CPS 499/573: Security and Safety in Autonomous Systems– Model and Analyze case studies using UPPAAL, Flow*, Matlab/Simulink tools Report

Rama Bhargavi Vempolu
Saivarshitha Thammera
Department of Computer Science
University of Dayton

Abstract

Four separate models are contained in the project: Ticket Vending Machine using UPPAAL, Railway Control System using UPPAAL, Mass-Spring-Damper using MATLAB/Simulink and Flow*, and Glycaemic Control using Flow*. Every model represents a distinct use of simulation tools and control systems. Using MATLAB/Simulink and Flow*, the Mass-Spring-Damper model provides a basic mechanical system for analyzing control strategies and dynamic responses. Formal verification is emphasized by the Railway Control System, which is modeled using UPPAAL, to guarantee the effectiveness and safety of vital transportation facilities. The Ticket Vending Machine, which is likewise modeled using UPPAAL, emphasizes transactional dependability and user engagement, demonstrating the effectiveness of the tool in handling concurrent operations in real-time systems. Lastly, the Glycaemic Control model explores biomedical applications, particularly blood sugar management, by using Flow*.

I INTRODUCTION

In the evolving landscape of system modeling and control, a multitude of simulation tools have been employed to address diverse challenges in various domains. This report presents a comprehensive examination of four benchmark models, each utilizing distinct tools to simulate and analyze different system dynamics. These models include the Mass-Spring-Damper using MATLAB/Simulink and Flow*, Railway Control System using UPPAAL, Ticket Vending Machine using UPPAAL, and Glycaemic Control using Flow*. The Mass-Spring-Damper system, implemented in MATLAB/Simulink and Flow*, serves as a fundamental example to explore dynamic response and control strategy in mechanical systems. The Railway Control System, modeled with UPPAAL, showcases the application of formal verification in enhancing the safety and efficiency of critical transportation infrastructures. In a similar vein, the Ticket Vending Machine model, also developed using UPPAAL, focuses on the intricacies of user interaction and transactional reliability, emphasizing the tool's ability to handle concurrent processes in real-time systems. Lastly, the Glycaemic Control model, realized through Flow*, ventures into the biomedical field, specifically targeting the management of blood sugar levels, thus demonstrating the tool’s proficiency in simulating complex biological processes. These models collectively highlight the versatility and necessity of various simulation tools in the design and validation of control systems across different fields, from mechanical to biomedical, underpinning the broader implications and applications of these technologies in system modeling and analysis.

II TIMED AUTOMATA

In this section, we explore the concept of timed automata, a critical framework in modeling and analyzing time-dependent systems. Timed automata are specialized forms of automata that are adept at handling systems where the timing of events is a crucial aspect of the system's behavior. A timed automaton is effectively a finite state machine augmented with a set of clocks, which are real-valued variables that progress uniformly over time. The formal definition of a timed automaton is as follows:

Definition 1: A timed automaton T is a tuple, T = ⟨Loc, C, Inv, Trans, Init⟩, where its components are defined as:

- Loc is a finite set of states or locations.
- C is a finite set of clocks, real-valued variables that increase at the same rate.
- Inv (Invariants): Each state in Loc is associated with an invariant from Inv. These invariants are conditions on the clocks that must hold true for the system to reside in that state.
- Trans (Transitions): A set of edges or transitions between states. Each transition is defined as a tuple ⟨l, l', g, r⟩, where l is the current state, l' is the next state, g is a guard condition based on clock values that enables the transition, and r is a set of clocks to be reset with this transition.
- Init (Initial State): A set of initial states and initial values for the clocks.

The semantics of a timed automaton T are defined in terms of its executions, which are sequences of states and time steps. An execution ρ of T is a sequence ρ = s0, s1, s2, ..., where s0 ∈ Init, and the transition from state si to state si+1 is determined by the transition relations in T. These transitions can be of two types: (a) a discrete jump from one state to another when a guard condition is satisfied and clocks are reset as specified, or (b) a time passage where the values of the clocks increase uniformly while staying within the state invariants.

Timed automata are particularly useful for modeling systems where timing constraints are critical, such as in network protocols, real-time systems, and embedded systems. Tools like UPPAAL have been designed specifically to model, simulate, and verify timed automata. In the context of the models

