Modeling Hybrid Systems Using Constraint Logic Programming

Ammar Mohammed

Department of Computer Science
University of Koblenz-Landau

May 14, 2008
Outline

1. Introduction
   - Hybrid Systems
   - Hybrid Automata

2. CLP
   - Introduction

3. CLP Modeling Approach
   - the Model
   - Controlling behaviors

4. Train Gate Controller Example
   - Model Execution

5. Conclusion
1. Introduction
   - Hybrid Systems
     - Hybrid Automata

2. CLP
   - Introduction

3. CLP Modeling Approach
   - the Model
   - Controlling behaviors

4. Train Gate Controller Example
   - Model Execution

5. Conclusion
Introduction

Hybrid Systems

- Discrete-event + Continuous evolution.
- Interacting components via Sync. + shared variables.

Hybrid Systems: \[\iff\] Constraints Systems describe

- Flows, Invariants, Guards, and synchronization.
- Certain parts of the State space.

Objective

to Model concurrent Hybrid Systems components using CLP
Hybrid Systems

- Discrete-event + Continuous evolution.
- Interacting components via Sync. + shared variables.

**Hybrid Systems:** Constraints Systems describe
- Flows, Invariants, Guards, and synchronization.
- Certain parts of the State space.

**Objective**

to Model concurrent Hybrid Systems components using CLP
1. Introduction
   - Hybrid Systems
   - Hybrid Automata

2. CLP
   - Introduction

3. CLP Modeling Approach
   - the Model
   - Controlling behaviors

4. Train Gate Controller Example
   - Model Execution

5. Conclusion
Hybrid Automata

Formal Definition

A hybrid automaton \( H = (X, Q, \text{Inv}_q, \phi_q, E, \text{Jump}, \text{Event}, \text{Init}) \) where:

- \( X \subseteq \mathbb{R}^n \)
- \( Q \) is a finite set of control locations.
- \( \text{Inv}_q \) a constraint predicate assigns a constraint on \( X \).
- \( \phi_q \) flow (continuous activity) predicate on variables \( X \).
- \( E \) discrete transition is augmented by:
  - \( \text{Jump} \) is a jump condition (guard action)
  - \( \text{Event} \) is a synchronization labels
- \( \text{Init} \) is the initial condition
Hybrid Automata

State: Changes

- **Discretely** when a transition is enabled
- **Continuously** by means of a finite time delay $t$.

Composition: $H_1$ and $H_2$

- $a$ is an event of both $\implies$ Both must sync on $a$-transitions.
- $a$ is an event only in $H_1$ $\implies$ each $a$-transitions synchronizes with a 0-duration time transition of $H_2$. 
Hybrid Automata

State: Changes
- **Discretely** when a transition is enabled
- **Continuously** by means of a finite time delay $t$.

Composition: $H_1$ and $H_2$
- $a$ is an event of both $\implies$ Both must sync on $a$-transitions.
- $a$ is an event only in $H_1$ $\implies$ each $a$-transitions synchronizes with a 0-duration time transition of $H_2$. 

Ammar M. (University of Koblenz)
Outline

1 Introduction
   - Hybrid Systems
   - Hybrid Automata

2 CLP
   - Introduction

3 CLP Modeling Approach
   - the Model
   - Controlling behaviors

4 Train Gate Controller Example
   - Model Execution

5 Conclusion
Constraint Logic Programming (CLP)

CLP Properties

- Declarative Nature.
- Expressivity.
- Efficiency.

Problems as CLP programs

- Formulate a problem as CSP (modeling)
- Solve the model

CLP(X) Languages: $X \in \{B,R,Q,FD\}$

EClipse-Prolog
Outline

1. Introduction
   - Hybrid Systems
   - Hybrid Automata

2. CLP
   - Introduction

3. CLP Modeling Approach
   - the Model
   - Controlling behaviors

4. Train Gate Controller Example
   - Model Execution

5. Conclusion
**Locations**

- Predicate over real variables Vars, Time and $T_0$, where $\text{Time, } T_0 \in \mathbb{R}^+$. 

  \[
  \text{automaton(Location, Vars, } T_0, \text{ Time)} : -
  \]

  \[
  c(\text{Vars}) \land c(\text{Inv.}) \land c(T_0, \text{Time}).
  \]

**Example**

\[
\text{train(far, X, } T_0, \text{ Time)} : -
\]

\[
X \geq 2000 - 50 \times \text{Time},
X \leq 2000 - 40 \times \text{Time},
\text{Time} \geq T_0, \ X \geq 1000.
\]
Locations

- Predicate over real variables $Vars, Time$ and $T_0$, where $Time, T_0 \in \mathbb{R}^+$.

$$\text{automaton}(\text{Location}, Vars, T_0, Time) :=-\ c(Vars) \land c(\text{Inv.}) \land c(T_0, Time).$$

Example

$$\text{train}(\text{far}, X, T_0, Time) :=-\ X \geq 2000 - 50 \times Time, \ X \leq 2000 - 40 \times Time, \ Time \geq T_0, \ X \geq 1000.$$
CLP Model

State Change

\[
\text{evolve}(\text{Automaton}, L, \text{NextL}, T_0, \text{Delay}, \text{Time}, \text{Event}, T_{new0}) :- \\
\text{discrete}(\text{Automaton}, L, \text{NextL}, T_0, \text{Delay}, \text{Time}, \text{Event}, T_{new0}); \\
\text{continuous}(\text{Automaton}, L, \text{NextL}, T_0, \text{Delay}, \text{Time}, \text{Event}, T_{new0}).
\]
**Discrete**

discrete(train, far, near, T₀, Delay, Time, Event, 0):-
Event \= app, Time =:= T₀+Delay.

