First results on nonlinear hybrid reachability combining interval Taylor method and IBEX library

M. Maiga[1,2], N. Ramdani[1], and L. Travé-Massuyès[2]

[1] Université d'Orléans, PRISME, 18020 Bourges, France
[2] CNRS, LAAS, 31400 Toulouse, France
(e-mail: mmaiga@laas.fr, nacim.ramdani@univ-orleans.fr, louise@laas.fr)

SWIM, Oldenburg, Germany
June 4-6, 2012

Outline

ANR-Project : MAGIC-SPS

- Goal : To develop guaranteed methods and algorithms for integrity control and preventive monitoring of systems
- Different work package :
  - WP1 : Modelling and identification of systems with bounded uncertainties ;
  - WP2 : Identifiability and diagnosability of systems with bounded uncertainties ;
  - WP3 : Preventive monitoring of continuous systems with bounded uncertainties ;
  - WP4 : Preventive monitoring of hybrid systems with bounded uncertainties ;
  - WP5 : Dissemination
- Project duration = october 2012 to december 2014
- Partners

ANR-Project : MAGIC-SPS

Our work package : WP4

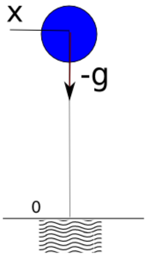
* Computing nonlinear hybrid reachability ;
* State estimation of HDS ;
* Feasibility of a fault prognosis for HDS

Outline

Hybrid System example : Bouncing ball

- Continuous dynamic (Free fall) $\rightarrow$ **Condition 1**

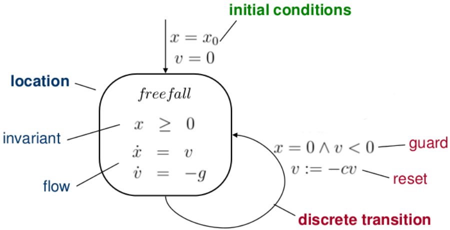
$$x \geq 0$$
$$\dot{x} = v$$
$$\dot{v} = -g$$

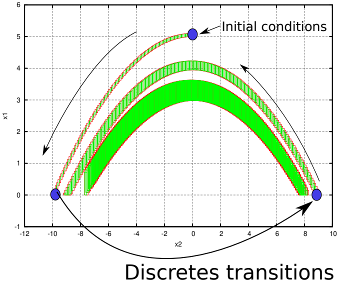


- Discrete dynamic (Bouncing) $\rightarrow$ **Condition 2**
  if $x = 0$ and $v < 0$; $v := -cv$
  - Velocity change direction
  - loss of velocity (deformation, friction)
  - $0 \leq c \leq 1$

Hybrid System example : Boucing ball

Condition 1 $\oplus$ Condition 2





$\mathbf{x}_0 \in [5, 5.1], \mathbf{v}_0 = 0$
$c = 0.8,\ g = 9.8,\ t \in [0, 4.5]$

Hybrid Reachability Computation

Hybrid automaton (Alur, *et al.*, 95)

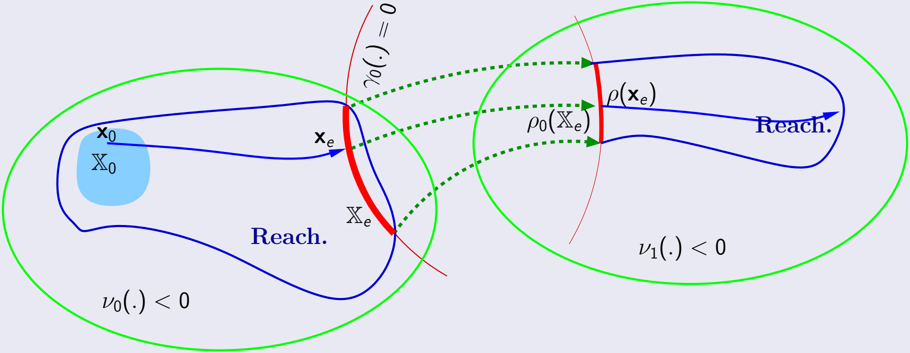
$$H = (\mathcal{Q}, \mathcal{D}, \mathcal{P}, \Sigma, \mathcal{A}, \mathrm{Inv}, \mathcal{F}),$$

$$\begin{aligned}
\mathrm{flow}(q): \quad & \dot{\mathbf{x}}(t) = f_q(\mathbf{x}, \mathbf{p}, t), \\
\mathrm{Inv}(q): \quad & \nu_q(\mathbf{x}(t), \mathbf{p}, t) < 0, \\
\\
e: \quad & (q \to q') = (q, \mathrm{guard}, \sigma, \rho, q'), \\
\mathrm{guard}(e): \quad & \gamma_e(\mathbf{x}(t), \mathbf{p}, t) = 0,
\end{aligned}$$

$$t_0 \leq t \leq t_N, \quad \mathbf{x}(t_0) \in \mathbb{X}_0 \subseteq \mathbb{R}^n, \quad \mathbf{p} \in \mathbb{P}$$