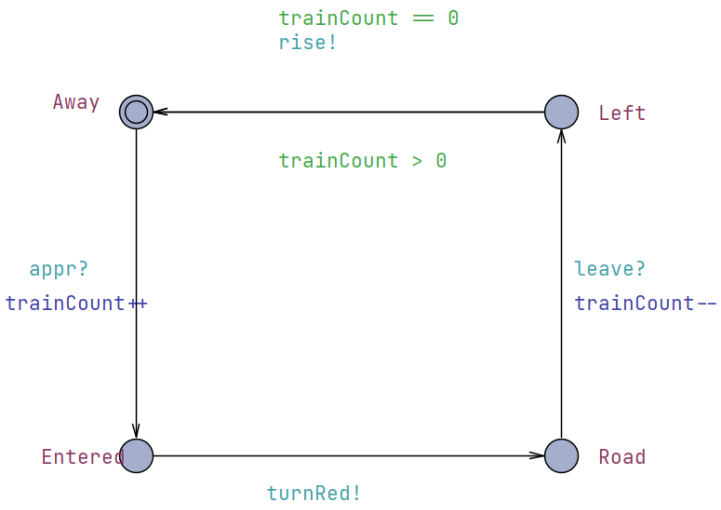
mentioned earlier, such as the Railway Control System and the Ticket Vending Machine, timed automata provide a robust framework for ensuring that timing constraints are met, and the system behaves as expected within the specified time frames.

III. TYPICAL BENCHMARKS

1. *Railway Control System using UPPAAL*

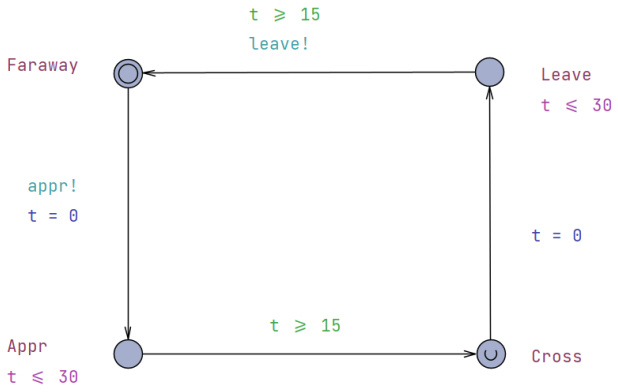
The model aims to simulate the interactions between trains, railway gates, and traffic lights at a crossing, emphasizing the critical aspect of timing in such systems. The model incorporates several elements, including train movements, sensor detections, light signaling, and gate operations, to ensure safe and efficient railway and road traffic management.

The UPPAAL model is structured around several key components:



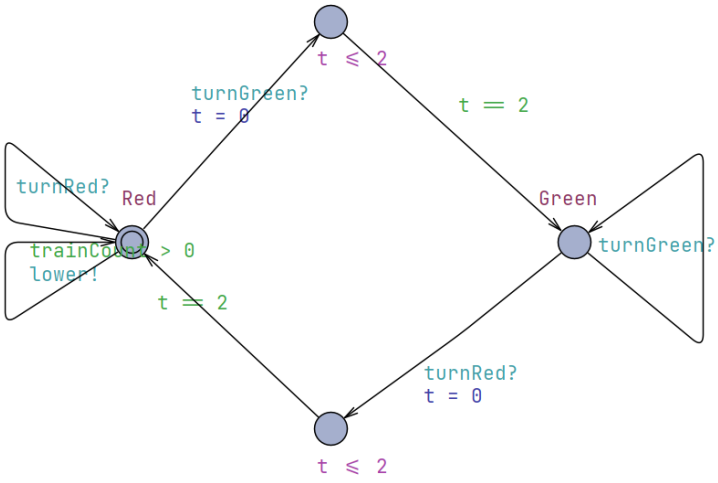
Global Declarations: The model defines global constants, variables, and channels. Notably, it specifies N as the number of trains, a trainId type for train identifiers, and trainCount to keep track of trains in the system. Communication channels like appr (approach), leave, rise, lower, turnGreen, and turnRed are established for synchronization between components.

Train Model (Trace Template): Each train is modeled with states representing its position: Approaching (Appr), Crossing (Cross), Leaving (Leave), and Faraway (Faraway). Transitions between these states are guarded by time constraints and synchronized with sensor and gate operations.

