CSEP590 – Model Checking and Software Verification
Summer 2003
Solution Set 3

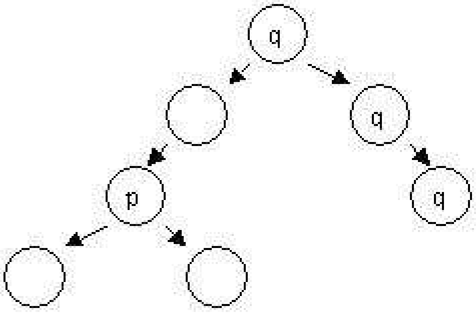
1. CTL equivalence/non-equivalence

Solutions:

a) $EFp \wedge EGq$, $EF(p \wedge EGq)$

Not equivalent

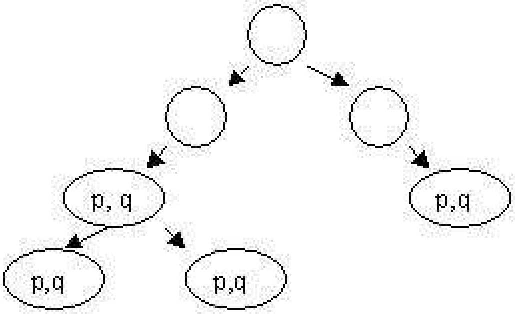
Counter-example: $EFp \wedge Egq$ satisfied, but not $EF(p \wedge EGq)$



b) $AFp \wedge AGq$, $AF(p \wedge AGq)$

Not equivalent

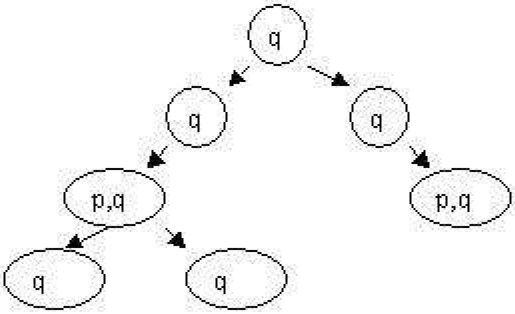
Counter-example: $AF(p \wedge AGq)$ satisfied, but not $AFp \wedge AGq$



c) $AFp \wedge AGq$, $AG(AFp \wedge q)$

Not equivalent

Counter-example: $AFp \wedge Agq$ is satisfied, but not $AG(AFp \wedge q)$



d) AFAGp ∧ AFAGq, AF(AGp ∧ AGq)

Equivalent

Justification:

i) AFAGp ∧ AFAGq ⇒ AF(AGp ∧ AGq)

Suppose AFAGp ∧ AFAGq holds, then there is a state somewhere in all future paths at which p is true, and all states on all paths from that state have p true as well. Furthermore, we know that there is a state somewhere in all future paths with q true, and that all states on all paths from that state have q true as well. Then we see that it must be true that somewhere on all future paths there must be "an intersection", that is, there must be a state where both p and q are true, and all paths from that state have both p and q true as well. Thus AF(AGp ∧ AGq) must also hold.

ii) AF(AGp ∧ AGq) ⇒ AFAGp ∧ AFAGq

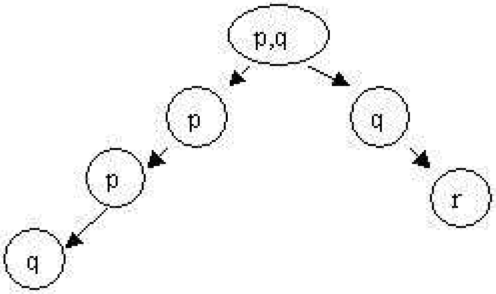
Suppose that AF(AGp ∧ AGq) holds, then it must be true that there is a state somewhere in all future paths at which p and q hold, and all states on all paths from that state have both p and q true as well. Thus, for each future path, we can choose the latter described state, and then it is true that p holds globally at that state, it is also true that q holds globally at that state. Therefore, AFAGp ∧ AFAGq must also hold.

Therefore, AFAGp ∧ AFAGq and AF(AGp ∧ AGq) are equivalent.

e) E[pUq] ∧ E[qUr], E[pUr]

Not equivalent

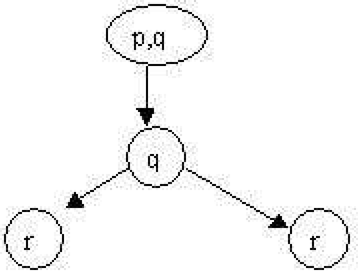
Counter-example: E[pUq] ∧ E[qUr] is satisfied, but not E[pUr]



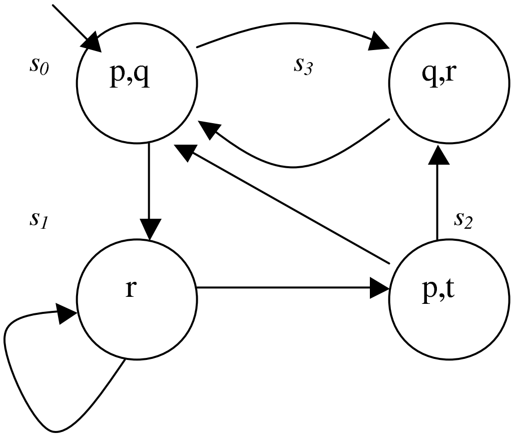
f) A[pUq] ∧ A[qUr], A[pUr]

Not equivalent

Counter-example: A[pUq] ∧ A[qUr] is satisfied, but not A[pUr]



2. CTL formulas and M



Solutions:

a) AFq

Holds.

q is true at s_0, and the future includes the present, thus all future paths contain q.

b) AG(EF($p \vee q$))

Holds.

