Lecture 13: Behavioural Software Modelling

2016-06-27

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
### Introduction and Vocabulary

### Principles of Design
- modularity
- separation of concerns
- information hiding and data encapsulation
- abstract data types, object orientation

### Software Modelling
- views and viewpoints, the 4+1 view
- model-driven/-based software engineering
- Unified Modelling Language (UML)
- **modelling structure**
  - class diagrams
  - object diagrams
  - object constraint logic (OCL)
- **modelling behaviour**
  - communicating finite automata
  - Uppaal query language
  - implementing CFA
  - an outlook on **UML State Machines**

### Design Patterns

### Testing: Introduction
Content

- **Communicating Finite Automata (CFA)**
  - concrete and abstract syntax,
  - networks of CFA,
  - operational semantics.

- **Transition Sequences**

- **Deadlock, Reachability**

- **Uppaal**
  - tool demo (simulator),
  - query language,
  - CFA model-checking.

- **CFA at Work**
  - drive to configuration,
  - scenarios,
  - invariants,
  - tool demo (verifier).

- **CFA vs. Software**
Communicating Finite Automata

presentation follows (Olderog and Dierks, 2008)
Channel Names and Actions

To define communicating finite automata, we need the following sets of symbols:

- A set \((a, b \in)\) Chan of channel names or channels.
- For each channel \(a \in\) Chan, two visible actions: \(a?\) and \(a!\) denote input and output on the channel \((a?, a! \notin\) Chan).\n- \(\tau \notin\) Chan represents an internal action, not visible from outside.
- \((\alpha, \beta \in)\) Act := \(\{a? | a \in\) Chan\} \cup \{a! | a \in\) Chan\} \cup \{\tau\} is the set of actions.

- An alphabet \(B\) is a set of channels, i.e. \(B \subseteq\) Chan.
- For each alphabet \(B\), we define the corresponding action set

\[
B?! := \{a? | a \in B\} \cup \{a! | a \in B\} \cup \{\tau\}.
\]

Note: Chan?! = Act.
Let \((v, w \in V)\) be a set of ((finite domain) integer) variables. By \((\varphi \in \Psi(V))\) we denote the set of integer expressions over \(V\) using function symbols \(+, -, \ldots, >, \leq, \ldots\).

A modification on \(v\) is

\[
\text{update: } v := \varphi, \quad v \in V, \quad \varphi \in \Psi(V).
\]

By \(R(V)\) we denote the set of all modifications.

By \(\vec{r}\) we denote a finite list \(\langle r_1, \ldots, r_n \rangle, n \in \mathbb{N}_0\), of modifications \(r_i \in R(V)\). \(\langle \rangle\) is the empty list \((n = 0)\). (reset vector) or (update vector)

By \(R(V)^*\) we denote the set of all such finite lists of modifications.
Definition. A communicating finite automaton is a structure

\[ \mathcal{A} = (L, B, V, E, \ell_{ini}) \]

where

- \((\ell \in) L\) is a finite set of **locations** (or **control states**),
- \(B \subseteq \text{Chan}\),
- \(V\): a set of data variables,
- \(E \subseteq L \times B^? \times \Phi(V) \times R(V)^* \times L\): a finite set of **directed edges** such that
  \[(\ell, \alpha, \varphi, \vec{r}, \ell') \in E \land \text{chan}(\alpha) \in U \implies \varphi = true.\]

Edges \((\ell, \alpha, \varphi, \vec{r}, \ell')\) from location \(\ell\) to \(\ell'\) are labelled with an **action** \(\alpha\), a **guard** \(\varphi\), and a list \(\vec{r}\) of **modifications**.

- \(\ell_{ini}\) is the **initial location**.
Example

ChoicePanel:
(simplified)

\[(\text{idle}, \text{WATER?}, \text{water\_enabled}, \text{false}, \text{water\_selected}) \in E\]

\[\text{water\_selected} \in \mathcal{L}\]

\[\text{water\_enabled} \Rightarrow \text{WATER?}\]

\[\text{soft\_selected} \Rightarrow \text{SOFT?}\]

\[\text{tea\_selected} \Rightarrow \text{TEA?}\]

\[\text{half\_idle} \Rightarrow \text{DOK?} \Rightarrow \text{OK!}\]

\[\text{water\_enabled} := \text{false}, \text{soft\_enabled} := \text{false}, \text{tea\_enabled} := \text{false}\]

\[(\text{half\_idle}, \text{OK!}, \text{true}, \text{tea\_enabled} := \text{false}, \Rightarrow , \text{idle})\]
Definition.