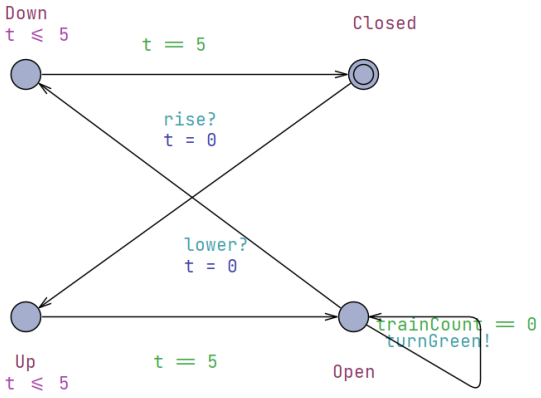


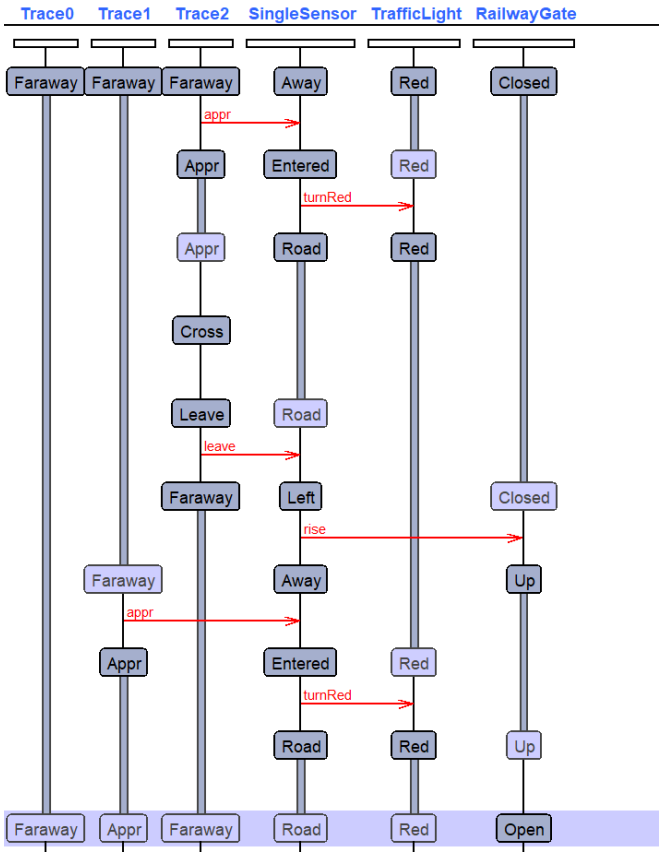
Sensor Model: The sensor detects train movements and updates trainCount. It has states like Road, Entered, Left, and Away, transitioning based on train movements and the number of trains at the crossing.

Light Model: It models a traffic light with states Green and Red. Transitions between these states are synchronized with the gate's status and train presence.



Gate Model: The gate's operational states are Up, Open, Down, and Closed. It reacts to train approaches and departures, coordinating with traffic lights for safe passage.





Verification Queries

Several verification queries are embedded in the model:

Safety: Queries like A[] Trace0.Cross imply Trace0.t <= 30 ensure trains don't stay at the crossing for too long. Others, such as A[] not (Trace0.Cross and Trace1.Cross), verify that two trains cannot be in the crossing simultaneously.

Timeliness: Queries such as E<> (RailwayGate.Down and RailwayGate.t <= 5) check if the gate can be lowered in a timely manner.

Operational Consistency: The model checks for scenarios like the traffic light being red when the gate is down (A[] (RailwayGate.Down imply TrafficLight.Red)) and ensures that the train count is within the expected range (A[] (trainCount >= 0 and trainCount <= N)).

Deadlock Avoidance: The query A[] not deadlock verifies that the system does not reach a state where no further actions are possible.

2. Ticket Vending Machine using UPPAAL

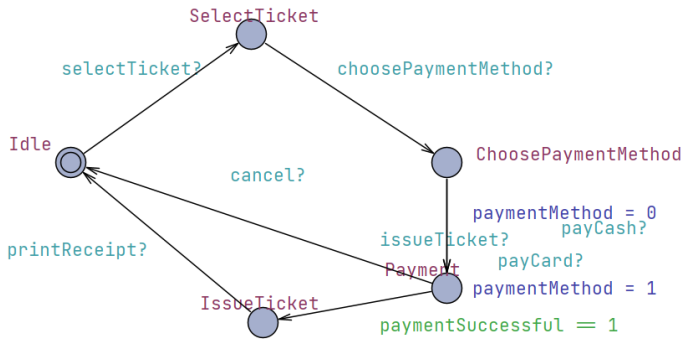
Introduction

TheTimed Automata model of a Ticket Vending Machine (TVM) system, designed and simulated using UPPAAL. The model aims to capture the interaction between a user and the TVM, focusing on the processes of ticket selection, payment method choice, payment processing, and ticket issuance. The model's primary goal is to ensure the system's reliability, efficiency, and deadlock-free operation.

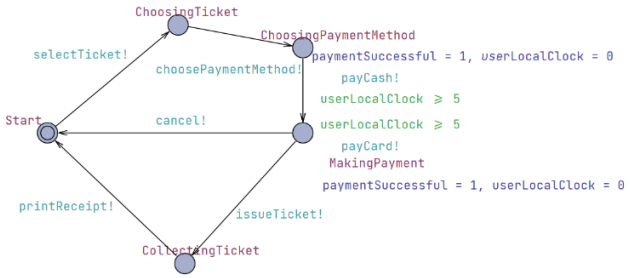
Model Description

The UPPAAL model for the TVM system consists of two main components:

TVM (Template for Ticket Vending Machine): This component models the states and transitions of the TVM, including Idle, SelectTicket, ChoosePaymentMethod, Payment, and IssueTicket. The transitions between these states are triggered by user actions and internal processes like payment validation.

