Control Software Synthesis for Discrete Time Linear Hybrid Systems

Prof. Enrico Tronci
Computer Science Department

Ph.D. Programme "Automatica, Bioengineering and Operations Research"
Course on "Hybrid systems: Computation and Control"

September 12–22, 2018
Department of Computer, Control, and Management Engineering
Sapienza University of Rome (Italy)
• Motivations and model-based design
• Discrete Time Linear Hybrid Systems (DTLHS)
• Control software synthesis problem
• QKS
• Results on Buck DC-DC converter
Motivations

- Cyber-Physical Systems are everywhere
- Safety- and Mission-critical
- Formal methods
  - Improve design quality
  - Decrease design costs

Embedded Systems

- 28 April 2016
  Japanese Hitomi spacecraft

- 20 August 2008
  Spanair Flight JK 5022

Mission-Critical
US$286 million lost

Safety-Critical
154 people died
every T seconds do {  // T controller sampling time
  x = AD(read(plant_state))
  if (!controllable_reg(x))  // Fault detection
    then Exception: Start fault isolation and recovery (FDIR)
  else // Nominal case: compute command u to send to actuator
    u = ctrl_law(x)
    write(DA(u))
}
Model-Based Design

Control Software Synthesis for DTLHS
Model-Based Design

- Plant Model
- Closed loop system specification
- Control Engineering + Software Engineering Tool
- Correct-By-Construction Control Software
  \[ WCET \leq \text{Sampling Time} \]
Modelling CPS with Hybrid Systems

- **Continuous** dynamics (time evolution)
- **Discrete** modes (control software)

Linear Hybrid Systems

- Timed automata [Alur&Dill, 94]
  \[ t = 1 \]
- Rectangular automata [Puri&Varaiya, 94]
  \[ t \in [t_1, t_2] \]
- Linear hybrid automata [Alur&al, 96]
- Piecewise affine hybrid systems [Bemporad&al, 00]
- Discrete time linear hybrid systems [Mari&Tronci, 07]

Reachability Undecidable [Henzinger&al, 98]
The problem (high level)

Find a control software implementation s.t. starting from any point in the controllable region, the closed loop system evolutions reach the specified goal region.
Hybrid Systems as DAEs

Example: The Buck DC/DC Converter

\[ \dot{i}_L = a_{1,1}i_L + a_{1,2}v_O + a_{1,3}v_D \]
\[ \dot{v}_O = a_{2,1}i_L + a_{2,2}v_O + a_{2,3}v_D \]

Discretisation step \( T \) (e.g., Euler)

\[ i'_L = (1 + Ta_{1,1})i_L + Ta_{1,2}v_O + Ta_{1,3}v_D \]
\[ v'_O = Ta_{2,1}i_L + (1 + Ta_{2,2})v_O + Ta_{2,3}v_D \]

Discrete time

\[ \bar{u} \rightarrow v_u = R_{off}i_u \]
\[ v_D = v_u - V_i \]

Continuous time

Algebraic

\[ q \rightarrow v_D = 0 \]
\[ q \rightarrow i_D \geq 0 \]
\[ u \rightarrow v_u = 0 \]
Discrete Time Linear Hybrid System

\[ \mathcal{H} = (X, U, Y, N) \]

Present state variables (\(X'\) are next state)

Input variables (controllable inputs)

Auxiliary variables (uncontrollable inputs)

Boolean combination of linear inequalities over 
\(X \cup U \cup Y \cup X'\)

Transition relation (next state)

Example: The Buck DC/DC Converter

\[ X = [i_L, v_O] \text{ real valued} \]
\[ X' = [i_L', v_O'] \text{ real valued} \]
\[ U = [u] \text{ boolean} \]
\[ Y = [i_u, v_u, i_D, v_D] \cup [q] \text{ real valued and bool} \]
\[ N(X, U, Y, X') \text{ defined by conj. of lin. inequal.} \]

\[ T = 10E-6 \]

\[ i_L' = (1 + Ta_{1,1})i_L + Ta_{1,2}v_O + Ta_{1,3}v_D \]
\[ v_O' = Ta_{2,1}i_L + (1 + Ta_{2,2})v_O + Ta_{2,3}v_D \]

\[ q \to v_D = 0 \]
\[ q \to i_D \geq 0 \]
\[ u \to v_u = 0 \]

