Formal Verification of Device State Chart Models

Original

Availability:
This version is available at: 11583/2402456 since:

Publisher:
IEEE Computer Society

Published
DOI:10.1109/IE.2011.36

Terms of use:
openAccess
This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)
Formal Verification of Device State Chart Models

Fulvio Corno, Muhammad Sanaullah
Outline

- Design process
- Formalisms
- Verification Methodology
- Results
- Conclusions
Design process for IE

- Intelligent environments gaining acceptance
  - More installations
  - Standard solutions
- Need more structured design process
  - Less “art”
  - More “engineering”
Reference model

- Wall switch
- Tangible
- PC
- Smartphone

User Interface

- Agents
- Fuzzy
- Rules
- Algorithm

Intelligence

- Access point
- Protocols
- Gateway
- Model
- Framework

Middleware

- Sensor
- Meter
- Actuator
- Bus
- Wearable
- Wireless

Devices
General Goals

- Adopt **formal representations** to allow a sound design process
- Enable validation and verification **throughout** the design process
- Integrate the solution in the Dog2.1 gateway toolset
## Adopted formalisms

<table>
<thead>
<tr>
<th>Level</th>
<th>Design artifact</th>
<th>Technique</th>
<th>Formalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>System requirements</td>
<td>Temporal Logics</td>
<td>UCTL</td>
</tr>
<tr>
<td>Intelligence</td>
<td>Intelligent algorithms</td>
<td>State machines</td>
<td>UML Statecharts</td>
</tr>
<tr>
<td>Middleware</td>
<td>Device categories</td>
<td>Ontology</td>
<td>DogOnt classes</td>
</tr>
<tr>
<td></td>
<td>System configuration</td>
<td>Ontology</td>
<td>DogOnt instances</td>
</tr>
<tr>
<td>Devices</td>
<td>Device models</td>
<td>State machines</td>
<td>UML Statecharts</td>
</tr>
<tr>
<td></td>
<td>Whole system behavior</td>
<td>Parallel state machines</td>
<td>UML Statecharts</td>
</tr>
</tbody>
</table>
The DogOnt ontology

Formalism
- UCTL
- UML Statecharts
- DogOnt classes
- DogOnt instances
- UML Statecharts
DogOnt instances: DimmerLamp

Formalism
- UCTL
- UML Statecharts
- DogOnt classes
- DogOnt instances
- UML Statecharts
- UML Statecharts
Overall system components

…to be continued…
Device modeling

- Ontologies are declarative formalisms: device properties
- For device behavior we need an operational formalism
  - Statecharts (Harel, 1987, now in UML 2.0)
Use cases

- Ontologies are declarative formalisms: device properties
- For device behavior we need an operational formalism
  - Statecharts (Harel, 1987, now in UML 2.0)
- We use Statecharts for
  - Modeling the behavior of each device type
  - Implementing the *Intelligent Algorithms* within the gateway
  - Building a whole-system model allowing simulation and emulation
- Statecharts have a formal semantics: formal verification is possible
Overall system components

...to be continued...
Overall system components

IE'2011, Nottingham UK

Formal Verification
Temporal logic

- **UCTL logic**
  - Branching-time
  - State-based and action-based

- **Operators**
  - Next (X,N)
  - Future (F)
  - Globally (G)
  - All (A)
  - Exists (E)
  - Until (U)

- **UMC Model Checker**
  - Supports Statecharts as a model

**Examples**

\[ AG[\text{openRequest}(T1)] \]
\[ A[\top \{\neg \text{openRequest}(T1)\} U \{\text{tsDone}(T1)\}] \top \]

\[ AG[\text{daDoorOpen}(DAExt)] \]
\[ A[\top \{\neg \text{daDoorOpen}(DAInner)\} U \{\text{extDoorClosed()}\}] \top \]
Overall system components

- System requirements
- DogOnt
- Intelligent Algorithms
- Real devices
- Gateway
- Sense & Control
- Load model
- System Configuration
- Whole Environment Model
- Whole System Model
- Device Statechart
- Composition
- Emulation
- Formal Verification Simulation
- Internal policies
But... (goal of this paper)

- Formal verification relies on the composition of device state charts
- Environment control relies on information in DogOnt device properties
- How to ensure their consistency?
- Solution: use formal verification, too
The problem
The problem

- Naming consistency for states
- Naming consistency for commands
- Naming consistency for notifications
- Acceptance of commands
- Reachability of declared states
- Generation of declared notification
- Range of numeric status variables
Approach

- From DogOnt, extract UCTL properties
- From DogOnt, build a synthetic environment for the device
- Integrate Device State Chart in the synthetic environment
- For every property
  - Run Model checker

Formal Verification

IE’2011, Nottingham UK

19
Approach

- From DogOnt, extract UCTL properties
- From DogOnt, build a synthetic environment for the device
- Integrate Device State Chart in the synthetic environment
- For every property
  - Run Model checker

Formal Verification
IE'2011, Nottingham UK

Building a closed system model, ready for verification

Device Statechart
Closed system model
Model Checking
OK
ERR

Environment
Device
Environment Generate Commands (EGC)
Environment Receive Notifications (ERN)
Approach

Example: DimmerLamp generated & verified properties

-- Action Properties
  -- the acceptance of all the commands in DSC
  EF {sending(stepDown)} true
  EF {sending(stepUp)} true
  EF {sending(set)} true
  EF {sending(off)} true
  EF {sending(on)} true

-- State Properties
  -- the reachability of all the states in DSC
  EF (offState)
  EF (onState)
  EF (LightIntensityState)
Experimental Results

- UCTL Model Checker
- Dog2.1 standard device classes
- Device classes verified: 11
- Number of verifies properties: 114
  - Some design errors found and corrected
- CPU time: < 1 sec / property

- Formally validated device statechart library in Dog2.1
Conclusions

- Engineering the Design Process for Intelligent Environments
- Formalisms and tools are needed
- Ontologies, Statecharts, Temporal Logics

http://elite.polito.it
http://domoticdog.sourceforge.net
fulvio.corno@polito.it