Modeling Hybrid Systems Using Constraint Logic Programming

Ammar Mohammed

Department of Computer Science
University of Koblenz-Landau

May 14, 2008

Outline

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

Hybrid Systems

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

Hybrid Systems: $\Longrightarrow$ Constraints Systems describe

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

Objective

to Model concurrent Hybrid Systems components using CLP

Introduction

Hybrid Systems

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

Hybrid Systems: $\Longrightarrow$ Constraints Systems describe

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

Objective

to Model concurrent Hybrid Systems components using CLP

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

Hybrid Automata

Formal Definition

A hybrid automaton H=(X,Q,Inv_q,ϕ_q,E,$Jump$,$Event$,$Init$) where:

- $X \subseteq R^n$
- Q is a finite set of control locations.
- Inv_q a constraint predicate assigns a constraint on X.
- ϕ_q flow (continuous activity) predicate on variables X.
- E discrete transition is augmented by:
  - *Jump* is a jump condition(guard action)
  - *Event* is a synchronization labels
- *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 $\Longrightarrow$ Both must sync on a-transitions.
- a is an event only in H_1 $\Longrightarrow$ 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 $\Longrightarrow$ Both must sync on a-transitions.
- a is an event only in H_1 $\Longrightarrow$ each a-transitions synchronizes with a 0-duration time transition of H_2

Outline

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

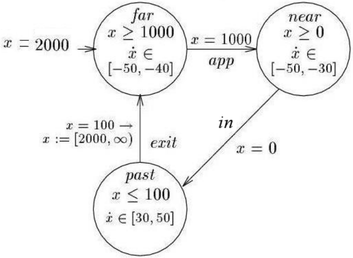
CLP Model

Locations

- Predicate over real variables Vars,Time and T_0, where Time,T_0 $\in R^+$.
  automaton(Location,Vars,T_0,Time):-
  c(Vars)$\wedge$ c(Inv.)$\wedge$c(T_0,Time).

Example

```
train(far,X,T0,Time):-
 X $>= 2000-50*Time,
 X $=<2000-40*Time,
 Time$>=T0, X$>=1000.
```



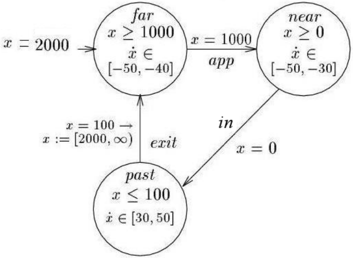
CLP Model

Locations

- Predicate over real variables Vars,Time and T_0, where Time,T_0 $\in R^+$.
  automaton(Location,Vars,T_0,Time):-
  c(Vars)$\wedge$ c(Inv.)$\wedge$c(T_0,Time).

Example

```
train(far,X,T0,Time):-
 X $>= 2000-50*Time,
 X $=<2000-40*Time,
 Time$>=T0, X$>=1000.
```



CLP Model

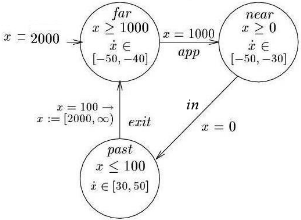
State Change

evolve(Automaton,L,NextL,T_0,Delay,Time,Event,T_{new0}) :-
discrete(Automaton,L,NextL,T_0,Delay,Time,Event,T_{new0});
continuous(Automaton,L,NextL,T_0,Delay,Time,Event,T_{new0}).

