Technische Universität München **Winter term 2015/16**
I7
Prof. J. Esparza / Dr. D. Suwimonteerabuth / Dr. C. Broadbent **24.01.2016**

Automata and Formal Languages — Homework 16

Due **Wednesday** 3rd February 2016 (TA: Christopher Broadbent)

Exercise 16.1

Let $k_1, k_2 \in \mathbb{N}_0$ be constants. Find a Presburger arithmetic formula, $\varphi(x, y)$, with free variables x and y such that $\mathcal{I} \models \varphi(x, y)$ iff $\mathcal{I}(x) \geq \mathcal{I}(y)$ and $\mathcal{I}(x) - \mathcal{I}(y) \equiv k_1 \pmod{k_2}$. Find a corresponding automaton for the case $k_1 = 0$ and $k_2 = 2$.

Exercise 16.2

Construct a finite automaton for the Presburger formula $\exists y\ x = 3y$.

Exercise 16.3

In this question we model access to a file by two processes using a variable *mode* that takes values over the domain $\{ \mathit{read}, \mathit{write}_1, \mathit{write}_2 \}$. The variable should be treatedas a conventional variable with the exception that when it has the value write_1 or write_2, it may only be set to its current value or the value *read*. Thus there are two distinct 'write locks' which will be used to model the case when the file is opened for writing by one of two distinct processes.

(i) Draw an automaton modelling the variable *mode*.

(ii) Consider the following two processes (for $i \in \{ 1, 2 \}$).

```
1 while true do
2 |  if mode = read then
3 |  |  mode ← write_i
4 |  if mode = write_i then
5 |  |  mode ← read
```

Draw an automaton modelling each process.

(iii) Draw the asynchronous product of the network of three automata: the automaton modeling the variable and the two automata for each of the two processes. This should model the behaviour of the two processes running concurrently.

(iv) Consider LTL with atomic propositions $\mathit{read}, \mathit{write}_1, \mathit{write}_2$ that hold in configurations of the system for which *mode* has the corresponding value.

Write down an LTL formula expressing that "each process writes to the file infinitely often". Draw a Büchi automaton for the *negation of* this property.

(v) Using the asynchronous product and this Büchi automaton, construct a Büchi automaton recognising computations of the system that violate the LTL property.

(vi) Adjusting the model and adding suitable new LTL atomic properties, write down an LTL formula expressing a reasonable fairness condition under which the system would satisfy the LTL specification from part (iv).

(vii) Suppose that the following processes (for $i \in \{ 1, 2 \}$) were used instead, where ? indicates a non-deterministic Boolean value. Would the LTL specification from part (iv) still be satisfied under the fairness contraint from part (vi)?

```
1 while true do
2 |  if mode = read then
3 |  |  mode ← write_i
4 |  |  while ? do
5 |  |  |  mode ← write_i
6 |  if mode = write_i then
7 |  |  mode ← read
```

Solution 16.1

The condition is equivalent to:

$$\mathcal{I}(x) - \mathcal{I}(y) = k_2 n + k_1' \text{ for some } n \in \mathbb{N}$$

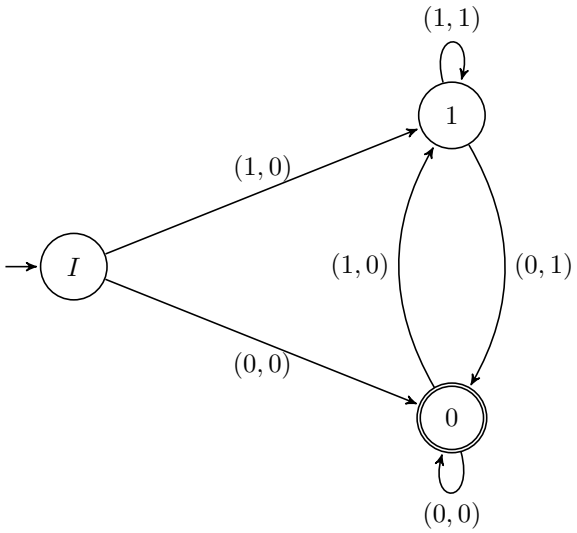
where we take $k_1' \equiv \pmod{k_2}$ with $0 \leq k_1' < k_2$. We can thus express the condition using the following Presburger formula:

$$\exists z\,(x = y + k_2.z + k_1')$$

We thus want to draw an automaton for the formula:

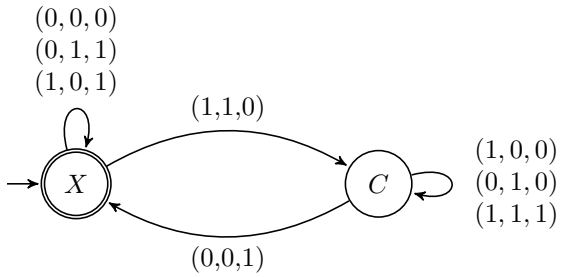
$$\exists z\,(x = y + 2z)$$

In order to help with this endeavour we first draw automata for the various syntactic subcomponents of the formula. The following automaton recognises $\{\ 2z \mid z \in \mathbb{N}\ \}$. Think: "$(z, 2z)$".



Here 1 indicates 'carry 1' and 0 indicates a zero carry bit.

The following automaton recognises $\{\ (a, b, a + b) \mid a, b \in \mathbb{N}\ \}$. Think "$(a, b, a + b)$".

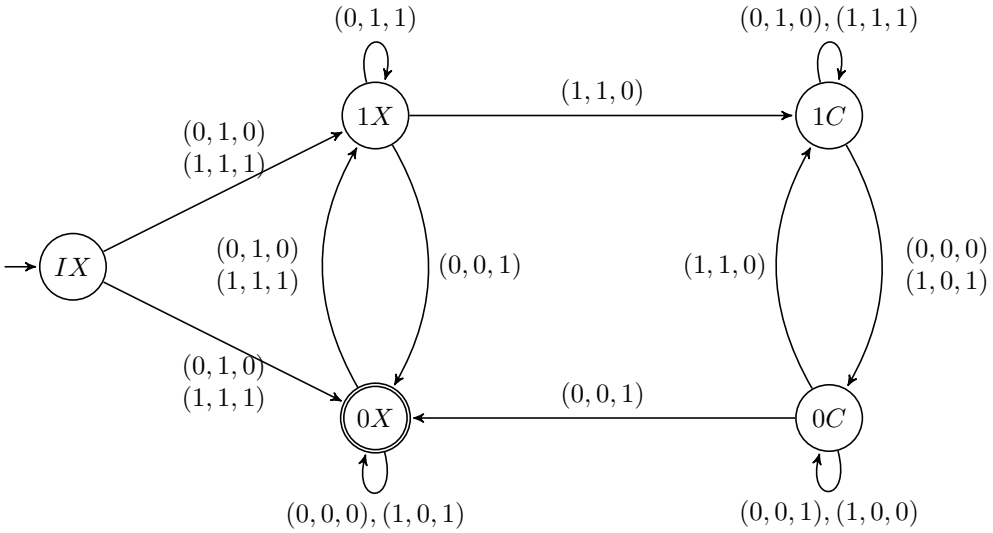


Here X indicates a zero carry bit and C indicates that the carry bit is set to 1.

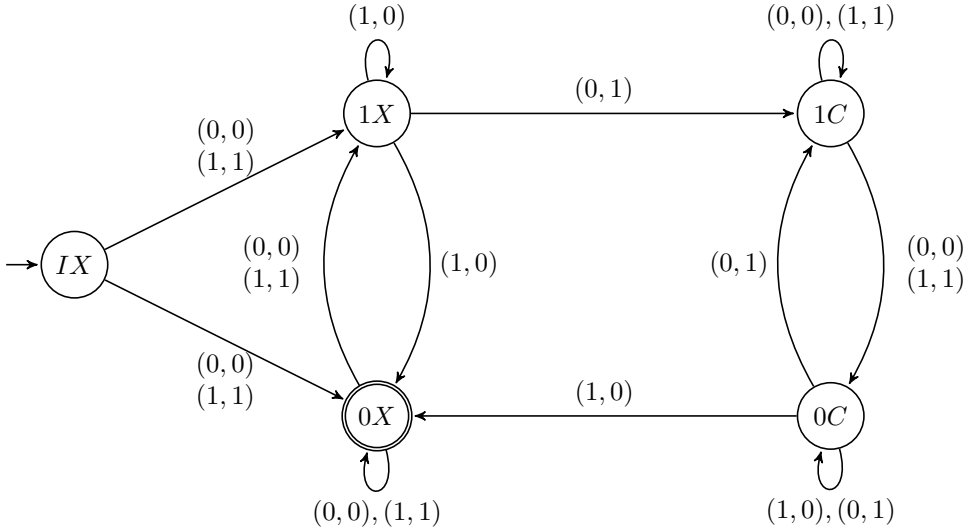
We now construct an automaton recognising the relation