Let $A_i = (L_i, B_i, V_i, E_i, \ell_{ini,i}), 1 \leq i \leq n$, be communicating finite automata.

The **operational semantics** of the **network** of FCA $C(A_1, \ldots, A_n)$ is the labelled transition system

$T(C(A_1, \ldots, A_n)) = (Conf, \text{Chan} \cup \{\tau\}, \{\overset{\lambda}{\rightarrow} | \lambda \in \text{Chan} \cup \{\tau\}\}, C_{ini})$

where

- $V = \bigcup_{i=1}^{n} V_i$,
- $Conf = \{\langle \ell, \nu \rangle | \ell_i \in L_i, \nu : V \rightarrow \mathcal{D}(V)\}$,
- $C_{ini} = \langle \ell_{ini}, \nu_{ini} \rangle$ with $\nu_{ini}(v) = 0$ for all $v \in V$.

The transition relation consists of transitions of the following two types.
• \( \nu : V \rightarrow \mathcal{D}(V) \) is a \textbf{valuation} of the variables,

• A valuation \( \nu \) of the variables canonically assigns an integer value \( \nu(\varphi) \) to each integer expression \( \varphi \in \Phi(V) \).

• \( \models \subseteq (V \rightarrow \mathcal{D}(V)) \times \Phi(V) \) is the canonical \textbf{satisfaction relation} between valuations and integer expressions from \( \Phi(V) \).

  \[ e.g. \quad \nu \models x > 10 \]

• **Effect of modification** \( r \in R(V) \) on \( \nu \), denoted by \( \nu[r] \):

  \[
  (\nu[v := \varphi])(a) := \begin{cases} 
  \nu(\varphi), & \text{if } a = v, \\
  \nu(a), & \text{otherwise}
  \end{cases}
  \]

  \( \nu : V \rightarrow \mathcal{D}(V) \)

• We set \( \nu[\langle r_1, \ldots, r_n \rangle] := \nu[r_1] \ldots [r_n] = (((\nu[r_1])[r_2]) \ldots)[r_n] \).

  That is, modifications are executed sequentially from left to right.
An internal transition \( \langle \vec{l}, \nu \rangle \xrightarrow{\tau} \langle \vec{l}', \nu' \rangle \) occurs if there is \( i \in \{1, \ldots, n\} \) and

- there is a \( \tau \)-edge \((\ell_i, \tau, \varphi, \vec{r}, \ell'_i) \in E_i\) such that
  - \( \nu \models \varphi \),
  - \( \vec{l}' = \vec{l}[\ell_i := \ell'_i] \),
  - \( \nu' = \nu[\vec{r}] \).

\[
\begin{align*}
\langle \vec{e}, \nu \rangle &= \langle (m, k), x = 8 \rangle \xrightarrow{\tau} \langle (n, k), x = 11 \rangle = \langle \vec{e}', \nu' \rangle \\
\langle \vec{e}, \nu \rangle &= \langle (m, k), x = 11 \rangle \xrightarrow{\tau} \langle (n, k), x = 27 \rangle
\end{align*}
\]
An internal transition \( \langle \vec{l}, \nu \rangle \xrightarrow{\tau} \langle \vec{l}', \nu' \rangle \) occurs if there is \( i \in \{1, \ldots, n\} \) and there is a \( \tau \)-edge \( (\ell_i, \tau, \varphi, \vec{r}, \ell'_i) \in E_i \) such that

- \( \nu \models \varphi \), \text{ "source valuation satisfies guard"}
- \( \vec{l}' = \vec{l}[\ell_i := \ell'_i] \), \text{ "automaton } i \text{ changes location"}
- \( \nu' = \nu[\vec{r}] \), \text{ "} \nu' \text{ is } \nu \text{ modified by } \vec{r} \text{"}

A synchronisation transition \( \langle \vec{l}, \nu \rangle \xrightarrow{b} \langle \vec{l}', \nu' \rangle \) occurs if there are \( i, j \in \{1, \ldots, n\} \) with \( i \neq j \) and there are edges \( (\ell_i, b!, \varphi_i, \vec{r}_i, \ell'_i) \in E_i \) and \( (\ell_j, b?, \varphi_j, \vec{r}_j, \ell'_j) \in E_j \) such that

- \( \nu \models \varphi_i \land \varphi_j \),
- \( \vec{l}' = \vec{l}[\ell_i := \ell'_i][\ell_j := \ell'_j] \),
- \( \nu' = (\nu[\vec{r}_i])[\vec{r}_j] \), \text{ "sender updates first"}