CLP Model

Discrete

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



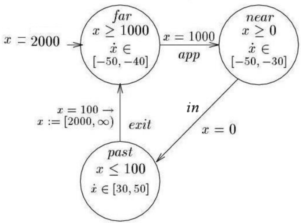
Continuous

```
continuous(train,L,L,T0,Delay,Time,Event,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

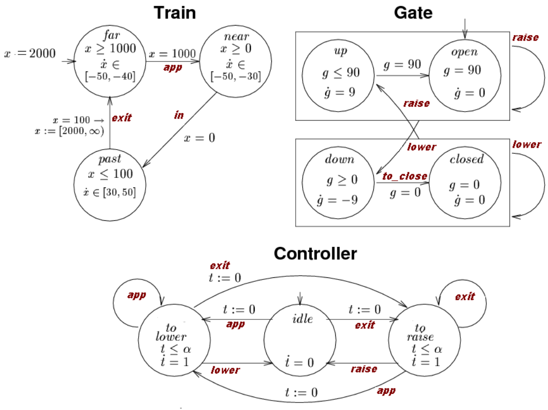
```
continuous(train,L,L,T0,Delay,Time,Event,Tnew0):-
 Event &\=app,Event &\=exit,Event &\=in ,
 Time >T0+Delay,Tnew0 is T0+Delay.
```

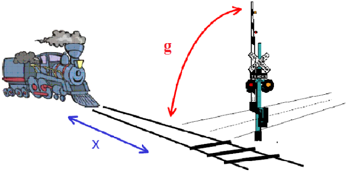
Outline

CLP Model:Controlling the concurrent Execution

driver($(L_1,T_{01}),(L_2,T_{02}),\ldots,(L_n,T_{0n})$):-
$automata_1(L_1,Var_1,T_{01},Time_1)$,
$automata_2(L_2,Var_2,T_{02},Time_2)$,
$\ldots$,
$automata_2(L_2,Var_2,T_{02},Time_2)$,
$miniEventTime(Time_1,Time_2,\ldots,Time_n,Delay)$,
$evolve(automata_1,L_1,NextL_1,T_{01},Delay,Time_1,Event,T_{new01})$,
$evolve(automata_2,L_2,NextL_2,T_{02},Delay,Time_2,Event,T_{new02})$,
$\ldots$,
$evolve(automata_n,L_n,NextL_n,T_{0n},Delay,Time_n,Event,T_{new0n})$,
$Time_1 \$>=T_{01}$, $Time_1 \$<=T_{01}+Delay$,
$Time_2 \$>=T_{02}$, $Time_2 \$<=T_{02}+Delay$,
$\ldots$, $Time_n \$>=T_{0n}$, $Time_n \$<=T_{0n}+Delay$,
driver($(NextL_1,T_{new01}),(NextL_2,T_{new02}),\ldots,(NextL_n,T_{new0n})$).

Train Gate Controller as CLP





Outline

Model Execution

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{14.9999999999999 .. 709.0} : G{0.0 .. 90.0} : 0
Event :to_close
Needed Time: 10.0

----

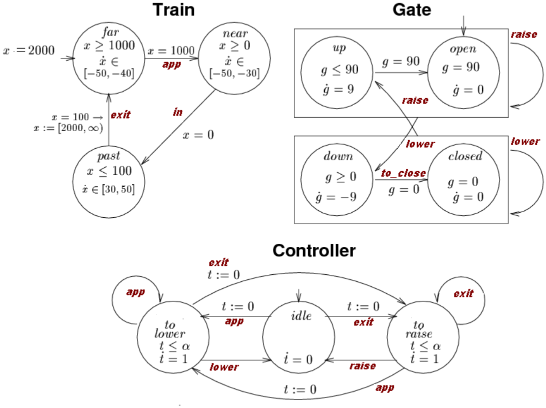
near : close : idle
X{0.0 .. 409.0} : 0 : 0
Event :in
Needed Time: 13.6333333333333

----

past : close : idle
X{0.0 .. 100.0} : 0 : 0
Event :exit
Needed Time: 3.33333333333333

----

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{9.99999999999977 .. 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

----

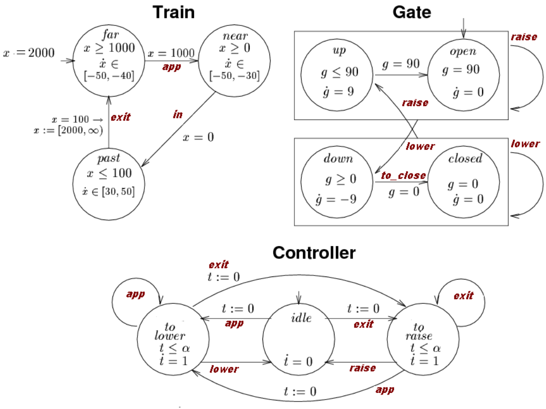
past : close : idle
X{0.0 .. 100.0} : 0 : 0
Event :exit
Needed Time: 3.33333333333333

----

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