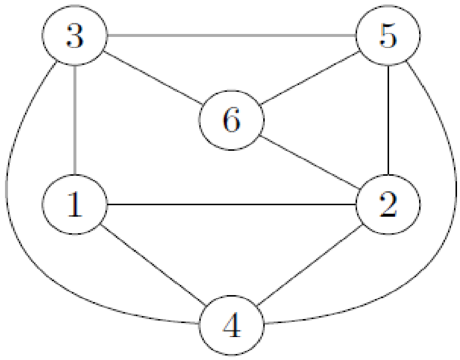
Answer – 1



a) ASP program below takes in a given graph input decides clique problem

Clique.lp-

```
v(1..6).
e(1, 2). e(1, 3). e(1, 4).
e(2, 1). e(2, 4). e(2, 5). e(2, 6).
e(3, 1). e(3, 4). e(3, 5). e(3, 6).
e(4, 1). e(4, 2). e(4, 3). e(4, 5).
e(5, 2). e(5, 3). e(5, 4). e(5, 6).
e(6, 2). e(6, 3). e(6, 5).

%since a clique is a subset of vertices V. C ⊆ V
k {c(X) : v(X)} k.
%unary predicate c, unary predicate v and binary predicate e
%x ∈  C and y ∈ C, if x ≠ y, then{x;y} ∈ E literal conversion to clingo
:- c(X), c(Y), X!=Y, v(X), v(Y), not  e(X,Y), not e(Y,X).

#show clique_are.
```

b) The number of k-clique for 3 is 6 and the rest are unsatisfiable. Since, at maximum only 3 vertices say 1,3,4 are adjacent to each other. If you try to extend by one vertex let's say 6 then it is not adjacent to 1 and 4. Similarly, if we try 5 then it is only adjacent to 3 and 4 not 1. Finally, for 2 the adjacent vertices are 1 and 4 but not 3. Hence the maximum clique number this graph has is 3.

Answer – 2

S	**Reduct P^S**	**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.	✗
{a,c}	d←a. d←c.	✗
{a,d}	d←a. d←b. d←c. a.	✓
{b,c}	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. a.	×
{b}	d←b. b.	×
{c}	d←c. c.	×
{d}	d←a. d←b. d←c. a. b. c.	×
{}	d←a. d←b. d←c. a. b. c.	×

Answer – 3

(a) The naïve solver did not produce a report after 5 minutes and was terminated. While the Unit propagation solver were satisfiable and generated a report with top line as SATISFIABLE (in 0.000689 s, sat-up) and the conflict driven clause learning method produced a similar report with just the difference in time period, SATISFIABLE (in 0.000576 s, sat-cdcl) using the given implementation of algorithms. Thus, the run time can be taken as sat-naïve, sat-up and sat-cdcl as timeout, 0.000689 s, 0.000576 s with CDCL being the fastest among them.

(b) The run time differs because naïve method would explode on the number of literals itself. While unit propagation is able to solve the problem using the watched literal scheme which still has a large structure but saves time in backtracking and has much faster unit propagation where it still propagates inside a clause of a formula. Since Unit propagation is still expensive if every clause has the same conflict. Thus, Conflict driven is the fastest Clause learning algorithm as it learns from a clause where the literals cause a conflict and avoids setting those literals to other interpretations.

Answer – 4

(a) Converting a propositional formula into an equisatisfiable CNF formula in the worst case requires exponential time (under the assumption P ≠ NP). – False. Normally, if it was "equivalence" condition then the worst-case scenario would be true but since it is "equisatisfiable" and it is preserved in the transformation it grows only linearly.

(b) There are decision problems that cannot be reduced to SAT (if so, name a concrete problem). – False. All NP problems can be reduced to SAT and others can be solved using various SAT solver algorithms which are just an extension of SAT or modern implementation.

(c) If I is closed under unit propagation relative to ϕ, then in order to close I ∪ {x} under unit propagation relative to ϕ it suffices to inspect the clauses c ∈ ϕ that watch $\bar{x}$. – True. To close x under I you have to suffice clause c ∈ ϕ that watch $\bar{x}$ and vice versa is true as well.

