Timed migration and interactions in distributed systems

Formal Methods Laboratory (FML)
Institute of Computer Science
Romanian Academy of Sciences
Iasi Branch

TEAM: Gabriel Ciobanu  gabriel@info.uaic.ro
      Bogdan Aman      baman@iit.tuiasi.ro

Diaspora 2016, Timisoara
27 aprilie 2016

Web page: http://iit.iit.tuiasi.ro/~fml/TiMo
Basic Features of TiMo

- Modelling distributed systems with time-related aspects
- Simple process algebra: locations + mobility + interaction + timers
- Local interaction (communication) and local clocks
- Communication of locations between processes
- Migration is no-urgent, modelling network delays
- Interaction (communication) is not delayed
- Discrete time semantics + maximal concurrency
Example: simple e-shops

In this scenario, we have a client process which initially resides in the *home* location, and wants to find an address of an e-shop where different kinds of electronic items (e-items) can be purchased.

To find out the address of a suitable e-shop, the client, within 3 time units, moves to the location *info* in order to acquire the relevant address.

The location *info* contains a broker who knows all about the availability of the e-shops stocking the desired e-item. In the first 5 time units the right e-shop is the one at the location *shopA*, and after that for 7 time units that at location *shopB*. 
Example: simple e-shops

The specification of the running example which captures the essential features of the scenario described previously can then be written down in the following way:

\[ SES = \text{home[Client]} \mid \text{info[Broker]} \mid \text{shopA[ ]} \mid \text{shopB[~]} \]

where:

\[ \text{Client} = \text{go}^{\Delta^3} \text{info} \cdot (\text{a}^{\Delta^2} \, ? \, (\text{shop}) \, \text{then go}^{\Delta^0} \, \text{shop} \]
\[ \text{else go}^{\Delta^0} \text{home} ) \]

\[ \text{Broker} = \text{a}^{\Delta^5} \, ! \, <\text{shopA}> \, \text{then 0} \]
\[ \text{else a}^{\Delta^7} \, ! \, <\text{shopB}> \]
\( a^{\Delta t} !<v> \) then P else Q

can send v over channel a if put<\textit{a@k}> is present for t time units and continue as P; if unsuccessful, continues as Q

\( a^{\Delta t} ?(u) \) then P else Q

can input some value if get<\textit{a@k}> is present for t time units, and substitute it for variable u within its body (u is bound within P, but not within Q); then continues as P; if unsuccessful, continues as Q
Mobility - intuition

\[ \text{go}^{At} \ u \ \text{then} \ P \]

waits for \( t \) time units before migrating and continuing as \( P \)

\( u \) can be assigned value dynamically through communication with other processes
Operational semantics rules - action

(MOVE)
\[ k \[ \text{go}^{\Delta t} m \text{ then } P \] \xrightarrow{k:m} m [S P] \]

(WAIT)
\[ k \[ \text{go}^{\Delta t} m \text{ then } P \] \xrightarrow{k} k [S \text{go}^{\Delta t-1} m \text{ then } P] \]

(COM)
\[ k \[ \text{a!}^{\Delta t} <v> \text{ then } P \text{ else } Q \mid \text{a?}^{\Delta t'} (u) \text{ then } P' \text{ else } Q' \] \xrightarrow{\text{a(v)}@k} k [S P \mid S \{v/u\}P'] \]
Operational semantics rules - time

\[(\text{TIME})\]

\[N \xrightarrow{\sqrt{k}} \phi_k(N)\]

- applicable if the other action rules cannot be applied at location \(k\) (maximal progress at a location \(k\))
- \(\phi_k(N)\) updates timer values, selects continuations, and removes \(S\)'s
Execution (operational semantics)

