Softwaretechnik / Software-Engineering

Lecture 10: Live Sequence Charts & RE Wrap-Up

2018-06-11

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

Topic Area Requirements Engineering: Content

• Introduction
• Requirements Specification
• Desired Properties
• Kinds of Requirements
• Analysis Techniques
• Documents
• Dictionary, Specification
• Specification Languages
• Natural Language
• Decision Tables
• Syntax, Semantics
• Completeness, Consistency, ...
• Scenarios
• User Stories, Use Cases
• Live Sequence Charts
• Syntax, Semantics
• Wrap-Up

VL 6...
VL 7...
VL 8...
VL 9...
VL 10...

Live Sequence Charts: TBA Construction

LSC Semantics: TBA Construction

Requirements Engineering Wrap-Up

• Requirements Analysis in a Nutshell
• Recall: Validation by Translation

Outlook: Formal Methods in Design & QA

Subject: Requirements Engineering: Course
In the following, we "only" need to construct the transitions' labels:

\[ q, \psi \vdash \text{progress transitions} \]

...
light up
on
dispenser
buttons
switched
prepare
user
e inserts 1
notify
user
false
•
\[ \begin{array}{c}
\tau_\text{E1} \\
\tau_i
\end{array} \]
\[ \begin{array}{c}
V,U \\
\text{off} \\
\text{on}
\end{array} \]
\[ \begin{array}{c}
\text{action} \\
\sigma \\
\cdot \\
\cdot \\
\cdot
\end{array} \]
\[ \begin{array}{c}
\text{event} \\
\text{state} \\
\text{configuration} \\
\text{interpretation}
\end{array} \]
LCSs vs. Software (or Systems)
A software $S$ is called compatible with LSC $L$ over $C$ and $E$ if and only if

• $\Sigma = (C \rightarrow B)$, $C \subseteq C$, i.e. the states comprise valuations of the conditions in $C$.

• $A = (B \rightarrow B)$, $E \subseteq B$, i.e. the events comprise valuations of $E_{i,j}$, $E_{i,j}$?

A computation path $\pi = \sigma_0 \alpha_1 \rightarrow \sigma_1 \alpha_2 \rightarrow \sigma_2 \cdot \cdot \cdot \in \llbracket S \rrbracket$ of software $S$ induces the word $w(\pi) = (\sigma_0 \cup \alpha_1)$, $(\sigma_1 \cup \alpha_2)$, $(\sigma_2 \cup \alpha_3)$, ..., we use $W_S$ to denote the set of words induced by $\llbracket S \rrbracket$, i.e.

$$W_S = \{ w(\pi) | \pi \in \llbracket S \rrbracket \}.$$
We say software which is Software Satisfies LSC

Requirements Engineering Wrap-Up

LSCs in Quality Assurance

into universal LSCs (requirements)

existential LSCs (scenarios)

Full LSC with

Activation Condition & Activation Mode

• Software and Software Specification, formally

LSCs vs. Software

Content
are the minimal (or $C$ and $M$) cuts of pre- and main-chart.

$\Theta$ (existential chart mode $MC(B(Lang \in + 2)')$).

$L$ (permissive is $LSC: buy\ water AC:$ true $\amalg$ invariant $\in\{am,ac\}$)

$\forall N \in m \exists W \in \exists (\\sim \Langle LSC: buy\ water AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\exists W \in \\sim \Langle LSC: power-on self\ test AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\forall W \in \exists \Langle LSC: power-on self\ test AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\exists W \in \exists (\\sim \Langle LSC: buy\ softdrink AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\forall W \in \exists \Langle LSC: buy\ softdrink AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\exists W \in \exists (\\sim \Langle LSC: buy\ water AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\forall W \in \exists \Langle LSC: buy\ water AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\exists W \in \exists (\\sim \Langle LSC: power-on self\ test AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\forall W \in \exists \Langle LSC: power-on self\ test AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\exists W \in \exists (\\sim \Langle LSC: buy\ softdrink AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\forall W \in \exists \Langle LSC: buy\ softdrink AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\exists W \in \exists (\\sim \Langle LSC: buy\ water AC:$ true $\amalg$ invariant $\in\{am,ac\}$