$$\{\ (y, z, y + 2z) \mid y, z \in \mathbb{N}\ \}$$

We do this with the aid of the two automata above. Think: $(y, z, y + b)$ where $b = 2z$. The '$b = 2z$' is enforced by the 1 and 0 components of the states. The $(y, z, y + b)$ is then enforced by the X and C components of the states.

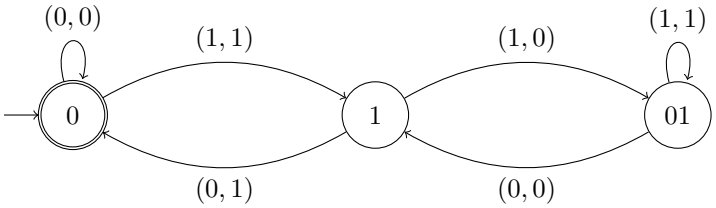


We then just project away the z component and change the order of the y and $y + b$ components to get an automaton for $\{ (y + b, y) \mid y \in \mathbb{N}, b = 2z \text{ for some } z \in \mathbb{N} \}$.



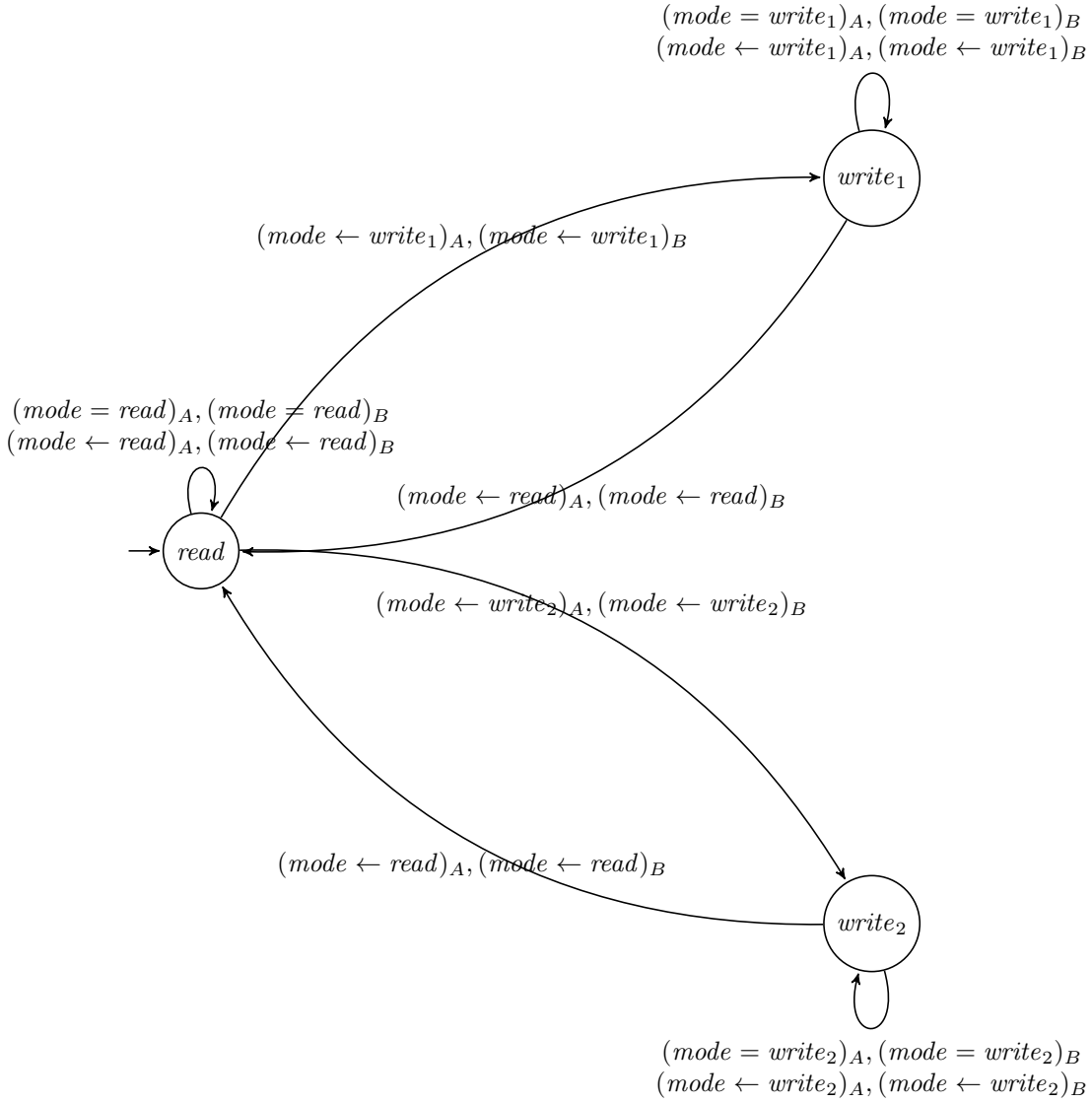
Solution 16.2

Here the number to be 'carried' is written in LSBF form as a label for the corresponding state.