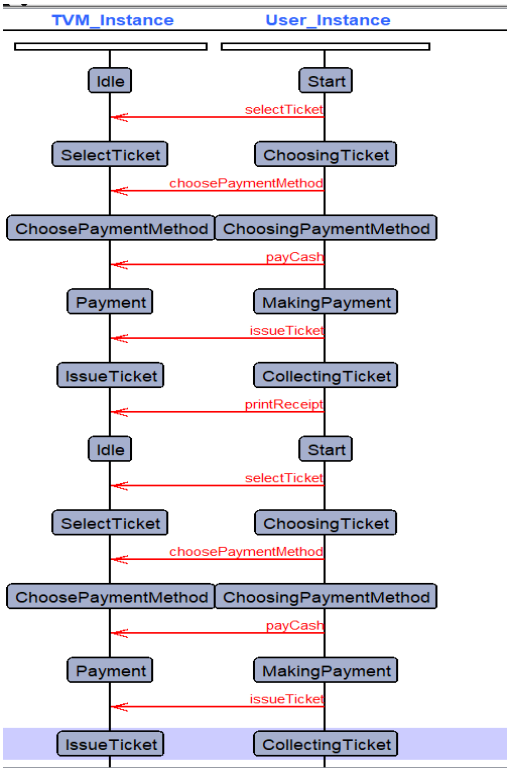


User (Template for User Interaction): This component represents the user interacting with the TVM. It includes states like Start, ChoosingTicket, ChoosingPaymentMethod, MakingPayment, and CollectingTicket. The user's transitions are synchronized with the TVM's operations, reflecting a realistic interaction scenario.



- Model Dynamics
- The TVM starts in the Idle state and transitions to SelectTicket upon user action.
- Once a ticket is selected, it moves to ChoosePaymentMethod, where the user chooses between cash or card payment.
- The Payment state handles the payment process, with different transitions for cash and card payments.

- On successful payment (paymentSuccessful == 1), the TVM issues the ticket and optionally prints a receipt.



Verification Queries

The model includes several critical verification queries:

Ticket Issuance Validation: Checks that a ticket is issued only after successful payment (e.g., A[] (TVM_Instance.IssueTicket imply paymentSuccessful == 1)).

Deadlock-Free Operation: Verifies the system is free from deadlocks, ensuring continuous operability.

Operational Feasibility: Confirms the system can reach a state where a ticket is issued (e.g., E<> TVM_Instance.IssueTicket).

Payment Method Effectiveness: Ensures that the selected payment method leads to successful ticket issuance (e.g., E<> (paymentMethod == 1 and TVM_Instance.IssueTicket)).

Payment Method Influence: Validates that the choice of payment method influences the payment success.

IV. HYBRID AUTOMATA

In this section, we focus on the application of hybrid automata for modeling two distinct systems: the Mass-Spring-Damper system and Glycaemic Control, both using the Flow* tool. Hybrid automata, known for their efficiency in modeling dynamical systems with both continuous physical dynamics and discrete transitions, are particularly well-suited for these complex systems.

Definition 1: A hybrid automaton H for these models can be structured as H = ⟨Loc, Var, Inv, Flow, Trans, Init⟩, where the components are defined as:

- Loc (Locations): A finite set of discrete locations representing different states or phases of the system.
- Var (Variables): A finite set of continuous, real-valued variables, with the state-space $Q = Loc \times R^n$. A state s in Q is a tuple s = (l, val(x)), where l is a location in Loc and val(x) is a valuation of x in Var.
- Inv (Invariants): A set of invariants associated with each location l in Loc, defining the permissible values of Var in that location.
- Flow: A set of ordinary differential inclusions for each continuous variable x in Var, describing the continuous dynamics in each location l.
- Trans (Transitions): A finite set of transitions between locations, each represented as a tuple $\tau = \langle l, l', g, u \rangle$, where l is the source location, l' is the target location, g is a guard condition, and u is an update map for state transition.
- Init (Initial State): An initial condition consisting of a set of locations in Loc and a formula over Var.

For the Mass-Spring-Damper system, the hybrid automaton captures the interaction between the mass, spring, and damper components, encompassing both the continuous motion (described by differential equations) and discrete transitions representing different system states or external interventions.

In the case of Glycaemic Control, the automaton models the complex interplay between glucose and insulin levels in the human body. This includes continuous physiological processes (like insulin metabolism and glucose absorption) and discrete events (such as insulin injections or dietary intake changes).

