Modeling and Verification of Real-time/Hybrid/Cyber-Physical Systems via Concurrent Co-inductive Constraint Logic Programming

Neda Saeedloei

Department of Computer Science
University of Texas at Dallas

May 24th 2011
Outline

1 Motivation
   - Incorporation of Real Time in Computation
   - Related Work
     - Temporal Logics
     - RTCTL

2 Background

3 Contribution
   - Co-inductive CLP(R) Framework for Verifying Real-time Systems
   - Timed Grammars
     - Practical Parser
   - Timed $\pi$-calculus
     - Operational Semantics in LP
   - Foundations of Cyber-Physical Systems (CPS)

4 Summary
Outline

1 Motivation
   - Incorporation of Real Time in Computation
   - Related Work
     - Temporal Logics
     - RTCTL

2 Background

3 Contribution
   - Co-inductive CLP(R) Framework for Verifying Real-time Systems
   - Timed Grammars
     - Practical Parser
   - Timed $\pi$-calculus
     - Operational Semantics in LP
   - Foundations of Cyber-Physical Systems (CPS)

4 Summary
Complex real-time systems are difficult to model and verify because they involve:

- Continuous time
- Perpetual execution
- Concurrency

Goal

- Developing techniques for modeling continuous time in real-time systems
  - Co-inductive logic programming
  - Constraint logic programming over reals (CLP(R))
Outline

1 Motivation
   - Incorporation of Real Time in Computation
   - Related Work
     - Temporal Logics
     - RTCTL

2 Background

3 Contribution
   - Co-inductive CLP(R) Framework for Verifying Real-time Systems
   - Timed Grammars
     - Practical Parser
   - Timed $\pi$-calculus
     - Operational Semantics in LP
   - Foundations of Cyber-Physical Systems (CPS)

4 Summary
Temporal Logics

- Formalisms for describing sequences of transitions between states in a reactive system
- Can be used for verifying discrete real-time systems
  - Time is not mentioned explicitly
- A powerful example of temporal logics: CTL*
- Properties like eventually or never are specified using special temporal operators
- Event $p$ will happen within at most $n$ time units is not simple to express

Cannot be used in a natural and efficient way to verify many types of interesting properties of real-time systems.
RTCTL

- Obtained by introducing bounds in the CTL temporal operators
- Can be used for verification of discrete real time systems
- Simple and effective way to allow the verification of time bounded properties
- Quantitative analysis on discrete-time models can be performed
  - Computing minimum/maximum delays
Continuous Real-Time

- Time is a continuous quantity
- By discretizing time certain aspects of real-time systems may not be modeled faithfully or at least in a natural fashion
- We model time as a continuous quantity rather than discretizing it
  - Constraint logic programming over reals
ω-Automata

- Nondeterministic finite state automata
- Acceptance condition modified suitably so as to handle infinite input words
- ω-automata accept ω-languages, i.e., a language consisting of infinite words
- A well-known type of ω-automata
  - Büchi automata
    - Some state from the set of final states must be traversed infinitely often
Timed Languages

- Behavior of a real-time system can be modeled by a timed word over the alphabet of events
- A timed word over an alphabet $\sum$ is an infinite sequence of pairs of the form $(\sigma_1, \tau_1)(\sigma_2, \tau_2)\ldots$ where
  - $\sigma_i$ is a symbol from the alphabet $\sum$
  - $\tau_i$ is a time-stamp associated with $\sigma_i$, such that $\tau_i \in R$ with $\tau_i > 0$ satisfying
    - Monotonicity: $\tau$ increases strictly monotonically, that is, $\tau_i < \tau_{i+1}$ for all $i \geq 1$
    - Progress: For every $t \in R$ there is some $i \geq 1$ such that $\tau_i > t$
A timed Büchi automaton is a tuple $< \Sigma, S, S_0, C, E, F >$ where

- $\Sigma$ is a finite alphabet
- $S$ is a finite set of states
- $S_0 \subseteq S$ is a set of start states
- $C$ is a finite set of clocks
- $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ gives the set of transitions
- $F$ is a set of final states
Example

Timed Automata
Example

Timed Automata

Motivation
Background
Contribution
Summary

Neda Saeedloei
Timed Automata are not Enough

- Using timed automata is a popular approach to designing, specifying and verifying real-time systems
- Equivalent to timed regular $\omega$-languages
- Timed automata are unsuitable for many complex (and useful) applications
- Timed automata are extended to pushdown timed automata
PTA are obtained from timed automata by adding:

- Stack
- Stack alphabet
- Stack operations, associated with each transition

Acceptance conditions for an infinite string for PTA:

- The stack must be empty in every final state
accepted timed words: $((a, t_a)^n (b, t_b)^n)^\omega$
Co-inductive CLP(R) Framework for Verifying Real-time Systems