Hybrid Reachability Computation

Set reachable in finite time



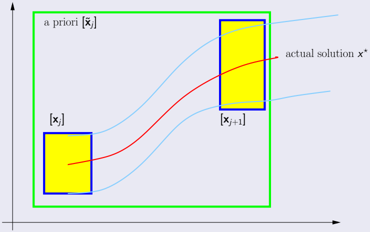
Outline

Guaranteed set integration with Taylor methods

(Moore,66) (Eijgenraam,81) (Lohner,88) (Rihm,94) (Berz,98) (Nedialkov,99)

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, \mathbf{p}, t), \quad t_0 \leq t \leq t_N, \; \mathbf{x}(t_0) \in [\mathbf{x}_0], \; \mathbf{p} \in [\mathbf{p}]$$

Time grid $\rightarrow \quad t_0 < t_1 < t_2 < \cdots < t_N$



- Analytical solution for $[\mathbf{x}](t), \; t \in [t_j, t_{j+1}]$

$$[\mathbf{x}](t) = [\mathbf{x}_j] + \sum_{i=1}^{k-1} (t - t_j)^i \mathbf{f}^{[i]}([\mathbf{x}_j], [\mathbf{p}]) + (t - t_j)^k \mathbf{f}^{[k]}([\tilde{\mathbf{x}}_j], [\mathbf{p}])$$

Guaranteed set integration with Taylor methods

(Moore,66) (Eijgenraam,81) (Lohner,88) (Rihm,94) (Berz,98) (Nedialkov,99)

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, \mathbf{p}, t), \quad t_0 \leq t \leq t_N, \, \mathbf{x}(t_0) \in [\mathbf{x}_0] \, , \, \mathbf{p} \in [\mathbf{p}]$$

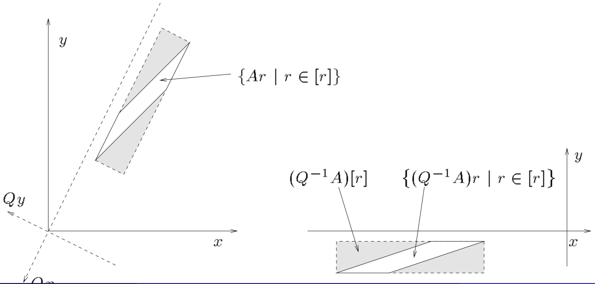
Mean-value approach

mean value forms + matrice preconditioning+ linear transforms

$$[\mathbf{x}](t) \in \left\{ \mathbf{v}(t) + \mathbf{A}^{\,a}(t)\mathbf{r}(t) \quad \middle| \quad \mathbf{v}(t) \in [\mathbf{v}](t), \;\, \mathbf{r}(t) \in [\mathbf{r}](t) \right\}.$$

---

a. Several methods

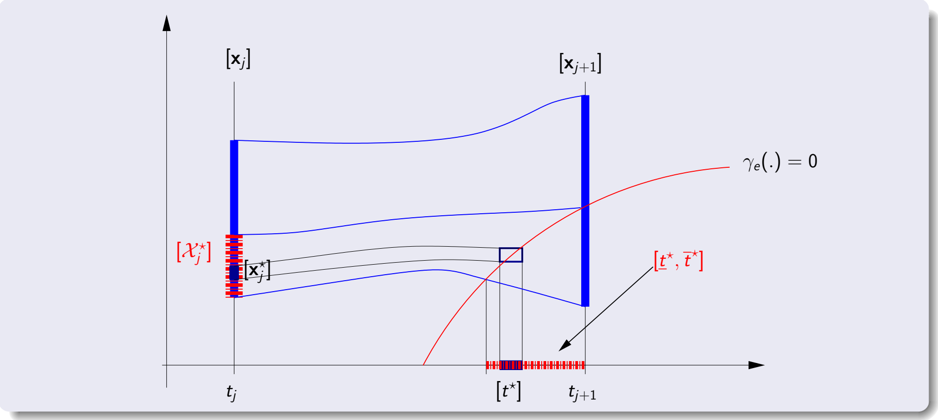


Outline

1. Introduction
2. Hybrid System
3. Interval Taylor Methods
4. Hybrid Transitions
5. IBEX library
6. Evaluation on Benchmarks
    - A simple illustrative exemple : 2 modes, continuous state dim=2
    - Benchmark 1
    - Benchmark 2
    - Benchmark 3
7. Conclusion