In both models, the execution semantics of the hybrid automaton H are defined by a sequence of states $\rho = s0 \rightarrow s1 \rightarrow s2 \rightarrow ...$, where each transition $si \rightarrow si+1$ is dictated by the hybrid automaton's transition relations. These transitions can be either discrete (instantaneous state updates) or continuous (state updates over a real-time interval).

The Flow* tool is employed to compute reachable sets for these models, allowing for the analysis and verification of their dynamic behavior under varying conditions. This approach facilitates a comprehensive understanding of the systems' responses to different inputs and scenarios, crucial for their effective control and management.

V. TYPICAL BENCHMARKS

1. *Mass Spring Damper using Flow**

The analysis of a Mass-Spring-Damper system modeled and simulated using the Flow* tool, a framework designed for hybrid reachability analysis. The focus is on examining the system's dynamics over time through reachability analysis, which determines the set of all states the system can reach from its initial state under certain conditions.

a. Model Specification

The Mass-Spring-Damper system is defined with three state variables: x (position), v (velocity), and t (time). The model uses a fixed-step method with a step size of 0.01 and a total simulation time of 30 units. The state variables are represented with a precision of 53 bits, and the remainder estimation is set to 1e−7. The system is designed to operate in a single mode, 'operating', without any jumps, reflecting continuous behavior over time.

b. System Dynamics

In the 'operating' mode, the system is governed by the following non-polynomial ordinary differential equations (ODEs):

- **x′=v:** The rate of change of position x is equal to the velocity
- **v′=(−((2)∗v)−((8)∗x))/2:** This represents the dynamics of the spring and damper, where the velocity v changes over time due to the spring force and damping.
- **t′=1:** Time progresses uniformly. The invariant condition t<=6 ensures that the system operates within this mode for a maximum of 6 time units.

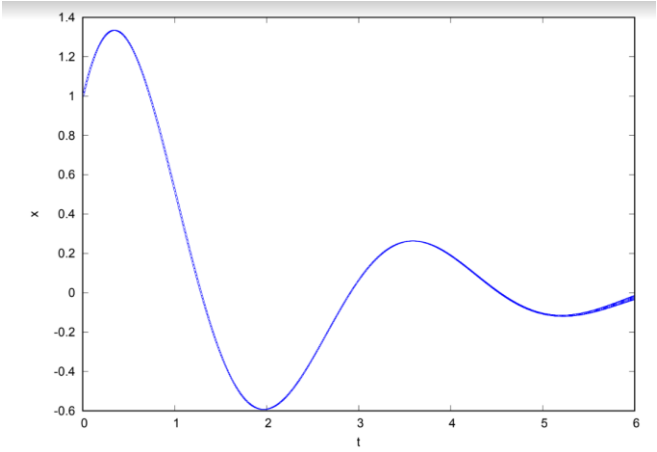
c. Initial Conditions

The initial conditions of the system are set as follows:

- Position x is initialized in the range [1, 1.001].
- Velocity v is within [2, 2.0001].
- Time t starts from [0, 0.01].

d. Analysis Setting and Output

The analysis employs adaptive orders for the Taylor model, ranging from a minimum of 2 to a maximum of 5, with a cutoff threshold of 1e−15. The output is configured to plot the reachability set in the t-x plane using Gnuplot and is saved under the name "Mass_Spring_Damper". The system does not involve discrete jumps (max jumps 0), indicating a purely continuous system behavior.



The graph plots the state variable x (presumably representing the displacement of the mass in the Mass-Spring-Damper system) against time t.

Here's a breakdown of the graph:

- The horizontal axis t represents time in some units (likely seconds), ranging from 0 to 6.
- The vertical axis x represents the displacement of the mass from its equilibrium position.
- The graph shows a typical behavior of a damped harmonic oscillator, which is what a Mass-Spring-Damper system essentially is. The oscillations represent the mass moving back and forth as the spring exerts force on it and the damper provides a force that resists the motion (damping).

From the initial condition, you can observe the following:

Initial Peak: The graph starts with an initial positive displacement x, suggesting that the mass was released from a point to the right of the equilibrium position (positive displacement) with an initial velocity (since the curve does not start from a peak).

Damping Effect: The amplitude of the oscillations decreases over time, which indicates the presence of damping in the system. This is consistent with the system's differential equation where the damping coefficient is represented by the term 2v.

Frequency of Oscillation: The time between successive peaks gives an indication of the natural frequency of the system. This is determined by the spring constant and the mass involved but will be affected by the damping factor.

Settling Time: The graph does not show the system coming to a complete rest within the depicted time frame, but the amplitude of oscillations is reducing, suggesting that it will eventually settle down to the equilibrium position as time goes on, assuming there's enough damping.

No Overdamping or Critical Damping: The presence of oscillations indicates that the system is not overdamped nor critically damped. Overdamping would not show oscillations, and critical damping would return to equilibrium in the shortest time without oscillations.