1 Motivation
   - Incorporation of Real Time in Computation
   - Related Work
     - Temporal Logics
     - RTCTL

2 Background

3 Contribution
   - Co-inductive CLP(R) Framework for Verifying Real-time Systems
   - Timed Grammars
     - Practical Parser
   - Timed π-calculus
     - Operational Semantics in LP
   - Foundations of Cyber-Physical Systems (CPS)

4 Summary

Neda Saeedloei
Modeling PTA with Co-inductive CLP(R)

- The underlying language is context free, not regular
- Accepted strings are infinite
- Clock constraints model real-time requirements

**Framework**

Logic programming extended with *co-induction* and *constraints over reals* is used to model PTA
Circular phenomena are quite common in Computer Science:

- Circular linked lists
- Graphs (with cycles)
- Controllers (run forever)
- Bisimilarity
- Interactive systems
- Automata over infinite strings/Kripke structures
- Perpetual processes

Numerous other examples can be found elsewhere (Barwise and Moss 1996)
Coinduction

- Infinite structures
  - Some of them can be represented by circular structures
  - Example: \( X = [1, 2, 1, 2, ...] \) can be represented by
  \[ X = [1, 2 | X] \]

- Infinite Proofs
  - Exhibit certain regularity such that coinduction can capture them

- Focus of our group: inclusion of coinductive reasoning techniques in LP and its applications
Induction vs Coinduction

- Induction is a mathematical technique for finitely reasoning about an infinite (countable) no. of things.
- Examples of inductive structures:
  - Naturals: 0, 1, 2, ...
  - Lists: [], [X], [X, X], [X, X, X], ...
- Three components of an inductive definition: (1) initiality, (2) iteration, (3) minimality
  - For example, the set of lists is specified as follows:
    An empty list [], is a list (initiality) ...(i)
    
    \[ H \mid T \] is a list if \( T \) is a list and \( H \) is an element (iteration) ...(ii)
    
    Minimal set that satisfies (i) and (ii) (minimality)
Co-induction is a mathematical technique for (finitely) reasoning about infinite things.

Two components of a coinductive definition: (1) iteration, (2) maximality

- For example, for a list:
  \([H \mid T]\) is a list if \(T\) is a list and \(H\) is an element (iteration). Maximal set that satisfies the specification of a list.

- This coinductive definition specifies all lists of infinite size.
## Mathematical Foundations

<table>
<thead>
<tr>
<th>Definition</th>
<th>Proof</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least fixed point</td>
<td>Induction</td>
<td>Recursion</td>
</tr>
<tr>
<td>Greatest fixed point</td>
<td>Coinduction</td>
<td>Corecursion</td>
</tr>
</tbody>
</table>

Motivation

Background

Contribution

Summary

Co-inductive CLP(R) Framework for Verifying Real-time Systems
Timed Grammars
Timed $\pi$-calculus
Foundations of Cyber-Physical Systems (CPS)
Operational Semantics

p :- p.
The query \(?- p. to succeed.

p([1 | T]) :- p(T).
The query \(?- p(X) to succeed with X= [1 | X].
Operational Semantics

- Nondeterministic state transition system
- States are pairs of:
  - A finite list of syntactic atoms [resolvent] (as in Prolog)
  - A set of syntactic term equations of the form $x = f(x)$ or $x = t$
- Transition rules
  - Definite clause rule
  - “Coinductive hypothesis rule”
    If a coinductive goal $G$ is called, and $G$ unifies with a call made earlier then $G$ succeeds.
Co-induction

Example: perpetual binary streams

```prolog
bit(0).
bit(1).
bitstream([H | T]) :- bit(H), bitstream(T).
?- X = [0, 1, 1, 0 | X], bitstream(X).
```

- Traditional logic program will not terminate.
Example: perpetual binary streams in Coinductive LP

```
:- coinductive stream/1.
stream( [ H | T ] ) :- num( H ), stream( T ).
um( 0 ).
um( s( N ) ) :- num( N ).

?- stream( [ 0, s( 0 ), s( s( 0 ) ) | T ] ).
MEMO: stream( [ 0, s( 0 ), s( s( 0 ) ) | T ] )
MEMO: stream( [ s( 0 ), s( s( 0 ) ) | T ] )
MEMO: stream( [ s( s( 0 ) ) | T ] )
MEMO: stream(T)

Answers:
T = [ 0, s(0), s(s(0)) | T ]
T = [ s(0), s(s(0)), s(0), s(s(0)) | T ]
T = [ s(s(0)) | T ]  
T = [ 0, s(0), s(s(0)) | X ] (where X is any rational list of numbers.)
```
Example of Modeling PTA with Co-inductive CLP(R)

trans(s0, (a, T), s1, Ci, Co, [], [1] ):-{Co=T}.
trans(s1, (a, T), s1, Ci, Co, P, [1|P]):-{Co=Ci}.
trans(s1, (b, T), s2, Ci, Co, [1|P], P):-{T-Ci<5, Co=Ci}.
trans(s2, (b, T), s2, Ci, Co, [1|P], P):-{Co=Ci}.
trans(s2, (b, T), s0, Ci, Co, [1|P], P):-{T-Ci<20, Co=Ci}.
Example of Modeling PTA with Co-inductive CLP(R)

```prolog
:- coinductive(driver/6).

driver([H | R], Si, T, Ci, Pi, [(H, T) | S]) :-
    trans(Si, (H, T), So, Ci, Co, Pi, Po),
    \{T2 > T\},
    driver(R, So, T2, Co, Po, S).
```