Computing flow/guards intersection

Time grid $\rightarrow \quad t_0 < t_1 < t_2 < \cdots < t_N$



Compute $[\underline{t}^\star, \overline{t}^\star] \times [\mathcal{X}_j^\star]$

Computing flow/guards intersection

Time grid $\rightarrow$ $t_0 < t_1 < t_2 < \cdots < t_N$

- $[\mathbf{x}](t) =$ Interval Taylor Series (ITS)$(t, [\mathbf{x}_j], [\tilde{\mathbf{x}}_j])$
- $\gamma([\mathbf{x}](t)) = 0$

$\Rightarrow$ $\gamma \circ \mathrm{ITS}(t, \mathbf{x}_j, [\tilde{\mathbf{x}}_j]) \rightarrow \psi(t, \mathbf{x}_j)$

To compute $[\underline{t}^\star, \overline{t}^\star] \times [\mathcal{X}_j^\star] \Rightarrow$ Solve CSP [1] $([t_j, t_{j+1}] \times [\mathbf{x}_j], \psi(.,.) = 0)$

1. Handbook of Constraint Programming, Rossi et al.,2006

Computing flow/guards intersection

Time grid $\rightarrow \quad t_0 < t_1 < t_2 < \cdots < t_N$

- $[\mathbf{x}](t) =$ Interval Taylor Series (ITS)$(t, [\mathbf{x}_j], [\tilde{\mathbf{x}}_j])$
- $\gamma([\mathbf{x}](t)) = 0$

$\Rightarrow \gamma \circ \text{ITS}(t, \mathbf{x}_j, [\tilde{\mathbf{x}}_j]) \rightarrow \psi(t, \mathbf{x}_j)$

To compute $[\underline{t}^\star, \overline{t}^\star] \times [\mathcal{X}_j^\star] \Rightarrow$ Solve CSP [1] $([t_j, t_{j+1}] \times [\mathbf{x}_j], \psi(.,.) = 0)$

How to solve this CSP ?

1. Handbook of Constraint Programming, Rossi et al.,2006

Hybrid Reachability

Ramdani & Nedialkov, 2011

- **Interval Taylor methods**

  ⇒ Analytical expressions for the boundaries of the continuous flows,

  ⇒ Controlling Wrapping effect

- **Interval constraint propagation techniques**

  ⇒ Solve event detection/localization problems

  ⇒ Flow/sets intersection with ALIAS [a] CSP solver.

a. http ://www-sop.inria.fr/coprin/logiciels/ALIAS/

This talk

- Use IBEX [a]
- Test this new interface on benchmarks !

a. http ://www.ibex-lib.org/

Outline

IBEX library (G. Chabert 2007)

IBEX library (G. Chabert 2007)

- Input IBEX = an interval vector XX (box) of dimension $dim(XX)$

  `XX=(XX(1);XX(2);.......;XX(dim(XX))`

IBEX library (G. Chabert 2007)

- Input IBEX $=$ an interval vector XX (box) of dimension $dim(XX)$

  `XX=(XX(1);XX(2);.......;XX(dim(XX))`

- build a symbolic box of dimension $dim(XX)$

```
const Symbol & Xx=env.add_symbol("Xx",dim(XX));
```

IBEX library (G. Chabert 2007)

- Input IBEX $=$ an interval vector XX (box) of dimension $dim(XX)$

  `XX=(XX(1);XX(2);.......;XX(dim(XX))`
- build a symbolic box of dimension $dim(XX)$

  ```
  const Symbol & Xx=env.add_symbol("Xx",dim(XX));
  ```
- Initialize rechearch domain of each variable

  ```
  space.box(1)=time;
  for(int ii=2;ii<=dim(XX);ii++)
  space.box(ii)=XX(ii);
  ```

IBEX library (G. Chabert 2007)

- Input IBEX $=$ an interval vector XX (box) of dimension $dim(XX)$

```
XX=(XX(1);XX(2);.......;XX(dim(XX))
```

- build a symbolic box of dimension $dim(XX)$

```
const Symbol & Xx=env.add_symbol("Xx",dim(XX));
```

- Initialize rechearch domain of each variable

```
space.box(1)=time;
for(int ii=2;ii<=dim(XX);ii++)
space.box(ii)=XX(ii);
```

- Solve CSP (Invariant and guard function) according to current location..

