Timed-pNets Semantic Model

Yanwen Chen
Supervisor: Yixiang Chen Eric Madelaine

ECNU INRIA

11/02/2014
Outline

- Motivation
- Basic definition: Logical clock, clock relations, timed specification
- Open/Close timed-pNets and the Compatibility
- Timed-pNets Tree
- Generate timed specification
- Conclusion and future work
Extend pNets by adding multi logical clocks and clock relations such that:

- build a timed model for the distributed systems that have no global physical clock.
- synchronous + asynchronous communications
Our proposal: Timed-pNets

- Logical clocks and clock relations are embedded into timed-pLTSs.
- Timed-pNets are built in terms of the communication between timed-pLTSs.
- These communications are represented by clock relations.
New proposal: logical clock and clock relations

- build logical clock: a sequence of timed-action occurrences
- add constraints between action occurrences

- clock filter and clock offset
Timed specification: a set of clocks with clock relations.

\[
\text{clock } C_a \\
\text{clock } C_b \\
\text{clock } C_c \\
\text{timed specification: } TS = \langle \{ C_a, C_b, C_c \}, \{ C_a < C_b, C_b = C_c \ldots \} \rangle
\]
Open/Close Timed-pNets node and the compatibility

Compatibility: $H_{Impl} \subseteq H \iff TS \ll TS'$. 
Timed-pNets Tree Structure

- hierarchical model
  - leaves are timed-pLTS,
  - nodes are synchronisation devices
  - holes are the timed specifications of the subsystems
Generate Timed Specification

- How to generate TS of a timed-pLTS

- How to generate TS of a timed-pNets node
Procedure of generate TS of a Timed-pLTS

1. If $\text{PreAct}(s_0) \notin \emptyset$, then $\text{PreAct}(s_0) < \text{PostAct}(s_0)$. 
   [Assign: $\text{PostActIndex}(s_0) \leftarrow \text{PreActIndex}(s_0) + 1$]

2. $\forall s \setminus s_0$, $\text{PreAct}(s) < \text{PostAct}(s)$, 
   [Assign: $\text{PostActIndex}(s) \leftarrow \text{PreActIndex}(s)$];

3. $\forall s$, if $\exists \alpha. s \xrightarrow{\alpha} s$ and the loop executes $N$ times, then 
   3.1 go inside the self-loop 
      3.1.1 from one preceding action occurrence: $\text{PreAct}(s) < \alpha_{-i}$.

...
Algo1: generate concrete rules for a timed-pLTS

- Analysis causality relations on clock occurrences
- set the assignment functions according to the general rules

<table>
<thead>
<tr>
<th>Transition</th>
<th>Action Occurrence Relations</th>
<th>Index Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_2$ $\xrightarrow{C_{\text{R}}} s_0$ $\xrightarrow{C_{\text{Cmd}}} s_1$</td>
<td>$!R.m &lt; ?Cmd.n$</td>
<td>$f(n) : n = m + 1$</td>
</tr>
<tr>
<td>$s_0$ $\xrightarrow{C_{\text{Cmd}}} s_1$ $\xrightarrow{\tau} s_2$</td>
<td>$?Cmd.n &lt; \tau_r$</td>
<td>$f(r) : r := n$</td>
</tr>
<tr>
<td>$s_0$ $\xrightarrow{C_{\text{Cmd}}} s_1$ $\xrightarrow{C_{\text{Notify}}} s_1$</td>
<td>$?Cmd.n &lt; !\text{notify}_i$</td>
<td>$i := i + 1$</td>
</tr>
<tr>
<td>$s_1$ $\xrightarrow{C_{\text{Notify}}} s_1$ $\xrightarrow{C_{\text{Notify}}} s_1$</td>
<td>$!\text{notify}<em>i &lt; !\text{notify}</em>{(i + 1)}$</td>
<td></td>
</tr>
<tr>
<td>$s_1$ $\xrightarrow{C_{\text{Notify}}} s_1$ $\xrightarrow{\tau} s_2$ $\xrightarrow{s_2}$</td>
<td>$\tau_r &lt; ?\text{ack}_j$</td>
<td>$f(j) : j = j + 1$</td>
</tr>
<tr>
<td>$\xrightarrow{?\text{ack}<em>j &lt; ?\text{ack}</em>{(j + 1)}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\xrightarrow{?\text{ack}_j &lt; !R.m}$</td>
<td></td>
<td>$f(m) : m = j;$</td>
</tr>
</tbody>
</table>
Initial value: \( m, n, r, i, j := 0; k := 1 \)
reset \( k := 1 \) when going out self-loop

