Assignment 9: Petri nets

ETH Zurich

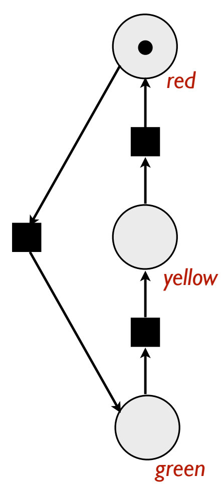
1 Modelling Systems as Petri Nets

1.1 Background

The following tasks are about *modelling* concurrent systems as Petri nets. Some have been adapted from [1].

1.2 Task

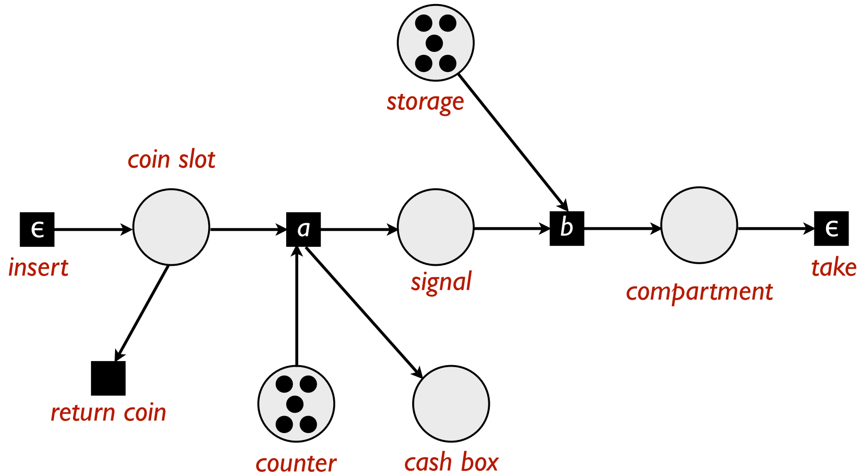
1. Consider the following Petri net N_1 that models a *traffic light*:



   Extend N_1 such that it models *two* traffic lights and satisfies the following:

   - *at most one* traffic light is ever on a green light; and
   - the two traffic lights *take turns* in moving to a green light (i.e. it should not be possible that one traffic light is perpetually red whilst the other repeatedly cycles through red-green-yellow).

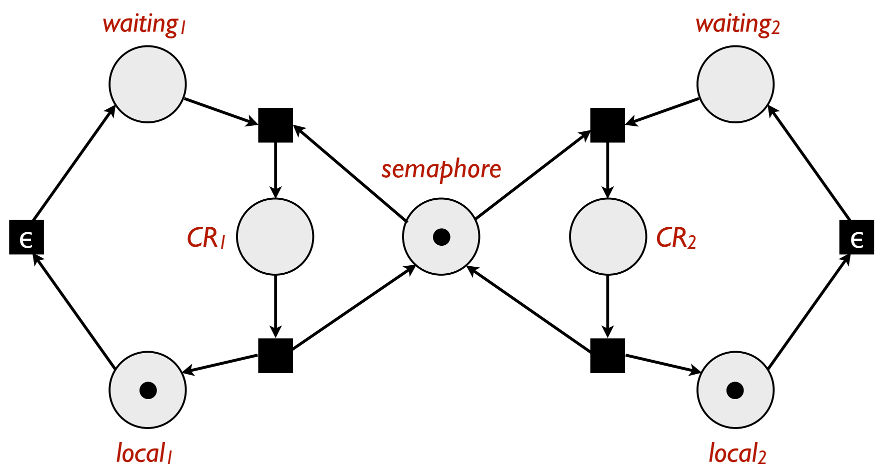
2. Consider the cookie vending machine Petri net we constructed in the lecture:



Extend the Petri net such that:

- at most one token can be in the coin slot place at any time; and
- at most one token can be in the signal place at any time.

