# Logic in Computer Science - Exam 2014

Nikolaj Dybdahl Rathcke (rfq695)

29 October 2014

# 1: Propositional Logic

## Question 1.1

### a(i)

We show the sequent $\vdash (p \to \neg q) \to q \to \neg p$ by natural deduction

| | | |
|---|---|---|
| 1 | $p \to \neg q$ | assumption |
| 2 | $q$ | assumption |
| 3 | $\neg\neg q$ | $\neg\neg I(2)$ |
| 4 | $\neg p$ | $MT(1,3)$ |
| 5 | $q \to \neg p$ | $\to I(2-4)$ |
| 6 | $(p \to \neg q) \to q \to \neg p$ | $\to I(1-5)$ |

### a(ii)

We show the sequent $\neg p \vee \neg\neg q \vdash p \to q$ by natural deduction

| | | |
|---|---|---|
| 1 | $\neg p \vee \neg\neg q$ | premise |
| 2 | $p$ | assumption |
| 3 | $\neg p$ | assumption |
| 4 | $\bot$ | $\neg E(2,3)$ |
| 5 | $q$ | $\bot E(4)$ |
| 6 | $\neg\neg q$ | assumption |
| 7 | $q$ | $\neg\neg E(6)$ |
| 8 | $q$ | $\vee E(1,3-5,6-7)$ |
| 9 | $p \to q$ | $\to I(2-8)$ |

## b(i)

We want to show the rule, $\frac{\phi \to \psi \vee \chi}{\phi \wedge \neg\psi \to \chi}$, is derivable by the following proof schema.

| | | |
|---|---|---|
| 1 | $\phi \to \psi \vee \chi$ | premise |
| 2 | $\phi \wedge \neg\psi$ | assumption |
| 3 | $\phi$ | $\wedge E_1(2)$ |
| 4 | $\psi \vee \chi$ | $\to E(3,1)$ |
| 5 | $\psi$ | assumption |
| 6 | $\neg\psi$ | $\wedge E_2(2)$ |
| 7 | $\bot$ | $\neg E(5,6)$ |
| 8 | $\chi$ | $\bot E(7)$ |
| 9 | $\chi$ | assumption |
| 10 | $\chi$ | $\vee E(4, 5-8, 9)$ |
| 11 | $\phi \wedge \neg\psi \to \chi$ | $\to I(2-10)$ |

## b(ii)

We want to show the rule, $\frac{\phi \wedge \psi \to \chi}{\phi \to \neg\psi \vee \chi}$, is derivable by the following proof schema.

| | | |
|---|---|---|
| 1 | $\phi \wedge \psi \to \chi$ | premise |
| 2 | $\phi$ | assumption |
| 3 | $\psi \vee \neg\psi$ | $LEM$ |
| 4 | $\psi$ | assumption |
| 5 | $\phi \wedge \psi$ | $\wedge I(2,4)$ |
| 6 | $\chi$ | $\to E(5,1)$ |
| 7 | $\neg\psi \vee \chi$ | $\vee I_2(6)$ |
| 8 | $\neg\psi$ | assumption |
| 9 | $\neg\psi \vee \chi$ | $\vee I_1(8)$ |
| 10 | $\neg\psi \vee \chi$ | $\vee E(3, 4-7, 8-9)$ |
| 11 | $\phi \to \neg\psi \vee \chi$ | $\to I(2-10)$ |

# Question 1.2

### a(i)

We want to determine if, $(p \to \neg q) \land (\neg q \to \neg p)$, is satisfiable and they are valid by the truth table

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

So we can conclude it is satisfiable but **not** valid.

### a(ii)

We want to determine if, $\neg(p \vee q) \land (\neg p \to q)$, is satisfiable and they are valid by the truth table

| $p$ | $q$ | $\neg(p \vee q)$ | $(\neg p \to q)$ | $\neg(p \vee q) \land (\neg p \to q)$ |
|---|---|---|---|---|
| F | F | T | F | F |
| F | T | F | T | F |
| T | F | F | T | F |
| T | T | F | T | F |

So we can conclude it is neither satisfiable or valid.

## b(i)

We fill out the 2 remaining definition for $U(\cdot)$.

$$U(\phi_1 \land \phi_2) \stackrel{def}{=} (U(\phi_1) \to U(\neg\phi_2)) \to \bot$$

$$U(\phi_1 \vee \phi_2) \stackrel{def}{=} U(\neg\phi_1) \to U(\phi_2)$$

Where we have used the previously defined $U(\neg\phi)$ in our definition for these two.

## b(ii)

We prove this by the following truth table and by rewriting $U(\cdot)$ to their translations, but keeping $U(\phi)$.

| $\phi_1$ | $\phi_2$ | $\phi_1 \vee \phi_2$ | $(U(\phi_1) \to \bot) \to U(\phi_2)$ | $\phi_1 \land \phi_2$ | $(U(\phi_1) \to U(\phi_2) \to \bot) \to \bot$ |
|---|---|---|---|---|---|
| F | F | F | F | F | F |
| F | T | T | T | F | F |
| T | F | T | T | F | F |
| T | T | T | T | T | T |

And we can see that the evaluations are the same.

# 2: Propositional Logic

## Question 2.1

### a

We want to determine if $\Phi_1$ and $\Phi_2$ are satisfiable, valid and/or a HornCNF formula.

We construct a truth table to determine satisfiability and validity.

| $p$ | $q$ | $\Phi_1 = (p \vee q) \wedge (\neg p \vee \neg q)$ | $\Phi_2 = p \wedge (\neg p \vee q) \wedge (\neg p \vee \neg q)$ |
|---|---|---|---|
| F | F | F | F |
| F | T | T | F |
| T | F | T | F |
| T | T | F | F |

Then, for $\Phi_1$, we see that

- It is satisfiable.
- It is **not** valid.
- It is **not** a HornCNF formula as it has two positive literals in the first clause.

For $\Phi_2$, we see that

- It is **not** satisfiable.
- It is **not** valid.
- It is a HornCNF formula.

