1.

File name: mod8 counter.gc

This model was created based on using boolean variables to represent bits. Each rule represents the idea that when one is added the least significant bit that is equal to zero becomes one and all less significant bits become zero. The initial state is set to zero and when all bits are one the next state is zero.

2.

The set of properties in the properties file under Q2.

There are eight properties here designed to test that progression to the next expected number is possible, enabling the counter to work as expected. These are liveness properties. Additionally, there is a property that checks it is impossible to skip one from the initial state of zero. This is a safety property as it tests that from this state a state cannot be skipped. Finally, there is a liveness property that tests for progression from the state where bit one is zero and bit zero is one to a state in which bit one is set to one.

3.

The last property mentioned in q2 passes vacuously this can be seen as the property ((b0&!b1)->F) also passes. The reason for the vacuous pass is that the property requires that b0 & !b1 is the initial state which it is not so the property becomes false -> something, which is always true. In order to fix this property it should be changed to AG((b0&!b1)->EX(b1)) so as to be applied at all points during execution.

4.

File name: mod8counter (bug).gc

This model was created by moving some of the guards inside in the form of if blocks. However, the program no longer works as intended as it is now able to remain in the same state. All properties still pass including the modified property from question 3.

5.

The set of properties in the properties file under Q5.

An additional set of properties have been added. These properties are safety properties that only fail in the event that a run is able to remain in the same state. A counter example in the bugged version would be that the counter can start in the initial state and then use rule 2 to remain there by making use of the skip option.

The earlier properties with the exception of the safety property have been changed to use AX instead of EX to force progression as is appropriate in a counter.

6.

The initial model does satisfy the new properties and property changes as it does not allow multiple rules to be applicable at once and none of these rules have the option to remain in the same state.

7.

File name: mod16 counter.gc

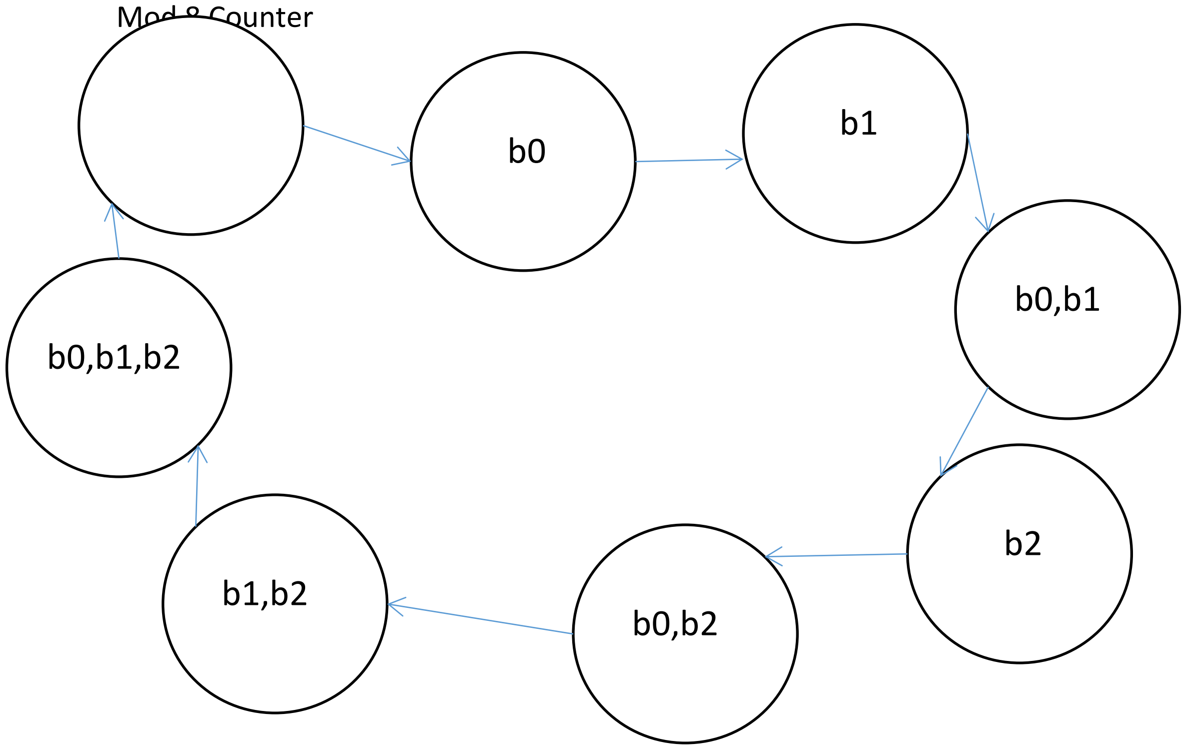
This model is extended to mod 16 by the addition of a new boolean variable to model bit 3. This has been initialised to zero as would be expected and a requirement to the final rule of this bit also being one has been added. Also, a new rule has been created to model the situations when this bit must be set to one. It is similar in nature to that of the corresponding rules for the other bits in that the requirement is that all bits less significant are already one.