3. Consider the Petri net we constructed for mutual exclusion:



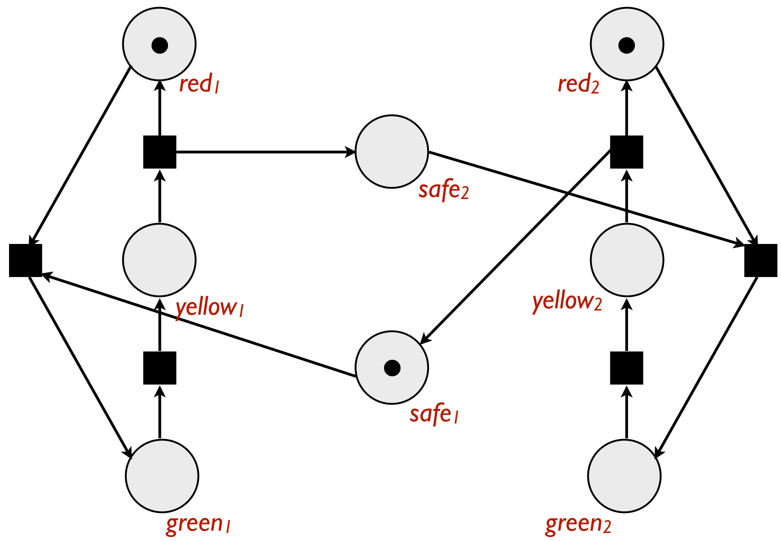
For each process i add a place noncritical$_i$ that holds a token if and only if that process i is not in its critical region.

4. Model as an elementary Petri net a gambling machine that has the following characteristics:

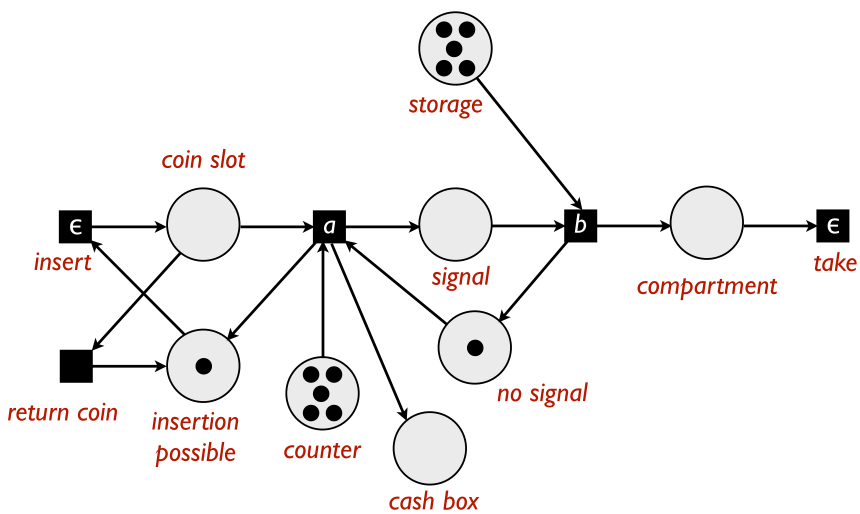
- a player can insert a coin, which should reach a "cash box";
- at this stage, the machine enters a state in which it pays out a coin from the (same) cash box an arbitrary number of times (including zero); and
- eventually, the machine stops giving out coins and becomes ready for another game.

1.3 Solutions

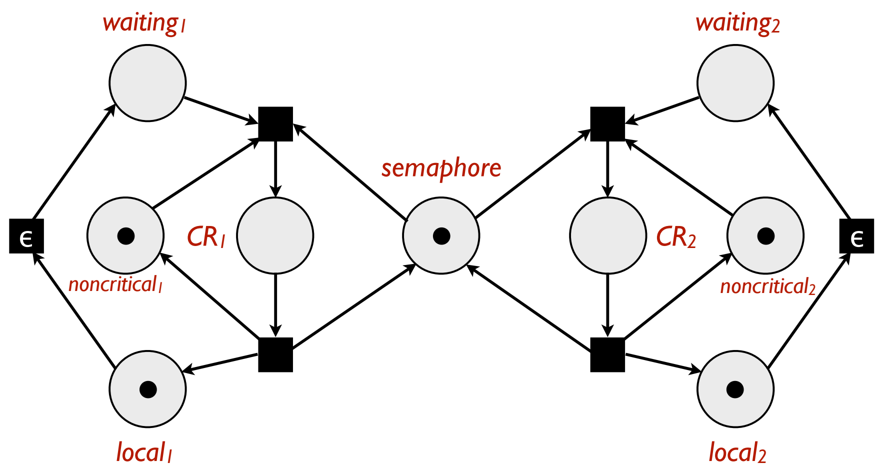
1. A possible extension of N_1:



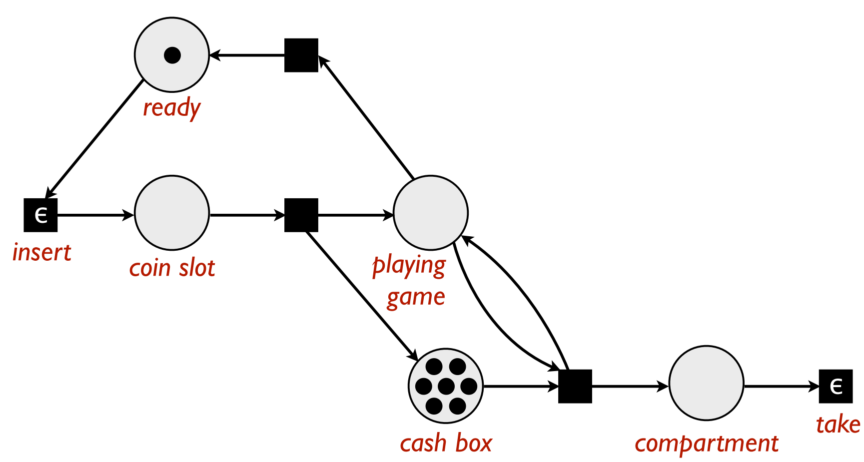
2. We add two new places which both contain a token in the initial marking:



3. A possible solution:



4. Below is a possible solution. The specification does not specify an initial number of coins, so we arbitrarily chose 7:



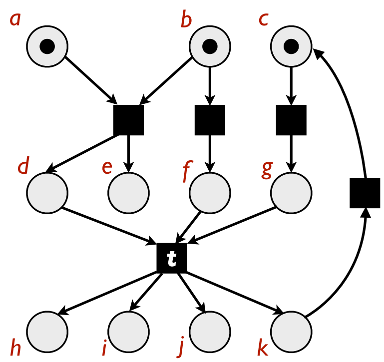
2 Reachability Graphs and Unfoldings

2.1 Background

These tasks have been partly adapted from [2], and are about the two semantics we assigned to Petri nets in the lecture: first, the semantics based on interleaving; second, the semantics based on true concurrency.

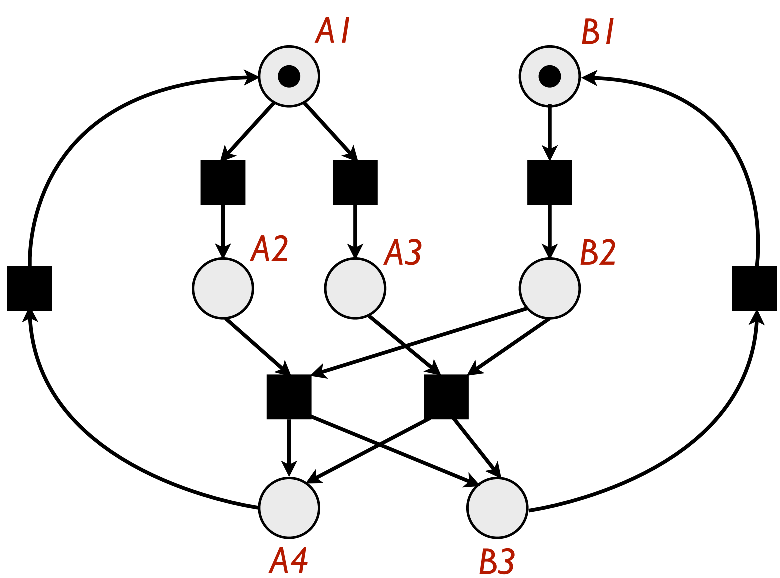
2.2 Task

1. Consider the following Petri net N_2:

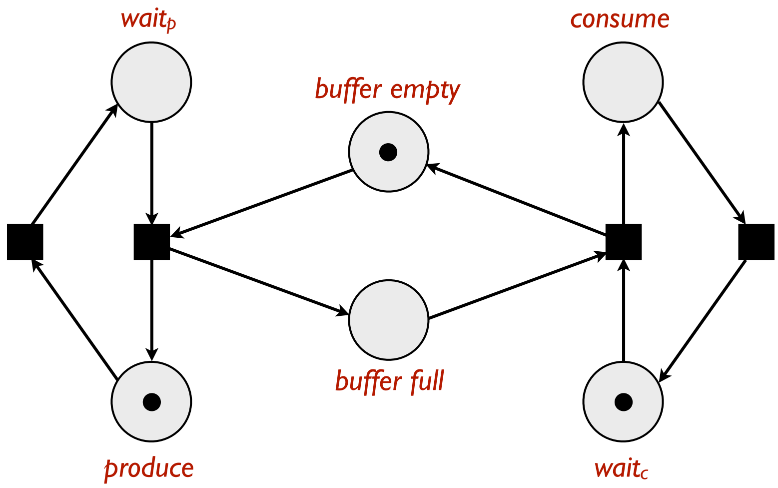


Construct the *unfolding* of N_2, and then use it to determine whether or not transition t can occur. Explain your reasoning.

