ECE/CS 584: Embedded and cyberphysical system verification Fall 2019

Dawei Sun, daweis2 *Homework 4: Reachability*

Due November 14^{th}

Typeset your solutions using LaTeX zip your writeup (.pdf) and code in a single file called `nedid-584-F19.zip` *and upload this file through Compass.*

Problem 1 (20 points). Consider the following decision problem Mult: Given binary numbers $m, n,$ and i, determine if the ith bit of the binary representation of $m \times n$ is 1. As a language we could define this as

$$\mathsf{Mult} = \{(m, n, i) \mid i\text{th bit of } m \times n \text{ is } 1\}$$

Prove that Mult $\in$ L (deterministic log space) by giving the pseudo-code of an algorithm and analyzing its memory requirements in terms of the number of (additional, non-input) bits it stores.

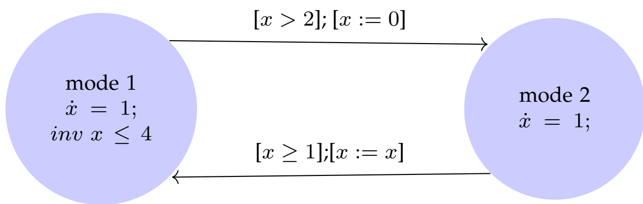
Solution Notations: **len**(m) is the number of digits in m. **MAX**, **MIN**, **mod**, **floor** are functions. The algorithm is as follows:

```
   input : m, n, i
   output: Decision
1  sum = 0;
2  for ri = 1 to i do
3  |   for ni = MAX(1, ri+1-len(m)) to MIN(ri, len(n)) do
4  |   |   mi = ri + 1 - ni;
5  |   |   sum = sum + m[mi] * n[ni];
6  |   end
7  |   if ri == i then
8  |   |   return (sum mod 2 == 1);
9  |   else
10 |   |   sum = floor(sum/2);
11 |   end
12 end
```

