Tasks for Actors

Frank S. de Boer
Main Problem

Modeling and analysis of real-time distributed software systems
Main Approach

Executable modeling language for concurrent objects
Main Research Context

EU STREP Project Credo (FP6) on
Modeling and analysis of evolutionary structures in
distributed services

Coordinator: F.S. de Boer (CWI)
Start date: 1-9-2006
End date: 1-9-2009
Main partners (involved in this work)

- Einar Broch Johnsen (UIO)
- Wang Yi (UU)
- Mahdi Jaghouri (CWI)
Concurrent Objects

Model:
- Objects represent dedicated processors (in distributed systems)
- Objects interact via asynchronous message passing
- Objects create processes for handling each incoming message
- Objects synchronize their processes

Analysis:
- Formal semantics
- Maude implementation
  - Simulation
  - Testing
  - Model-Checking

Main challenge:

Behavioral interfaces for modeling and analysis of real-time scheduling policies for concurrent objects
A

No

- inter-object (return)
- intra-object (suspended processes)

synchronization
Technical Overview

- Timed Automata
- Task Automata
- Actors
- Tasks for Actors
- Conclusion
Timed Automata

Clocks  Real-valued
States  Delay:
  ▶ Invariant
Transitions  Instantaneous actions:
  ▶ Enabling condition
  ▶ Reset
Semantics Timed Automata

Configuration $\langle s, c \rangle$

- $s$: a state of the automaton
- $c$: clock assignment

Transitions:

**Delay** $\langle s, c \rangle \rightarrow^{\delta} \langle s, c + \delta \rangle$
provided $c + \delta \models I$

**Instantaneous Action** $\langle s, c \rangle \rightarrow^{a} \langle s', c[X := 0] \rangle$
provided $c \models e$

Timed Traces $(\delta_1, a_1), \ldots, (\delta_n, a_n), \ldots$
Analysis

Model-checking: Reduction to finite state-space
Task Automata

Extension of timed automata with dynamic task generation.

- Tasks are associated with states and specified by
  - worst and best execution times
  - deadlines
- Tasks are *scheduled by queuing*
  (e.g., *shortest deadline first*)
Operational semantics

Configuration $\langle s, c, q \rangle$

- $s$: a state of the automaton
- $c$: clock assignment
- $q$: task queue $(T, w, b, d)$
  - $w$: worst case execution time
  - $b$: best case execution time
  - $d$: deadline
Given a transition $s \xrightarrow{a} s'$ with $L(s') = T(w, b, d)$ we have

$$\langle s, c, (T_1, w_1, b_1, d_1), \ldots, (T_n, w_n, b_n, d_n) \rangle \xrightarrow{a} \langle s', c', (T_1, w_1, b_1, d_1), \ldots, (T, w, b, d), \ldots, (T_n, w_n, b_n, d_n) \rangle$$
Delay

\[
\langle s, c, (T_1, w_1, b_1, d_1), \ldots, (T_n, w_n, b_n, d_n) \rangle \rightarrow_{\delta} \langle s, c', (T_1, w'_1, b'_1, d'_1), \ldots, (T_n, w_n, b_n, d'_n) \rangle
\]

where

- \(w'_1 = w_1 - \delta\)
- \(b'_1 = b_1 - \delta\)
- \(d'_i = d_i - \delta\)
- \(c' = c + \delta\)

Termination condition: \(b_1 \leq 0\).
Schedulability Analysis

Schedulability analysis = Reachability analysis
Results

Note: Upperbound of the queue $= \sum d_i/w_i$

- Non-preemptive scheduling is decidable
- Scheduling is decidable for fixed execution times
- Schedulability in general is undecidable
Actors

Semantics of message handlers $m = S$:

Internal Action $\langle S, q \rangle \xrightarrow{\tau} \langle S', q \rangle$

Output $\langle m; S, q \rangle \xrightarrow{m} \langle S, q \rangle$

Input Enabledness $\langle S, q \rangle \xrightarrow{m} \langle S, q \cdot m \rangle$

Message Handling $\langle \text{nil}, m \cdot q \rangle \xrightarrow{\tau} \langle S_m, q \rangle$

Interleaving $A \xrightarrow{\tau} A' \quad \ldots, A, \ldots \rightarrow \ldots, A', \ldots$

Communication $\quad \begin{array}{c} A \xrightarrow{m} A', \quad B \xrightarrow{m} B' \\ \ldots, A, B, \ldots \rightarrow \ldots, A', B' \ldots \end{array}$
Extending Actors with Task Scheduling

- Timed automata specifications $T_m$ of message handlers (output actions: $m(d)$)
- Scheduling (e.g., shortest deadline first)
Schedulability Analysis

Analysis of a single actor wrt a timed automaton specification $D$ (driver) of the environment
(input actions: $m(d)$)
Operational Model

States \( \langle s, s', c, (T_1, c_1, d_1), \ldots, (T_n, c_n, d_n) \rangle \)

- \( s \) in Driver
- \( s' \) in \( T_1 \)
- \( c \): clock assignment
- \( c_i \leq d_i \)

Transitions

- Interleaving of instantaneous (input and output) actions
- Synchronization on delay
Summary

Construction of the Task Automaton:

\[ T_{m_1}, \ldots, T_{m_n}, D \Rightarrow T_A \]

where

- \( T_{m_i} \): TA of method \( m_i \) of actor \( A \)
- \( D \): Driver
Modular Analysis: Design by Contract

Possible use Driver $D$

Actual use Use case $U$

Compatibility by refinement (trace inclusion):

$$U \preceq D$$

Verification by deadlock analysis of synchronous product:

$$U \parallel D$$

(assuming $D$ is deterministic)
Conformance Testing

Conformance by refinement (trace inclusion):

\[ S \sqsubseteq \Pi_A D_A \]

Falsification:

\[ \text{Traces}(S) \setminus \text{Traces}(\Pi_A D_A) \neq \emptyset \]

Test case

\[ (t_1, R_1), \ldots, (t_n, R_n) \]

- \( t_i \): Transition in \( \Pi_A D_A \)
- \( R_i \): Alternative transitions (in \( \Pi_A D_A \))

A deadlock in the synchronous product \( T \parallel S \) generates a counter-example
What Next?

- Application to the ASK system (Almende)
- Actors2Objects (synchronization)
- Real-time extension of concurrent objects
- Software Families: EU FET IP HATS project on *Highly Adaptable and Trustworthy Software Using Formal Models*
- Distributed Implementation: Objective C
References

▶ Credo: http://credo.cwi.nl.

▶ E. B. Johnsen and O. Owe. 
An Asynchronous Communication Model for Distributed Concurrent Objects. 
Software and Systems Modeling.

▶ E. Fersman, P. Krcal, P. Pettersson, and W. Yi. 
Task automata: Schedulability, decidability and undecidability. 
Information and Computation.

Schedulability of asynchronous real-time concurrent objects. 
Journal of Logic and Algebraic Programming.

▶ F.S. de Boer, T. Chothia and M. M. Jaghoori. 
Modular Schedulability Analysis of Concurrent Objects in Creol. 
FSEN 2009, LNCS.