2. For the Petri net below, iteratively construct its unfolding until there are 9 transitions:



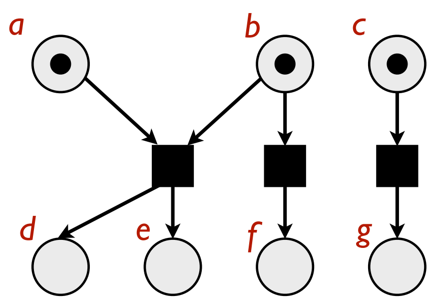
3. Consider the Petri net below that models a producer-consumer scenario for a bounded buffer of capacity 1:



Construct a reachability graph for the Petri net, and prove that the buffer is never both full and empty.

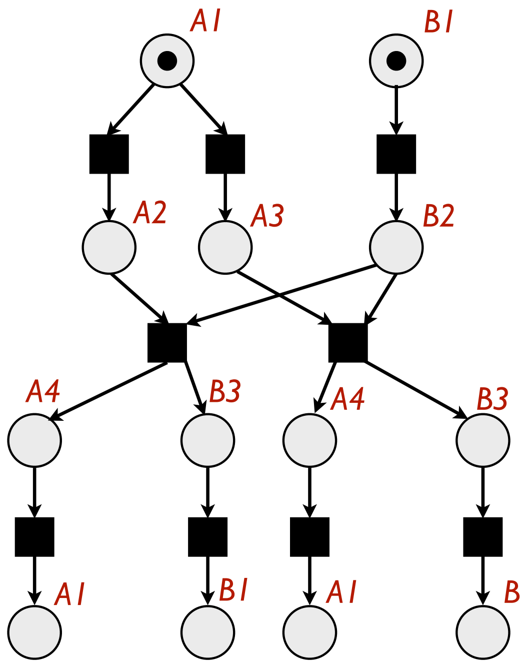
2.3 Solutions

1. Note that the unfolding of N_2 is finite:



As the unfolding represents all the possible computations of N_2, and does not contain a transition labelled t, then we conclude that transition t in N_2 will never occur.

2. The unfolding, cut off after nine transitions, is as below:

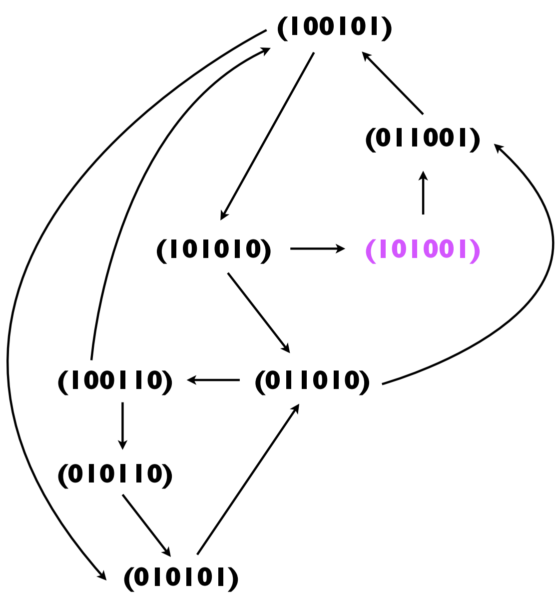


(Note that there are other solutions, e.g. if reachable markings are searched for in a "depth-first" manner.)

3. Let a marking M of the Petri net be expressed by the vector:

$$(\ M(\text{produce}) \ M(\text{wait}_p) \ M(\text{buffer\_empty}) \ M(\text{buffer\_full}) \ M(\text{consume}) \ M(\text{wait}_c) \ ) \, .$$

For such an encoding, we get the following reachability graph:



The buffer is never both full and empty because the graph contains no marking with $M(\text{buffer\_empty}) = M(\text{buffer\_full}) = 1$ or $M(\text{buffer\_empty}) = M(\text{buffer\_full}) = 0$.

References

[1] Wolfgang Reisig. Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013.

[2] Javier Esparza and Keijo Heljanko. Unfoldings: A Partial-Order Approach to Model Checking. Springer, 2008.