**Continuous**

continuous(train, L, L, T₀, Delay, Time, Event, Tₙₑ𝑤₀):-
Event \= app, Event = exit, Event \= in,
Time > T₀+Delay, Tₙₑ𝑤₀ is T₀+Delay.
CLP Model

Discrete

discrete(train, far, near, T₀, Delay, Time, Event, 0):-
Event = app, Time = T₀ + Delay.

Continuous

continuous(train, L, L, T₀, Delay, Time, Event, T_{new₀}) :-
Event \neq app, Event \neq exit, Event \neq in,
Time > T₀ + Delay, T_{new₀} is T₀ + Delay.
Outline

1. Introduction
   - Hybrid Systems
   - Hybrid Automata

2. CLP
   - Introduction

3. CLP Modeling Approach
   - the Model
   - Controlling behaviors

4. Train Gate Controller Example
   - Model Execution

5. Conclusion
driver( (L₁, T₀₁), (L₂, T₀₂), ...,(Lₙ, T₀ₙ)) :-
  automata₁(L₁, Var₁, T₀₁, Time₁),
  automata₂(L₂, Var₂, T₀₂, Time₂),
  ...,
  automata₂(L₂, Var₂, T₀₂, Time₂),
  miniEventTime(Time₁, Time₂, ... ,Timeₙ, Delay),
  evolve(automata₁, L₁, NextL₁, T₀₁, Delay, Time₁, Event, Tₙₑｗ₀₁),
  evolve(automata₂, L₂, NextL₂, T₀₂, Delay, Time₂, Event, Tₙₑｗ₀₂),
  ...,
  evolve(automataₙ, Lₙ, NextLₙ, T₀ₙ, Delay, Timeₙ, Event, Tₙₑｗ₀ₙ),
  Time₁ ≥ T₀₁, Time₁ ≤ T₀₁ + Delay,
  Time₂ ≥ T₀₂, Time₂ ≤ T₀₂ + Delay,
  ... , Timeₙ ≥ T₀ₙ, Timeₙ ≤ T₀ₙ + Delay,
  driver( (NextL₁, Tₙₑｗ₀₁), (NextL₂, Tₙₑｗ₀₂), ...,(NextLₙ, Tₙₑｗ₀ₙ)).
Outline

1. Introduction
   - Hybrid Systems
   - Hybrid Automata

2. CLP
   - Introduction

3. CLP Modeling Approach
   - the Model
   - Controlling behaviors

4. Train Gate Controller Example
   - Model Execution

5. Conclusion
far : open : idle
X{1000.0 .. 2000.0} : 90 : 0
Event :app
Needed Time: 25.0
---------
near : open : to_lower
X{515.0 .. 1000.0} : 90 : Tz{0.0 .. 9.7}
Event :lower
Needed Time: 9.7
---------
near : down : idle
X{14.999999999999 .. 709.0} : G{0.0 .. 90.0} : 0
Event :to_close
Needed Time: 10.0
---------
near : close : idle
X{0.0 .. 409.0} : 0 : 0
Event :in
Needed Time: 13.63333333333
---------
past : close : idle
X{0.0 .. 100.0} : 0 : 0
Event :exit
Needed Time: 3.33333333333
---------
far : close : to_raise
X{1515.0 .. 2000.0} : 0 : Tz{0.0 .. 9.7}
Event :raise
Needed Time: 9.7
Model Execution

far : open : idle
X{1000.0 .. 2000.0} : 90 : 0
Event :app
Needed Time: 25.0

near open : to_lower
X{510.0 .. 1000.0} : 90 : Tz{0.0 .. 9.8}
Event :lower
Needed Time: 9.8

near : down : idle
X{9,999999999999977 .. 706.0} : G{0.0 .. 90.0} : 0
Event :to_close
Needed Time: 10.0

near : close : idle
X{0.0 .. 406.0} : 0 : 0
Event :in
Needed Time: 13.5333333333333

past : close : idle
X{0.0 .. 100.0} : 0 : 0
Event :exit
Needed Time: 3.3333333333333

far : close : to_raise
X{1510.0 .. 2000.0} : 0 : Tz{0.0 .. 9.8}
Event :raise
Needed Time: 9.8

far : up : idle
X{1010.0 .. 1608.0} : G{0.0 .. 90.0} : 0
Conclusion

- CLP model for hybrid systems
- Concurrent execution using constraints over time and events.
- Executable model for analysis