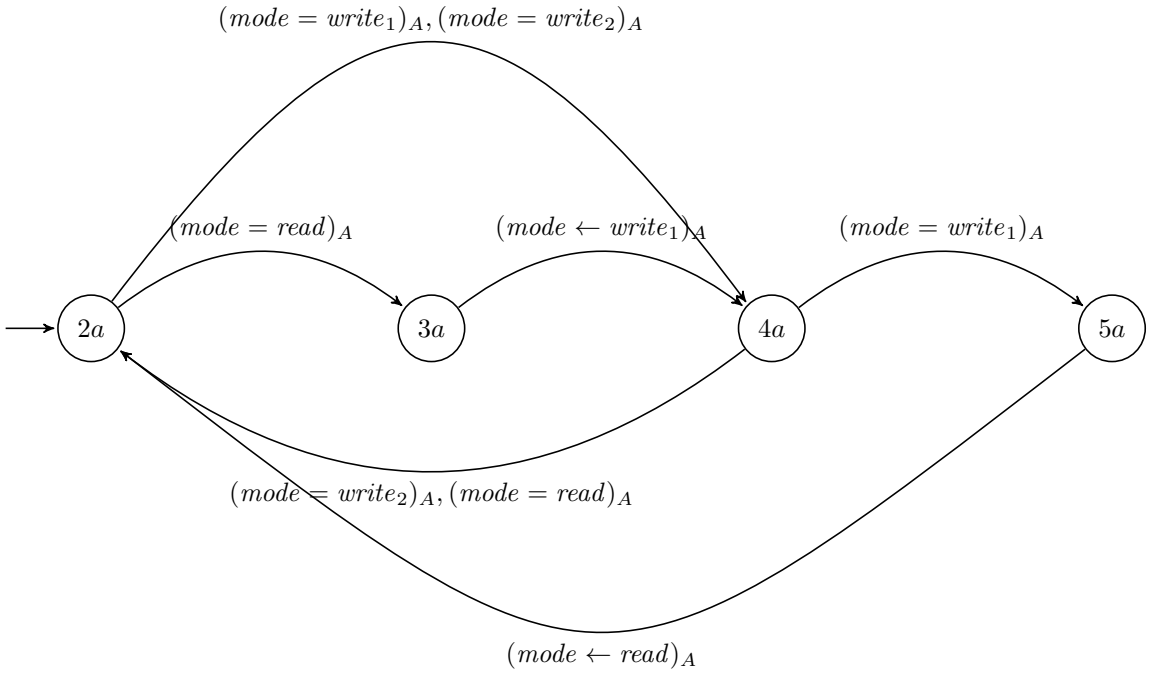


Solution 16.3

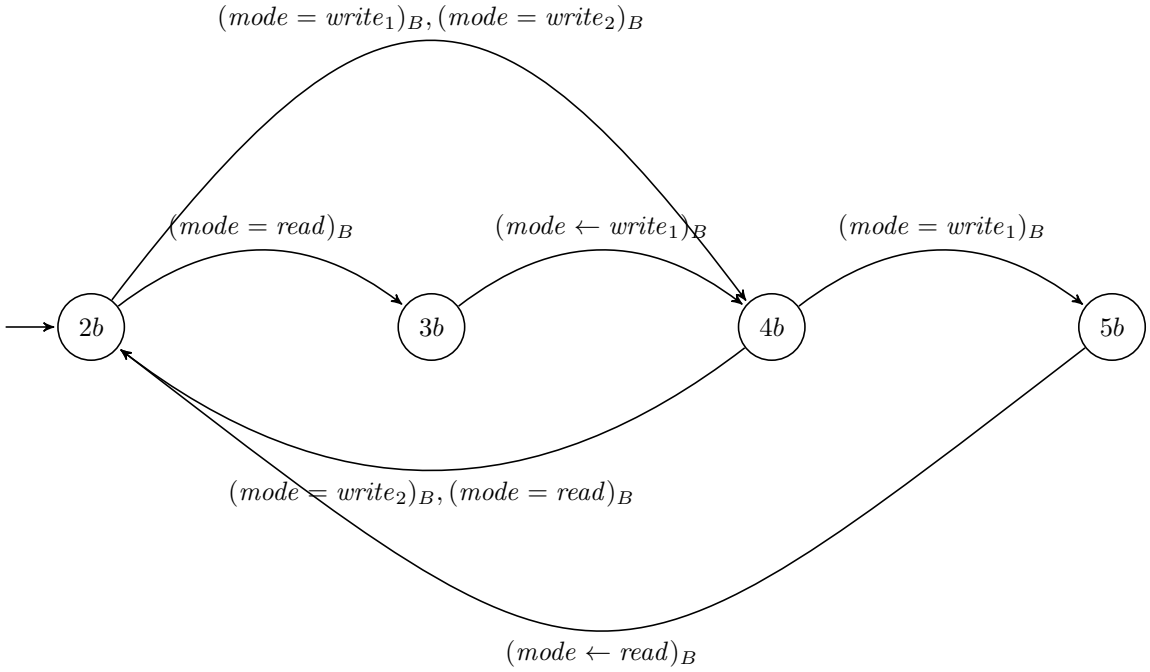
Here is an automaton modelling the variable with the given constraints. We assume that the variable is initialized to 'read'. We employ two copies of each test/set operation in order to allow the two processes to interface with the variable independently.