- **Input**
  - Can be fully specified, e.g., [a,a,a,b,b,b, ...]
  - Can be partially specified, e.g., [a,X,a,Y,b,b, ...]
  - Can be unspecified, e.g., X

- **Output**
  - Concrete legal behavior of the system
  - Sequences of time-stamped events
    - Time-stamps are not concrete, but related by set of constraints
  - More general than what you normally expect
Example of Modeling PTA with Co-inductive CLP(R)

\[
[(a,0), (a,2), (b,4), (b,16),...]
\%
\text{is legal}
\%
\text{(will unify with the output of the program)}

\[
[(a,0), (a,2), (b,6), (b,16),...]
\%
\text{is not legal}

\[
[(a,0), (a,2), (b,4), (b,8), (b,16),...]
\%
\text{is not legal}
\]
Application: The Generalized Railroad Crossing (GRC) Problem

- Several tracks and an unspecified number of trains traveling in both directions
- A gate at the railroad crossing, operated (by a controller), in a way that guarantees
  - Safety: The gate must be down while one or more trains are in the crossing
  - Utility: The gate goes down only if a train is approaching
Motivation

Background

Contribution

Summary

Co-inductive CLP(R) Framework for Verifying Real-time Systems
Timed Grammars
Timed π-calculus
Foundations of Cyber-Physical Systems (CPS)

GRC

Neda Saeedloei
Verifying Properties

Given a property Q to be verified

- Specify its negation as a logic program, notQ
- If the query notQ fails w.r.t. the logic program that models the system, the property Q holds.
- If the query notQ succeeds, the answer provides a counterexample to why the property Q does not hold.
Verifying Safety and Utility

unsafe(N) :- driver(s0, s0, 0, 0, 0, X, N, R),
    append(C, [ (in(_), _) | D ], R),
    append(A, [ (up(_), _) | B ], C),
    not_member((down, _), B).

unutilized(N) :- driver(s0, s0, 0, 0, 0, X, N, R),
    append(A, [ (down, _) | B ], R),
    find_first_up(B, C),
    not_member((in(_), _), C).
## Verification Time

**Table:** safety and utility verification times

<table>
<thead>
<tr>
<th>Number of tracks</th>
<th>safety</th>
<th>utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.006</td>
<td>0.006</td>
</tr>
<tr>
<td>2</td>
<td>0.065</td>
<td>0.072</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0.587</td>
</tr>
<tr>
<td>4</td>
<td>5.666</td>
<td>5.634</td>
</tr>
<tr>
<td>5</td>
<td>60.013</td>
<td>60.430</td>
</tr>
<tr>
<td>6</td>
<td>426.300</td>
<td>453.544</td>
</tr>
</tbody>
</table>
Motivation

Incorporation of Real Time in Computation
Related Work
  - Temporal Logics
  - RTCTL

Background

Contribution

Co-inductive CLP(R) Framework for Verifying Real-time Systems
  - Timed Grammars
  - Timed $\pi$-calculus

Practical Parser
Operational Semantics in LP
Foundations of Cyber-Physical Systems (CPS)

Summary

Neda Saeedloei
Motivation

- For real-time systems timed regular languages may not be powerful enough
- Timed context-free languages might be needed
We propose timed grammars
- Simple and natural method for describing timed languages
- Describe words that have real-time constraints placed on the times at which the word’s symbols appear

Equivalence of PTA and $\omega$-TCFGs

Modeling $\omega$-TCFGs with
- Definite clause grammars (DCGs)
- Constraints over reals (CLP(R))
- Co-induction

Complex real-time systems can be directly (and naturally) modeled as co-inductive CLP(R) programs
Timed Context-Free Grammars Examples

\[
S \rightarrow a \ \{ c := 0 \} \ S
\]
\[
S \rightarrow b \ \{ c < 5 \}
\]
\[
S \rightarrow a \ \{ c := 0 \} \ R
\]
\[
R \rightarrow a \ R
\]
\[
R \rightarrow b \ \{ c < 5 \}
\]
Timed Context-Free $\omega$-Grammars ($\omega$-TCFGs)

Timed Context-free grammars with co-recursive grammar rules (i.e., recursive rules that need not have base cases)

Example

\[
\begin{align*}
S &\rightarrow R \ S \\
R &\rightarrow a \ \{c := 0\} \ T \ b \ \{c < 20\} \\
T &\rightarrow a \ T \ b \\
T &\rightarrow a \ b \ \{c < 5\}
\end{align*}
\]
Incorporation of co-induction and CLP(R) into DCGs allows modeling of ω-TCFGs, this model serves as a practical parser for the ω-TCFL recognized by the ω-TCFG.