### b

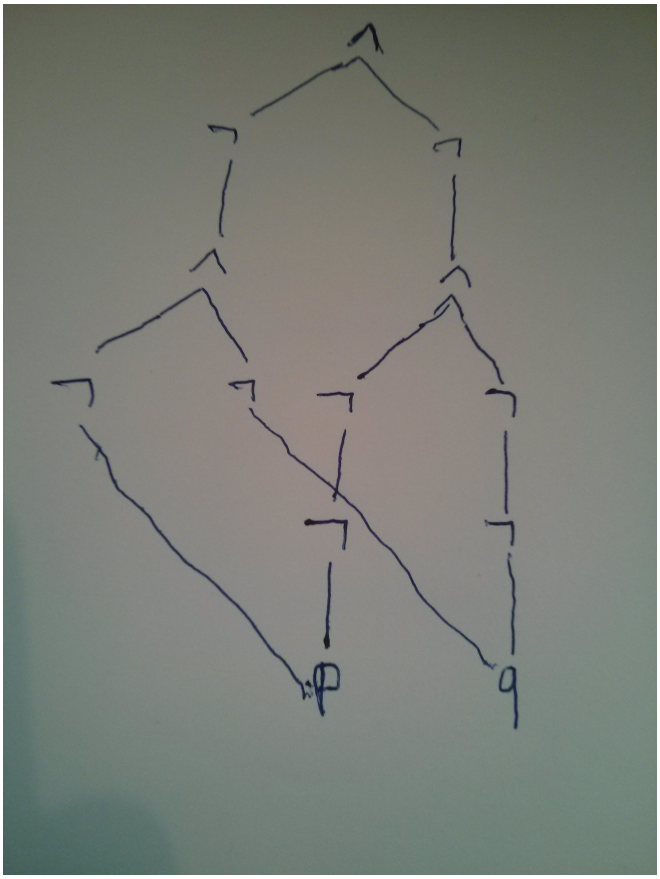
We write the translated formula $T(\Phi_1)$.

From [**H&R**, page 69] we can rewrite the formula to

$$\begin{aligned} T(\Phi_1) &= (\neg(\neg T(\phi_1) \wedge \neg T(\phi_2))) \wedge (\neg(\neg\neg T(\phi_1) \wedge \neg\neg T(\phi_2))) \\ &= (\neg(\neg T(p) \wedge \neg T(q))) \wedge (\neg(\neg\neg T(p) \wedge \neg\neg T(q))) \\ &= (\neg(\neg p \wedge \neg q)) \wedge (\neg(\neg\neg p \wedge \neg\neg q)) \end{aligned}$$

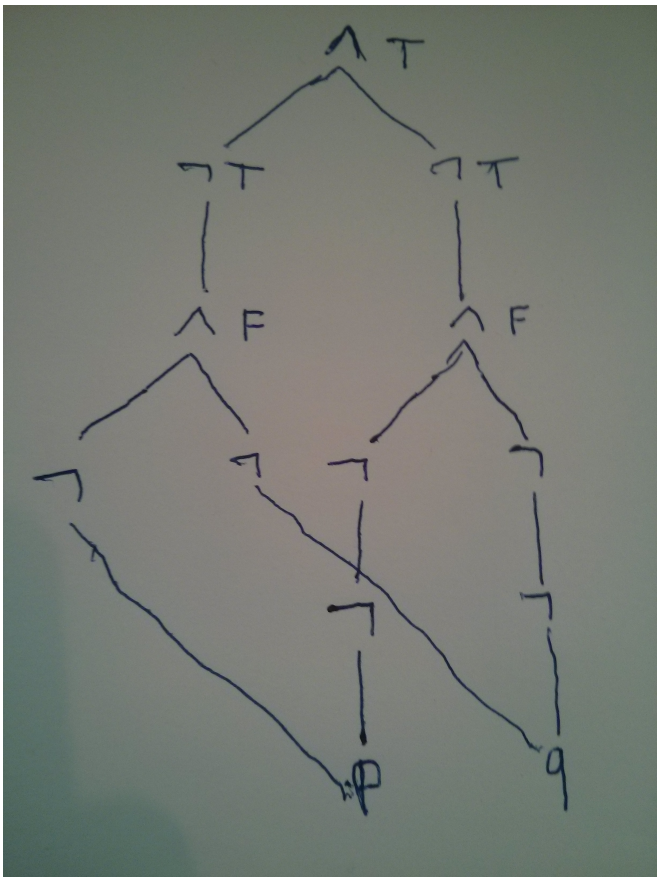
**c**

In the picture below, the DAG can be seen.



**d**

After running the linear SAT solver, we get the following




Since we can not determine anything from 2 false "and"'s, we are stuck and therefore it is undecidable.

## Question 2.2

We want to determine if the statements (a-f) are true or false by providing proof or a counter example.

**a**

True. If $\Phi$ is valid, then it is by definition also satisfiable as satisfiability requires one evaluation to be true and validity requires all evaluations to be true, so this is easily seen to be true.

**b**

False. Demonstrated by the following example

$$\Phi = p \wedge q$$

Where it is true for $p = T$ and $q = T$ (satisfiable), but not for $p = F$ and $q = F$.

**c**

True. As $\neg\phi$ says that one evaluation of $\neg\Phi$ is true, this means that not all evaluations of $\Phi$ are true, and thus it is not valid.

**d**

True. If $\Phi$ is not always true, that means there is one evaluation to false, which means $\neg\Phi$ is true and is then satisfiable.

**e**

True. If $\neg\Phi$ is always true, that means all evaluations of $\Phi$ are false and it is therefore not valid.

**f**

False. As demonstrated by the following example

$$\Phi = p \wedge q$$

Where an evaluation of $\Phi$ where $p = F$ and $q = F$ is not true and $\Phi$ is not valid.
And the evaluation of $\neg\Phi$ where $p = T$ and $q = T$ is not true and $\neg\Phi$ is not valid.