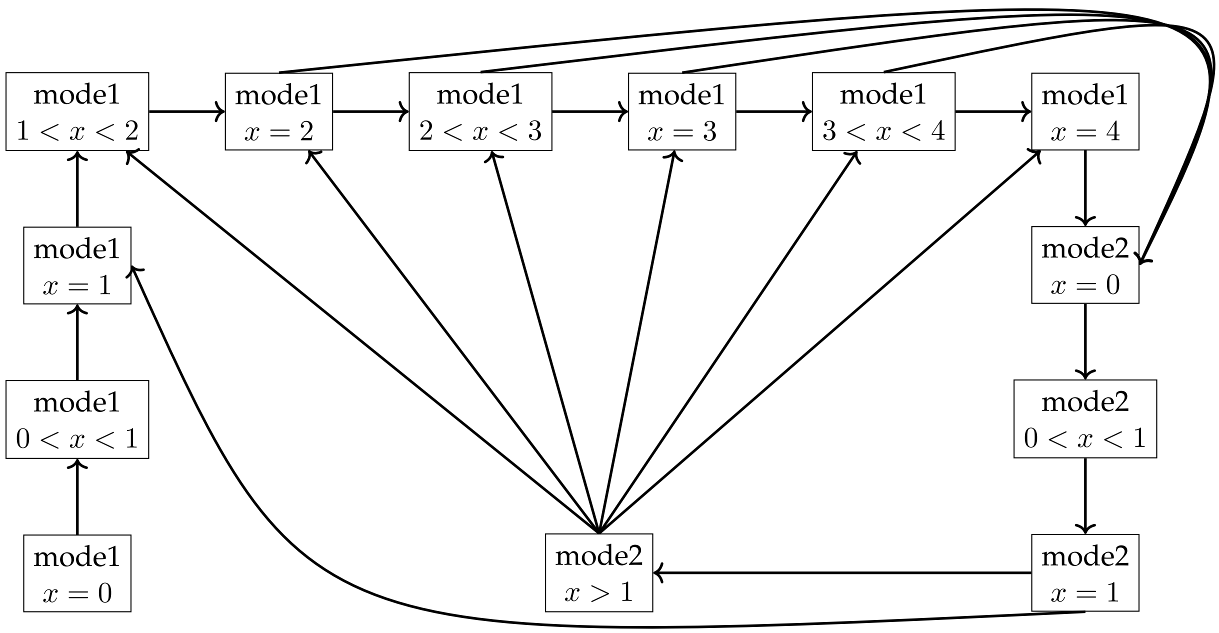
Algorithm 1: Binary Multiplication

Analysis: Denote N as the number of digits in m and n, i.e. $N = \textbf{len}(m) + \textbf{len}(n)$. Next, we analyze the size of every intermediate variable. Clearly, $ri \leq i \leq N + 1$, and thus size of ri is $O(log(N))$. For mi and ni the analysis is similar. Next, we analyze the maximal value of sum. We claim that $sum < 2N$ and use induction to prove that. First, when $ri = 0$, $sum < 2N$ clearly holds. If $sum < 2N$ holds when $ri = k - 1$, then after Line 10 is executed we have $sum < N$. Because during the execution of Line 3-6 the increament of sum is at most N, we have $sum < 2N$ holds when $ri = k$. Therefore, $sum < 2N$ always holds and thus we know the size of sum is $O(log(N))$.

Problem 2. (10 points) Construct the region automaton corresponding to the following timed automaton.

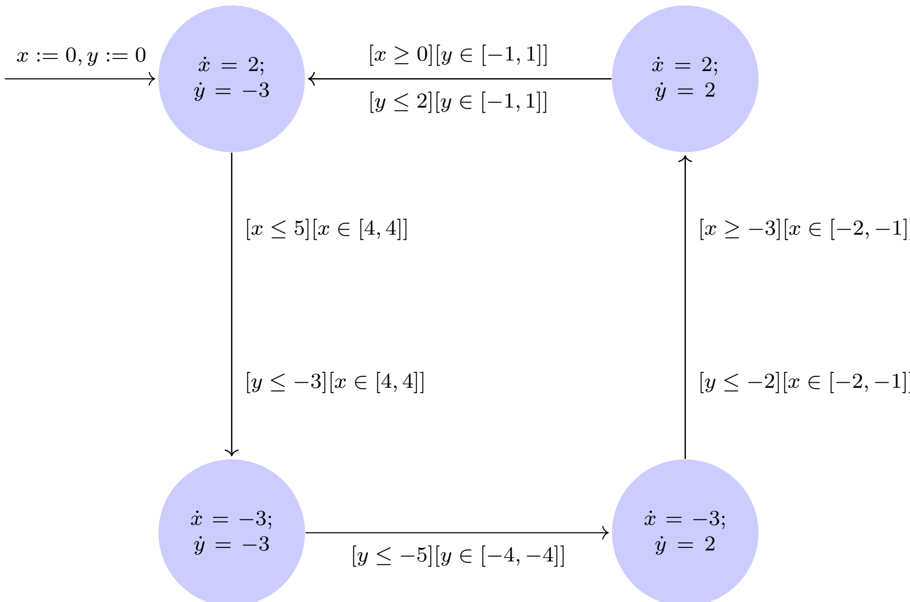


Solution Region automaton:



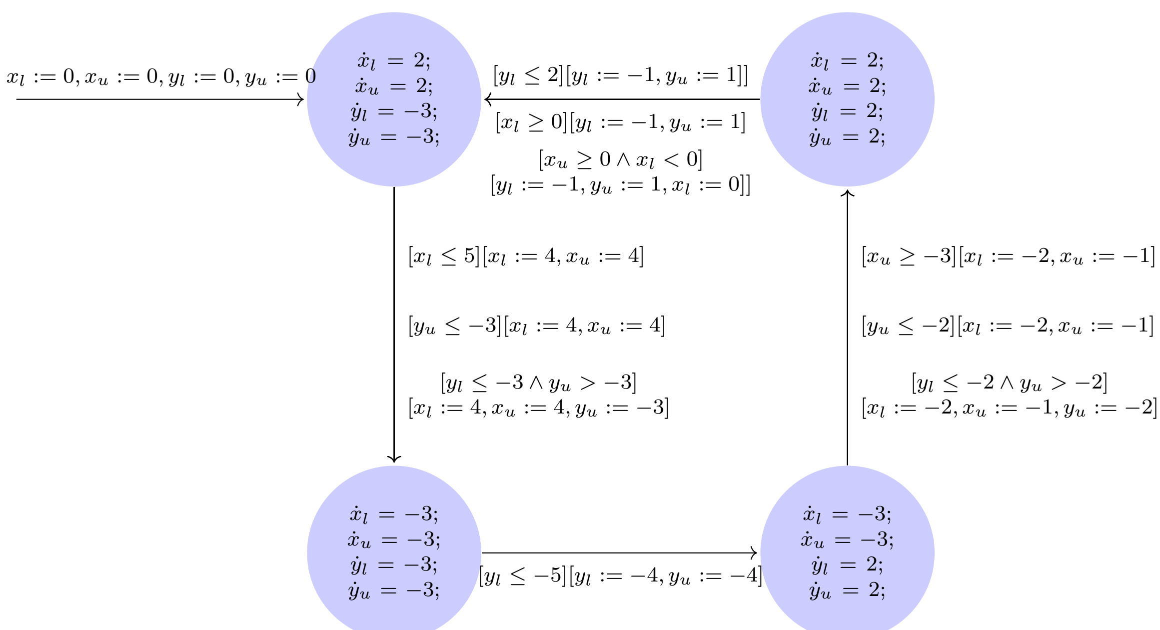
Problem 3 (10 points). (a) Convert the rectangular initialized hybrid automaton of Figure **??** to a timed automaton (clocks may be initialized to constant intervals). The first expression on the arrows are the preconditions/guards and the second expresssion is the effect or the reset function. No reset implies that the state variables are not reset.

(b) Plot an execution of the original hybrid automaton and the corresponding execution of the



timed automaton.

Solution



-
-

Problem 4 (20 points). Consider a system with two leaky tanks T_1 and T_2 and an inflow pipe P which can feed to either of the tanks. The inflow rate from P, when on, is f_{in}, and the outflow rates from the tanks (independent of any inflow) are f_1 and f_2. These rates are measured in terms of the rate of drop (and rise) of the water levels in the tanks. The controller for P is designed such that within δ time of the level in tank i dropping below h_i, the pipe P is turned to feed T_i.