Answer – 5

I	Clauses and Watched Literals		
	p V q V r V s	p V $\bar{q}$ V $\bar{t}$	p V t
	p,q	p, $\bar{q}$	p,t

$\bar{p}$	r,q	$\bar{t}, \bar{q}$	p,t
t		$\bar{t}, \bar{q}$	
$\bar{q}$	r,s		
$\bar{r}$	r,s		
s			

I worked out extra interpretation steps just to reach the end of the problem and result satisfiability.

Answer – 6

(a) **Proof**: It is not valid and unsatisfiable since the knowledge world cannot know and unknow the same formula, i.e., Happy because knowledge is true in all possible world. So if e,w satisfies **K** Happy then for all w' ϵ e, e,w' ⊨ Happy. But if e,w satisfies **K** ¬Happy then for all w' ϵ e, e,w' ⊨ ¬Happy. Since only either are satisfiable or (**K** Happy V **K** ¬Happy) is valid, thus both are unsatisfiable together, i.e (**K** Happy ^ **K** ¬Happy) is not valid/satisfiable.

(b) **Proof:** Since e,w are in interpretation and we suppose e,w ⊨ ¬**K** (Happy V Sad); we need to show that e,w ⊨ ¬**K** Happy. Therefore two possible worlds are {¬Happy, ¬Sad} and {¬Happy}. Then for all w' ϵ e, e,w' ⊨ ¬Happy ^ ¬Sad , therefore e,w' ⊨ ¬Happy. Thus for all all w' ϵ e, e,w' ⊨ ¬Happy. Hence e,w ⊨ ¬**K** Happy. Here, we know that some world satisfies ¬Happy ^ ¬Sad. That means that not all world satisfies Happy V Sad. That means that a world exists which does not falsify Happy. Therefore, some world exists which satisfies ¬Happy. And thus, the formula is satisfiable but not valid.

Answer – 7

(a) $\Box\forall a \forall x \forall y \left([a]Holding(x) \leftrightarrow \left(Holding(x)\ \wedge\ (a \neq putOn(y)\ V\ (\ y = T\ \wedge\ a \neq putOnTable))\right)\ V\ (On(x,y) \wedge a = pickUp(x))\ \right)$

(b) Let Σ contain SSA On(x,y) and Holding(x) from (a), other STRIPS model known formula of pickup, putOn and putOnTable,etc.
Also, abbreviating Holding(x) by H(x), pickUp(x) by pu(x), putOn(x) by po(x), and putOnTable(x) by pot(x).
We try regression as follows,

$$\phi \wedge \Sigma \models [pu(B)][po(C)]On(B,C)$$

$$\text{Iff, } \phi \models R\big([pu(B)][po(C)]On(B,C)\big)$$

$$\phi \models R\left([pu(B)]\, \gamma\, On \begin{array}{ccc} a & x & y \\ po(C) & B & C \end{array}\right)$$

$$\phi \models R\left([pu(B)]\left(H(B) \wedge \big(po(C) = po(C) \vee (C = T \wedge po(C) = pot)\big)\right)\right)$$

Since, simplifying as (C = T ∧ po(C) = pot) is trivially false, whereas po(C) = po(C) is trivially true, so the whole formula can be implied to H(B). Continuing the regression,

$$\phi \vDash R\ ([pu(B)]H(B))$$

$$\phi \vDash R\ \gamma\ H \begin{matrix} a & x \\ pu(B) & B \end{matrix}$$

$$\phi \vDash \Big(H(B)\ \wedge\ \big(pu(B) \neq po(y)\, V\, (\, y = T\ \wedge\ pu(B)\ =\ pot)\big)\Big)\ V\ \big(On(B,y) \\ \wedge\ pu(B) = pu(B)\big)$$

$$\phi \vDash \cancel{\Big(H(B)\ \wedge\ \big(pu(B) \neq po(y)\, V\, (\, y = T\ \wedge\ pu(B)\ =\ pot)\big)\Big)}\ V\ \big(On(B,y) \\ \wedge\ pu(B) = pu(B)\big)$$

Since considering second part for easier simplification pu(B) = pu(B) is trivially true and On(B,A) is true for initial state where A ∈ y then formula is valid and ϕ entails the regression making the resultant formula valid even if not H(B).

(c) Clear(x) is very useful predicate and is used as a precondition for STRIPS formalization. Although in SSA it can be made redundant by substituting the formula

$$(Holding(x) \vee \neg\exists y On(y,x))$$

Here we assume that when the robot is holding a block makes it clear on the top or no other block exists that is On the block in question.