Stability: The system appears to be stable as the oscillations are decaying, and there is no indication of growing amplitudes, which would suggest instability.

2. *Glycaemic control using Flow**

The analysis of a glycaemic control system using the Flow* tool, which is adept at performing reachability analysis for systems described by hybrid automata. Glycaemic control systems are used to understand and manage the glucose-insulin dynamics in the human body, and are particularly relevant for the treatment and management of diabetes.

a. Model Specification

The glycaemic control model is defined using state variables G (glucose level), X (insulin action), I (insulin level), t (time), and tl (local time for transition logic). The model is simulated over a period of 720 units of time (presumably minutes), with a fixed-step size of 0.05 units. The remainder estimation is set at $1e{-}1$, and the precision for the state variables is set to 53 bits. The system is configured to operate within 10 discrete transitions (max jumps), and the output is directed to a file named "glucose_control_I".

b. System Dynamics

The glycaemic control system is described by three modes, each with its own set of non-polynomial ordinary differential equations (ODEs) and invariants:

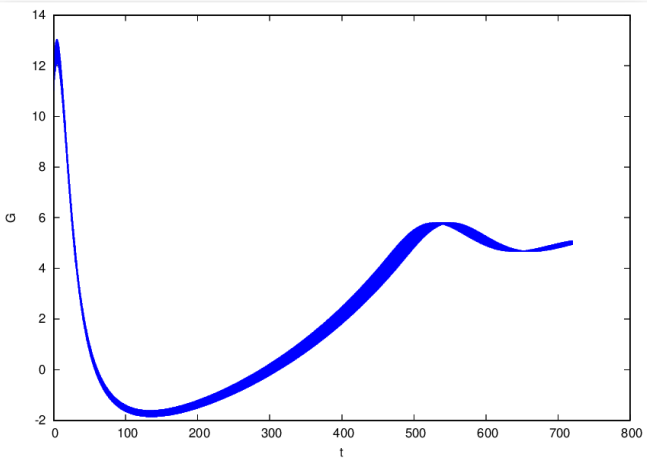
- **Mode l1**: Represents the state where glucose levels are at or below a threshold of 4 units. The ODEs model the change in glucose, insulin action, and insulin level over time. The invariant conditions for this mode restrict the glucose level $G \leq 4$ and time $t \leq 720$.
- **Mode l2**: Describes an intermediate range of glucose levels between 4 and 8 units. The ODEs are similar to those in mode l1, with a modification in the insulin level's ODE to include a term proportional to glucose G and a constant decay. The invariants ensure $4 \leq G \leq 8$ and $t \leq 720$.
- **Mode l3**: Applies when glucose levels are high, at or above 8 units. The ODEs follow the pattern of the previous modes, with a different constant rate for the insulin level's ODE. The invariants specify $G \geq 8$ and $t \leq 720$.

c. Transitions and Initial Conditions

Transitions between modes are governed by guards based on the glucose level G and the local transition time tl. Resets are applied to the local time tl upon each transition.
The initial conditions place the system in mode l3 with glucose levels between 11 and 12 units, no initial insulin action X, a fixed initial insulin level I, and both time variables t and tl starting from zero.

d. Analysis Setting and Output

The analysis employs gnuplot for visualization, with adaptive orders for the Taylor model between 2 and 5, and a cutoff threshold of $1e{-}12$. The system prints the reachability set in the t-G plane, providing insights into the glucose level over time.



The graph provided is a visualization of glucose levels (G) over time (t) generated from a hybrid reachability analysis of a glycaemic control system using Flow*. Here's what the graph illustrates:

- The horizontal axis (t) represents time, which in the context of the model, likely spans 720 minutes, reflecting a 12-hour period which is a common timeframe for observing blood glucose fluctuations.
- The vertical axis (G) represents glucose levels.

Key observations from the graph are:

1. **Initial High Glucose Level**: The graph starts with an initial glucose level between 11 and 12, which is considerably high, indicating a hyperglycemic state.
2. **Rapid Decrease**: There is a steep decline in the glucose level initially, indicating an effective insulin action or absorption phase, which could be the body's natural insulin response or an administered dose taking effect.
3. **Stabilization and Fluctuation**: After the initial drop, the glucose level stabilizes but still shows fluctuations within a certain range. This could reflect the interplay between insulin action, glucose absorption, and perhaps meal intake or other physiological factors affecting glucose levels.
4. **Transitions between Modes**: The changes in the slope of the curve may correspond to the transitions between modes defined in the system. When glucose reaches the thresholds of 4 or 8, it triggers a change in the model's dynamics, which is reflected in the graph as changes in the behavior of the glucose level over time.
5. **Overall Control Trend**: The graph suggests that while the glucose level is brought down from a high point and is prevented from escalating back to that level, there is still variability that may need to be addressed for tighter glycaemic control.
6. **Ending Glucose Level**: Towards the end of the 720-minute period, the glucose level appears to be rising slightly. This could suggest a waning insulin effect or other metabolic changes increasing glucose levels, which may require additional insulin administration or

other interventions.