## Question 2.3

**a**

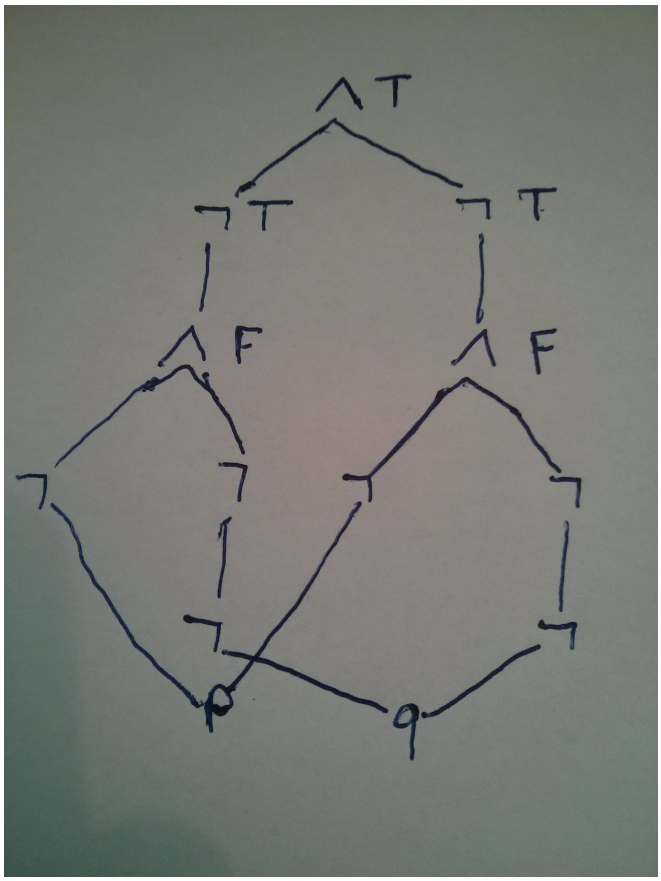
Consider the formula

$$\Phi = \neg(p \vee \neg q) \wedge \neg(p \vee \neg q)$$

It is a HornCNF formula as there are no clauses with more than one positive literal. Then we get the following translation $T(\Phi)$.

$$T(\Phi) = (\neg(\neg p \wedge \neg\neg q)) \wedge (\neg(\neg p \wedge \neg\neg q))$$

We can now draw the DAG and run the linear solver, so we get




Which gets stuck when the two "and"'s are false.

# 3: Predicate Logic

## Question 3.1

### i

This translates into the following predicate logic formula

$$\forall x(P(x) \land \neg\forall y(L(y) \rightarrow K(x,y)))$$

### ii

This translates into the following predicate logic formula

$$\exists x\exists y(P(x) \land L(y) \land K(x,y) \land \forall z(L(z) \land K(x,z) \rightarrow y = z))$$

## 3.2

### a

We want to prove the sequent, $\forall x \forall y(R(x,y) \rightarrow \neg R(y,x)) \vdash \forall x \forall y(R(x,y) \rightarrow \neg(x = y))$ with natural deduction for predicate logic.

| | | |
|---|---|---|
| 1 | $\forall x \forall y(R(x,y) \rightarrow \neg R(y,x))$ | premise |
| 2 | $x_0$ | |
| 3 | $\forall y(R(x_0,y) \rightarrow \neg R(y,x_0))$ | $\forall E(1)$ |
| 4 | $y_0$ | |
| 5 | $R(x_0,y_0) \rightarrow \neg R(y_0,x_0)$ | $\forall E(3)$ |
| 6 | $R(x_0,y_0)$ | assumption |
| 7 | $\neg R(y_0,x_0)$ | $\rightarrow E(6,5)$ |
| 8 | $x_0 = y_0$ | assumption |
| 9 | $R(x_0,x_0)$ | $= E(8,6; \phi \stackrel{def}{=} R(x_0,x))$ |
| 10 | $\neg R(x_0,x_0)$ | $= E(8,7; \phi \stackrel{def}{=} \neg R(x,x_0))$ |
| 11 | $\bot$ | $\neg E(9,10)$ |
| 12 | $\neg(x_0 = y_0)$ | $\neg I(8-11)$ |
| 13 | $R(x_0,y_0) \rightarrow \neg(x_0 = y_0)$ | $\rightarrow I(6-12)$ |
| 14 | $\forall y R(x_0,y) \rightarrow \neg(x_0 = y)$ | $\forall I(4-13)$ |
| 15 | $\forall x \forall y(R(x,y) \rightarrow \neg(x = y))$ | $\forall I(2-14)$ |

### b

For it to be consistent, we need to find a model that satisfies both $\phi$ and $\psi$.
Consider the model $\mathcal{M} = (A, \cdot, L^{\mathcal{M}})$ with universe $A = a, b, c$ and $L^{\mathcal{M}} = (a,b), (b,c), (c,a)$.
This means $\forall v \forall w(R(v,w) \rightarrow \neg R(w,v))$ is satisfiable as you can only get $[[F \rightarrow T]], [[F \rightarrow F]], [[T \rightarrow \neg F]]$ which are all true - or $\mathcal{M} \vdash \phi$.
Furthermore $\forall v \exists w(R(v,w))$ is satisfiable as you can always pick an element $w$ no matter the element $v$ - or that $\mathcal{M} \vdash \psi$.
This means that our answer uses completeness as we can not prove anything is wrong.

# 4: Linear-Time Temporal Logic

## Question 4.1

We find an LTL formula equivalent $S\phi$ given by

$$G((\phi \rightarrow X(\neg\phi) \wedge XX(\phi)) \wedge F(\phi))$$

Now, we want to show that the formal semantics of our formula is equivalent to $S$. First, lets rewrite the $G$ according to its formal semantics

$$\forall i \geq 1.\pi^i \models \phi$$

Then we rewrite the implication and the right side of the last `and`

$$\forall i \geq 1.[\pi^i \models \phi \text{ implies } \pi \models \psi \text{ and } [\exists j \geq 1.\pi^{i+j} \models \phi]]$$

Now we rewrite the last part - $X(\neg\phi) \wedge XX(\phi)$ which is the part that corresponds to $\psi$

$$\forall i \geq 1.[(\pi^i \models \phi \text{ implies } [\text{not}(\pi^{i+1} \models \phi) \text{ and } \pi^{i+2} \models \phi]) \text{ and } [\exists j \geq 1.\pi^{i+j} \models \phi]]$$

Which corresponds to the semantics given in the assignment text.