IBEX library (G. Chabert 2007)

```
 switch(mode)
  {

index_contraint[1]=env_.add_ctr(G(Xx[1],Xx[2],....,Xx[n])=0);

       index_contraint[2]=env.add_ctr(G(Xx[1],Xx[2],....,Xx[n])=0);



       index_contraint[m]=env.add_ctr(G(Xx[1],Xx[2],....,Xx[n])=0);


  vector<const Constraint*> vec_constraint;

      vec_constraint.push_back(&env.constraint(index_contraint[1....m])); vector of Constraint


    CSP csp(vec_constraint,space); Create a system of constraints (list of constraints)
    HC4 hc(csp); propagation with a system of constraints

      RoundRobin rr(csp.space, seuiB); Create a bisector with round-robin heuristic
```

Paver paver(hc,rr, seuiP) ; a classical branch & bound algorithm

```
      paver.explore();start research potential solutions
      paver.report(); report all solutions find after exploration
  }
```

IBEX library (G. Chabert 2007)

The output of Ibex :

Each solution of CSP is given by paver.box(i, j)(i=number of contractor[2], j[3]=jth box solution) which is an interval vector

2. i=1
3. j=1...Nsol

IBEX library (G. Chabert 2007)

- For guard solving, time $=[t_j, t_{j+1}] \to [\underline{t}^\star, \overline{t}^\star] \times [\mathcal{X}_j^\star]$

IBEX library (G. Chabert 2007)

- For guard solving, time $=[t_j, t_{j+1}] \to [\underline{t}^\star, \overline{t}^\star] \times [\mathcal{X}_j^\star]$
- For invariant solving, time $=[t_j] \to [\mathcal{X}_j^{inv}]$

Outline

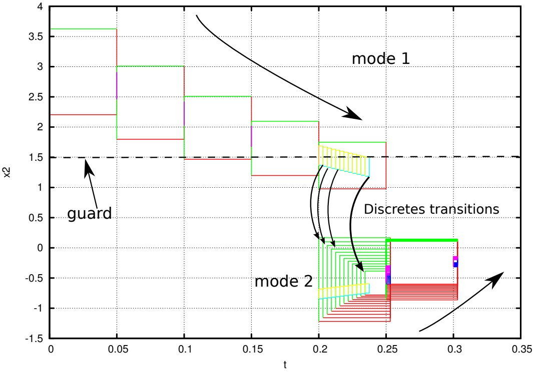
Outline

Example

$q = 1, 2$ $e = 1 \to 2$:

$$\left\{\begin{array}{rl} \text{flow}(1): & f_1(x_1, x_2) = (x_2, -px_2 - g\sin(x_1)) \\ \text{inv}(1): & \nu_1(x_1, x_2) = x_2 - 1.5 \\ \text{flow}(2): & f_2(x_1, x_2) = (x_2, -3px_2 - g\sin(x_1)) \\ \text{inv}(2): & \nu_2(x_1, x_2) = -\nu_1(x_1, x_2) \\ \text{guard}(1): & \gamma_1(x_1, x_2) = \nu_1(x_1, x_2) \\ \text{reset}(1): & \rho_1(x_1, x_2) = (\alpha_1 x_1, \alpha_2 x_2) \end{array}\right.$$

avec $\alpha_1 = -1$, $\alpha_2 \in [-2.05, -2]$, $g = 10$, $p \in [6, 6.3]$ et $x_0 \in [-0.9, -0.8] \times [3, 3.5]$.



Small comparison about CPU times

$$\left\{\begin{array}{rl} \text{flow}(1): & f_1(x_1,x_2) = (x_2, -px_2 - g\sin(x_1)) \\ \text{inv}(1): & \nu_1(x_1,x_2) = \cos(x_1) - x_2/10 - 0.7 \\ \text{flow}(2): & f_2(x_1,x_2) = (x_2, -3px_2 - g\sin(x_1)) \\ \text{inv}(2): & \nu_2(x_1,x_2) = -\nu_1(x_1,x_2) \\ \text{guard}(1): & \gamma_1(x_1,x_2) = \nu_1(x_1,x_2) \\ \text{reset}(1): & \rho_1(x_1,x_2) = (\alpha_1 x_1, \alpha_2 x_2) \end{array}\right. \tag{1}$$

with $\alpha_1 = -1$, $\alpha_2 \in [-2.05, -2]$, $g = 10$, $p \in [6, 6.3]$ and $x_0 \in [-0.9, -0.8] \times [3, 3.5]$.