F. Mari, I. Melatti, I. Salvo, and E. Tronci
Model Based Synthesis of Control Software from System Level Formal Specifications

Control Software Synthesis for DTLHS
Quantized Feedback Control

- Feedback control problem for LTSs
- Feedback control problem for DTLHSs
- Quantized feedback control problem
Feedback Control Problem for LTSs

**Labelled Transition System (LTS)**

- **States**: $S = \{-1, 0, 1\}
- **Actions**: $A = \{0, 1\}
- **Transition Relation**: $T : S \times A \rightarrow \mathbb{B}$

**Controller for LTS**

- **Controller Function**: $K : S \times A \rightarrow \mathbb{B}$
- **Admissible Actions**: $orall s \in S \forall a \in A \text{ if } K(s, a) \text{ then } a \in \text{Adm}(S, s)$
- **Domain of Controller**: $\text{Dom}(K) = \{s \in S | \exists a K(s, a)\} \quad \text{Dom}(K) = \{-1, 0, 1\}$
Feedback Control Problem for LTSs

Closed loop system

LTS \((S, A, T^{(K)})\)

\[ T^{(K)}(s, a, s') = T(s, a, s') \land K(s, a) \]

LTS control problem

\((S, I, G)\)

\[ I = \{-1, 0, 1\} \subseteq S \]
\[ G = \{0\} \subseteq S \]
Feedback Control Problem for LTSs

Distance from goal $J$

Path $\pi_1$

$J(S, G, \pi_1) = +\infty$

Path $\pi_2$

$J(S, G, \pi_2) = \min\{n \mid n > 0 \land \pi_2^S(n) \in G\} = 1$

Worst case distance:

$J_{\text{strong}}(S, G, s) = \sup_{a \in \text{Adm}(S, s)} \sup_{\pi \text{ starting with } a} J(S, G, \pi)$

Best case distance:

$J_{\text{weak}}(S, G, s) = \sup_{a \in \text{Adm}(S, s)} \inf_{\pi \text{ starting with } a} J(S, G, \pi)$
Solution to LTS control problem

Let

\[ \mathcal{P} = (\mathcal{S}, I, G) \] be an LTS control problem

\( K \) a controller for \( S \) s.t. \( I \subseteq \text{Dom}(K) \)

Then

\( K \) is a **strong solution** to \( P \) iff

\[ \forall s \in \text{Dom}(K) \quad J_{\text{strong}}(S^{(K)}, G, s) \text{ is finite} \]

\( K \) is a **weak solution** to \( P \) iff

\[ \forall s \in \text{Dom}(K) \quad J_{\text{weak}}(S^{(K)}, G, s) \text{ is finite} \]
Feedback Control Problem for DTLHSs

Dynamics of DTLHS $\mathcal{H}$

\[
LTS(\mathcal{H}) = (\mathcal{D}_X, \mathcal{D}_U, \tilde{N})
\]

\[
\tilde{N} : \mathcal{D}_X \times \mathcal{D}_U \times \mathcal{D}_X \to \mathbb{B}
\]

subject to

\[
\tilde{N}(x, u, x') = \exists y \in \mathcal{D}_Y \ N(x, u, y, x')
\]

DTLHS control problem and solution

\[
(\mathcal{H}, I, G) \rightarrow \text{LTS control problem } (LTS(\mathcal{H}), I, G)
\]

A controller $K : \mathcal{D}_X \times \mathcal{D}_U \to \mathbb{B}$ is strong (weak) solution to $(\mathcal{H}, I, G)$ iff it is a strong (weak) solution to $(LTS(\mathcal{H}), I, G)$
Example: The Buck DC/DC Converter