## Question 4.2

**a**

We want to show the equivalence $GS\phi \equiv S\phi$ using the semantics for $S$. We start with the semantics for $GS\phi$ and rewrite, so

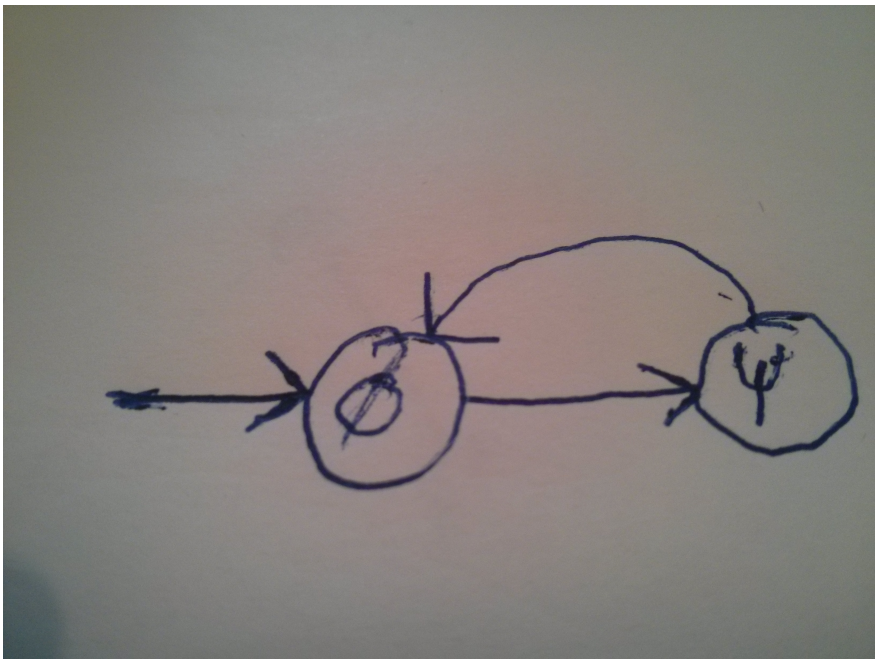
$$\begin{aligned}
&GS\phi \\
&\Leftrightarrow \forall k \geq 1.[pi^k \models \forall i \geq 1.[(\pi^i \models \phi \text{ implies } [\text{not}(\pi^{i+1} \models \phi) \text{ and } \pi^{i+2} \models \phi]) \text{ and } [\exists j \geq 1.\pi^{i+j} \models \phi]]] \\
&\Leftrightarrow \forall i \geq 1.[(\pi^i \models \phi \text{ implies } [\text{not}(\pi^{i+1} \models \phi) \text{ and } \pi^{i+2} \models \phi]) \text{ and } [\exists j \geq 1.\pi^{i+j} \models \phi]] \\
&\Leftrightarrow S\phi
\end{aligned}$$

Where the last step before we conclude the equivalence is due to the easily seen equivalence $GG\phi \equiv G\phi$.

## Question 4.3

### a

This equivalence does not hold as demonstrated below



Where $S\phi \vee S\psi$ (for both actually) but $S(\phi \vee \psi)$ does not as it holds in every state.

### b

This equivalence does not hold as demonstrated by the same model shown in (4.3a). The reasoning is the same, as both $S\phi$ and $S\psi$ holds, but not $S(\phi \vee \psi)$.

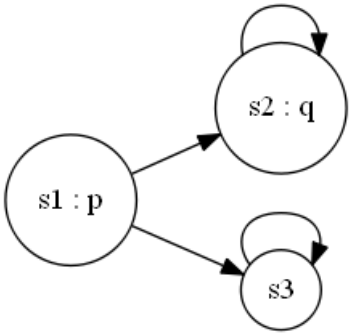
### c

**The assignment text says** $S\phi \wedge S\phi \equiv S[\phi \wedge \psi]$.
This means, again, we can use our model from (4.3a) to show the equivalence does not hold as $S\phi$ holds, and thus $S\phi \wedge S\phi$ holds. However, there is not state where $\phi \wedge \psi$ and $S(\phi \wedge \psi)$ can not hold.

# 5: Computation Tree Logic

## Question 5.1

Below is seen a model where both statements hold

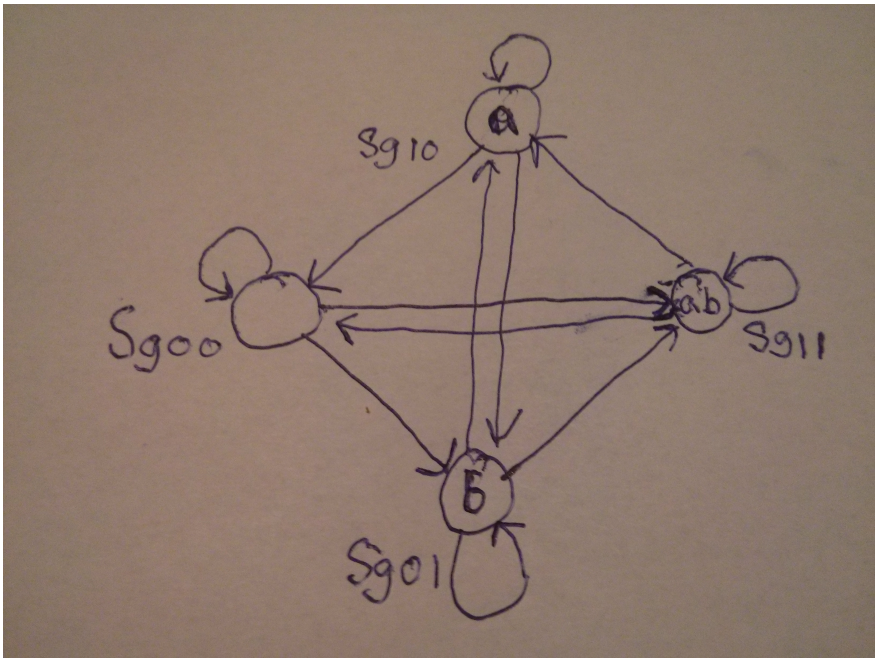



The model holds as $p$ holds until there exists a next where $q$ holds (transition from s1 to s2). However, $q$ does hold for all next states from s1 (transition from s1 to s3).
In appendix A1, you can see the NuSMV code and terminal output. The code also be seen in `5-1.smv`. These conclude that the model satisfies both properties.

