CS 181u Applied Logic HW 4
Due Tuesday March 7, 2023

Anirudh Satish, Shaheen Cullen-Baratloo

Problem 1. Write CTL formulas that correspond to the following properties:

a. It is possible for p and q to hold at the same time.

   $EFp \wedge q$

b. From each reachable state it is possible to reach a state where p holds.

   $AGEFp$

c. Whenever p holds, eventually q will hold.

   $AG(p \implies AFq)$

d. Whenever p holds, eventually r will hold and q will hold until r holds.

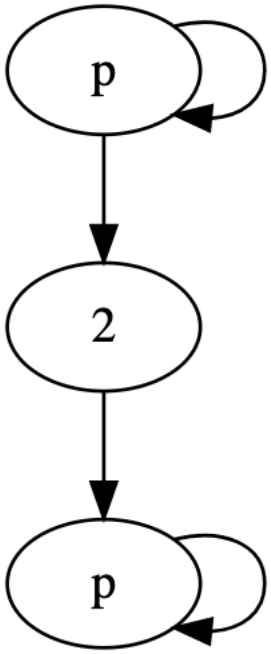
   $AG(p \implies AqUr)$

Problem 2. Show that each of the CTL operators, EG, AF, EF, EU, and AU, can be written by using only itself, EX, AX, and Boolean operations.

- $EG\phi := \phi \wedge (EXEG\phi)$
- $AF\phi := (AXAF\phi) \vee \phi$
- $EF\phi := (EXEF\phi) \vee \phi$
- $\psi EU\phi := \phi \vee (\psi \wedge (EX\psi EU\phi))$
- $\psi AU\phi := \phi \vee (\psi \wedge (AX\psi AU\phi))$

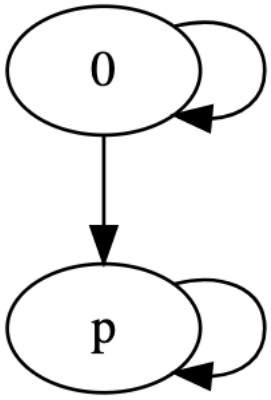
Problem 3. For each pair of LTL formula and CTL formula, give a transition system that satisfies one formula but not the other, and indicate which formula it satisfies.

a. $FG\ p$ and $AF\ AG\ p$



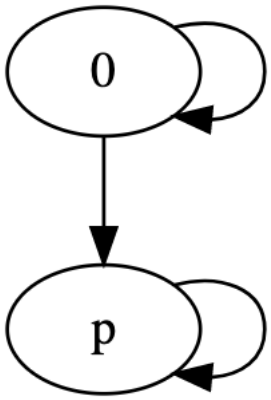
The initial state is the top one. This does satisfy the LTL formula, since all paths either stay at the initial state forever or reach the bottom eventually, and in either case you have p forever. However, the initial state does not satisfy AGp, and there is a path that stays at the initial state forever, so $AFAGp$ is not satisfied.

b. $FG\ p$ and $EF\ EG\ p$



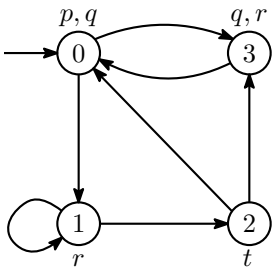
The initial state is the top one. This does not satisfy the LTL formula, as the path 0^w does not satisfy FGp. However, from the initial state, there is a possible path that takes you to state p, which satisfies EGp.

c. $GF\ p$ and $AG\ EF\ p$



The initial state is the top one. This does not satisfy the LTL formula, as the path 0^w does not satisfy GFp. However, from the initial state, both possibilities satisfy EGp, as there is always a possibility of going to state p.

Problem 4. For the transition system, $\mathcal{M}$, write a νSMV model that captures the behavior of $\mathcal{M}$. Then, for each of the following CTL formulas, ϕ, given below, use νSMV to (1) determine if $\mathcal{M} \models \phi$ and (2) exhibit a path that satisfies ϕ or show that none exist.



Using νSMV, we determined whether or not the transition system given above satisfied each of the CTL formulas given below.

a. $AF\ q$

$\mathcal{M} \models AF\ q$. The start state has q true, therefore it is true that every path eventually has q as true. One example path is 01^w

b. $AG\ (EG\ (p \vee r))$

$\mathcal{M} \not\models AG\ (EG\ (p \vee r))$. If we take the path $(012)^w$, we see that at state 2, $EG(p \vee r)$ does not hold since neither p nor r hold at 2, so no path exists where $G(p \vee r)$ holds. Therefore, since 2 is a reachable state, $AG\ (EG\ (p \vee r))$ cannot hold.

c. $EX\ (EX\ r)$

$\mathcal{M} \models EX\ (EX\ r)$. An example path is 01^w

d. $AG\ AF\ q$

$\mathcal{M} \not\models AG\ AF\ q$. From state 0, the path 01^w does not ever hit q, so $AF\ q$ does not hold. This means $AG\ AF\ q$ does not hold.

e. $p\ AU\ q$

$\mathcal{M} \models p\ AU\ q$. The start state has q. So, it is trivial that for all paths, p holds until q.

