T-79.231 **Autumn 2002**
Parallel and Distributed Digital Systems
Answers to tutorial 6
Oct 28th 2002

1. Let PH be the net given in the problem statement. $PH \models f$, iff the LTL formula f holds for all inifinite executions in the net PH.

   a) $PH \models \Box\neg((E(ph_1) \wedge E(ph_2)) \vee (E(ph_2) \wedge E(ph_3)) \vee (E(ph_3) \wedge E(ph_1)))$
   Two neighbours cannot dine at the same time, written as a Maria LTL formula:

   ```
   []!(((1 subset place E) && (2 subset place E))
                || ((2 subset place E) && (3 subset place E))
                || ((3 subset place E) && (1 subset place E)));
   ```

   b) Make the counter claim that $ph_(1)$ never dines: $PH \not\models \Box\neg(E(ph_1))$
   The counter claim is untrue $\Rightarrow$ ph_1 dines in some state.
   The property written as a Maria LTL formula:

   ```
   []!(1 subset place E);
   ```

   N.B.: The $\Diamond$ operator requires that the property holds for all traces from the initial state, so we will have: $PH \not\models \Diamond E(ph_1)$

   c) The philosopher $ph_(1)$ may starve if, whichever the state he is in, the philosopher cannot get to dine eventually. (In other words, so as ph_1 not to starve, he has to get to dine infinitely many times.)
   $PH \not\models \Box\Diamond E(ph_1)$
   This formula does not hold in the traces ending in a deadlock, written as a Maria LTL formula:

   ```
   []<>(1 subset place E);
   ```

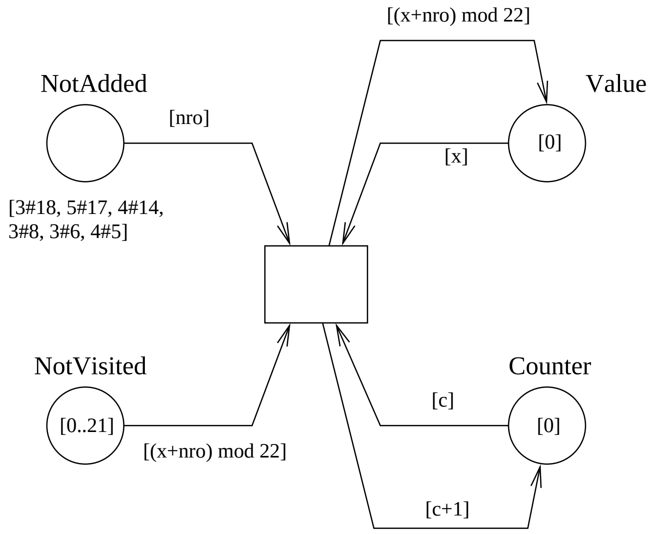
   d) The formula holds because in all states where the philosopher ph_1 is dining the forks f_1 and f_2 are also taken.
   $PH \models \Box(E(ph_1) \Rightarrow \neg(F(f_1) \vee F(f_2)))$

   ```
   [](1 subset place E => !(1 subset place F || 2 subset place F));
   ```

   e) The formula does not hold since in a system deadlock state all forks are taken but the philosopher ph_1 is not dining.
   $PH \not\models \Box(\neg(F(f_1) \vee F(f_2)) \Rightarrow E(ph_1))$

   ```
   []!((1 subset place F || 2 subset place F) => 1 subset place E)
   ```

2. a) The algebraic net solution is given below:



b) The Maria description is given below:

```
typedef unsigned(0..22) Int;
place NotAdded Int : 3#18,5#17,4#14,3#8,3#6,4#5;
place NotVisited Int : Int a (a <= 21) : a;
place Value Int : 0;
place Counter Int : 0;
trans add
  in {
    place NotAdded : num;
    place NotVisited : (x+num)%22;
    place Value : x;
    place Counter : c;
  }
  out {
    Value:      (x+num)%22;
    Counter:    c+1;
  };
```