Goal
keeping output voltage close
enough to given reference value, e.g. 5

\((\mathcal{H}, I, G)\) control problem with

\[ I \equiv (|i_L| \leq 2) \land (0 \leq v_O \leq 6.5) \]

\[ G \equiv (|v_O - 5| \leq 0.01) \land (|i_L| \leq 2) \]
Quantized Feedback Control Problem

DTLHS

\[ \mathcal{H} = (\{x\}, \{0, 1\}, \emptyset, N) \]

\[ N(x, u, y, x') = \{ \bar{u} \rightarrow x' = 0.5x, u \rightarrow x' = 1.5x \} \]

\[ x \in [-3.5, 3.5] \]

AD Quantization

FSM as seen from the control software
Quantized feedback control problem for DTLHSs

Input
- DTLHS control problem \((\mathcal{H}, I, G)\)
- Quantization function due to \textit{AD conversion} \(\Gamma\)

Output
- Quantized control function \(\hat{K} : \Gamma(\mathcal{D}_x) \times \Gamma(\mathcal{D}_u) \rightarrow \mathbb{B}\)
- Control function \(K(x, a) = \hat{K}(\Gamma(x), \Gamma(a))\)
QKS

Quantized Controller Software Generator

DTLHS
Desired Controllable Region
Goal Region
AD/DA n. of bits

SOL
Control Software
Actual Controllable Region
Robustness by Construction
Guaranteed WCET

Control Software
Microcontroller

D/A
A/D

Plant (physical system)

Observable state of plant

Action

Stems from undecidability of DTLHS control problem
[ICTAC 12]

Idea: Halting problem for two counter machine reduced to DTLHS control problem

mclab qks
QKS Workflow

1. **Step 1: Control Abstraction Computation**
   - Input: Implementation Specification (Quantization Schema)
   - Plant Model (DTLHS)
   - System Level Formal Specification (Liveness and Safety)

2. **Step 2: Symbolic Strong Controller Synthesis**
   - Finite LTS Control Problem
   - Most General Optimal Controller

3. **Step 3: C Code Generation from OBDD**
   - Control Software
Control Abstraction: Idea

\[ H = (\{x\}, \{0, 1\}, \emptyset, \mathbb{N}) \quad N(x, u, y, x') = \{ \bar{u} \rightarrow x' = 0.5x, u \rightarrow x' = 1.5x \} \]

AD
Quantization
\[ x \in [-3.5, 3.5] \]

FSM M for H modelling quantization

MaxCtrAbs
Control Abstraction: Idea

\( \mathcal{H} = \{x\}, \{0, 1\}, \emptyset, N \)  
\( N(x, u, y, x') = \{ u \rightarrow x' = 0.5x, u \rightarrow x' = 1.5x \} \)

**Quantization**

\( x \in [-3.5, 3.5] \)

**Control abstraction step 1:** Restrict to safe transitions
Control Abstraction: Idea

\[ H = \{x\}, \{0, 1\}, \emptyset, N \]  
\[ N(x, u, y, x') = \{ \bar{u} \rightarrow x' = 0.5x, u \rightarrow x' = 1.5x \} \]

**AD Quantization**

\[ x \in [-3.5, 3.5] \]

Control abstraction step 2: Remove self-loop if it eventually disappears.
Main Theorem

Control Problem (H, I, G) + Quantization AD

- Compute AD MaxCtAbs W (FSM)
  - Check if from each quantized initial state s in AD(I) a quantized goal state in AD(G) is reachable in W (Weak Controller).
  - Use [Tronci - ICFEM98].
  - No: No Solution
  - Yes: Unknown

- Compute AD MinCtAbs P (FSM)
  - Check if there exists a restriction (Strong Controller) K of P such that in KP any path from a state in AD(I) (quantized initial state) reaches a state in AD(G) (quantized goal state) within a finite number of steps.
  - Use [Cimatti - AIPS98].
  - No: Unknown
  - Yes: Solution: \( \lambda x u. K(AD(x), AD(u)) \)
Strategy

Compute MinCtrAbs.
This yields a **sufficient condition** for existence of a solution to \((H, I, G)\).
Unfortunately computing the Minimum Control Abstraction is undecidable
(since it entails solving a reachability problem on linear hybrid systems).
We look for a **small enough** (i.e., **deterministic enough**) control abstraction.