\[
k \left[ \text{go}^{\Delta_2} m \cdot \text{a?}^{\Delta_1} (u).P \right] \mid m \left[ \text{a!}^{\Delta_4} <h> \right]
\]
\[
k \left[ \text{go}^{\Delta_2} m \cdot \text{a?}^{\Delta_1} (u).P \right] \mid m \left[ \text{a!}^{\Delta_3} <h> \right]
\]
\[
k \left[ \text{go}^{\Delta_1} m \cdot \text{a?}^{\Delta_1} (u).P \right] \mid m \left[ \text{a!}^{\Delta_3} <h> \right]
\]
\[
k \left[ \text{go}^{\Delta_0} m \cdot \text{a?}^{\Delta_1} (u).P \right] \mid m \left[ \text{a!}^{\Delta_3} <h> \right]
\]
\[
m \left[ \text{a?}^{\Delta_1} (u).P \mid \text{a!}^{\Delta_3} <h> \right]
\]
\[
m \left[ \text{a?}^{\Delta_1} (u).P \mid \text{a!}^{\Delta_2} <h> \right]
\]
\[
m \left[ \{h/u}\{h/u}\} \mid 0 \right]
\]
\[
x \text{ then } P \text{ else } 0 \quad \text{is} \quad x \cdot P
\]
\[
x \text{ then } 0 \text{ else } 0 \quad \text{is} \quad x
\]
Cumulative effect of actions at location k:

\[
\begin{align*}
N & \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_n} \sqrt{k} \xrightarrow{\Psi} M
\end{align*}
\]

- If N is well formed (e.g. no 'S's)
- Then M is well formed

And for \(\Psi\) the multiset of all actions \(\lambda_i\)
Access permissions

Security aspects expressed by access permissions

- **put\textless b@k\textgreater** to send to channel b at location k
- **get\textless b@k\textgreater** to receive from channel b at location k
- **Γ** set of access permissions of a migrating process

Let \( \gamma(k,m)(\Gamma) \) denote the change of access permissions of \( \Gamma \) when moving from \( k \) to \( m \)

Example:

\[
\text{Put}_{b@k}^{-}(\Gamma) = \Gamma - \{\text{put}\textless b@k\textgreater\}
\]
Safe access permissions

AIM:

... to verify that migrating process has sufficient access permissions to enable participation in all potential future communications, and never an unauthorized attempt happens during network evolutions ... 

METHOD:

use judgements of the form:

\[ \Gamma \models_k P \]

meaning that:

\[ \Gamma \] are safe access permissions for \( P \) when it is started up at location \( k \)
Main results

THEOREM 1 (soundness)
Having safe access permissions is preserved over the operational semantics rules

THEOREM 2 (safety of communications)
Processes with safe access permissions are not prevented from participating in communications with other processes

THEOREM 3 (completeness)
Processes without safe access permissions can be placed in a context which blocks a potential communication.
Automatic Analysis in PAT

PAT is an extensible framework for developing domain-specific model checkers.
TiMo@PAT Framework

Modularized design

Over 2640 registered users, over 600 organizations

TiMo@PAT: automatic verification of TiMo systems
Property Definition

Constraints

Bounded constraint: \( expr \preceq x \), where \( \preceq \in \{<, \leq\} \)

Optimal constraint: \( \max(expr) \) or \( \min(expr) \);

Reachability property: \( \top \leadsto K \)

Process migration property: \( P@loc \);

Bounded Liveness Property

E.g., if client is able to arrive at location paying within 10 time units.

Optimized Reasoning

E.g., to find an execution path for process client to arrive at location paying with the shortest time
Example

A travel shop system

Three processes: client, agent and query

Three locations: home, travel shop and server

\[ TS = \text{home[client(130)] | server[agent(100)] | server[query]}; \]

\[
\text{client(init : int)} = \text{go}^{\Delta 1} \text{travelshop} \rightarrow \text{flight?(price : int)} \\
\rightarrow \text{go}^{\Delta 1} \text{home} \rightarrow \text{client(init - price)};
\]

\[
\text{agent(balance : int)} = \text{data?(price : int)} \rightarrow \text{go}^{\Delta 1} \text{travelshop} \\
\rightarrow \text{flight!(price)} \rightarrow \text{agent(balance + amount)};
\]

\[
\text{query} = \text{data!(100)} \rightarrow \text{query};
\]
<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$\top \leadsto (\text{init}<em>{\text{client}} = 70 \land \text{balance}</em>{\text{agent}} = 170)$</td>
<td>The balance of client is 70 and the balance of agent is 170.</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$\top \leadsto \text{client}@\text{bank}$</td>
<td>client is able to arrive at bank.</td>
</tr>
<tr>
<td>$BL_1$</td>
<td>$R_1 \uplus t_{\text{client}} \leq 10$</td>
<td>$R_1$ is satisfied within 10 ticks for client.</td>
</tr>
<tr>
<td>$BL_2$</td>
<td>$R_2 \uplus t_{\text{client}} \leq 5$</td>
<td>$R_2$ is satisfied within 5 ticks for client.</td>
</tr>
<tr>
<td>$OR_1$</td>
<td>$R_1 \uplus \text{min}(t_{\text{client}})$</td>
<td>The shortest time of $R_1$ for client.</td>
</tr>
<tr>
<td>$OR_2$</td>
<td>$R_2 \uplus \text{min}(t_{\text{client}})$</td>
<td>The shortest time of $R_2$ for client.</td>
</tr>
<tr>
<td>$DF$</td>
<td>deadlock free</td>
<td>A deadlock state is undesired.</td>
</tr>
</tbody>
</table>
In order to allow a quantitative examination of behaviours, we add probabilities to TiMo, resulting in the new language pTiMo (probabilistic TiMo).