## Question 5.2

### a

We name the states $S = \{s_{g00}, s_{g01}, s_{g10}, s_{g11}\}$ with the transitions and labeling as shown below.



**b**

The NuSMV code can be found in `5-2-b.smv` or in appendix A2. It corresponds to the model as there are 4 states implemented and there are 3 transitions to from each each state to another as implemented.

**c**

A CTL formula has been given and implemented as a specification in `5-2-b.smv` or in A2. This gives us that the specification is true for our model.

## Question 5.3

We want to compute the set $S = [[a \rightarrow E[aUAG(b \vee c)]]]$ using the SAT algorithm.

We note that

$$SAT(a) = \{0, 3, 4, 7\}$$
$$SAT(b) = \{0, 1, 5\}$$
$$SAT(c) = \{0, 2, 6\}$$

Then we compute $SAT(a \rightarrow E[aUAG(b \vee c)])$.

$$\begin{aligned} SAT(\phi) :&= SAT(a \rightarrow E[aUAG(b \vee c)]) \\ &= SAT(\neg a \vee E[aUAG(b \vee c)]) \\ &= SAT(\neg a) \cup SAT(E[aUAG(b \vee c)]) \\ &= S - SAT(a) \cup SAT_{EU}(a, AG(b \vee c) \\ (*) \quad &= (S - \{0, 3, 4, 7\}) \cup \{0, 1, 2, 3, 4\} \\ &= \{0, 1, 2, 3, 4, 5, 6, 8\} \end{aligned}$$

(*) is obtained from the following calculations.

$$
\begin{aligned}
W &:= SAT(a) \\
&= \{0,3,4,7\} \\
X &:= S \\
Y &:= SAT(AG(b \vee c)) \\
&= SAT(\neg EF \neg (b \vee c)) \\
&= S - SAT(EF \neg (b \vee c)) \\
&= S - SAT(E[\top U \neg (b \vee c)) \\
&= S - SAT_{EU}(\top, \neg (b \vee c)) \\
(**) \quad &= S - \{0,3,4,5,6,7,8\} \\
&= \{1,2\} \\
X &:= Y = \{1,2\} \\
Y &:= \{1,2\} \cup (\{0,3,4,7\} \cap \text{pre}_{\exists}(\{1,2\})) \\
&= \{1,2\} \cup (\{0,3,4,7\} \cap \{0,1,2,3\}) \\
&= \{1,2\} \cup \{0,3\} \\
&= \{0,1,2,3\} \\
X &:= Y = \{0,1,2,3\} \\
Y &:= \{0,1,2,3\} \cup (\{0,3,4,7\} \cap \text{pre}_{\exists}(\{0,1,2,3\})) \\
&= \{0,1,2,3\} \cup (\{0,3,4,7\} \cap \{0,1,2,3,4\}) \\
&= \{0,1,2,3\} \cup \{0,3,4\} \\
&= \{0,1,2,3,4\} \\
X &:= Y = \{0,1,2,3,4\} \\
Y &:= \{0,1,2,3,4\} \cup (\{0,3,4,7\} \cap \text{pre}_{\exists}(\{0,1,2,3,4\})) \\
&= \{0,1,2,3,4\} \cup (\{0,3,4,7\} \cap \{0,1,2,3,4,5\}) \\
&= \{0,1,2,3,4\} \cup \{0,3,4\} \\
&= \{0,1,2,3,4\}
\end{aligned}
$$

(**) is obtained from the following calculations.

$$
\begin{aligned}
W &:= SAT(\top) \\
&= S \\
X &:= S \\
Y &:= SAT(\neg(b \vee c)) \\
&= S - SAT(b \vee c)) \\
&= S - (SAT(b) \cup SAT(c)) \\
&= S - (\{0,1,5\} \cup \{0,2,6\}) \\
&= S - (\{0,1,2,5,6\} \\
&= \{3,4,7,8\} \\
X &:= Y = \{3,4,7,8\} \\
Y &:= \{3,4,7,8\} \cup (S \cap \text{pre}_{\exists}(\{3,4,7,8\})) \\
&= \{3,4,7,8\} \cup (S \cap \{0,3,4,5,6,7,8\} \\
&= \{3,4,7,8\} \cup \{0,3,4,5,6,7,8\} \\
&= \{0,3,4,5,6,7,8\} \\
X &:= Y = \{0,3,4,5,6,7,8\} \\
Y &:= \{0,3,4,5,6,7,8\} \cup (S \cap \text{pre}_{\exists}(\{0,3,4,5,6,7,8\})) \\
&= \{0,3,4,5,6,7,8\} \cup \{0,3,4,5,6,7,8\} \\
&= \{0,3,4,5,6,7,8\}
\end{aligned}
$$

Which means our resultare the states $= \{0,1,2,3,4,5,6,8\}$.