Compute MaxCtrAbs.
This yields a **necessary condition** for existence of a solution to \((H, I, G)\).
Computing the Maximum Control Abstraction is decidable and easier than
Computing MinCtrAbs. Thus, in the following we focus on computing MinCtrAbs.
Main Alg: Computing Control Abstraction

**Input:** A quantization AD, a DTLHS $H = (X, U, Y, N)$, a control problem $(H, I, G)$.

**Output:** OBDDs for: $N$ (transition relation of MinCtrAbs), $I$ (quantization of I), $G$ (quantization of G)

```plaintext
minCtrAbs(AD, H, I, G) {
1: X = [x_1, ..., x_n]; X' = [x_1', ..., x_n']; U = [u_1, ..., u_r]; N(X, U, Y, X') = 0; I(X) = 0; G(X) = 0;
2: forall (s in AD(STATE)) do { // MILP (… X …) is feasible = EXISTS X ( … )
3:     if (s is the quantization of an initial state) I = I ∪ {s}; // MILP1 add initial state
4:     if (s is the quantization of a goal state) G = G ∪ {s}; // add goal state
5:     forall (u in AD(CTR)) do {
6:         if (action u from a state X with quantization s may lead to an unsafe state) // MILP5 skip unsafe
7:             {continue;}
8:         if (action u from a state X with quantization s may lead to a selfloop) // MILP2 add selfloop
9:             {N = N ∪ {(s, u, s)};
10:        forall (i = 1, ..., n) do {
11:            m_i = min value for u-successor of x_i ; // MILP3 min reach x
12:            M_i = max value for u-successor of x_i ; } // max reach x
13:        Over_Img(s, u) = \prod_{i=1}^{n} [AD(m_i), AD(M_i)]; // Overapprox of 1-step reachable x
14:        forall (s' in Over_Img(s, u)) do {
15:            if (s != s' and s' is a (quantized) u-successor of s) // MILP4
16:                N = N ∪ {(s, u, s'); } // add transition (s, u, s')
17:        } } // end exploration
18: return (N, I, G); }
```

Control Software Synthesis for DTLHS
Main Alg: MILPs

Discrete state $s$ is the quantization of an initial state =
$$\exists X \ [I(X) \land AD(X) \implies s] = \text{MILP}(I(X) \land AD(X) \implies s) \text{ is feasible} = \text{MILP1}$$

Discrete state $s$ is the quantization of a goal state =
$$\exists X \ [G(X) \land AD(X) \implies s] = \text{MILP}(I(X) \land AD(X) \implies s) \text{ is feasible}$$

Action $u$ from a state $X$ with quantization $s$ may lead to an unsafe state =
$$\text{MILP}(N(X, U, Y, X') \land AD(X) = s \land AD(U) = u \land X' \text{ is not in STATE}) \text{ is feasible} = \text{MILP5}$$

Action $u$ from a state $X$ with quantization $s$ may lead to a selfloop =
$$\text{SelfLoop}(s, u) = \text{MILP2}$$

$$m_i = \min \text{ value for } u\text{-successor of } x_i =$$
$$m_i = x_i^{''}, \text{ where } X^{''} = [x_1^{''}, ..., x_n^{''}] \text{ is a solution to the MILP}$$
$$(\min, x_i', N(X, U, Y, X') \land AD(X) = s \land AD(U) = u) = \text{MILP3}$$

$$M_i = \max \text{ value for } u\text{-successor of } x_i =$$
$$M_i = x_i^{''}, \text{ where } X^{''} = [x_1^{''}, ..., x_n^{''}] \text{ is a solution to the MILP}$$
$$(\max, x_i', N(X, U, Y, X') \land AD(X) = s \land AD(U) = u)$$