Accordingly, pTiMo models are no longer labelled transitions systems (LTSs), like in TiMo, but instead they are labelled discrete-time Markov chains (DTMCs).
Adding quantitative aspects: \text{pTiMo}

Typical properties:

\text{TiMo}: "can a given system reach a certain state before $t_1$ time steps have elapsed at location $l_1$?"

versus

\text{pTiMo}: "what is the probability that a given system reaches a certain state before $t_1$ time steps have elapsed at location $l_1$?"
pTiMo: a few issues

It is by no means straightforward to derive a probabilistic formalism based on TiMo, given that:

- the behaviour of TiMo models is best defined in terms of complete computational steps and active location selections

- there are several sources of non-determinism, associated with movement, communication, and location selection
Dealing with sources of non-determinism

To create pTiMo, we treat each source of non-determinism in the following manner:

1. split complete computational steps into:
   - a part containing only potential movements
   - a part containing only potential communications

2. define discrete probability distributions for each part, individually

3. define discrete probability distributions for location selections

4. combine the resulting probability distributions into joint distributions
Probabilistic logic PLTM for pTiMo

As a means of investigating the behaviour of pTiMo networks, we define a new logic, named PLTM (Probabilistic Logic for Timed Mobility).

PLTM includes features such as:

- properties for short-run and long-run behaviour
- explicit references to locations and processes
- temporal constraints over local clocks, both finite and infinite
- complex action guards over multisets of transitions (i.e., complete computational steps)
Examples of PLTM properties

Some properties which can be verified in PLTM:

- "with probability greater than 0.5, the process $P_1$ will communicate at location $l_1$, on channels $a_1$ or $a_3$, before 3 time steps have elapsed at location $l_1$, and 4 time steps have elapsed at location $l_2$"

- "the long-run probability that no movement occurs during a complete transition is less than 0.3"
A real-time extension of TiMo named rTiMo, uses real-time and explicit timeouts

It is useful for expressing certain temporal properties of multi-agent systems with migration and time constraints.

In rTiMo, the discrete transitions caused by performing actions with timeouts are alternated with continuous transitions.
rTiMo: a real-time extension of TiMo

We established a formal relationship between rTiMo and timed safety automata.

- allows the use of model checking capabilities provided by UPPAAL to verify several temporal properties of distributed networks with migrating and communicating processes described in rTiMo.
rTiMo: a real-time extension of TiMo

**THEOREM.** Given an rTiMo network $N$, there exists a timed safety automaton $A_N$ with a bisimilar behaviour. Formally, $N \sim A_N$.

**COROLLARY.** For an rTiMo network the reachability problem is decidable.
Concluding remarks

TiMo models distributed systems with time and mobility

- It has a simple syntax, but can model complex systems with respect to time and mobility.
- Timing constraints for migration and communication.
- Local clocks and maximal parallelism of actions.
- Security aspects expressed by dynamic access permissions (access permissions may be lost or gained by processes when moving).
- An operational semantics and formal results.
- A sound and complete system for safe communication and migration in open networks.
Concluding remarks

- pTiMo allows probabilistic behaviour of TiMo networks by solving the non-determinism involved in movement, communication and selection of active locations.

- PLTM: a probabilistic temporal logic for pTiMo
  - check properties with explicit reference to locations and processes
  - impose temporal constraints over local clocks (i.e., finite or infinite upper bounds, for each location independently)
  - define complex action guards over multisets of actions found in other logics

- A link between rTiMo and timed automata allows model checking by UPPAAL to verify temporal properties of distributed networks with migration and communication.


Bogdan Aman (IIT, Iasi)


Manchun Zheng (NUS, Singapore)

Thank You!