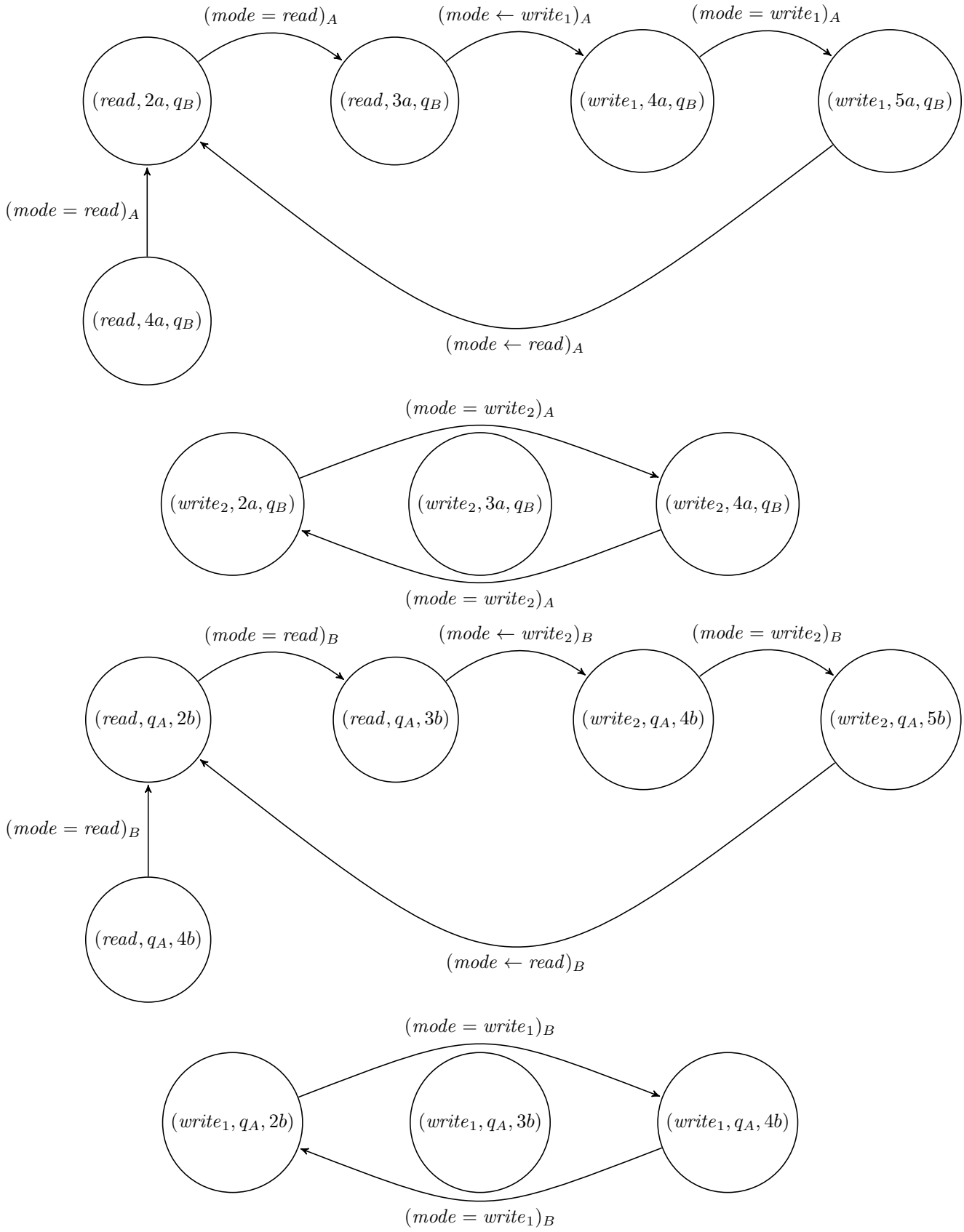
Here is an automaton modelling the first process.



Here is a similar automaton modelling the second process.



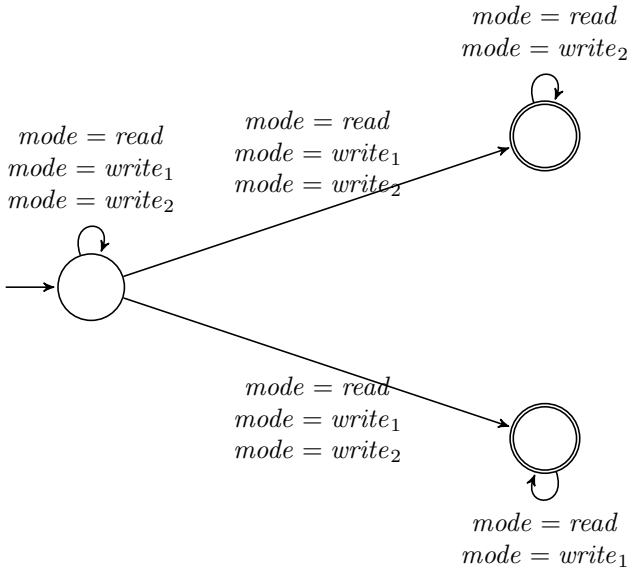
We now describe the asynchronous product of this network of three automata. (Don't worry, any such example in the exam will be much smaller so you will be able to easily draw it directly!) The initial state of the asynchronous product is $(read, 2a, 2b)$. The asynchronous product then contains all states *reachable* via the following edges. Note that the following *defines* the edges in the graph (q_A ranges over the states of the first-process, and q_B over those of the second), but to actually *draw* the graph one would have to this definition for every possible q_A and q_B (that yield a state reachable from the initial state). Of course in the initial state $(read, q_A, q_B)$ it is the case that $q_A = 2a$ and $q_B = 2b$.