# 6: Binary Decision Diagrams

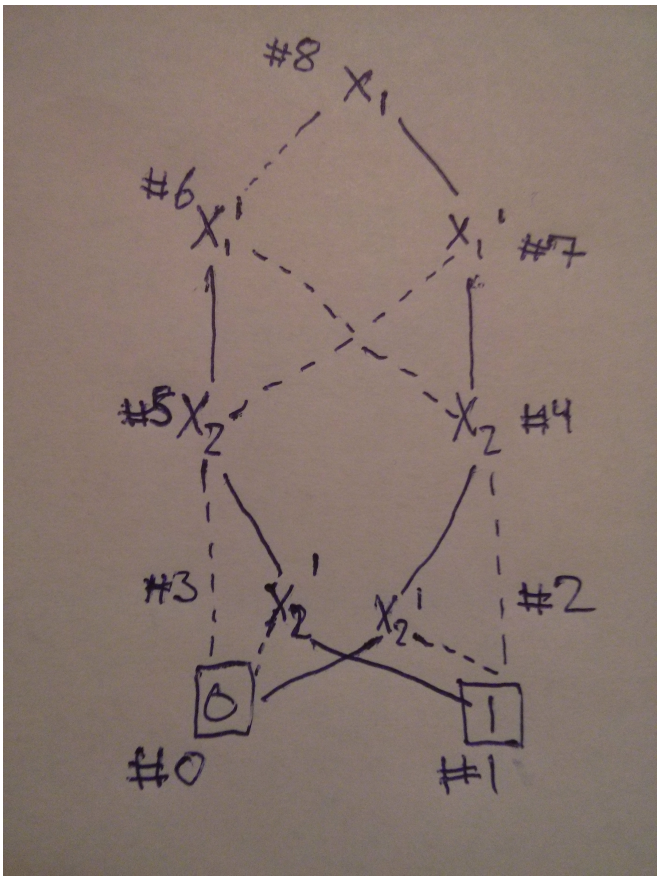
## Question 6.1

### a(i)

From appendix A3, The OBDD is provided using the `reduce` algorithm

**a(ii)**

From the OBDD in (a(i)), it allows us to create the following ROBDD.

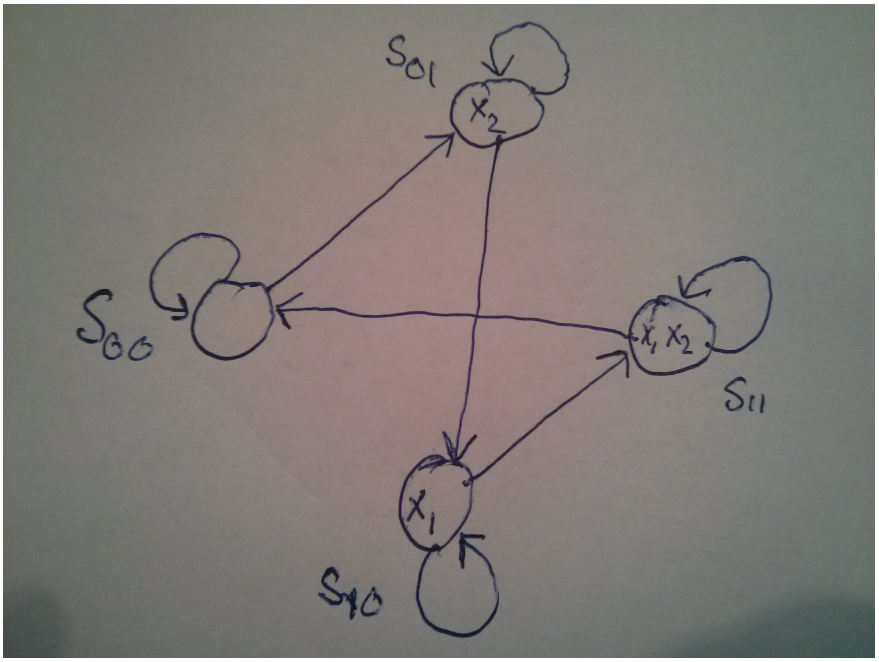


**b**

We create the following truth table

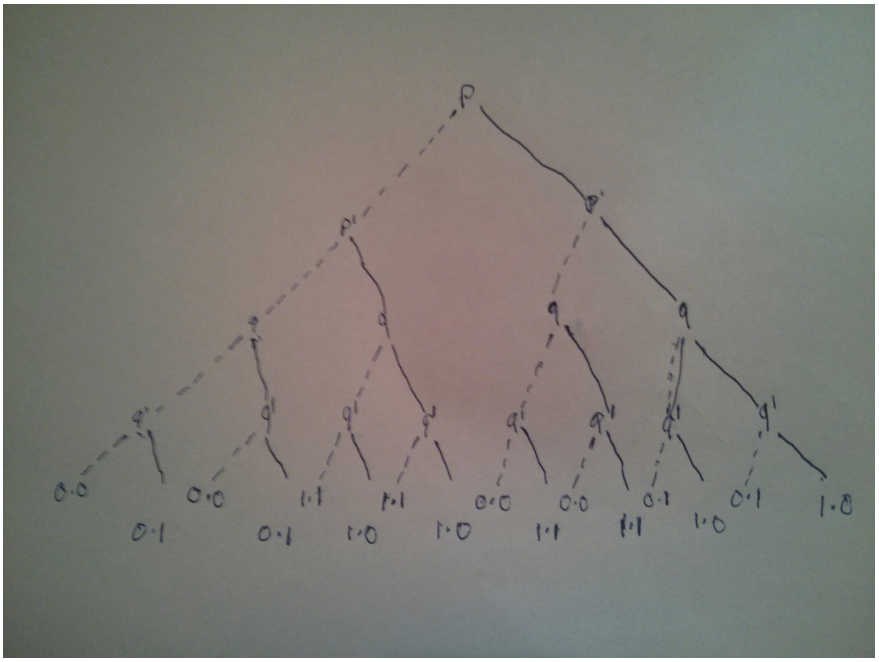
| $x_1$ | $x_1'$ | $x_2$ | $x_2'$ | $\rightarrow$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

This means we can produce our CTL model, by making the transitions where the truth table produces 1 as seen below

