Softwaretechnik / Software-Engineering

Lecture 7: Formal Methods for Requirements Engineering

2017-05-29

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
  - Working Definition: Software
  - Live Sequence Charts
  - Syntax, Semantics
- Discussion
### (A Selection of) Analysis Techniques

<table>
<thead>
<tr>
<th>Analysis Technique</th>
<th>Focus</th>
<th>current situation</th>
<th>desired situation</th>
<th>innovation consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis of existing data and documents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Questioning with (closed/structured/open)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interview</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modelling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experiments</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototyping</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Participative development</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Ludewig and Lichter, 2013)

---

### Topic Area Requirements Engineering: Content

- **VL 6**
  - Introduction
  - Requirements Specification
    - Desired Properties
    - Kinds of Requirements
    - Analysis Techniques

- **VL 7**
  - Decision Tables
    - Syntax, Semantics
    - Completeness, Consistency,...

- **VL 8**
  - Scenarios
    - User Stories, Use Cases
  - Working Definition: Software
  - Live Sequence Charts
  - Syntax, Semantics

- **VL 9**
  - Discussion
Requirement Documents are important - e.g., for
- negotiation, design & implementation, documentation,
testing, delivery, re-use, re-implementation.

A Requirements Specification should be
- correct, complete, relevant, consistent, neutral, traceable, objective.
Note: vague vs. abstract.

Requirements Representations should be
- easily understandable, precise, easily maintainable, easily usable

Distinguish
- hard / soft,
- functional / non-functional,
- open tacit

It is the task of the analyst to elicit requirements.
- Natural language is inherently imprecise, counter-measures:
  - natural language patterns.
- Do not underestimate the value of a good dictionary.

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
- Working Definition: Software
  - Live Sequence Charts
  - Syntax, Semantics
- Discussion
Decision Tables

- (Basic) Decision Tables
- Syntax, Semantics
- ...for Requirements Specification
- ...for Requirements Analysis
- Completeness
- Useless Rules
- Determinism
- Domain Modelling
- Conflict Axiom
- Relative Completeness
- Vacuous Rules
- Conflict Relation
- Collecting Semantics
- Discussion

Logic
Decision Tables: Example

\[\begin{array}{ccc}
T & r_1 & r_2 & r_3 \\
\hline
c_1 & \times & \times & \ast \\
c_2 & \times & \ast & \ast \\
c_3 & \ast & \ast & \ast \\
a_1 & \ast & \ast & \ast \\
a_2 & \ast & \ast & \ast \\
\end{array}\]

Decision Table Syntax

- Let \(C\) be a set of conditions and \(A\) be a set of actions s.t. \(C \cap A = \emptyset\).
- A decision table \(T\) over \(C\) and \(A\) is a labelled \((m + k) \times n\) matrix.

\[
T: \text{decision table} \\
c_1: \text{description of condition } c_1 \quad v_{1,1} \quad \cdots \quad v_{1,n} \\
\vdots \quad \vdots \quad \ddots \quad \vdots \\
c_m: \text{description of condition } c_m \quad v_{m,1} \quad \cdots \quad v_{m,n} \\
a_1: \text{description of action } a_1 \quad w_{1,1} \quad \cdots \quad w_{1,n} \\
\vdots \quad \vdots \quad \ddots \quad \vdots \\
a_k: \text{description of action } a_k \quad w_{k,1} \quad \cdots \quad w_{k,n}
\]
• Let $C$ be a set of conditions and $A$ be a set of actions s.t. $C \cap A = \emptyset$.

• A decision table $T$ over $C$ and $A$ is a labelled $(m + k) \times n$ matrix 

<table>
<thead>
<tr>
<th>$T$: decision table</th>
<th>$r_1$</th>
<th>$\cdots$</th>
<th>$r_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$ description of condition $c_1$</td>
<td>$v_{1,1}$</td>
<td>$\cdots$</td>
<td>$v_{1,n}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$c_m$ description of condition $c_m$</td>
<td>$v_{m,1}$</td>
<td>$\cdots$</td>
<td>$v_{m,n}$</td>
</tr>
<tr>
<td>$a_1$ description of action $a_1$</td>
<td>$w_{1,1}$</td>
<td>$\cdots$</td>
<td>$w_{1,n}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$a_k$ description of action $a_k$</td>
<td>$w_{k,1}$</td>
<td>$\cdots$</td>
<td>$w_{k,n}$</td>
</tr>
</tbody>
</table>

- where
  - $c_1, \ldots, c_m \in C$,  
  - $a_1, \ldots, a_k \in A$,  
  - $v_{1,1}, \ldots, v_{m,n} \in \{-, \times, *\}$ and  
  - $w_{1,1}, \ldots, w_{k,n} \in \{-, \times\}$.