3. Mass Spring Damper using MATLAB/Simulink

This report provides an analysis of a Mass-Spring-Damper system modelled using MATLAB/Simulink. The system is a classic example of a second-order linear system used in engineering to study dynamic behaviour, control strategies, and vibration analysis.

a. System Definition

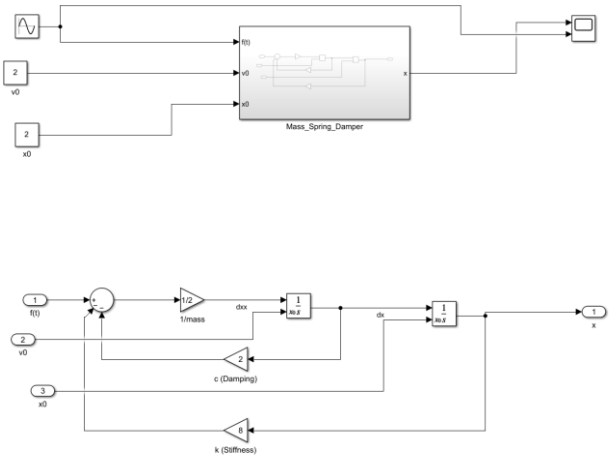
The system comprises a mass (m), a damper (c), a spring (k), and an external forcing function f(t). The differential equation governing the system's motion is given by:

$$mx''+cx'+kx=f(t)$$
or equivalently,
$$x''=m1(f(t)-cx'-kx)$$

b. System Parameters

For the model under consideration, the parameters are defined as follows:

- Mass (m): 2 kg
- Damping coefficient (c): 2 N·s/m
- Spring constant (k): 8 N/m
- Forcing function ($f(t)$): 2sin(2t) N



c. Simulink Model

The Simulink model comprises several interconnected blocks representing the system's physical components and their interactions:

- **Function Generator**: Represents the external force $f(t)$, set to 2sin(2t) to simulate a sinusoidal forcing function with an amplitude of 2 N and a frequency that corresponds to a 2t argument in the sine function.

- **Mathematical Operations**: The model uses an integrator block to simulate the second derivative of the mass's position (x''), which is the acceleration. It integrates the result to get the velocity (x') and integrates again to obtain the position (x).

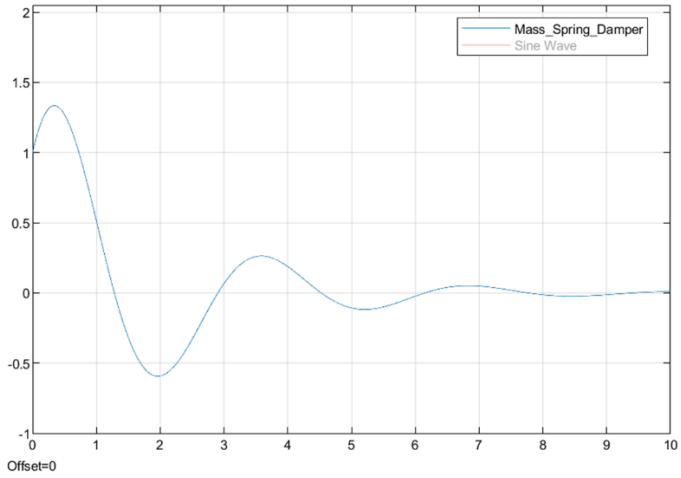
- **Gain Blocks**: The blocks labeled 1/*mass*, c (Damping), and k (Stiffness) apply the respective inverse mass, damping coefficient, and spring constant to the corresponding variables.

- **Summation Blocks**: Used to combine the forces acting on the mass according to the governing differential equation. They account for the restoring force of the spring, the damping force, and the external force.

- **Scope**: Visualizes the output, which is the displacement x of the mass over time.

d. Observations and Analysis

- The Simulink model is set up to simulate the response of the mass to the sinusoidal external force over time.
- The integrator blocks and the feedback loops correctly implement the second-order differential equation that describes the system's dynamics.
- The scope output will display the periodic response of the system, which includes both transient and steady-state behaviour due to the sinusoidal forcing function.