ALIAS [4]	26 s
IBEX [5]	0.204 s

4. PIV 2GHz
5. Core i5 2.4GHz

Outline

Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)

Bergman minimal model : (G, I, X)

$$\begin{aligned} \frac{dG}{dt} &= -p_1 G - X(G + G_B) + g(t) \\ \frac{dX}{dt} &= -p_2 X + p_3 I \\ \frac{dI}{dt} &= -n(I + I_b) + \frac{1}{V_I} i(t) \end{aligned}$$

initial conditions :

$$G(0) \in [-2, 2] \quad X(0) = 0 \quad I(0) = \in [-0.1, 0.1]$$

$p_1 = 0.01, p_2 = 0.025, p_3 = 1.3.10{-}5, V_I = 12, n = 0.093, G_B = 4.5, I_b = 15.$

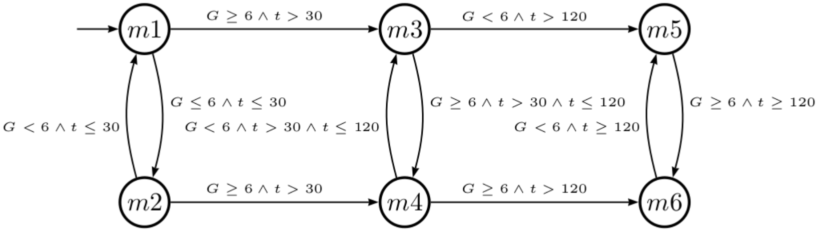
the goal of this benchmark[6] is to compute the reacheable set over the time horizon $t \in [0, 360]$

6. Bench proposed by Xin Chen and Sriram Sankaranarayanan

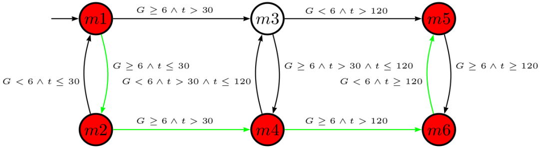
Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)

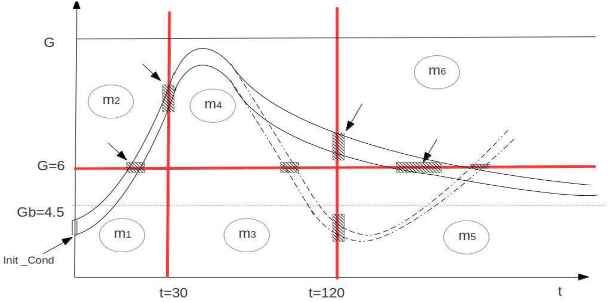
Model 1

$$i(t) = \begin{cases} 1 + \frac{2G(t)}{9} & G(t) < 6 \\ \frac{50}{3} & G(t) \geq 6 \end{cases} \qquad g(t) = \begin{cases} \frac{t}{60} & t \leq 30 \\ \frac{120-t}{180} & t \in [30, 120] \\ 0 & t \geq 120 \end{cases}$$