• Columns $(v_{1,i}, \ldots, v_{m,i}, w_{1,i}, \ldots, w_{k,i}), 1 \leq i \leq n$, are called rules.

• $r_1, \ldots, r_n$ are rule names.

• $(v_{1,i}, \ldots, v_{m,i})$ is called premise of rule $r_i$.

• $(w_{1,i}, \ldots, w_{k,i})$ is called effect of $r_i$.

### Decision Table Semantics

Each rule $r \in \{r_1, \ldots, r_n\}$ of table $T$

<table>
<thead>
<tr>
<th>$T$: decision table</th>
<th>$r_1$</th>
<th>$\cdots$</th>
<th>$r_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$ description of condition $c_1$</td>
<td>$v_{1,1}$</td>
<td>$\cdots$</td>
<td>$v_{1,n}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$c_m$ description of condition $c_m$</td>
<td>$v_{m,1}$</td>
<td>$\cdots$</td>
<td>$v_{m,n}$</td>
</tr>
<tr>
<td>$a_1$ description of action $a_1$</td>
<td>$w_{1,1}$</td>
<td>$\cdots$</td>
<td>$w_{1,n}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$a_k$ description of action $a_k$</td>
<td>$w_{k,1}$</td>
<td>$\cdots$</td>
<td>$w_{k,n}$</td>
</tr>
</tbody>
</table>

is assigned to a propositional logical formula $F(r)$ over signature $C \cup A$ as follows:

• Let $(v_1, \ldots, v_m)$ and $(w_1, \ldots, w_k)$ be premise and effect of $r$.

• Then

$$F(r) := \underbrace{F(v_1, c_1) \land \cdots \land F(v_m, c_m)}_{= F_{\text{pre}}(r)} \land \underbrace{F(w_1, a_1) \land \cdots \land F(w_k, a_k)}_{= F_{\text{eff}}(r)}$$

where

$$F(v, x) = \begin{cases} 
  x, & \text{if } v = \times \\
  \neg x, & \text{if } v = - \\
  \text{true}, & \text{if } v = * 
\end{cases}$$
### Decision Table Semantics: Example

\[ F(r) := F(v_1, c_1) \land \cdots \land F(v_m, c_m) \land F(v_1, a_1) \land \cdots \land F(v_k, a_k) \]

\[ F(v, x) = \begin{cases} x & \text{if } v = \times \\ \neg x & \text{if } v = - \\ \text{true} & \text{if } v = * \end{cases} \]

<table>
<thead>
<tr>
<th>( F )</th>
<th>( r_1 )</th>
<th>( r_2 )</th>
<th>( r_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( - )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( \times )</td>
<td>( - )</td>
<td>( * )</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>( - )</td>
<td>( - )</td>
<td>( * )</td>
</tr>
<tr>
<td>( a_1 )</td>
<td>( \times )</td>
<td>( - )</td>
<td>( - )</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>( - )</td>
<td>( - )</td>
<td>( - )</td>
</tr>
</tbody>
</table>

- \( F(r_1) = \neg x \land \neg x \land \neg c_3 \land \neg \neg a_3 \land \neg \neg a_2 \)
- \( F(r_2) = c_3 \land \neg c_3 \land c_2 \land a_1 \land \neg a_2 \)
- \( F(r_3) = \neg c_3 \land \neg a_3 \land \neg a_2 \land \neg c_3 \land \neg a_1 \land \neg a_2 \)

---

### Decision Tables as Requirements Specification
We can use decision tables to model (describe or prescribe) the behaviour of software!

**Example:**

Ventilation system of lecture hall 101-0-026.

- We can observe whether button is pressed and whether room ventilation is on or off, and whether (we intend to) start ventilation or stop ventilation.
- We can model our observation by a boolean valuation \( \sigma : C \cup A \rightarrow B \), e.g., set
  \[
  \sigma(b) := \text{true, if button pressed now and} \quad \sigma(b) := \text{false, if button not pressed now.}
  \]
  \[
  \sigma(go) := \text{true, we plan to start ventilation and} \quad \sigma(go) := \text{false, we plan to stop ventilation.}
  \]
- A valuation \( \sigma : C \cup A \rightarrow B \) can be used to assign a truth value to a propositional formula \( \varphi \) over \( C \cup A \).
  As usual, we write \( \sigma \models \varphi \) iff \( \varphi \) evaluates to true under \( \sigma \) (and \( \sigma \not\models \varphi \) otherwise).
- Rule formulae \( F(r) \) are propositional formulae over \( C \cup A \) thus, given \( \sigma \), we have either \( \sigma \models F(r) \) or \( \sigma \not\models F(r) \).
- Let \( \sigma \) be a model of an observation of \( C \) and \( A \).
  We say, \( \sigma \) is **allowed** by decision table \( T \) if and only if there exists a rule \( r \) in \( T \) such that \( \sigma \models F(r) \).