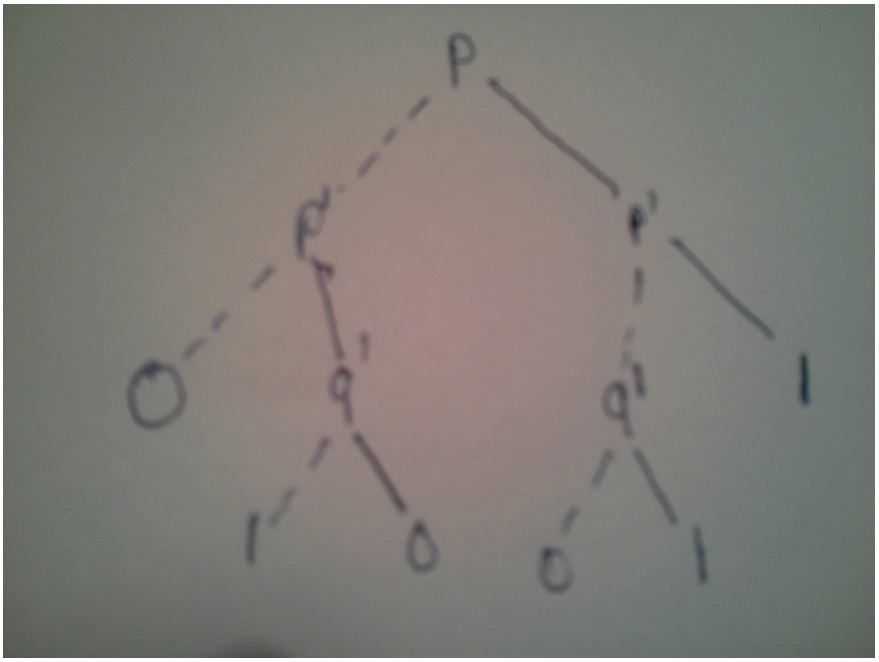


## Question 6.2

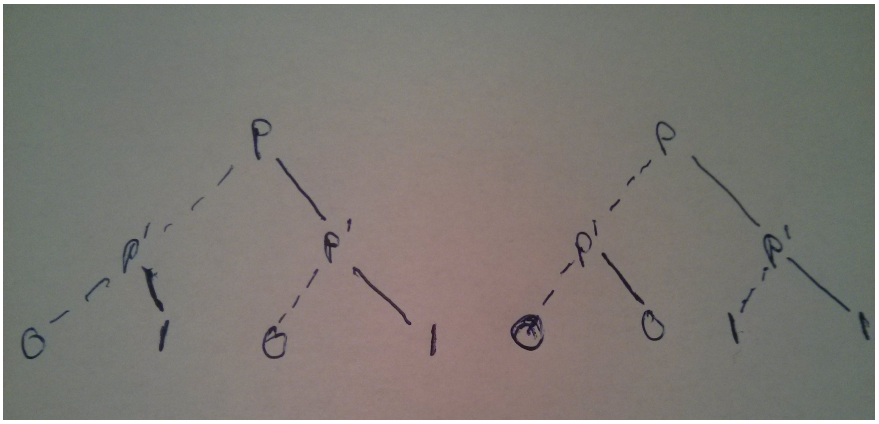
We want to compute $exists(p', exists(q', apply(\cdot, B_{\rightarrow}, B_{Y'})))$ and by computing $apply(\cdot, B_{\rightarrow}, B_{Y'}$. $BY'$ is simply the given $B_Y$ tree, but with the variables primed, hence we split $B_{\rightarrow}$ into two subproblem and we can apply the operation $\cdot$ directly and get $B_{app}$.



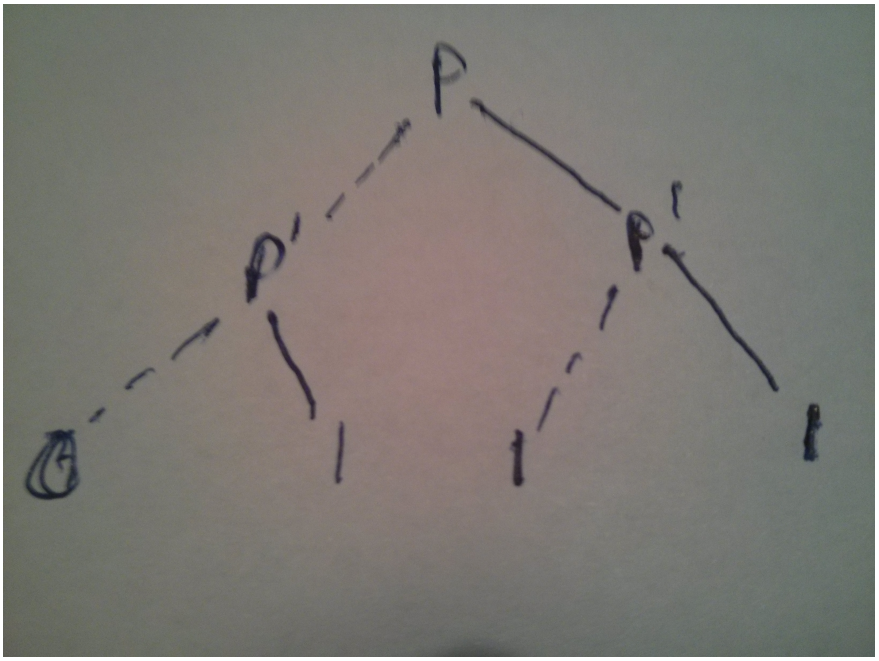
Which is reduced to



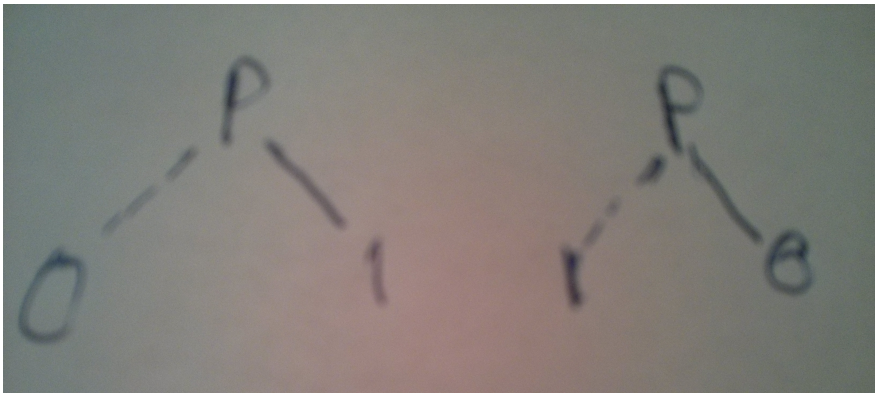
Now we calculate $exists(q', B_{app})$.
We now find $restict(0, q', B_{app})$ and $restict(1, q', B_{app})$ respectively.



