phi \vDash R\Big(\gamma_{Holding}{}^{a}_{pickUp(b)}{}^{x}_{b}\wedge((putOn(C) = putOn(C)) \vee \ldots)\Big)$

Iff $\phi \vDash ((\mathrm{pickUp}(B) = \mathrm{pickUp}(B))\wedge\big(putOn(C) = putOn(C)\big) \vee \ldots$

$\phi \vDash ((\mathrm{pickUp}(B) = \mathrm{pickUp}(B))\wedge\big(putOn(C) = putOn(C)\big) \vee\ldots$

$[\mathrm{pickUp}(B)][[\mathrm{putOn}(c)]\mathrm{On}(B,C)$ is always satisfied, so it is valid.

c) If clear(x) redundant?

In my opinion, it is redundant.

$$((\forall\ y\ \neg\mathrm{On}(y,x)) \bigwedge \neg\mathrm{Holding}(x)) \rightarrow clear(x)$$