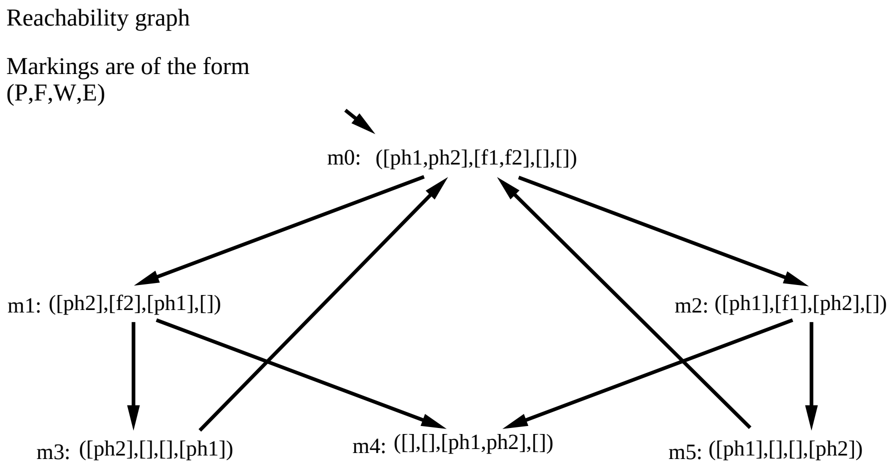
c) We can add a reject statement to the net which will stop reachability analysis if we have found a solution. As the requirement was that the solution should be found in exactly 22 steps, the presented `reject` statement will stop the reachability graph generation when the token in place `Counter` reaches the value 22. If we removed the `fatal` keyword, the generation would continue and the analyzer would report the states where the reject condition was true.

```
reject (place Counter equals 22) && fatal;
```

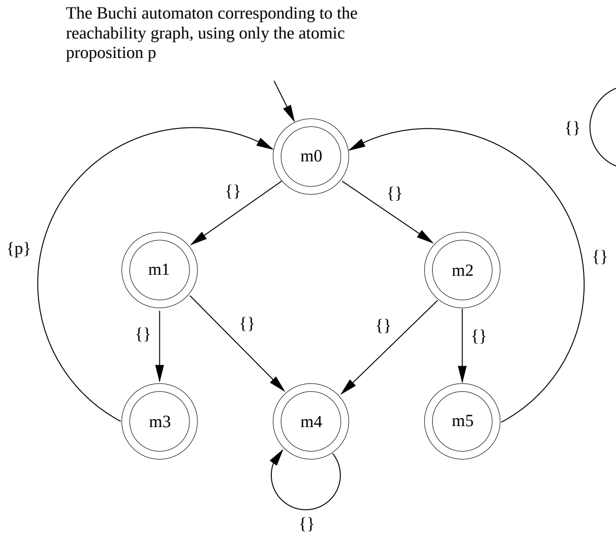
d) Reachability analysis is too powerful a tool for this problem. In reachability analysis, all aready reached states are stored, and their number can be exponential to the size of the input. Using the solution methods for combinatorial problems we can solve using a linear time w.r.t. the size of the input.

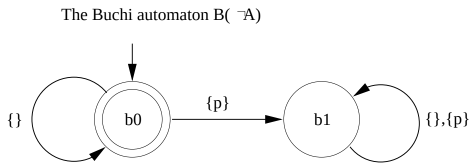
3. Being beyond the scope of the course, this example is used for demonstation purposes only.

   a) Reachability graph of the net:



   b) The Büchi automaton $B(G)$ corresponding to the reachability graph (using only the atomic proposition p and the Büchi automaton $B(\neg A$ corresponding to the negation of formula $A = \Diamond p$:





c) The product automaton $(B(G) \cap B(\neg A))$ is presented below. The product automaton has an infinite run visiting the accepting state infinitely often: $(m_0, b_0) \rightarrow (m_2, b_0) \rightarrow (m_5, b_0) \rightarrow (m_0, b_0) \ldots$ Because of this, the system and the negation of the formula have shared infinite behaviour and thus the original formula $\Diamond p$ does not hold in the system.

The product of automata B(G) and B(¬ A)