This can be seen by noting that states s_0, s_1, s_2, and s_3 all satisfy EF($p \vee q$) – there is some state reachable from those states where either p or q is satisfied.

c) EX(EXr))

Holds.

Look at path s_0, s_1, s_1 - this path shows the existence of a state following s_0, immediately after which there is a state with r true.

d) AG(AFq))

Does not hold.

To see this consider the path s_0, s_1, s_1, s_1, s_1, s_1, (s_1 repeating)…

e) AGEXE[($p \vee r$)Uq]

Holds.

To see this, notice that EXE[($p \vee r$)Uq] holds for all states:

s_0 – next state is s_1, then E[($p \vee r$)Uq] holds as s_1, s_2, s_0

s_1 – next state is s_1, then E[($p \vee r$)Uq] holds as s_1, s_2, s_0

s_2 – next state is s_3, then E[($p \vee r$)Uq] holds as s_3, s_0

s_3 – next state is s_0, then E[($p \vee r$)Uq] holds as s_0, s_3

Corrected Solution:

f) AF(A[($p \rightarrow r$)Uq]

Holds.

A[($p \rightarrow r$)Uq] is equivalent to A[($\neg p \vee r$)Uq]

To see this, we show that $A[(\neg p \vee r)Uq]$ all paths from s_0 satisfy this formula. The trick is that formally "A[p U q]" means that on all paths, p occurs *0 or more* times until q. Then we just note that q is asserted in state s_0, and so $A[(\neg p \vee r)Uq]$ holds on every path.

3. CTL formulas for English properties

Solution:

a) *"The event p always precedes the event q."*
$\neg E[\neg p \; U \; (q \wedge \neg p)]$

b) *"After p, q is never true."*
$AG(p \rightarrow AXAG\neg q)$

c) *"Between the events q and r, p is never true."*
$[AG(q \rightarrow \neg EF(p \wedge EFr))] \wedge [AG(r \rightarrow \neg EF(p \wedge E))]$

4. Pseudo-code for TRANSLATE

Solution:

```
function translate(formula F) {
          case (F) {
             F is T : return T;
             F is (bottom) : return ¬T;
             F is an atomic proposition : return F;
             F is ¬ F1 : return ( ¬TRANSLATE(F) );
             F is F1 ´ F2 :  return (TRANSLATE(F1) ´ TRANSLATE(F2) );
             F is F1 ` F2 : return (¬(TRANSLATE(¬F1) ´ TRANSLATE(¬F2)));
             F is F1 → F2 : return (TRANSLATE(¬F1 ` F2) );
             F is AX F1 : return (TRANSLATE(¬EX¬F1) );
             F is EX F1 : return (EX (TRANSLATE(F1)));
             F is A[F1 U F2]: return (A[TRANSLATE(F1) U TRANSLATE(F2)]);
             F is E[F1 U F2]: return (E[TRANSLATE(F1) U TRANSLATE(F2)]);
             F is EF F1 : return ( E [T U TRANSLATE(F1)] );
             F is EG F1 : return (TRANSLATE(¬AF¬ F1) );
             F is AF F1 : return ( A [T U TRANSLATE(F1)] );
             F is AG F1 :  return (TRANSLATE(¬EF¬ F1) );
          }
}
```

5. Microwave modeling

$AG(Start \rightarrow AF\ Heat)$

Solutions:

a) Formula meaning

- "In all states, it is true that if start holds in a state, the in some state on all future paths from that state, heat will eventually hold also"
- We're checking that if start is pressed, then the heat will eventually turn on.

b) Equivalent to $\neg \mathrm{EF}(\mathrm{Start} \wedge \mathrm{EG}\neg \mathrm{Heat})$

$\mathrm{AG}(\mathrm{Start} \rightarrow \mathrm{AF\ Heat}) = \neg \mathrm{EF}\ (\neg(\mathrm{Start} \rightarrow \mathrm{AF\ Heat}))$ ~ Translate AG to EF
$= \neg\ \mathrm{EF}\ (\neg(\neg \mathrm{Start} \vee \mathrm{AF\ Heat}))$ ~ Substitute →
$= \neg\ \mathrm{EF}\ (\mathrm{Start} \wedge (\neg \mathrm{AF\ Heat}))$ ~ DeMorgan's law
$= \neg\ \mathrm{EF}\ (\mathrm{Start} \wedge \mathrm{EG}\neg \mathrm{Heat})$ ~ Translate AF to EG

c) Does $M,1 \models \phi$ hold?

Subformula	**Satisfied States**
Heat	4, 7
$\neg$ Heat	1, 2, 3, 5, 6
EG $\neg$ Heat	1, 2, 3, 5
Start	2, 5, 6, 7
Start $\wedge$ EG$\neg$Heat	2, 5
EF (Start $\wedge$ EG$\neg$Heat)	1, 2, 3, 4, 5, 6, 7
$\neg$ EF (Start $\wedge$ EG$\neg$Heat)	none

So, the formula does not hold for state 1.