8.

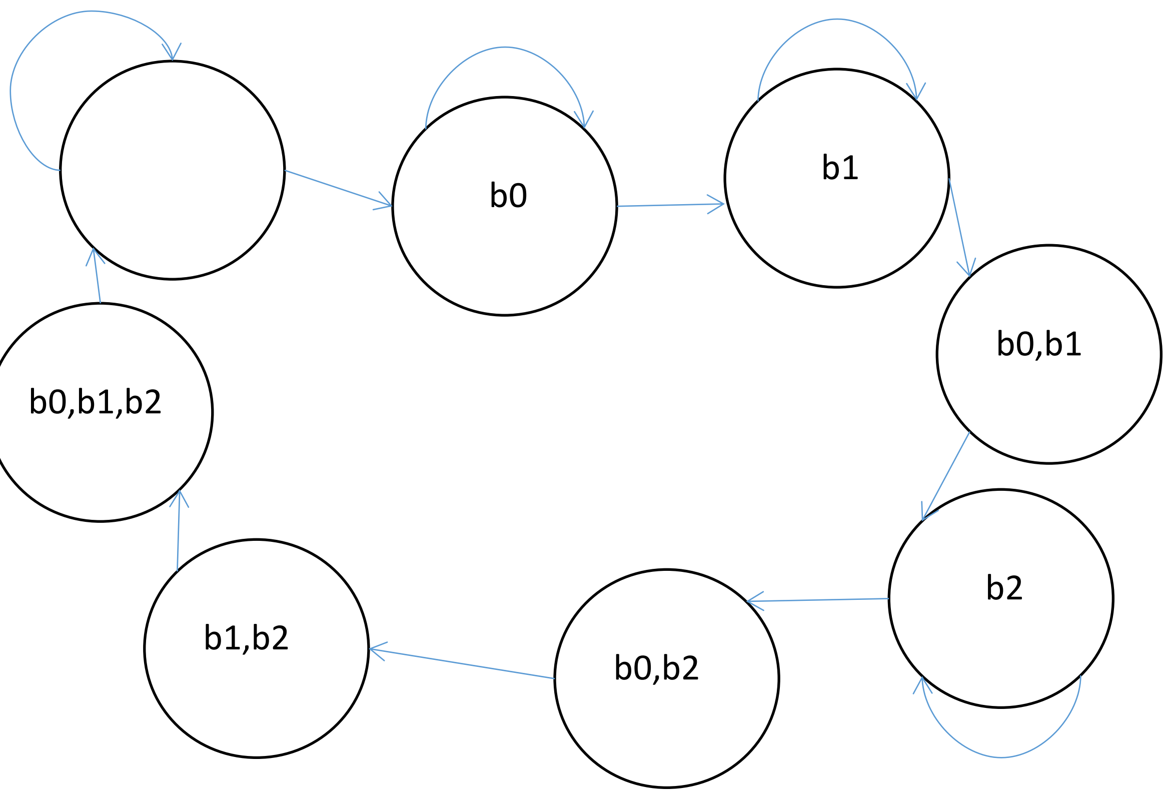
All the properties still pass. The number progression properties now indicate two different progressions. For example the property AG((b0 & b1 & b2)->AX(!b0 &!b1 &!b2)) models progression from 7 to 8 and from 15 to 0. The additional property AG((b0 & b1 & b2 & !b3)->AX(!b0 &!b1 &!b2 & b3)) designed to specifically represent progression from 7 to 8 would fail in the mod 8 counter were there a bit 3 which always remained set to zero. The set of altered properties for all numbers in the mod 16 file is shown in the properties file.