This style of communication is known under the names \text{"} \textbf{rendezvous}, \text{"} \textbf{synchronous}, \text{"} \textbf{blocking} \text{"} communication (and possibly many others).
A transition sequence of \( \mathcal{C}(A_1, \ldots, A_n) \) is any (in)finite sequence of the form

\[
\langle \vec{\ell}_0, \nu_0 \rangle \overset{\lambda_1}{\rightarrow} \langle \vec{\ell}_1, \nu_1 \rangle \overset{\lambda_2}{\rightarrow} \langle \vec{\ell}_2, \nu_2 \rangle \overset{\lambda_3}{\rightarrow} \ldots
\]

with

- \( \langle \vec{\ell}_0, \nu_0 \rangle = \mathcal{C}_{\text{ini}} \),

- for all \( i \in \mathbb{N} \), there is \( \overset{\lambda_{i+1}}{\rightarrow} \) in \( \mathcal{T}(\mathcal{C}(A_1, \ldots, A_n)) \) with \( \langle \vec{\ell}_i, \nu_i \rangle \overset{\lambda_{i+1}}{\rightarrow} \langle \vec{\ell}_{i+1}, \nu_{i+1} \rangle \).
Example

ChoicePanel: (simplified)

User:

$\langle (\text{idle}, 1), \frac{w}{s} = 1, e = 0 \rangle \xrightarrow{\text{WATER}} \langle (\text{water}, 1), \frac{w}{s} = 1, e = 0 \rangle \xrightarrow{\tau} \langle (\text{tea}, 1), \frac{u}{s} = 1, e = 0 \rangle \xrightarrow{\tau} \langle (\text{C50}, 1), \frac{u}{s} = 1, e = 0 \rangle$

* Note: \text{SOFT}

else: guard tea_enabled not satisfied, and no comm. partners for C50 or E1
Deadlock, Reachability

- A configuration $\langle \ell, \nu \rangle$ of $C(A_1, \ldots, A_n)$ is called **deadlock** if and only if there are no transitions from $\langle \ell, \nu \rangle$, i.e. if

$$\neg(\exists \lambda \in \Lambda \exists \langle \ell', \nu' \rangle \in \text{Conf} \bullet \langle \ell, \nu \rangle \xrightarrow{\lambda} \langle \ell', \nu' \rangle).$$

The network $C(A_1, \ldots, A_n)$ is said to **have a deadlock** if and only if there is a configuration $\langle \ell, \nu \rangle$ which is a deadlock.

- A configuration $\langle \vec{\ell}, \nu \rangle$ is called **reachable** (in $C(A_1, \ldots, A_n)$) if and only if there is a transition sequence of the form

$$\langle \vec{\ell}_0, \nu_0 \rangle \xrightarrow{\lambda_1} \langle \vec{\ell}_1, \nu_1 \rangle \xrightarrow{\lambda_2} \langle \vec{\ell}_2, \nu_2 \rangle \xrightarrow{\lambda_3} \cdots \xrightarrow{\lambda_n} \langle \vec{\ell}_n, \nu_n \rangle = \langle \vec{\ell}, \nu \rangle.$$

A location $\ell \in L_i$ is called **reachable** if and only if any configuration $\langle \vec{\ell}, \nu \rangle$ with $\ell_i = \ell$ is reachable, i.e. there exist $\vec{\ell}$ and $\nu$ such that $\ell_i = \ell$ and $\langle \vec{\ell}, \nu \rangle$ is reachable.
Uppaal

(Larsen et al., 1997; Behrmann et al., 2004)
Tool Demo
Consider $\mathcal{N} = C(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ over data variables $V$.

- **basic formula**:
  
  $$
  atom ::= \mathcal{A}_i.\ell \mid \varphi \mid \text{deadlock}
  $$

  where $\ell \in L_i$ is a location and $\varphi$ an expression over $V$.

- **configuration formulae**:

  $$
  term ::= atom \mid \text{not} \ term \mid term_1 \text{ and } term_2
  $$

- **existential path formulae**:

  $$
  e\text{-formula} ::= \exists \Diamond \ term \quad \text{(exists finally)}
  \quad \text{exists finally}
  \mid \exists \Box \ term \quad \text{(exists globally)}
  \quad \text{exists globally}
  $$

- **universal path formulae**:

  $$
  a\text{-formula} ::= \forall \Diamond \ term \quad \text{(always finally)}
  \quad \text{always finally}
  \mid \forall \Box \ term \quad \text{(always globally)}
  \quad \text{always globally}
  \mid term_1 \rightarrow term_2 \quad \text{(leads to)}
  $$