- General method of Converting ω-TCFGs to co-inductive CLP(R) programs
  - The generated LP models the ω-TCFG as a collection of DCG rules
  - Each rule is extended with clock expressions
Example: Parser

\[ s(T, Ci, Co) \rightarrow r(T, Ci, Co1), \{T2 > T\}, s1(T2, Co1, Co). \]

\[ r(T, Ci, Co) \rightarrow [(a, T)], \{Ci = T, T2 > T\}, \]
\[ t(T2, Ci, Co1), \{T3 > T2\}, \]
\[ [(b, T3)], \{T3 - Ci < 20\}. \]

\[ t(T, Ci, Co) \rightarrow [(a, T)], \{T2 > T\}, t(T2, Ci, Co1), \]
\[ \{T3 > T2\}, [(b, T3)], \{Co = Co1\}. \]

\[ t(T, Ci, Co) \rightarrow [(a, T)], \{T2 > T\}, [(b, T2)], \]
\[ \{T2 - Ci < 5, Co = Ci\}. \]
Co-inductive CLP(R) Framework for Verifying Real-time Systems
Timed Grammars
Timed $\pi$-calculus
Foundations of Cyber-Physical Systems (CPS)

Timed Context-Free $\omega$-Grammars Modeled as Co-inductive CLP(R) Programs

- Check whether a particular timed string will be accepted or not
- Systematically generate all possible timed strings that can be accepted
- Verify system properties by posing appropriate queries
Motivation

Background

Contribution

Summary

Co-inductive CLP(R) Framework for Verifying Real-time Systems

Timed Grammars

Timed $\pi$-calculus

Foundations of Cyber-Physical Systems (CPS)

Timed Context-Free $\omega$-Grammar Example

\[
\begin{align*}
C & \rightarrow \text{approach}\{c := 0\} \ L \ \text{exit}\{c := 0\} \ \text{raise}\{c < 1\} \ C \\
C & \rightarrow \text{approach}\{c := 0\} \ L \ N \ \text{exit}\{c := 0\} \ \text{raise}\{c < 1\} \ C \\
L & \rightarrow \text{lower}\{c < 1\} \\
L & \rightarrow \text{approach} \ \text{lower}\{c < 1\} \ \text{exit} \\
N & \rightarrow \text{approach} \ \text{exit} \\
N & \rightarrow \text{approach} \ \text{exit} \ N \\
N & \rightarrow \text{exit} \ \text{approach} \\
N & \rightarrow \text{exit} \ \text{approach} \ N
\end{align*}
\]

Neda Saeedloei
Equivalence of PTA and $\omega$-CFGs

\[
S \rightarrow R \ S \\
R \rightarrow a \ \{c := 0\} \ T \ b \ \{c < 20\} \\
T \rightarrow a \ T \ b \\
T \rightarrow a \ b \ \{c < 5\}
\]
Outline

1. Motivation
   - Incorporation of Real Time in Computation
   - Related Work
     - Temporal Logics
     - RTCTL

2. Background

3. Contribution
   - Co-inductive CLP(R) Framework for Verifying Real-time Systems
   - Timed Grammars
     - Practical Parser
   - Timed π-calculus
     - Operational Semantics in LP
   - Foundations of Cyber-Physical Systems (CPS)

4. Summary

Neda Saeedloei
Motivation

- π-calculus was introduced with the aim of modeling concurrent/mobile processes
- It is not equipped to model concurrent real-time systems and reason about their behavior
  - Several extensions of π-calculus with time have been proposed
  - All these approaches discretize time rather than represent it faithfully as a continuous quantity
Outline

- Extending $\pi$-calculus with real time by adding clocks
  - Powerful formalism for describing concurrent real-time systems and reasoning about their behaviors
- Developing operational semantics for the proposed timed $\pi$-calculus
- Developing the notion of timed bisimilarity and its properties (not presented here)
  - e.g., expansion theorem for real-time, concurrent, mobile processes
- Implementation based on co-induction, coroutining, and constraint logic programming over reals of operational semantics
- Application Example
Design Decisions

- Associating time-stamps to all messages
- Adding clocks
- Adding clock operations
  - Clock resets
  - Clock constraints
- Representing messages by triples of the form \( \langle m, t_m, c \rangle \)
Syntax

\[
\begin{align*}
C & ::= C_c \ C_r \\
C_c & ::= (Clock \sim x)C_c \mid (Clock - t \sim x)C_c \mid \epsilon \\
C_r & ::= (Clock := 0)C_r \mid \epsilon \\
\sim & ::= < \mid > \mid \leq \mid \geq \mid = \\
M & ::= C\bar{x}\langle y, t_y, c \rangle.P \mid Cx(\langle y, t_y, c \rangle).P \mid C_{\tau}.P \mid 0 \mid M + M' \\
P & ::= M \mid P \mid P' \mid !P \mid \nu z\ P \mid [x = y] P
\end{align*}
\]
Examples

Example 1

The expression $x(\langle m, t_m, c \rangle). (c - t_m \geq 5) \bar{y}\langle n, t_n, c \rangle$ represents a process that receives a message $m$ on channel $x$ and sends a message $n$ on channel $y$ with the delay of at least 5 units of time.