Kripke structures:

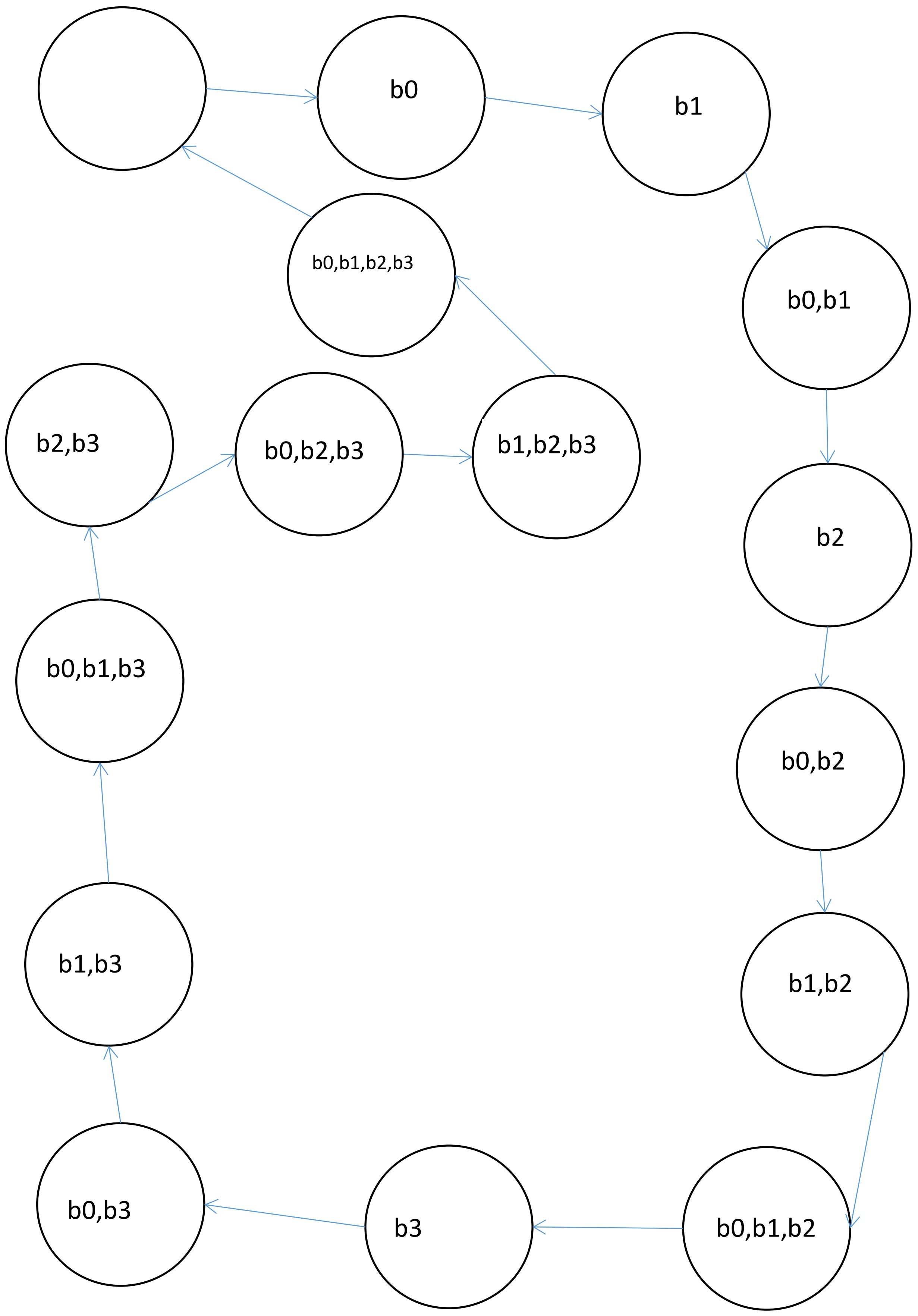
Mod 8 Counter



Mod 8 counter (bug)



Mod 16 counter



b0
b1
b0,b1
b2
b0,b2
b1,b2
b0,b1,b2
b3
b0,b3
b1,b3
b0,b1,b3
b2,b3
b0,b2,b3
b1,b2,b3
b0,b1,b2,b3