- **formulae (or queries)**:

  $$
  F ::= e\text{-formula} \mid a\text{-formula}
  $$
The satisfaction relation

\[ \langle \vec{\ell}, \nu \rangle \models F \]

between configurations

\[ \langle \vec{\ell}, \nu \rangle = \langle (\ell_1, \ldots, \ell_n), \nu \rangle \]

of a network \( C(A_1, \ldots, A_n) \) and formulae \( F \) of the Uppaal logic is defined inductively as follows:

- \( \langle \vec{\ell}, \nu \rangle \models \text{deadlock} \) iff \( \langle \vec{\ell}, \nu \rangle \) is a deadlock configuration
- \( \langle \vec{\ell}, \nu \rangle \models A_i.\ell \) iff \( \ell_{0,i} \ell_i = \ell \)
- \( \langle \vec{\ell}, \nu \rangle \models \varphi \) iff \( \nu \models \varphi \)
- \( \langle \vec{\ell}, \nu \rangle \models \text{not term} \) iff \( \langle \vec{\ell}, \nu \rangle \not\models \text{term} \)
- \( \langle \vec{\ell}, \nu \rangle \models \text{term}_1 \text{ and term}_2 \) iff \( \langle \vec{\ell}, \nu \rangle \models \text{term}_1 \) and \( \langle \vec{\ell}, \nu \rangle \models \text{term}_2 \)
Example: Computation Paths vs. Computation Tree

**ChoicePanel:**

```
idle → SOFT? → water_enabled
water_selected → TEA? → tea_enabled
tea_selected → request_sent → half_idle

water_enabled := false, soft_enabled := false, tea_enabled := false
```

**User:**

```
DOK?
OK!
water_enabled := false, soft_enabled := false, tea_enabled := false
DTEA!
DWATER!
DSOFT!
```

**WATER**

\[ \langle \text{water-selected, l}, \begin{array}{c}
we = 1 \\
se = 1 \\
te = 0
\end{array} \rangle \]

\[ \langle \text{request-sent, l}, \begin{array}{c}
we = 1 \\
se = 1 \\
te = 0
\end{array} \rangle \]

\[ \langle \text{half-idle, l}, \begin{array}{c}
we = 1 \\
se = 1 \\
te = 0
\end{array} \rangle \]

**SOFT**

\[ \langle \text{soft-selected, l}, \begin{array}{c}
we = 1 \\
se = 1 \\
te = 0
\end{array} \rangle \]

\[ \langle \text{request-sent, l}, \begin{array}{c}
we = 1 \\
se = 1 \\
te = 0
\end{array} \rangle \]

\[ \langle \text{half-idle, l}, \begin{array}{c}
we = 1 \\
se = 1 \\
te = 0
\end{array} \rangle \]
Example: Computation Paths vs. Computation Tree

ChoicePanel:

- idle
  - SOFT?
    - soft_enabled
      - tea_enabled
        - request_sent
      - soft_selected
        - request_sent
  - WATER?
    - water_enabled
      - water_selected
        - request_sent
    - request_sent

User:

- C50!
  - l
  - WATER!
  - E1!
  - SOFT!
  - TEA!

WATER
\[ \langle (\text{idle}, l), \begin{array}{c} \text{we = 1} \\ \text{se = 1} \\ \text{te = 0} \end{array} \rangle \]

SOFT
\[ \langle (\text{soft_selected}, l), \begin{array}{c} \text{we = 1} \\ \text{se = 1} \\ \text{te = 0} \end{array} \rangle \]

\[ \tau \]

\[ \langle (\text{request_sent}, l), \begin{array}{c} \text{we = 1} \\ \text{se = 1} \\ \text{te = 0} \end{array} \rangle \]

\[ \tau \]

\[ \langle (\text{half_idle}, l), \begin{array}{c} \text{we = 1} \\ \text{se = 1} \\ \text{te = 0} \end{array} \rangle \]
Example: Computation Paths vs. Computation Graph

ChoicePanel:

User:

\[ \langle (\text{water\_selected}, l), \begin{array}{c} we = 1 \\ se = 1 \\ te = 0 \end{array} \rangle \]

\[ \langle (\text{request\_sent}, l), \begin{array}{c} we = 1 \\ se = 1 \\ te = 0 \end{array} \rangle \]

\[ \langle (\text{half\_idle}, l), \begin{array}{c} we = 1 \\ se = 1 \\ te = 0 \end{array} \rangle \]

\[ \langle (\text{soft\_selected}, l), \begin{array}{c} we = 1 \\ se = 1 \\ te = 0 \end{array} \rangle \]

\[ \langle (\text{idle}, l), \begin{array}{c} we = 1 \\ se = 1 \\ te = 0 \end{array} \rangle \]
Exists finally:

- $\langle \vec{l}, \nu_0 \rangle \models \exists \Diamond \text{term}$

iff

$\exists \text{ path } \xi \text{ of } C \text{ starting in } \langle \vec{l}, \nu_0 \rangle$

$\exists i \in \mathbb{N}_0 \cdot \xi^i \models \text{term}$

"some configuration satisfying term is reachable"

Example: $\langle \vec{l}, \nu_0 \rangle \models \exists \Diamond \varphi$
Satisfaction of Uppaal Queries by Configurations

 Exists globally:

- \( \langle \vec{\ell}_0, \nu_0 \rangle \models \exists \square \text{term} \)

iff

\( \exists \) path \( \xi \) of \( C \) starting in \( \langle \vec{\ell}_0, \nu_0 \rangle \)

\( \forall i \in \mathbb{N}_0 \bullet \xi^i \models \text{term} \)

"on some computation path, all configurations satisfy term"

Example: \( \langle \vec{\ell}_0, \nu_0 \rangle \models \exists \square \varphi \)
Satisfaction of Uppaal Queries by Configurations

- **Always globally:**

  \[ \langle \vec{\ell}_0, \nu_0 \rangle \models \forall \Box \text{term} \quad \text{iff} \quad \langle \vec{\ell}_0, \nu_0 \rangle \not\models \exists \Diamond \neg \text{term} \]

  “not (some configuration satisfying \(\neg\text{term}\) is reachable)”
  
or: “all reachable configurations satisfy \text{term}”

- **Always finally:**

  \[ \langle \vec{\ell}_0, \nu_0 \rangle \models \forall \Diamond \text{term} \quad \text{iff} \quad \langle \vec{\ell}_0, \nu_0 \rangle \not\models \exists \Box \neg \text{term} \]

  “not (on some computation path, all configurations satisfy \(\neg\text{term}\))”
  
or: “on all computation paths, there is a configuration satisfying \text{term}”
Satisfaction of Uppaal Queries by Configurations

Leads to:

- \( \langle \vec{l}_0, \nu_0 \rangle \models term_1 \rightarrow term_2 \) iff 
  \( \forall \) path \( \xi \) of \( N \) starting in \( \langle \vec{l}_0, \nu_0 \rangle \) \( \forall i \in \mathbb{N}_0 \) •
  \( \xi^i \models term_1 \implies \xi^i \models \forall \Box term_2 \)

“on all paths, from each configuration satisfying \( term_1 \),
a configuration satisfying \( term_2 \) is reachable” (response pattern)

Example: \( \langle \vec{l}_0, \nu_0 \rangle \models \varphi_1 \rightarrow \varphi_2 \)

Diagram:
**Definition.** Let $\mathcal{N} = C(A_1, \ldots, A_n)$ be a network and $F$ a query.

(i) We say $\mathcal{N}$ satisfies $F$, denoted by $\mathcal{N} \models F$, if and only if $C_{ini} \models F$.

(ii) The model-checking problem for $\mathcal{N}$ and $F$ is to decide whether $\mathcal{N}, F \in \models$.

**Proposition.** The model-checking problem for communicating finite automata is **decidable**.
• **Communicating Finite Automata (CFA)**
  - concrete and abstract syntax,
  - networks of CFA,
  - operational semantics.

• **Transition Sequences**

• **Deadlock, Reachability**

• **Uppaal**
  - tool demo (simulator),
  - query language,
  - CFA model-checking.

• **CFA at Work**
  - drive to configuration,
  - scenarios,
  - invariants,
  - tool demo (verifier).

• **CFA vs. Software**
CFA and Queries at Work
**Model Architecture — Who Talks What to Whom**

- **Shared variables:**
  - `bool water_enabled, soft_enabled, tea_enabled;`
  - `int w = 3, s = 3, t = 3;`

- **Note:** Our model does not use scopes (“information hiding”) for channels. That is, ‘Service’ could send ‘WATER’ if the modeler wanted to.
Shared variables:

- bool water_enabled, soft_enabled, tea_enabled;
- int w = 3, s = 3, t = 3;

Note: Our model does not use scopes (“information hiding”) for channels. That is, ‘Service’ could send ‘WATER’ if the modeler wanted to.
• **Question:** Is it (at all) possible to have no water in the vending machine model? (Otherwise, the design is definitely broken.)

• **Approach:** Check whether a configuration satisfying

\[ w = 0 \]

is reachable, i.e. check

\[ \mathcal{N}_{VM} \models \exists \diamondsuit w = 0. \]

for the vending machine model \( \mathcal{N}_{VM} \).
References
References