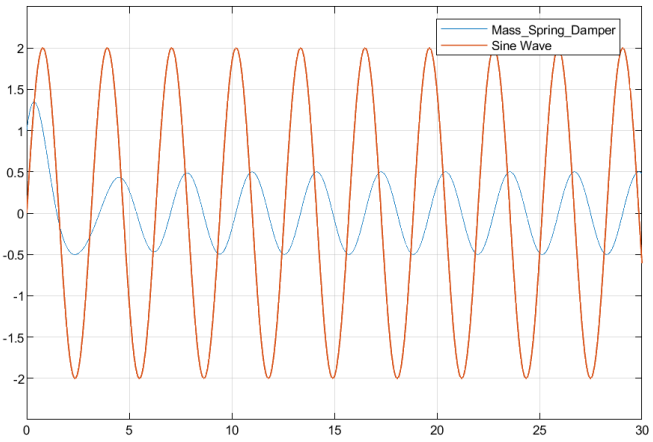


The graph is a simulation output from MATLAB/Simulink of a Mass-Spring-Damper system without using sin function.
Here is the breakdown of the graph:

- The horizontal axis represents time, and it seems to span from 0 to 10 units, which could be seconds or any other unit of time depending on the simulation settings.
- The vertical axis represents the displacement of the mass (denoted as x) in the Mass-Spring-Damper system. The units here could be meters or any unit of length, again depending on the simulation settings.
- The initial peak in the graph indicates the system's response to the initial conditions or the immediate effect of the applied force.
- The oscillatory pattern shows the natural response of the system, which is the characteristic of a damped harmonic oscillator. The amplitude of the oscillation decreases over time, which is indicative of the damping effect in the system.
- The system does not appear to be critically damped because it oscillates about the equilibrium position before settling

down. If it were critically damped, it would return to equilibrium without oscillating.

- The frequency of the sinusoidal input seems to be close to the natural frequency of the system, as indicated by the significant initial displacement. However, since the system eventually settles, it is not in a state of resonance, which would have been indicated by increasing or sustained large amplitude oscillations.
- Towards the end of the simulation, the amplitude of oscillation diminishes, showing that the system is approaching steady-state behavior where the oscillations due to the initial conditions have largely been damped out.



The graph is the output of a MATLAB/Simulink simulation for a Mass-Spring-Damper system with an overlay of the applied sinusoidal force.
Here's an interpretation of the graph:

- The horizontal axis represents time, spanning from 0 to 30 units. These units are typically in seconds for such simulations but can be scaled as needed.
- The vertical axis represents the amplitude, which could be in units of force for the applied sine wave and displacement for the mass-spring-damper response.
- There are two plots on the graph:
    - The blue plot labeled "Mass_Spring_Damper" likely shows the displacement of the mass over time.
    - The orange plot labeled "Sine Wave" represents the sinusoidal force applied to the system, with the equation $f(t)=2\sin(2t)$.

Key observations from the graph:

1. **Resonance Phenomenon**: The orange sine wave seems to be the driving force, and its frequency appears to match or is close to the natural frequency of the Mass-Spring-Damper system, causing resonance, which is characterized by the large amplitude oscillations of the blue plot.
2. **Steady-State Oscillation**: The system's response (blue plot) reaches a steady-state relatively quickly, indicating a stable system. The steady-state response is periodic and has the same frequency as the driving sine wave.
3. **Amplitude Comparison**: The amplitude of the system's response is larger than that of the forcing function, again suggesting that the system is operating at or near resonance, which typically amplifies the response amplitude compared to the forcing amplitude.
4. **Phase Difference**: There is a noticeable phase difference between the system's response and the driving force. In resonant conditions, the displacement of a damped harmonic oscillator typically lags the driving force by a phase angle approaching $2\pi/2$ (90 degrees).
5. **Damping Effect**: The response does not show an unbounded increase in amplitude, which means the system is damped. The damping force dissipates energy, preventing the amplitude from growing infinitely, which would happen in an undamped resonant system.
6. **No Visible Decay**: Over the 30-unit time span, the amplitude of the system's response does not appear to decay, which suggests that the damping is not strong enough to dissipate the energy input by the sinusoidal force over the displayed time frame.

VI. CONCLUSIOIN

This report illustrates the remarkable versatility and precision of simulation tools across diverse domains, from mechanical systems to biomedical applications. The Mass-Spring-Damper system modeled in MATLAB/Simulink and Flow* offers fundamental insights into dynamic behavior and control strategies. The Railway Control System and the Ticket Vending Machine, both simulated using UPPAAL, emphasize the critical role of timed automata in ensuring safety and efficiency in transportation and transactional reliability in real-time systems. Lastly, the Glycaemic Control model, developed with Flow*, ventures into the complex interplay of biological processes, highlighting the tool's capability in simulating intricate physiological dynamics. Collectively, these models underscore the integral role of simulation tools in advancing our understanding and management of complex systems, with broad implications across various fields including engineering, transportation, and healthcare.

REFERENCES

https://ths.rwth-aachen.de/research/projects/hypro/benchmarks-of-continuous-and-hybrid-systems/

https://www.mathworks.com/help/simscape/ug/mass-spring-damper-in-simulink-and-simscape.html