Example 2

Consider a system which is composed of two processes $P$ and $Q$ that run in parallel. Moreover, there is a clock $c$ that can be accessed by both $P$ and $Q$ which should be reset before the parallel execution begins. The timed $\pi$-calculus expression presenting this scenario is $(c := 0) \tau.(P | Q)$. 
Actions

\[ \alpha_t \ ::= \ C_r, \bar{x}\langle y, t_y, c \rangle \mid C_r, x(\langle y, t_y, c \rangle) \mid C_r, \bar{x}(\langle y, t_y, c \rangle) \mid C_r, \langle \tau, t \rangle \]

- \( P \xrightarrow{C_r,\bar{x}\langle y, t_y, c \rangle} Q \) : \( P \) sends \( \langle y, t_y, c \rangle \) via \( x \), and evolves to \( Q \).
- \( P \xrightarrow{C_r,x(\langle y, t_y, c \rangle)} Q \) : \( P \) receives any message \( \langle w, t_w, d \rangle \) and becomes \( Q\{w/y, t_w/t_y, d/c\} \).
- \( P \xrightarrow{C_r,\bar{x}(\langle y, t_y, c \rangle)} Q \) : \( P \) emits a private name along with its time-stamp and a clock on port \( x \), and becomes \( Q \).
- \( P \xrightarrow{C_r,\langle \tau, t \rangle} Q \) : \( P \) takes an internal action at time \( t \).
- The set of clocks that should be reset in each transition is specified by \( C_r \).
Timed $\pi$-calculus Operational Semantics

TAU \[ \frac{[C_c]}{C_c,\tau \cdot P \xrightarrow{\tau} P} \]

OUT \[ \frac{[C_c]}{C_c,\tilde{x}(y, t_{y}, c) \cdot P \xrightarrow{\tilde{x}(y, t_{y}, c)} P} \]

INP \[ \frac{\[C_c((d/c))\]}{C_c,\tilde{x}(z, t_{z}, c) \cdot P \xrightarrow{\tilde{x}(z, t_{z}, c)\{y/z, t_{y} / t_{z}, d/c\}} P(y / z, t_{y} / t_{z}, d / c)} y \notin fn(vzP), d \notin c(P) \]

MAT \[ \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'} \]

SUM \[ \frac{P \xrightarrow{\alpha, p} P'}{P + Q \xrightarrow{\alpha, p} P'} \]

PAR \[ \frac{P \xrightarrow{\alpha} P', Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P' \mid Q'} \]

COM \[ \frac{C_c,\tilde{x}(y, t_{y}, c) \cdot P \between C_c,\tilde{x}(y, t_{y}, c) \cdot Q}{P \between Q \xrightarrow{C_c,\tilde{x}(y, t_{y}, c)} P' \between Q'} \]

CLOSE \[ \frac{P \xrightarrow{\alpha, p} P', Q \xrightarrow{\alpha, q} Q'}{P \between Q \xrightarrow{\alpha, p, q} \nu z(P' \between Q')} \]

RES \[ \frac{P \xrightarrow{\alpha, \nu zP} \nu zP}{P \xrightarrow{\alpha} p} \]

OPEN \[ \frac{P \xrightarrow{\alpha, \nu yP} \nu yP}{P \xrightarrow{\alpha, \nu yP} p} y \neq x \]

REP-ACT \[ \frac{P \xrightarrow{\alpha} P'}{P \xrightarrow{\alpha} P' \parallel P} \]

REP-COM \[ \frac{P \xrightarrow{\alpha, p} P', C_c,\tilde{x}(y, t_{y}, c) \cdot P''}{P \parallel P' \xrightarrow{\alpha, \tilde{x}(y, t_{y}, c)\{P' \parallel P''\}} P''} \]

REP-CLOSE \[ \frac{P \xrightarrow{\alpha, p} P', C_c,\tilde{x}(z, t_{z}, c) \cdot P''}{P \parallel P' \xrightarrow{\alpha, \tilde{x}(z, t_{z}, c)\{P' \parallel P''\}} P''} z \notin fn(P) \]
Motivation
Background
Contribution
Summary

Co-inductive CLP(R) Framework for Verifying Real-time Systems
Timed Grammars
Timed π-calculus
Foundations of Cyber-Physical Systems (CPS)

