1. Give a Büchi automaton that corresponds to the LTL property $G F p$.

Given the transition system $T = (S, I, R)$ where $I = \{0\}$, $S = \{0, 1, 2\}$ and $R = \{(0, 1), (1, 2), (2, 1), (2, 2)\}$, assume that the only state which satisfies the atomic proposition $p$ is 1. Show the Büchi automaton $A_T$ that corresponds to this transition system (based on the construction given in the lecture notes), and the product automaton $A_T \times A_f$.

If there is one, show an accepting run of the product automaton and show the path in the transition system $T$ which corresponds to this run and satisfies the LTL formula $G F p$.

Does the transition system $T$ satisfy the property $F G \neg p$? Why?

2. Construct a Büchi automaton that corresponds to the LTL property $p U (r \land X r)$ using the LTL-Büchi automata translation algorithm. Show the intermediate steps (like the example in the lecture notes).

3. You are given the following program:

```plaintext
initial: a=False and b=False;

P:: a:=true;

Q:: b:=true;

R:: await(a) a:=!a;

cobegin P || Q || R coend
```

The transition relation set for the above program is $T = \{t_P, t_Q, t_R\}$ where each transition is executed only once. The program starts at an initial state where $a$ and $b$ are both False. The transition $t_P$ sets $a$ to true, the transition $t_Q$ sets $b$ to true, and the transition $t_R$ toggles $a$ after $a$ becomes true. Each transition is executed once and then the whole program terminates.

The property we want to check is $F(a)$. (It is in $LTL_X$, hence it is invariant under stuttering).

(a) List the visible transitions according to this property.

(b) For each pair of transitions state if they are dependent or independent.

(c) Draw the graph for the reachable state space and choose ample sets for each state in the reachable state space (satisfying the constraints C0, C1, C2 and C3 given in the lecture notes).
(d) Construct the corresponding reduced state space using the ample sets computed above.

4. Consider the following leader election protocol which identifies a leader among \( N \) processes in a ring topology connected by FIFO message queues. Each process can only send messages in a clockwise manner. Processes can be either in active or relaying state, and each process starts in the active state. Each process has a unique process id which is a natural number.

```plaintext
active:
d := process-id;
while true do
    send(d);
    receive(e);
    if (e == process-id) then
        announce elected;
    endif
    if (d > e) then
        send(d);
    else
        send(e);
    endif;
    receive(f);
    if (f == process-id) then
        announce elected;
    endif;
    if (e >= max(d,f)) then
        d := e;
    else
        goto relay;
    endif
endwhile
relay:
while true do
    receive(d);
    if (d == process-id) then
        announce elected;
    endif
    send(d);
endwhile
```

Implement the above algorithm in in Promela (the input language of Spin) and avoid invalid end states. Check these properties on your solution using the Spin model checker: 1) There is always at most one leader. 2) Eventually always a leader will be elected. 3) The elected leader will be the process with the highest process id. 4) The maximum total amount of messages sent in order to elect the leader is at most \( 2N\lceil \log_2 N \rceil + N \). 5) Each process executes the loop in the active state at most twice.

The automaton for an LTL formula can be generated by `./spin -f "LTL formula"` (use the negation of the property you wish to verify). Concatenate the output of this to your Promela specification and then follow the steps below to check the property on your specification:
• ./spin -a specification-file
• cc -o run pan.c
• ./run -a

Generate a counter-example if the property is violated. Try to explain the behavior. (You can use "./spin -t -p specification-file" to print the counter example).

Experiment with increasing number of processes. How does the verification (or falsification) time change with the increasing number of processes?

Turn in your Promela specification and the outputs generated by Spin.