The code for our νSMV model is below.

```
MODULE s()


VAR
    state   : {0,1,2,3};
    statevalue1 : {p,r,t};
    statevalue2 : {p,q,r,t};

ASSIGN
```

```
    init(state) := 0;
    init(statevalue1) := p;
    init(statevalue2) := q;

    next(state)    :=
        case
            state = 0 : {1,3};
            state = 1 : {1,2};
            state = 2 : {0, 3};
            state = 3 : {0};
        esac;

    next(statevalue1)    :=
        case
            state = 0 : p;
            state = 1 : r;
            state = 2 : t;
            state = 3 : r;
        esac;

    next(statevalue2)    :=
        case
            state = 0 : q;
            state = 1 : r;
            state = 2 : t;
            state = 3 : q;
        esac;



MODULE main

VAR
    ts : s();



CTLSPEC
  AF(ts.statevalue2 = q)
CTLSPEC
  AG(EG (ts.statevalue1 = p | ts.statevalue2 = p | ts.statevalue1 = r | ts.stateval
CTLSPEC
  EX (EX (ts.statevalue1 = r | ts.statevalue2 = r))
CTLSPEC
  AG (AF (ts.statevalue2 = q))
```

```
CTLSPEC
  A [(ts.statevalue1 = p | ts.statevalue2 = p) U (ts.statevalue2 = q)]
```

Problem 5. Consider the νSMV stack model that we covered in class. The code is given on the next page for your convenience. You may refer to the lecture slides for further explanation of the stack model.

For each of the following verification conditions, write a νSMV specification, add the specification to the provided `stack.smv` file, and confirm that running νSMV verifies the property.

i. LTL property: The stack pointer is never negative and never larger than the size of the stack buffer.
```
specification G (s.top >= 0 & s.top <= 5) is true
```

ii. LTL property: The stack is never simultaneously full and empty.
```
specification G (s.full -> !s.empty) is true
```

iii. CTL property: In all possible states, it is always the case that if the stack happens to be empty, then it is possible that the stack eventually becomes full.
```
specification AG (s.empty -> EF s.full) is true
```

iv. LTL property: If the user only ever pushes then the stack eventually fills up.
```
specification ( G action = push -> F s.full) is true
```

v. CTL property: At any time, if the user pops when the stack does not have any elements in it then the resulting pop value will be `NULL`.
```
specification AG ((action = pop & s.empty) -> AX pop_val = NULL) is true
```

vi. LTL property: At any time, if the user pops when the stack has at least one value in it, then the resulting pop value will not be `NULL`.
```
specification G ((action = pop & !s.empty) -> X pop_val != NULL) is true
```

vii. LTL property: It is always the case that if x is pushed on the stack and the stack is not full, then if in the next state the user decides to pop then in the state after that the popped value is x.
```
specification G (((action = push & !s.full) & push_val = x) -> ( X action
= pop -> X ( X pop_val = x))) is true
```

viii. CTL property: Starting from any point in time, it is possible that the stack will eventually hold three items.
```
specification AG (EF s.top = 3) is true
```

ix. CTL property: If the stack pointer is 1 and then pop is executed twice in a row, then the stack will be empty.
```
specification AG (((s.top = 1 & action = pop) & AX action = pop) -> AX (AX
s.empty)) is true
```

x. LTL property: If the user pushes an x followed directly by a y, and the stack is not full during either of those push actions, then if the user immediately pops, the resulting value will be y and if the user immediately pops again, the resulting value will be x.
```
specification G ((((((action = push & push_val = x) & X action = push) &
```

```
X push_val = y) & !s.full) & X !s.full) -> ( X ( X action = pop) -> (( X
( X ( X pop_val = y)) & X ( X ( X action = pop))) -> X ( X ( X ( X pop_val
= x)))))) is true
```

```
#define SIZE 5
MODULE stack(action, push_val, pop_val)
  VAR
    top : 0 .. SIZE;
    buffer : array 0 .. SIZE - 1 of {x, y, z};
  DEFINE
    full := top = SIZE;
    empty := top = 0;
  ASSIGN
    init(top) := 0;
    next(top) :=
      case
        (action = push) & (top < SIZE)  : top + 1;
        (action = pop) & (top > 0) : top - 1;
        TRUE : top;
      esac;
    next(pop_val) :=
      case
        action = pop & !empty : buffer[top - 1];
        TRUE : NULL;
      esac;
    next(buffer[0]) := top = 0 & action = push ? push_val : buffer[0];
    next(buffer[1]) := top = 1 & action = push ? push_val : buffer[1];
    next(buffer[2]) := top = 2 & action = push ? push_val : buffer[2];
    next(buffer[3]) := top = 3 & action = push ? push_val : buffer[3];
    next(buffer[4]) := top = 4 & action = push ? push_val : buffer[4];
MODULE main
  VAR
    pop_val : {NULL, x, y, z};
    push_val : {x,y,z};
    action : {push, pop};
    s : stack(action, push_val, pop_val);
```