<table>
<thead>
<tr>
<th>Transition</th>
<th>Action Occurrence Relations</th>
<th>Index Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_2 \xrightarrow{\text{?}C_{\text{Cmd}_1}} s_1 )</td>
<td>!( R_m &lt; ?\text{Cmd}_n )</td>
<td>( f(n) : n = m + 1 )</td>
</tr>
<tr>
<td>( s_0 \xrightarrow{\text{?}C_{\text{Cmd}_2}} s_1 )</td>
<td>?( \text{Cmd}_n &lt; \tau_r )</td>
<td>( f(r) : r := n )</td>
</tr>
<tr>
<td>( s_0 \xrightarrow{\text{?}C_{\text{Notify}_1}} s_1 )</td>
<td>?( \text{Cmd}_n &lt; \text{!notify}_i )</td>
<td>( i := i + 1 )</td>
</tr>
<tr>
<td>( s_1 \xrightarrow{\text{?}C_{\text{Notify}_2}} s_1 )</td>
<td>!( \text{notify}<em>i &lt; \text{!notify}</em>{i+1} )</td>
<td>( f(r) : r = n )</td>
</tr>
<tr>
<td>( s_1 \xrightarrow{\text{?}C_{\text{Notify}_3}} s_2 )</td>
<td>!( \text{notify}_i &lt; \tau_r )</td>
<td>( f(j) : j = j + 1 )</td>
</tr>
<tr>
<td>( s_2 \xrightarrow{\text{?}C_{\text{Ack}_j}} s_2 )</td>
<td>( \tau_r &lt; \text{!ack}_j )</td>
<td>( f(m) : m = j )</td>
</tr>
<tr>
<td>( s_2 \xrightarrow{\text{?}C_{\text{Notify}_4}} s_1 )</td>
<td>( \text{!ack}<em>j &lt; \text{!ack}</em>{j+1} )</td>
<td>( )</td>
</tr>
<tr>
<td>( s_1 \xrightarrow{\text{?}C_{\text{Notify}_5}} s_1 )</td>
<td>( \text{!ack}_j &lt; \text{!R}_m )</td>
<td>( )</td>
</tr>
</tbody>
</table>
task: given a set of natural numbers, how to induce its general formula?
We proof: $C_{\alpha}^{\{P(i)\}} \prec C_{\beta}^{\{P'(i)\}} \iff \forall i \in \mathbb{N}, \alpha_{-}(P(i)) \prec \beta_{-}(P'(i))$

so that we have:

- $C_{\alpha} \prec C_{\beta}^{\{2i-1\}} \iff \forall i \in \mathbb{N}, \alpha_{i} \prec \beta_{-}(2i - 1)$
- $C_{\alpha}^{\{2i-1\}} \prec C_{\beta}^{\{2i\}} \iff \forall i \in \mathbb{N}, \alpha_{-}(2i - 1) \prec \beta_{-}(2i)$
generate TS of a Timed-pNets node

Case 1

\[ \forall_1 \quad < \ldots, C_{\alpha_1}, \ldots, C_{\beta_1}, \ldots> \rightarrow C_{a_{g1}} \]
\[ \forall_2 \quad < \ldots, C_{\alpha_2}, \ldots, C_{\beta_2}, \ldots> \rightarrow C_{a_{g2}} \]

- \( C_{a_{g1}} = C_{a_{g1}} \) if and only if \( C_{\alpha_1} = C_{\alpha_2} \land C_{\beta_1} = C_{\beta_2} \).
- \( C_{a_{g1}} < C_{a_{g1}} \) if and only if \( C_{\alpha_1} < C_{\alpha_2} \land C_{\beta_1} < C_{\beta_2} \).

Case 2

\[ \forall_3 \quad < \ldots, C_{\beta_1}, \ldots, C_{\gamma_1}, \ldots> \rightarrow C_{a_{g3}} \]
\[ \forall_4 \quad < C_{\alpha_1}, C_{\beta_2}, \ldots, \ldots, \ldots \rightarrow C_{a_{g4}} \]

- \( C_{a_{g3}} = C_{a_{g4}} \) if and only if \( C_{\beta_1} = C_{\beta_2} \).
- \( C_{a_{g3}} < C_{a_{g4}} \) if and only if \( C_{\beta_1} < C_{\beta_2} \).

Case 3

\[ \forall_5 \quad < \ldots, C_{\beta_1}, \ldots \rightarrow C_{a_{g5}} \]
\[ \forall_6 \quad < \ldots, C_{\beta_2}, \ldots, C_{\gamma_1}, \ldots \rightarrow C_{a_{g6}} \]

- \( C_{a_{g5}} = C_{a_{g6}} \) if and only if \( C_{\beta_1} = C_{\beta_2} \).
- \( C_{a_{g5}} < C_{a_{g6}} \) if and only if \( C_{\beta_1} < C_{\beta_2} \).

Case 4

\[ \forall_5 \quad < \ldots, C_{\beta_1}, \ldots, \ldots \rightarrow C_{a_{g5}} \]
\[ \forall_8 \quad < \ldots, C_{\beta_2}, \ldots, \ldots \rightarrow C_{a_{g8}} \]

- \( C_{a_{g5}} = C_{a_{g8}} \) if and only if \( C_{\beta_1} = C_{\beta_2} \).
- \( C_{a_{g5}} < C_{a_{g8}} \) if and only if \( C_{\beta_1} < C_{\beta_2} \).
Example

### Case 1

- $C_{\text{Notify}g_1} \prec C_{\text{Notify}g_2}$
- $C_{\text{Notify}g_1} \prec C_{\text{Ack}g_3}$
- $C_{\text{Cmd}g_5} \prec C_{Rg_6}$
- $C_{\text{Cmd}g_5} \prec C_{Rg_6}$
- $C_{\text{Req}g_7} \prec C_{\text{Cmd}g_5}$
- $C_{\text{Req}g_7} \prec C_{\text{Req}g_7}$
- $C_{\text{Req}g_7} \prec C_{Tg_8}$
Conclusion

- Our model is flexible—logical clock, without physical common clock
- Flexible to compose by adding relations of clocks in TS
future work

- Current work
  - Prove the completeness of the 4 cases to generating TS of timed-pNets
  - Detect wrong composition by checking the relation conflicts

- Future work
  - Extend to duration-pNets so that we can specify the actions that include execution time
• E.A. Lee. Cyber physical systems: Design challenges.
• Frederic Mallet. CCSL: specifying clock constraints with UML/MARTE.
• Julien Deantoni and Frederic Mallet. TimeSquare: Treat your Models with Logical Time.
Questions