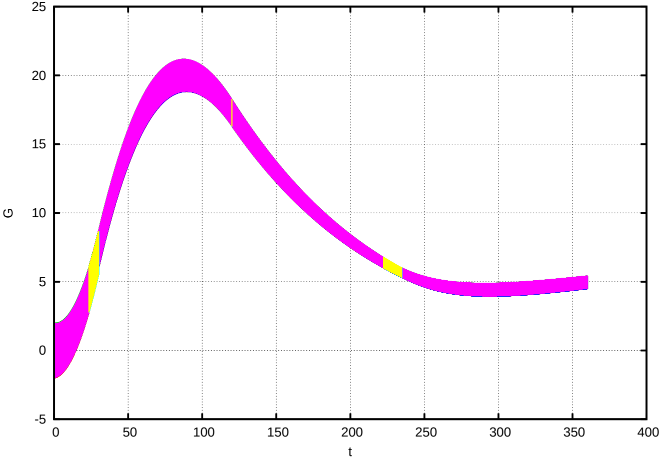


Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



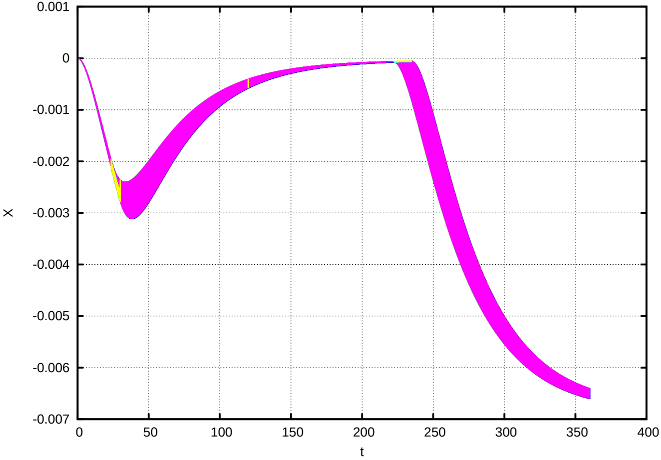


Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



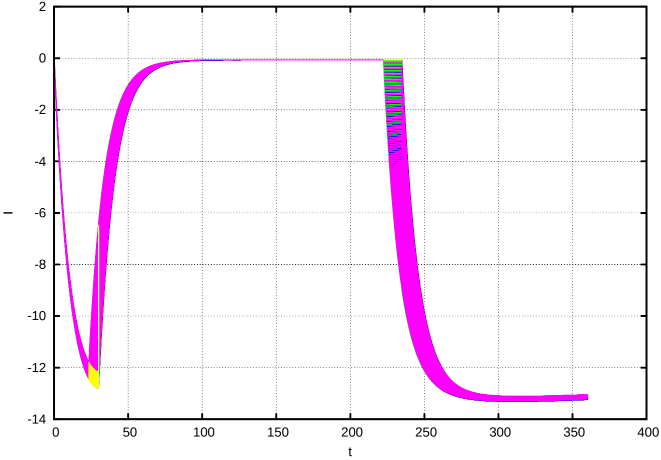
CPU time = 1m11.500s core i5, 64 bits

Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



CPU time = 1m11.500s core i5, 64 bits

Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



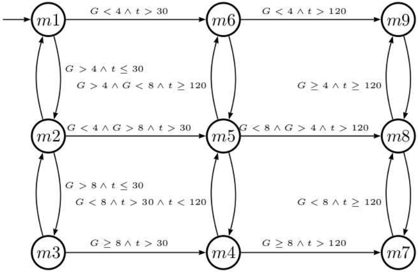
CPU time = 1m11.500s core i5, 64 bits

Outline

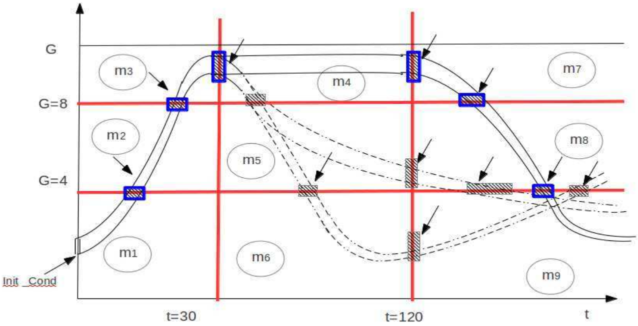
Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)

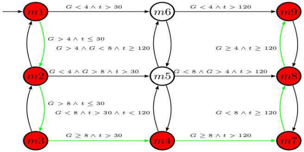
Model 2

$$i(t) = \begin{cases} \frac{25}{3} & G(t) \leq 4 \\ \frac{25(G(t)-3)}{3} & G(t) \in [4, 8] \\ \frac{125}{3} & G(t) \geq 8 \end{cases} \qquad g(t) = \begin{cases} \frac{t}{60} & t \leq 30 \\ \frac{120-t}{180} & t \in [30, 120] \\ 0 & t \geq 120 \end{cases}$$

