Specification and Verification of Trustworthy Component-Based Real-Time Reactive Systems

Authors:
Vasu Alagar and Mubarak Mohammad
Concordia University
Montréal, Canada

Presented by:
Mubarak Mohammad
Agenda

- Context
- Motivation
- Contributions:
  - A formal methodology for developing trustworthy RTRS
  - Automatic generation of component behavior
- Modeling Checking
- Example
- Conclusion
Real-Time Reactive Systems (RTRS)
Trustworthiness

- A *trustworthy* system is a system that can be depended upon for quality of service.

- RTRS are required to be trustworthy due to:
  - Their non-terminating behavior
  - The critical contexts it operate in

- In order to trust, the *credentials* of trust should be defined and examined:
  - Safety
  - Security
Component-Based Development (CBD)

- **Advantages** [1]:
  - Reusability
  - Managing design complexity
  - Reducing time and effort
  - Increasing productivity

- **Trustworthy component**: a component that guarantees safe and secure interactions.

Motivation

- The design of RTRS should rely on rigorous formal model to be formally verifiable.

- Provide a formal approach for the development of trustworthy component-based RTRS.
Formal Methodology

- Verification-oriented design methodology that involves:
  1. Formal specification of component structure and functional/nonfunctional (trustworthiness) properties [2];
  2. Automatic generation of component behavior; and
  3. Verification of functional/nonfunctional component behavior using model checking.

Composite Component Template

Component Template

Structure
- Data Parameters
- Services
- Interface Types
- Frame
- Architecture Types
- Connector Types

Contract
- Data Constraints
- Data Security
- Service Security
- User Identity
- Reactivity
- Time Constraints

Composition

Structural Composition

Contract Composition

Automatic Generation of Behavior
- Timed Automata
- RT PROMELA
- Others...

Model Checking
- UPPAAL
- KRONOS
- SPIN
- Others...

Properties
- Safety
- Data Security
- Liveness
- Service Security
- Data flow Security
UPPAAL Modeling Language [3]

- **Time Automata** ($L, l_0, K, A, E, I$)
  - $L$ is a set of locations denoting states;
  - $l_0$ is the initial location;
  - $K$ is a set of clocks;
  - $A$ is a set of actions, events causing transitions;
  - $E$ is a set of edges, transition specifications; and
  - $I$ is a function assigning clock constraints to locations as invariants.

Transformation Rules

Component Template

Structure
- Services
- Data Parameters
- Interface Types, Frame, Architecture Types, and Connector Types

Contract
- Data Constraints
- Data Security
- Service Security
- Reactivity
- Time Constraints

UPPAAL Template
- Locations (L)
- Actions (A)
- Edges (E)
- Expressions: 1-Select 2-Guard 3-Sync 4-Update
- Invariants (I)
- Clocks (K)

Create a location for every request for service
Create an action for every request for service or request from service
Create an edge for every request for service or request from service
Set values of parameters in the Update expression
Create an edge for every response from the service
Create an invariant for every time constraint
Create a clock for every time constraint
Model Checking

```plaintext
A[] forall (i : int[1,2]) LM.user==i => QuantityParameter==0 imply DataSecurity(i,1)==true
A[] forall (i : int[1,2]) C.user==i & C.switchOFF imply EventSecurity(i,1)==true
A[] C.user==2 imply not C.switchOFF
E<>C.switchOFF
A[] C.openValve imply quantity>=Max
A[] C.openPump imply quantity<Min
C.controlLevel & quantity<=Min --> quantity<Min & quantity<Max
```
Example

Events = \{e_1: Stimulus, e_2: Response, e_3: Request\},

Data Parameters(e_1) = \{d: Int\},

Reactivity(e_1) = \{e_2, e_3\},

Data Constraint(e_1, e_2): d > 10,

Data Constraint(e_1, e_3): d \leq 10,

Time Constraint(e_1, e_2) = [0, 5],

Time Constraint(e_1, e_3) = [0, 5]

Select: x: int
Guard: EventSecurity(user, e_1)
Sync: e_1?
Update: c_1 := 0,
\[ d = (DataSecurity(user, d)? x: Null) \]
Invariant: c_1 \leq 5

Guard: d > 10 && EventSecurity(user, e_2)
Sync: e_2!

Guard: d \leq 10 && EventSecurity(user, e_3)
Sync: e_3!
Conclusion

- We plan to evaluate our method on problems from different domains where safety and security are critical.

- We are investigating the requirements of a trustworthy ADL.

- We are building a visual interface tool for designing trustworthy RTRS.