Discrete state $s'$ is a (quantized) $u$-successor of $s =$
$$\text{MILP} \ (N(X, U, Y, X') \land AD(X) = s \land AD(U) = u \land AD(X') = s') \text{ is feasible} = \text{MILP4}$$
Main Alg: Checking Self-loops

SelfLoop(s, u) {

For each real valued state component $x_i$, do  // check gradient of $x_i$
    let $w_i$ be the min elongation of $x_i$, that is the solution to
    MILP(min, $x_i' - x_i$, $N(X, U, Y, X') \land AD(X) = s \land AD(U) = u$);
    let $W_i$ be the max elongation of $x_i$, that is the solution to
    MILP(max, $x_i' - x_i$, $N(X, U, Y, X') \land AD(X) = s \land AD(U) = u$);

If for some $i [(w_i \neq 0) \land (W_i \neq 0) \land (w_i$ and $W_i$ have the same sign)]
then return 0  // any long enough sequence of $u$ actions will drive
    // state component $x_i$ outside of $AD^{-1}(s)$
else return 1  // unable to show that self loop can be eliminated
}
QKS Workflow

**Step 1:** Control Abstraction Computation

- Input
  - Implementation Specification (Quantization Schema)
  - Plant Model (DTLHS)
  - System Level Formal Specification (Liveness and Safety)

**Step 2:** Symbolic Strong Controller Synthesis

**Step 3:** C Code Generation from OBDD

Control Software Synthesis for DTLHS
From OBDDs \((N, I, G)\) we compute symbolically an OBDD \(K(x, u)\) for the Strong Controller using the algorithm in [Cimatti – AIPS98].
QKS Workflow

1. **Step 1: Control Abstraction Computation**
   - Input: Implementation Specification (Quantization Schema)
   - Plant Model (DTLHS): System Level Formal Specification (Liveness and Safety)

2. **Step 2: Symbolic Strong Controller Synthesis**
   - Finite LTS Control Problem
   - Most General Optimal Controller

3. **Step 3: C Code Generation from OBDD**
   - Control Software

Control Software Synthesis for DTLHS
Using the algorithm in [Tronci – ICFEM98] From K we generate a C implementation F(x) for K(x, u) s.t. K(x, F(x)) holds for any state x in the controllable region.
Glimpse on Control SW Generation

Ctr Software generated from OBDD for K.

```
char obdd_in_C(char *x) {
    char return_bit = 1;
    L_0x53: if (x[1] == 1) goto L_0x4f;
             else {return_bit = !return_bit; goto L_0x52;}
    L_0x52: if (x[2] == 1) goto L_0x4b; else goto L_1;
    L_0x4f: if (x[2] == 1) goto L_1; else goto L_0x4b;
    L_0x4b: if (x[3] == 1) goto L_1; else goto L_0;
    L_1: return return_bit;
    L_0: return_bit = !return_bit; return return_bit;
}
```

Software Implementation of Ctr K

WCET = A * <size of longest path in K OBDD> * <number of U bits>, where:
A = Time to compute an if-then-else and a goto. Thus: WCET <= A*X_BITS*U_BITS
Results: Setting

Parameters:  
\[ T = 10^{-6}\text{s}, \]
\[ L = 2 \times 10^{-4}\text{H}, \quad r_L = 0.1 \text{ Ohm}, \]
\[ C = 5 \times 10^{-5}\text{F}, \quad r_C = 0.1 \text{ Ohm}, \]
\[ R = 5 \pm 25\% \text{ Ohm}, \quad V_i = 15 \pm 25\% \text{ V}, \]
\[ V_{\text{ref}} = 5 \text{ V}, \quad p = 0.01\text{V} \text{ (converter precision)} \]