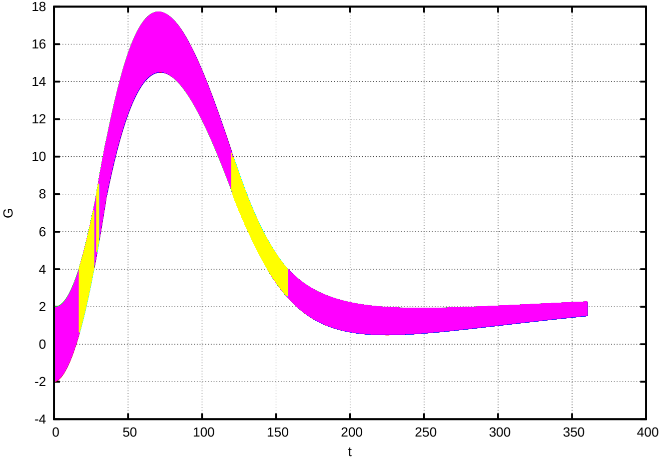


Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



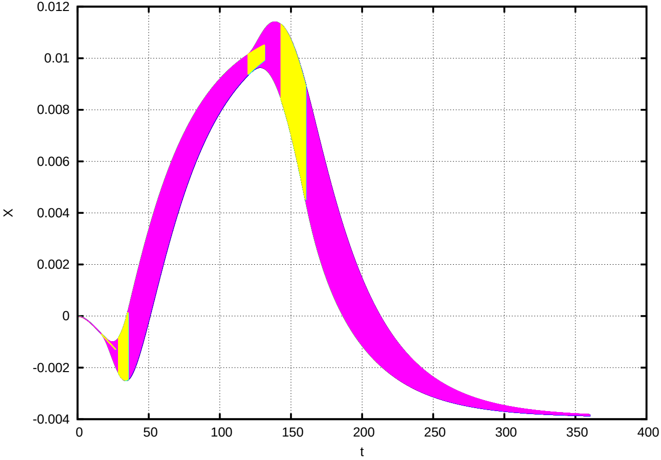


Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



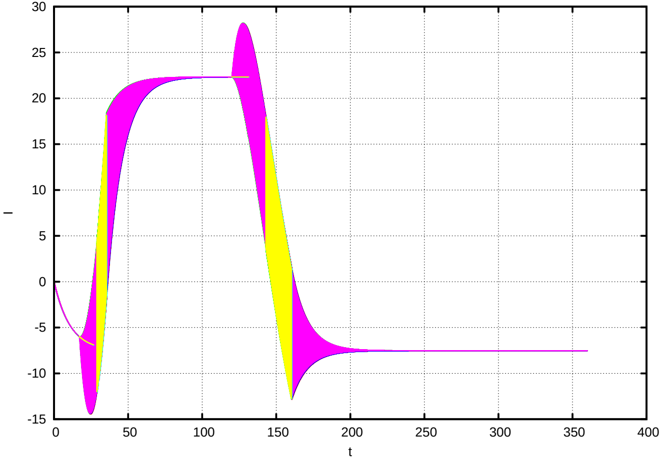
CPU time = 4m15.652s core i5, 64 bits

Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



CPU time = 4m15.652s core i5, 64 bits

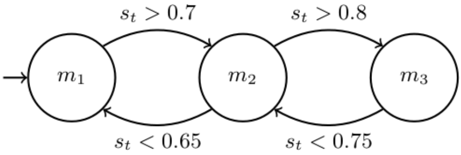
Evaluation on Benchmarks : Glycemic Control in Diabetic Patients (Type 1 diabetes)



CPU time = 4m15.652s core i5, 64 bits

Outline

Evaluation on Benchmarks : Vehicle Model



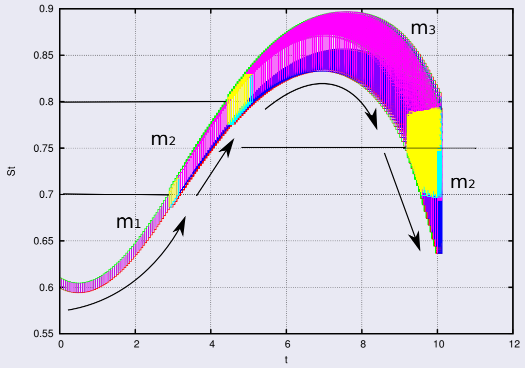
$$\frac{dx}{dt} = vc_t; \frac{dy}{dt} = vs_t; \frac{dv}{dt} = u_1$$
$$\frac{dc_t}{dt} = \sigma v^2 s_t; \frac{ds_t}{dt} = -\sigma v^2 c_t; \frac{d\sigma}{dt} = u_2$$

$$x \in [1, 1.2] \quad y \in [1, 1.2] \quad v \in [0.8, 0.81]$$
$$s_t \in [0.6, 0.61] \quad c_t \in [0.7, 0.71] \quad \sigma = [0, 0.05]$$

the goal of this benchmark[7] *is to compute the reacheable set over the time horizon* $t \in [0, 10]$

7. Bench proposed by Xin Chen and Sriram Sankaranarayanan

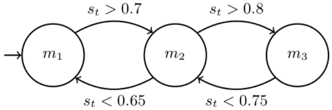
Evaluation on Benchmarks : Vehicle Model