$\forall W \in \exists \Langle LSC: buy\ water AC:$ true $\amalg$ invariant $\in\{am,ac\}$
Example: vending Machine

- Requirement: Buy Water
  - LSC: buy water
  - AC: true
  - AM: invariant
  - I: strict

User

CoinValidator

ChoicePanel

Dispenser

C

50

pWATER

¬

(C50 ∨ E1 ∨ pSOFT ∨ pTEA ∨ pFILLUP)

water in stock

dWATER OK

¬

(dSoft ∨ dTEA)

We (only) accept the software if,

(i) Whenever we insert 0.50 €,
(ii) and press the 'water' button (and no other button),
(iii) and there is water in stock,
(iv) then we get water (and nothing else).

- Negative scenario: A Drink for Free
  - LSC: only one drink
  - AC: true
  - AM: invariant
  - I: permissive

User

Vend. Ma.

E

1

pSOFT

SOFT

¬

C50 ∧ ¬ E1

false

We don't accept the software if it is possible to get a drink for free.

(i) Insert one 1 euro coin.
(ii) Press the 'softdrink' button.
(iii) Do not insert any more money.
(iv) Get two softdrinks.

- Live Sequence Charts
- TBA Construction
- LSCs vs. Software
- Software and Software Specification, formally
- Software satisfies Software Specification
- Full LSC (without pre-chart)
- Activation Condition & Activation Mode
- (Slightly) Advanced LSC Topics
- Full LSC with pre-chart
- LSCs in Requirements Engineering
- strengthening existential LSCs (scenarios)
- into universal LSCs (requirements)
- LSCs in Quality Assurance
- Requirements Engineering Wrap-Up
- Requirements Analysis in a Nutshell
- Recall: Validation by Translation
- Outlook: Formal Methods in Design & QA
How to Prove that a Software Satisfies an LSC?

- LSC: buy softdrink
  - AC: true
  - AM: invariant
  - User
  - Vend. Mach.
  - E

- LSC: get change
  - AC: true
  - AM: invariant
  - User
  - Vend. Mach.
  - C

Software $S$ satisfies existential LSC $L$ if there exists $\pi \in [S]$ such that $L$ accepts $w(\pi)$. Prove $S | = L$ by demonstrating $\pi$.

Note: Existential LSCs may hint at test-cases for the acceptance test!

Pushing Things Even Further (Harel and Marelly, 2003)
The new software needs to be a 1:1 re-implementation of the old software; if not properly specified, the new software may need to adhere to requirements of the old software. To communicate with your developers, re-use needs to be based on re-reading the code without specification. If specification is not present, re-use needs to be based on reading the code. Every observation is a feature.

Two broad directions: teach formalism and prepare for use.

**Method**

- **VL 10...**
- **VL 8...**
- **VL 7...**
- **VL 6...**
- **VL 5...**
- **VL 4...**
- **VL 3...**
- **VL 2...**
- **VL 1...**

**Syntax, Semantics**

- **LSC: buy water**
- **LSC: start ventilation**
- **LSC: stop ventilation**

**Introduction**

- **Requirements Engineering Wrap-Up**
- **Definition:**

- **TBA**

**Kinds of Requirements**

- **Syntax, Semantics**
- **Dictionary, Specification**
- **Universal**
- **Existential**
- **Induced by a computation path of**

**Requirements Specification**

- **Instance lines, messages, conditions, local invariants; mode: hot or cold.**
- **Live Sequence Charts**
- **LTSA**
- **LTL**
- **PDL**
- **TCTL**
- **ECTL**
- **Introduction**
- **Analysis Techniques**
- **Requirements Engineering Wrap-Up**

**Tell Them What You’ve Told Them. . .**
Example: Software Specification

\[ S_1 = (M.C \mid \mid \mid C.M \mid \mid \mid M.C) \]

\[ S_2 = (M.C) \] \text{ or } \( (C.M) \)

\[ S_3 = (C.M) \]

References