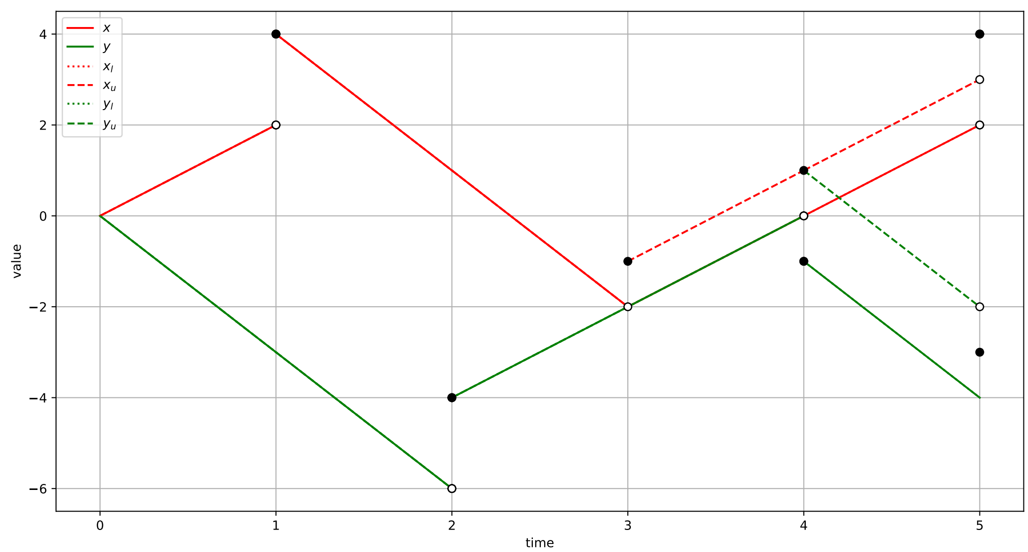


Figure 1: plot of execution

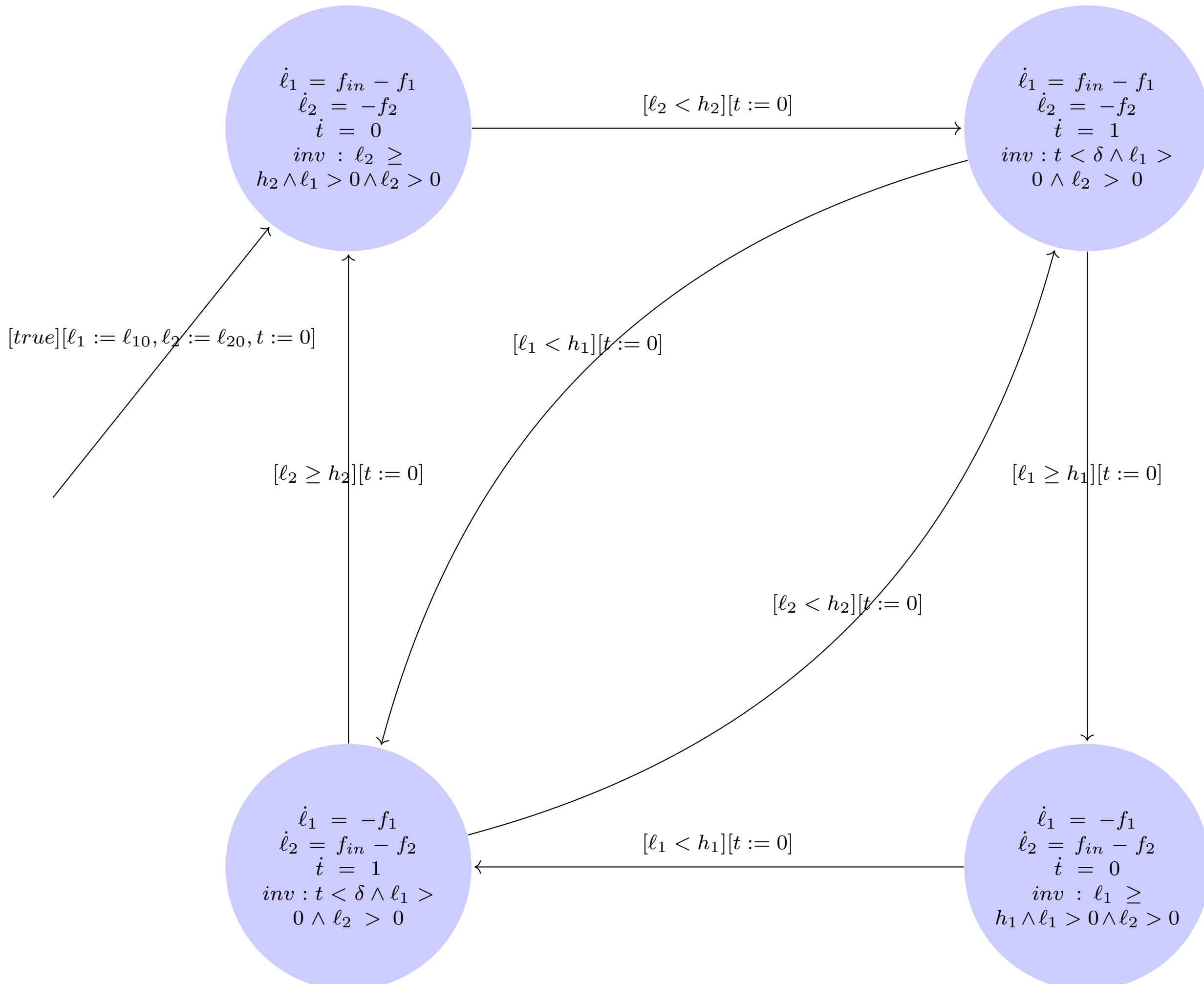
(a) Model the system as a hybrid automaton. Show the circle-arrow representation.

(b) Under what conditions does the model display Zeno behavior ?

(c) Under what conditions can it be guaranteed that neither tank becomes empty? Prove it.

Solution

(a) Variables: ℓ_1 the level of tank 1, ℓ_2 the level of tank 2, t a timer. Constants: ℓ_{10}, ℓ_{20} the initial value of ℓ_1, ℓ_2.