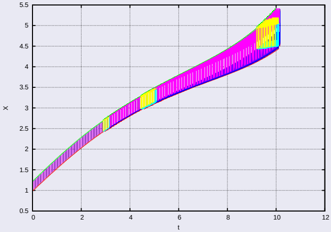
CPU time = 24.678 s core i5, 64 bits

$x \in [1, 1.2] \quad y \in [1, 1.2] \quad v \in [0.8, 0.81]$

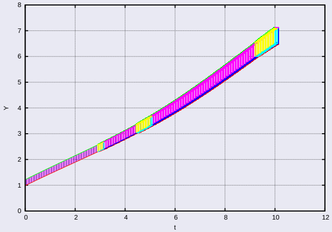
$s_t \in [0.6, 0.61] \quad c_t \in [0.7, 0.71] \quad \sigma = 0.05$



Evaluation on Benchmarks : Vehicle Model



(a) X

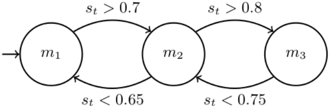


(b) Y

CPU time = 16.885 s core i5, 64 bits

$$\frac{dx}{dt} = vc_t; \frac{dy}{dt} = vs_t; \frac{dv}{dt} = u_1$$

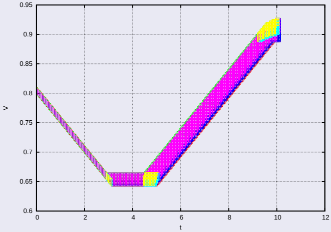
$$\frac{dc_t}{dt} = \sigma v^2 s_t; \frac{ds_t}{dt} = -\sigma v^2 c_t; \frac{d\sigma}{dt} = u_2$$



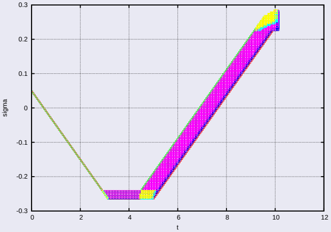
$x \in [1, 1.2] \quad y \in [1, 1.2] \quad v \in [0.8, 0.81]$

$s_t \in [0.6, 0.61] \quad c_t \in [0.7, 0.71] \quad \sigma = 0.05$

Evaluation on Benchmarks : Vehicle Model



(c) V

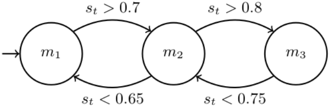


(d) Sigma

CPU time = 24.678 s core i5, 64 bits

$$\frac{dx}{dt} = vc_t; \frac{dy}{dt} = vs_t; \frac{dv}{dt} = u_1$$

$$\frac{dc_t}{dt} = \sigma v^2 s_t; \frac{ds_t}{dt} = -\sigma v^2 c_t; \frac{d\sigma}{dt} = u_2$$



$$x \in [1, 1.2] \quad y \in [1, 1.2] \quad v \in [0.8, 0.81]$$

$$s_t \in [0.6, 0.61] \quad c_t \in [0.7, 0.71] \quad \sigma \in [0.05, 0.05]$$

Evaluation on Benchmarks : CPU times

Bench	Ss	NM	NT	NET	CPU times
G M 1	3	6	10	4	1m11.500s
G M 2	3	9	16	6	4m15.652s
Vehicle model	6	3	4	3	0m24.678s

- Ss = Size of system (continuous state vector dimension).
- NM= Number of Modes.
- NT= Number of Transitions.
- NET= Number of Enabled Transitions (with initials conditions given).

Outline

Concluding remarks

Conclusion

- Analytical expression for the continuous flows
- Interval constraint programming for solving flow/guards intersection
- CSP solving IBEX

Concluding remarks

Conclusion

- Analytical expression for the continuous flows
- Interval constraint programming for solving flow/guards intersection
- CSP solving IBEX

$\rightarrow$ Controlling the number of the box in the list

Concluding remarks

Conclusion

- Analytical expression for the continuous flows
- Interval constraint programming for solving flow/guards intersection
- CSP solving IBEX

$\rightarrow$ Controlling the number of the box in the list

$\rightarrow$ Merging boxes without over-approximation (A Rauh, et al., 2006) (Benazera and Louise, 2009)

Concluding remarks

Conclusion

- Analytical expression for the continuous flows
- Interval constraint programming for solving flow/guards intersection
- CSP solving IBEX

→ Controlling the number of the box in the list

→ Merging boxes without over-approximation (A Rauh, et al., 2006) (Benazera and Louise, 2009)

→ Use VNODE-LP (N. Nedialkov, 2010)