Timed π-calculus Operational Semantics

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transition Rules for TAU, OUT, INP, MAT, SUM, PAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU</td>
<td>$[C_c] C_c C_r \tau.P \xrightarrow{C_r,(\tau,t)} P$</td>
</tr>
<tr>
<td>OUT</td>
<td>$[C_c] C_c C_r \bar{x}(y, t_y, c).P \xrightarrow{C_r, \bar{x}(y, t_y, c)} P$</td>
</tr>
<tr>
<td>INP</td>
<td>$[C_c{d/c}] C_c C_r x((z, t_z, c)).P \xrightarrow{C_r{d/c}, x((y, t_y, d))} P{y/z, t_y/t_z, d/c}$</td>
</tr>
<tr>
<td>MAT</td>
<td>$P \xrightarrow{\alpha_t} P' \quad [x = x]P \xrightarrow{\alpha_t} P'$</td>
</tr>
<tr>
<td>SUM</td>
<td>$P \xrightarrow{\alpha_t} P' \quad P + Q \xrightarrow{\alpha_t} P'$</td>
</tr>
<tr>
<td>PAR</td>
<td>$P \xrightarrow{\alpha_t} P' \quad P \parallel Q \xrightarrow{\alpha_t} P' \parallel Q$</td>
</tr>
</tbody>
</table>

Table: Timed π-calculus Transition Rules for TAU, OUT, INP, MAT, SUM, PAR

Neda Saeedloei
Operational Semantics in Logic Programming

Syntax of the Language in LP

\[ A ::= \text{out}((C, \mathcal{N}, \mathcal{M}), \mathcal{P}) \mid \text{in}((C, \mathcal{N}, \mathcal{M}), \mathcal{P}) \mid \text{tau}((C, T), \mathcal{P}) \mid \text{zero} \mid \text{choice}(\mathcal{P}, \mathcal{P}) \mid \text{par}(\mathcal{P}, \mathcal{P}) \mid \text{rep}(\mathcal{P}) \mid \text{nu}(\mathcal{N}, \mathcal{P}) \mid \text{match}(\mathcal{N} = \mathcal{N}, \mathcal{P}) \]

\[ \mathcal{C} ::= \text{reset}(\mathcal{C}N) \mid \text{const}(\mathcal{C}N \sim \mathcal{R}) \mid \text{const}(\mathcal{C}N - T \sim \mathcal{R}) \]

\[ \mathcal{D} ::= \text{proc}(\mathcal{P}N, \mathcal{P}) \]

\[ \mathcal{M} ::= (\mathcal{N}, T, \mathcal{C}N) \]
Example: 1-Track GRC

Motivation

Background

Contribution

Summary

Co-inductive CLP(R) Framework for Verifying Real-time Systems

Timed Grammars

Timed $\pi$-calculus

Foundations of Cyber-Physical Systems (CPS)

Neda Saeedloei
Example: 1-Track GRC

\[
\text{train} \equiv v \, pc \, \text{ch1<pc, \(t_p\), \(t\)}. \\
(t := 0) \, pc<\text{approach, \(t_a\), \(t\)}. \\
(t > 2) \, (\tau, \, t_i). \\
(\tau, \, t_o). \\
(t < 5) \, pc<\text{exit, \(t_e\), \(t\)}.
\]

\[
\text{proc(train, }
\text{nu(out(ch1, (pc, \(tp\), \(t\))),}
\text{in(reset(p), pc, (approach, \(ta\), \(t\))),}
\text{tau((t>2)(t<3), ti),}
\text{tau(to),}
\text{out((t<5), pc, (exit, te, t))})
\]

Neda Saeedloei
Example: 1-Track GRC

controller ≜ ch1(<pc, t, c>).pc(<x_1, t_1, c>).
(c = 1)(c := 0) ch2<lower, t_l, c>.
pc(<x_2, t_2, c>).c-t_2 < 1)(c := 0) ch2<raise, t_r, c>

proc(controller,
in(ch1, (pc, tp, c)),
in(pc, (x_1, t_1, c)),
out((c=1)(c:=0), ch2, (lower, t_l, c)),
in(pc, (x_2, t_2, c)),
out((c<1)(c:=0), ch2, (raise, t_r, c)))
Example: 1-Track GRC

\[
\text{gate} \equiv \text{ch2}(x, t_x, g).
\]
\[
([x = \text{lower}] (g < 1) (\tau, t_d) + \\
[x = \text{raise}] (g > 1) (g < 2) (\tau, t_u))
\]

\[
\text{proc(gate,}
\]
\[
\text{in(ch2, (x,tx,g)),}
\]
\[
\text{choice(match(x=lower, tau((g<1), td)),}
\]
\[
\text{match(x=raise, tau((g>1)(g<2), tu))))}
\]
Co-inductive CLP(R) Framework for Verifying Real-time Systems
Timed Grammars
Timed $\pi$-calculus
Foundations of Cyber-Physical Systems (CPS)

