Transforming Live Sequence Charts into Statecharts

Riandi Wiguna
rian.wiguna@mail.mcgill.ca
School of Computer Science
McGill University
April 13, 2004
Overview

1. Recap of Play-In/Play-Out
2. Recap of LSCs (Live Sequence Charts)
3. LSC Specification in AToM3
4. LSC to Statechart Transformation
5. Transformed Microwave Functions
6. DChart Demo
Recap of Play-In/Play-Out

The Play-In/Play-Out Approach is a way to easily generate and test LSCs (Live Sequence Charts). LSCs model all desired system reactions, providing a complete design for the system.

A full LSC specification of a system can be transformed into statecharts.
Recap of Play-In/Play-Out

This presentation is based off “Synthesizing State-Based Object Systems from LSC Specifications” by David Harel and Hillel Kugler, with additional information from:

1. “DCharts, a Formalism for Modeling and Simulation Based Design for Reactive Software Systems” by Thomas Huning Feng

2. “Can Behavioral Requirements be Executed? (And why would we want to do so?)” by David Harel

3. “Specifying and Executing Behavioral Requirements: The Play-In/Play-Out Approach” by David Harel and Rami Marelly
Recap of LSCs

- Modified MSCs (Message Sequence Charts)

- Two different kinds of LSCs:
  - Universal (Solid Border)
    - Model system reactions that *must* happen
    - Pre-Chart is condition for main chart actions
    - Exiting these prematurely causes a system error/crash
    - Drive system execution during Play-Out
  - Existential (Dashed Border)
    - Model system reactions that *may* happen
    - Must be able to run to completion in at least one scenario
    - Monitored during Play-Out

(from pgs. 4-6 of “Specifying and Executing...”)
Universal Chart

if ( )
then {

}

else if ( ) {


Temp:=Obj1.val + Obj2.val

Assignment

Hot Message

Obj3method()

Condition (system error on False)

if ( )
then {
}

else if ( ) {

}

Condition=True

Pre-Chart

Obj4method()
**LSC to Statechart Transformation**

- **Goal of this example:** transform LSC for the 'Popcorn' button into language of Statecharts
- **We'll use multiple Statecharts**
Note transition 3: “/Power->POP_ACTIVE” starts chain of object notification
LSC to Statechart Transformation

1. Create one statechart for each unique object in Universal LSC
2. For each statechart:
   1. Create default state
   2. Create one state for each action requiring the object
   3. Chain states together with transitions.
   4. Create one transition from state at end of chain to default state
   5. Label transitions with above actions and “ACTIVE” notification
LSC to Statechart Transformation

3. Use orthogonal components if object is in more than one Universal LSC*

4. Check Bad\(\max\) \(\text{Bad}\), set of all supercuts without successors or that lead to those without successors*

*We didn't do these in the example
Transformed Microwave Functions

LSCs:
1. Add1Min
2. Clear
3. Popcorn
4. Defrost
5. Start
6. Stop
7. OpenDoorWhileOven
   Active

Statecharts:
1. +1Min.
2. Clear
3. Popcorn
4. Defrost
5. Start
6. Stop
7. Power
8. Door
9. Timer
10. Oven
Results of Transformation

In the following statecharts, please assume all states before “ACTIVE” notices have transitions to the default states.
Add1Min

Reddenbacher

+1 Min.

Timer

Click()

timerAdd(min=True, 1)

updateTimer()
+1Min.

Add1Min:

/Timer->timerAdd
(min=True, 1)

Click()
/Timer->1M_ACTIVE

Clear:

Click()
/Timer->CLEAR_ACTIVE
Reddenbacher

Start
Door
Timer
Oven

Click()

Door.open=False AND Timer.counter! = 0 AND Oven.active=False

ovenStart()

startCountdown()
Start

Click()

[Door->getOpen() == False]

/Oven->ovenStart()

[Timer->getCounter() != 0]

[Oven->getActive() == False]

[Door->START_ACTIVE]

Stop

Click()

[Door->getOpen() == False]

/Oven->ovenStop()

[Timer->getCounter() != 0]

[Oven->getActive() == True]

[Door->STOP_ACTIVE]
OpenDoorWhileOvenActive

Reddenbacher

Door

Timer

Oven

Door.open=False AND Oven.active=True

ovenStop() -> stopCountdown()
Demonstration & Questions

Final Questions?
References


2. Harel, David. “Can Behavioral Requirements be Executed? (And why would we want to do so?)”

3. Harel, David and Hillel Kugler. “Synthesizing State-Based Object Systems from LSC Specifications”.