(b) When $\ell_1 < h1 \wedge \ell_2 < h2$, the model may display Zeno behavior (via the two bend arrows in the firure)?

(c) A sufficient condition is "$\ell_{10} > h_1 > \delta f_1, \ell_{20} > h_2 > \delta f_2, (f_{in} - f_1)(f_{in} - f_2) \geq f_1 f_2, f_{in} > 0$". Proof: Without loss of generality, we assume initially the pipe is filling tank1. When $t = \frac{\ell_{20}-h_2}{f_2}$, the level of tank2 is h_2. So, the pipe will transit to tank2 at $t_1 \in [\frac{\ell_{20}-h_2}{f_2}, \frac{\ell_{20}-h_2}{f_2} + \delta]$. Then, when $t = t_1 + \frac{\ell_{10}+(f_{in}-f_1)t_1-h_1}{f_1} > t_1 + \frac{(f_{in}-f_1)t_1}{f_1}$, the level of tank1 will be h_1. So between $[t_1, t_1 + \frac{(f_{in}-f_1)t_1}{f_1}]$, the pipe is filling tank2. It is easy to check that between $[0, t_1 + \frac{(f_{in}-f_1)t_1}{f_1}]$, neither tank gets empty because $h_1 > \delta f_1, h_2 > \delta f_2$. At time $t_1 + \frac{(f_{in}-f_1)t_1}{f_1}$, $\ell_1 = \ell_{10} + (f_{in} - f_1)t_1 - f_1\frac{(f_{in}-f_1)t_1}{f_1} = \ell_{10}$, and $\ell_2 = \ell_{20} - f_2 t_1 + (f_{in} - f_2)\frac{(f_{in}-f_1)t_1}{f_1} \geq \ell_{20}$ because $(f_{in} - f_1)(f_{in} - f_2) \geq f_1 f_2$. Ath this point, ℓ_1 and ℓ_2 satisfies the condition again. So by the inductive theorem, we know this condition is an invariant, and thus neither tank will be empty.

Problem 5 (40 points). Implement a basic reachability analysis algorithm for the billiards problem (from homework 2). To fix notations, let the state of the ith ball in the billiard table be $(x_i, y_i, v_i, u_i) \in^4$, where v_i and u_i are the velocities along x and y directions. Let $a, b > 0$ be the length and the width of the table. The algorithm should take as input:

1. The number $n > 0$ of balls.
2. The (uncertain) initial position intervals $X_i \subsetneq [0, a], Y_i \subsetneq [0, b]$, for each $i \in [n]$, where X_i is the uncertainty in initial x position and y is the uncertainty in initial y position of the ith ball.
3. The (fixed) initial velocities $v_{0,i}$ and $u_{0,i}$.
4. A set of unsafe locations specified as squares for one or more of the balls, just like the specification of the initial states.
5. A time bound T.

The output of the algorithm should be:

1. The set of reachable states of the balls starting from the specified initial positions, up to the time bound T.
2. A safety check answer that either shows that (a) the ball(s) does not reach the specified unsafe set, or (b) finds a particular initial configuration which makes the ball reach the unsafe set.

For simplicity, use rectangles (or hyperrectangles) to represent the reachable states of each ball.

Your reach set computer should propagate the uncertainties in the position correctly as the balls *bounce* on the walls of the billiard table. Assume that corners belong to one of the sides. However, note that because of the uncertainty in initial position, you will have to consider multiple bounce transitions that may occur for all the states described by a rectangle.

You may ignore ball-ball collisions. Bonus points for handling collisions properly with appropriate explanations. Note that if you ignore collisions, then you can compute the reachset of each ball independently.

You may use one of the reachability analysis tools like C2E2 [**?**], Flow* [**?**], CORA [**?**], or SpaceEx [**?**] *at your own risk*. That is, we will not be able to provide detailed consulting or bugfixes.

This problem will be graded based on a 10 minute demo where you will have to run the program and answer some questions.

Solution Some key observations:

- The interactions between the walls and the balls are linear. Starting from a convex initial set, the reachbale set at time t is also a convex set and the vertices corresponds to the vertices of the intial set.