Example: 1-Track GRC

\[
\text{train}(X, Y, W, Tc, Si) :-
\]
\[
(H = \text{approach}, \{Tc2 = W\};
\]
\[
H = \text{in}, \{W - Tc > 2, Tc2 = Tc\};
\]
\[
H = \text{out}, \{Tc2 = Tc\};
\]
\[
H = \text{exit}, \{W - Tc < 5, Tc2 = Tc\},
\]
\[
\{W2 > W\},
\]
\[
\text{train\_trans}(Si, H, So),
\]
\[
\text{freeze}(X, \text{train}(Xs, Ys, W2, Tc2, So)),
\]
\[
((H = \text{approach}; H = \text{exit}) \rightarrow Y = [(H, W)| Ys];
\]
\[
Y = Ys),
\]
\[
X = [(H, W)| Xs].
\]
Example: 1-Track GRC

controller([(H, W)| Xs], Y, Sc) :-
    freeze(Xs, controller(Xs, Ys, Sc3)),
    (H = approach, M = lower, \{W2 > W, W2 - W = 1\};
    H = exit, M = raise, \{W2 > W, W2 - W < 1\}),
    controller_trans(Sc, H, Sc2),
    controller_trans(Sc2, M, Sc3),
    Y = [(M, W2)| Ys].

gate([(H, W)| Xs], Sg) :-
    freeze(Xs, gate(Xs, Sg3)),
    (H = lower, M = down, \{W2 > W, W2 - W < 1\};
    H = raise, M = up, \{W2 > W, W2 - W > 1, W2 - W < 2\}),
    gate_trans(Sg, H, Sg2), gate_trans(Sg2, M, Sg3).

main(A, B, C) :-
    freeze(A, (freeze(C, gate(C, s0)),
    controller(B, C, s0))), train(A, B, 0, 0, s0).
Internal Transitions of GRC Components

\[
\begin{align*}
\text{train-trans}(s0, \text{ approach}, & \quad s1). \\
\text{train-trans}(s1, \text{ in}, & \quad s2). \\
\text{train-trans}(s2, \text{ out}, & \quad s3). \\
\text{train-trans}(s3, \text{ exit}, & \quad s0). \\
\text{c-trans}(s0, \text{ approach}, & \quad s1). \\
\text{c-trans}(s1, \text{ lower}, & \quad s2). \\
\text{c-trans}(s2, \text{ exit}, & \quad s3). \\
\text{c-trans}(s3, \text{ raise}, & \quad s0). \\
\text{g-trans}(s0, \text{ lower}, & \quad s1). \\
\text{g-trans}(s1, \text{ down}, & \quad s2). \\
\text{g-trans}(s2, \text{ raise}, & \quad s3). \\
\text{g-trans}(s3, \text{ up}, & \quad s0).
\end{align*}
\]
Outline

1 Motivation
   - Incorporation of Real Time in Computation
   - Related Work
     - Temporal Logics
     - RTCTL

2 Background

3 Contribution
   - Co-inductive CLP(R) Framework for Verifying Real-time Systems
   - Timed Grammars
     - Practical Parser
   - Timed $\pi$-calculus
     - Operational Semantics in LP
   - Foundations of Cyber-Physical Systems (CPS)

4 Summary

Neda Saeedloei
CPS consist of perpetually and concurrently executing physical and computational components.

The presence of physical components require the computational components to deal with continuous quantities.
CPS Characteristics Summary

- Perform discrete computations
- Deal with continuous physical quantities
- Run forever
- They are concurrent
Design Challenges of CPS

- Dealing with continuous quantities in computations
  - typical approaches discretize them, e.g., time
- Operational modeling/analysis of perpetual computations is not well understood
  - Co-induction have been introduced to formally model rational, infinite computations
- Concurrency is reasonably well understood
- However, concurrency combined with continuous quantities and perpetual computations makes modeling of CPS difficult
Problem
A formalism that can model discrete and continuous quantities together with concurrent, perpetual execution is lacking

Goal
Faithfully modeling CPS and reasoning about them

Our Thesis
Logic programming extended with co-induction, constraints over reals and coroutining is an excellent formalism for modeling CPS and reasoning about them.
Modeling CPS

- Communicating hybrid \( \omega \)-automata as underlying model
  - State machines modeled as *logic programs*
  - Physical quantities are represented as continuous quantities (i.e., not discretized)
    - The constraints imposed on them by CPS physical interactions are faithfully modeled with *CLP(R)*
- Non-terminating nature handled via *co-inductive LP*
- The communication/concurrency is handled by *coroutining*

So each hybrid \( \omega \)-automaton modeled as a co-inductive CLP(R) program

The multiple co-inductive CLP(R) programs execute concurrently modeled as co-routined logic programs
Traditional Example of CPS: Reactor Temperature Control System

Motivation
Background
Contribution
Summary

Co-inductive CLP(R) Framework for Verifying Real-time Systems
Timed Grammars
Timed $\pi$-calculus
Foundations of Cyber-Physical Systems (CPS)
LP Realization of Reactor Temperature Control System

r1(out1, add1, in1, W, Ti, To, T) :- \{ W - Ti \geq T, \ To = Ti \}.