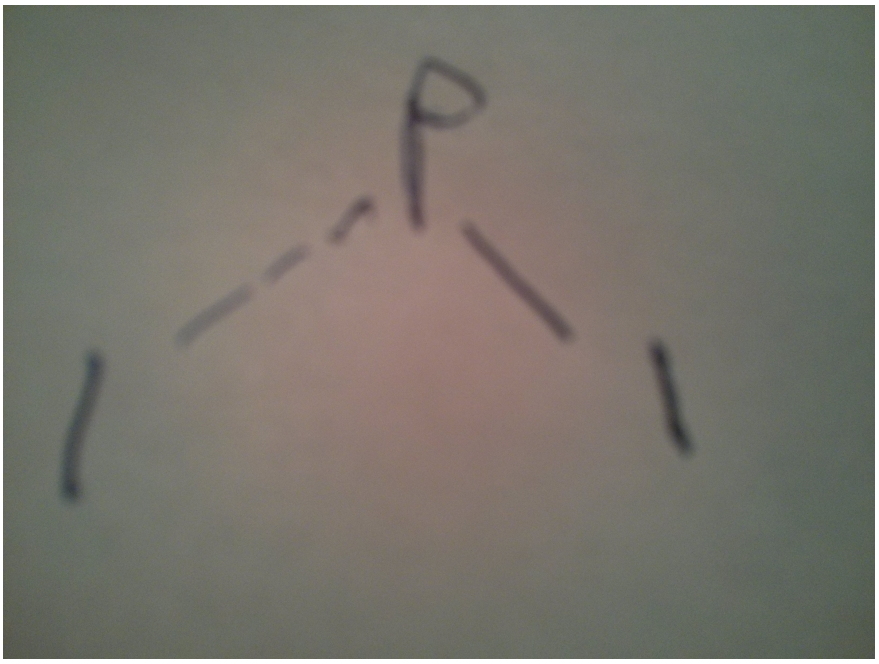
And calculate $B_{app2} = apply(+, restict(0, q', B_{app}, restict(1, q', B_{app}))$



Now we find $exists(p', B_{app2})$ with $restict(0, p', B_{app2})$ and $restict(1, p', B_{app2})$ respectively




And lastly we find $apply(+, restict(0, q', B_{app}, restict(1, q', B_{app}))$




Which is the BDD that satisfying $EX(p \leftrightarrow \neg q)$.

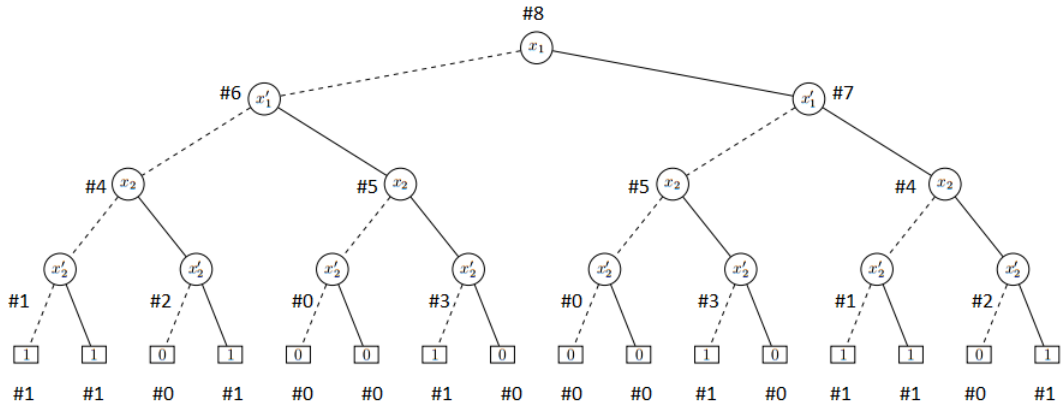
# A1

```
MODULE main
VAR
st : {st1, st2, st3};

ASSIGN
  init(st) := st1;

  next(st) :=
    case
      st = st1 : {st2,st3};
      st = st2 : st2;
      st = st3 : st3;
      esac;

DEFINE
p := st = st1;
q := st = st2;

SPEC A[p U EX q]
SPEC A[p U AX q]

-- specification A [ p U (EX q) ]   is true
-- specification A [ p U (AX q) ]   is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  st = st1
  q = FALSE
  p = TRUE
-- Loop starts here
-> State: 1.2 <-
  st = st3
  p = FALSE
-> State: 1.3 <-
```

# A2

```
MODULE main
VAR
st : {sg00, sg01, sg10, sg11};

ASSIGN
  init(st) := {sg00, sg01, sg10, sg11};

  next(st) :=
    case
      st = sg00 : {sg00,sg01,sg11};
      st = sg01 : {sg01,sg10,sg11};
      st = sg10 : {sg00,sg01,sg10};
      st = sg11 : {sg00,sg10,sg11};
      esac;

DEFINE
a := st = sg10 | st = sg11;
b := st = sg01| st = sg11;

SPEC (a & EX !b) | (!a & EX b) | (a & EX b) | (!a & EX !b)

-- specification ((((a & EX !b) | (!a & EX b)) | (a & EX b)) | (!a & EX !b)) is
   true
```

# A3


#8
$x_1$
#6
$x_1'$
#7
$x_1'$
#4
$x_2$
#5
$x_2$
#5
$x_2$
#4
$x_2$
$x_2'$
#1
#2
#0
#3
#0
#3
#1
#2
1
1
0
1
0
0
1
0
0
0
1
0
1
1
0
1
#1
#1
#0
#1
#0
#0
#1
#0
#0
#0
#1
#0
#1
#1
#0
#1