Safety Bounds:  
\[ |i_L| \leq 4, \quad -1 \leq v_O \leq 7, \quad |i_u| \leq 10^3, \quad |i_D| \leq 10^3, \quad |v_u| \leq 10^7, \quad |v_D| \leq 10^7. \]

\[ I = \{(i_L, v_O) \mid |i_L| \leq 2, \quad 0 \leq v_O \leq 6.5\}, \quad G = \{(i_L, v_O) \mid |i_L| \leq 2, \quad |v_O - V_{\text{ref}}| \leq p\} \]
Results: Ctrl Abs + Ctrl SW

Table shows CPU Time (s) needed to compute a near-optimal control law and its C implementation (K). All computations run within 200MB RAM.

Main Alg returns UNK for b=8, SOL for all other cases. Thus we know, on a formal ground, that for b=10 our synthesized controller works correctly on the desired set of initial states.

**Arcs:** Arcs in MinCtrAbs (our close to minimum Control Abstraction)

**MaxLoops:** Loops in MaxCtrAbs.

**LoopFrac:** Fraction of self loops in MaxCtrAbs that is also in MinCtrAbs.

Experiments on an Intel 3.0 Ghz Dual Quad Core Linux Pc with 4GB of RAM

<table>
<thead>
<tr>
<th>b</th>
<th>CPU (s)</th>
<th>Arcs</th>
<th>MaxLoops</th>
<th>LoopFrac</th>
<th>Controller Synthesis</th>
<th>Total CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2.50e+03</td>
<td>1.35e+06</td>
<td>2.54e+04</td>
<td>0.00323</td>
<td>0.00e+00</td>
<td>1.07e+02</td>
</tr>
<tr>
<td>9</td>
<td>1.13e+04</td>
<td>7.72e+06</td>
<td>1.87e+04</td>
<td>0.00440</td>
<td>1.00e+02</td>
<td>1.24e+03</td>
</tr>
<tr>
<td>10</td>
<td>6.94e+04</td>
<td>5.14e+07</td>
<td>2.09e+04</td>
<td>0.00781</td>
<td>7.00e+02</td>
<td>2.75e+03</td>
</tr>
<tr>
<td>11</td>
<td>4.08e+05</td>
<td>4.24e+08</td>
<td>2.29e+04</td>
<td>0.01417</td>
<td>5.00e+03</td>
<td>7.00e+03</td>
</tr>
</tbody>
</table>

WCET(b=10) = IF THEN ELSE TIME*STATE_BITS*CTR_BITS = 0.5*10^-7*20*1 = 10^-6
Avg Execution Time (s) for MILP problems in Main Alg. This is quite small since each MILP all MILPs have about the same size (plant model).

Number of calls to MILP problems in Main Alg. MILP4 most called one. This is closer to \( \text{STATE\_BITS}^2 \times \text{CTR\_BITS} \) than to \( \text{STATE\_BITS}^2 \times \text{CTR\_BITS} \). This shows effectiveness of Over Img in main Alg.
Results: Controllable Regions

8 bits

9 bits

10 bits

11 bits

Don't cares offer optimization opportunities
Results: Ctrl SW Performances

Transient – 11 bit quantization
State of the art: ~ 2 ms
Automatic Synthesis: ~ 0.3 ms

Ripple – 11 bit quantization
State of the art: ~ 50 mV
Automatic Synthesis: ~ 4 mV
Conclusions

• QKS: Automatic synthesis of quantized control software for DTLHSs

• Correct-by-construction (e.q., quantization taken into account, no arithmetical overflow, …)

• Known controllable region

• Robust by construction w.r.t. variations in plant parameters

• Guaranteed Worst Case Execution Time (WCET) of control software
Future

- Fully symbolic approaches (e.g., based on Fourier-Motzkin quantifier elimination)
- Methods to decrease WCET
- Statistical model checking approaches to counteract state explosion