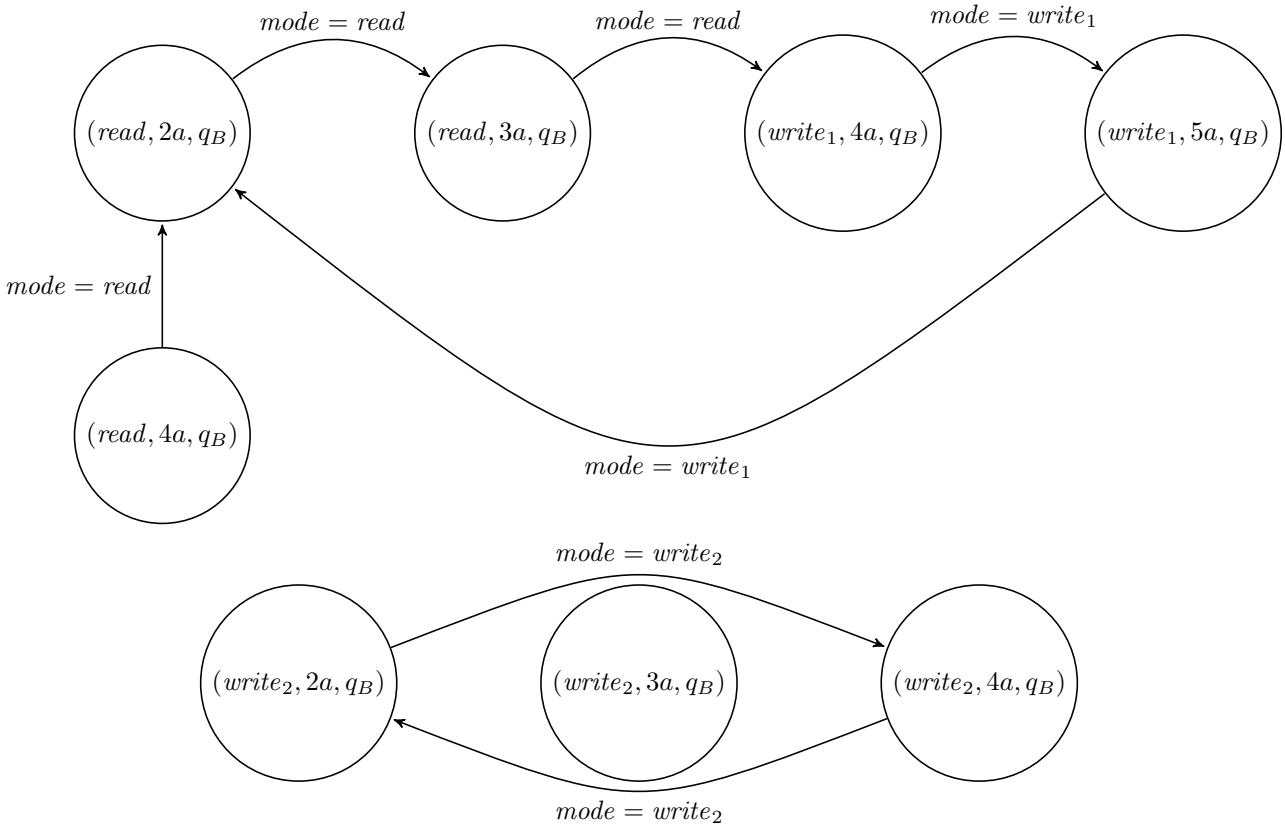
We can express 'each process writes to a file infitely often' by the LTL formula:

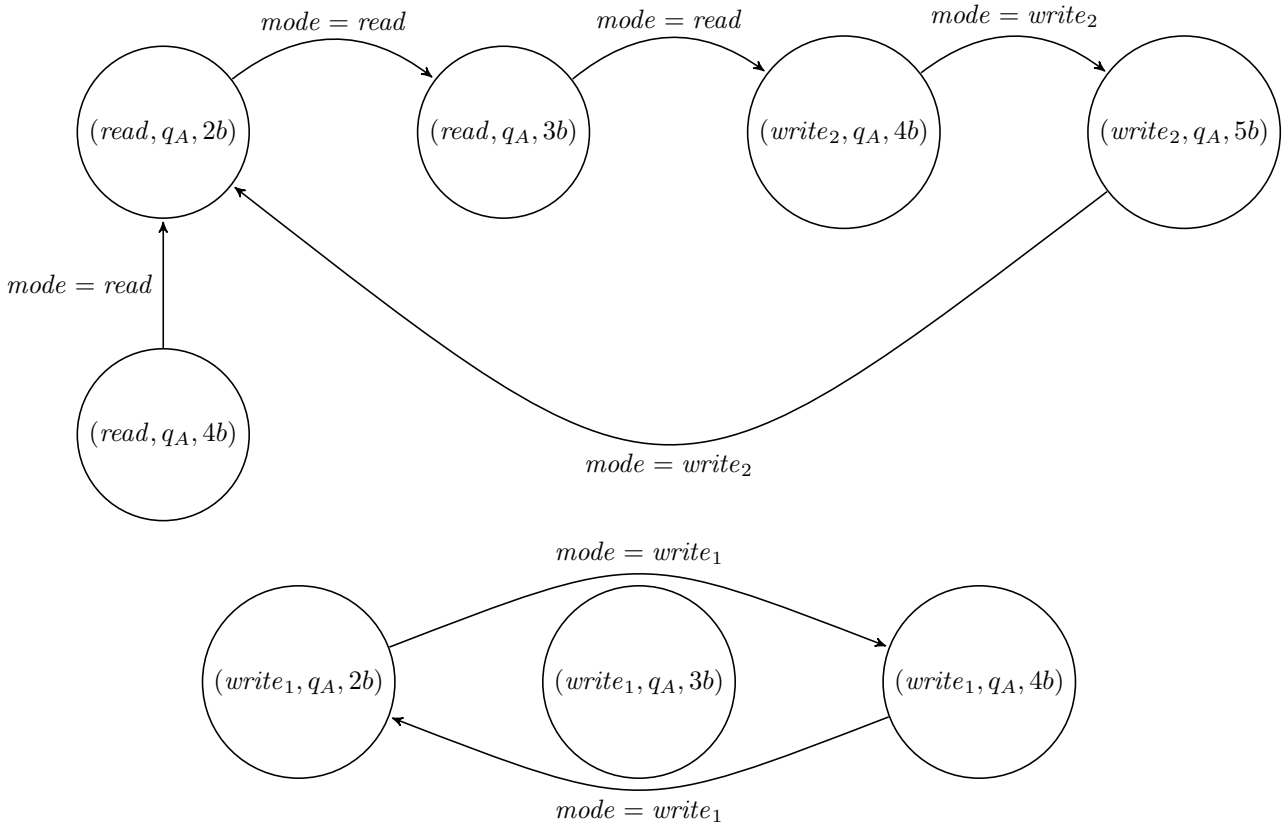
$$GF(mode = write_1) \ \wedge \ GF(mode = write_2)$$

The following Büchi automaton recognises sequences of values for the variable *mode* that *violate* this property:



We can then take the product of this Büchi automaton together with an automaton recognising all sequences of values of *mode* associated with computations of the system. The latter can easily be obtained by changing the edge labels of the asynchronous product so that each transition is labelled with the value of *mode* at the state from which the transition originates.





The system can in fact violate the property. Our model allows a run in which only one process ever makes transitions. If only the first process makes transitions, then $write_2$ will never occur and so the property will be violated. This is manifest by the fact that the product of the system with the Büchi automaton above is non-empty.

However, the system does satisfy the following property:

$$(GF1 \wedge GF2) \Rightarrow (GF(mode = write_1) \ \wedge \ GF(mode = write_2))$$

where 1 and 2 are new atomic propositions respectively asserting that a transition is due to process 1 or due to process 2. The condition $(GF1 \wedge GF2)$ is called a *fairness condition*.

Note that the program in (vii) will violate the property even assuming the fairness constraint. This is because process 1 (for example) can maintain the write-lock forever in the while-loop in lines 4 and 5 in the case when ? always resolves to *true*. This can occur even if both processes are allowed to make transitions.