r1(in1, remove1, out1, W, Ti, To, T) :- \{ To = W \}.

r2(out2, add2, in2, W, Ti, To, T) :- \{ W - Ti \geq T, \ To = Ti \}.

r2(in2, remove2, out2, W, Ti, To, T) :- \{ To = W \}.
LP Model of Reactor Temperature Control System

\[
c(norod, add1, rod1, Pi, Po, W, Ti1, Ti2, To1, To2, F) :-
\begin{align*}
  & (F == 1 \rightarrow \{Ti=Ti1\}; \{Ti=Ti2\}), \\
  & \{Pi<550, Po=550, \exp(e, (W-Ti)/10)=5, To1=W, To2=Ti2\}.
\end{align*}
\]

\[
c(rod1, remove1, norod, Pi, Po, W, Ti1, Ti2, To1, To2, F) :-
\begin{align*}
  & \{Pi>510, Po=510, \exp(e, (W-Ti1)/10)=5, To1=W, To2=Ti2\}.
\end{align*}
\]

\[
c(norod, add2, rod2, Pi, Po, W, Ti1, Ti2, To1, To2, F) :-
\begin{align*}
  & (F == 1 \rightarrow \{Ti=Ti1\}; \{Ti=Ti2\}), \\
  & \{Pi<550, Po=550, \exp(e, (W-Ti)/10)=5, To1=Ti1, To2=W\}.
\end{align*}
\]

\[
c(rod2, remove2, norod, Pi, Po, W, Ti1, Ti2, To1, To2, F) :-
\begin{align*}
  & \{Pi>510, Po=510, \exp(e, (T-Ti2)/10)=9/5, To1=Ti1, To2=W\}.
\end{align*}
\]

\[
c(norod, _, shutdown, Pi, Po, W, Ti1, Ti2, To1, To2, F) :-
\begin{align*}
  & (F == 1 \rightarrow \{Ti=Ti1\}; \{Ti=Ti2\}), \\
  & \{Pi<550, Po=550, \exp(e, (W-Ti)/10)=5, To1=Ti1, To2=Ti2\}.
\end{align*}
\]
LP Model of Reactor Temperature Control System

:- coinductive(rod1/6).
rod1([(H, W)| Xs], Si1, Si2, Ti1, Ti2, T) :-
  (H = add1; H = remove1) ->
  (H = add1 -> freeze(Xs,rod1(Xs, So1, Si2, To1, Ti2, T));
   freeze(Xs,rod1(Xs, So1, Si2, To1, Ti2, T);
   rod2(Xs, So1, Si2, To1, Ti2, T)));,
  r1(Si1, H, So1, W, Ti1, To1, T);
  H = shutdown, {W - Ti1 < T, W - Ti2 < T}).

:- coinductive(rod2/6).
rod2([(H, W)| Xs], Si1, Si2, Ti1, Ti2, T) :-
  (H = add2; H = remove2) ->
  (H = add2 -> freeze(Xs,rod2(Xs, Si1, So2, Ti1, To2, T));
   freeze(Xs,rod1(Xs, Si1, So2, Ti1, To2, T);
   rod2(Xs, Si1, So2, Ti1, To2, T)));,
  r2(Si2, H, So2, W, Ti2, To2, T);
  H = shutdown, {W - Ti1 < T, W - Ti2 < T}).
Co-inductive CLP(R) Framework for Verifying Real-time Systems

Timed Grammars

Timed $\pi$-calculus

Foundations of Cyber-Physical Systems (CPS)

LP Model of Reactor Temperature Control System

```prolog
:- coinductive(contr/7).
contr(X, Si, W, Pi, Ti1, Ti2, Fi) :-
(H=add1; H=remove1; H=add2; H=remove2; H=shutdown),
\{W2 > W\},
freeze(X,contr(Xs, So, W2, Po, To1, To2, Fo)),
c(Si,H,So,Pi,Po,W,Ti1,Ti2,To1,To2,Fi),
((H=add1; H=remove1) -> Fo = 1; Fo = 2),
((H=add1; H=remove1; H=add2; H=remove2) ->
 X = [(H, W)| Xs]; X = [(H, W)]).

main(S, W, T) :-
{W - Tr1 = T, W - Tr2 = T},
freeze(S, (rod1(S, s0, s0, Tr1, Tr2, T);
 rod2(S, s0, s0, Tr1, Tr2, T))),
contr(S, s0, W, 510, Tc1, Tc2, 1).
```

Neda Saeedloei
Summary

• Techniques for incorporation of continuous time in computation
  • Co-inductive CLP(R) framework for modeling and verification of real-time systems
  • Timed Grammars
    • Practical parsers
  • Timed $\pi$-calculus
    • Operational Semantics in LP
  • Foundations of CPS

• Future work
  • Incorporation of continuous time in traditional model checkers