\[
T \text{: room ventilation}
\begin{array}{|c|c|c|}
\hline
b & r_1 & r_2 & r_3 \\
\hline
\text{button pressed?} & \times & \times & \times \\
\text{ventilation off?} & \times & \times & \times \\
\text{ventilation on?} & \times & \times & \times \\
\text{start ventilation} & \times & \times & \times \\
\text{stop ventilation} & \times & \times & \times \\
\hline
\end{array}
\]

\[
F(r_1) = b \land \text{off} \land \neg\text{on} \land go \land \neg stop
\]
\[
F(r_2) = b \land \neg\text{off} \land \text{on} \land \neg go \land stop
\]
\[
F(r_3) = \neg b \land \text{true} \land \text{true} \land \neg go \land \neg stop
\]

(i) **Assume**: button pressed, ventilation off, we (only) plan to start the ventilation.

\[
\sigma = \{ b \mapsto \text{true}, \text{off} \mapsto \text{false}, \text{on} \mapsto \text{false}, \text{go} \mapsto \text{false}, \text{stop} \mapsto \text{false} \}
\]

\[
\sigma \not\models F(r_3)
\]

\[
\sigma \models F(r_1)
\]

\[
\sigma \models F(r_2)
\]
### Example

<table>
<thead>
<tr>
<th>$T$: room ventilation</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>button pressed?</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>off</td>
<td>ventilation off?</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>on</td>
<td>ventilation on?</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>go</td>
<td>start ventilation</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>stop</td>
<td>stop ventilation</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
</tbody>
</table>

$F(r_1) = b \land \text{off} \land \neg \text{on} \land \text{go} \land \neg \text{stop}$

$F(r_2) = b \land \neg \text{off} \land \text{on} \land \neg \text{go} \land \text{stop}$

$F(r_3) = \neg b \land \text{true} \land \text{true} \land \neg \text{go} \land \neg \text{stop}$

(i) **Assume**: button pressed, ventilation off, we (only) plan to start the ventilation.

- Corresponding valuation: $\sigma_1 = \{ b \mapsto \text{true}, \text{off} \mapsto \text{true}, \text{on} \mapsto \text{false}, \text{start} \mapsto \text{true}, \text{stop} \mapsto \text{false} \}$.
- Is our intention (to start the ventilation now) allowed by $T$? Yes! (Because $\sigma_1 \models F(r_1)$)

(ii) **Assume**: button pressed, ventilation on, we (only) plan to stop the ventilation.

- Corresponding valuation: $\sigma_2 = \{ b \mapsto \text{true}, \text{off} \mapsto \text{false}, \text{on} \mapsto \text{true}, \text{start} \mapsto \text{false}, \text{stop} \mapsto \text{true} \}$.
- Is our intention (to stop the ventilation now) allowed by $T$? Yes. (Because $\sigma_2 \models F(r_2)$)

(iii) **Assume**: button not pressed, ventilation on, we (only) plan to stop the ventilation.

- Corresponding valuation:
- Is our intention (to stop the ventilation now) allowed by $T$? No!

---

### Decision Tables as Specification Language

- **Decision Tables** can be used to **objectively** describe desired software behaviour.

- **Example**: Dear developer, please provide a program such that
  - in each situation (button pressed, ventilation on/off),
  - whatever the software does (action start/stop)
  - is allowed by decision table $T$.

<table>
<thead>
<tr>
<th>$T$: room ventilation</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>button pressed?</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>off</td>
<td>ventilation off?</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>on</td>
<td>ventilation on?</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>go</td>
<td>start ventilation</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>stop</td>
<td>stop ventilation</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
</tbody>
</table>
• Decision Tables can be used to objectively describe desired software behaviour.

• Another Example: Customer session at the bank:

<table>
<thead>
<tr>
<th>T1: cash a cheque</th>
<th>r1</th>
<th>r2</th>
<th>else</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>c3</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>a1</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- clerk checks database state (yields σ for c1, ..., c3);
- database says: credit limit exceeded over 500 €, but payment history ok;
- clerk cashes cheque but offers new conditions (according to T1).

Decision Tables as Specification Language

Requirements on Requirements Specifications

- A requirements specification should be:
  - correct — it correctly represents the wishes/needs of the customer.
  - complete — all requirements (existing in somebody's head, or a document, or ...) should be present.
  - relevant — things which are not relevant to the project should not be constrained.
  - consistent, free of contradictions — each requirement is compatible with all other requirements; otherwise the requirements are not realisable.

- neutral, abstract — a requirements specification does not constrain the realisation more than necessary,
  - traceable, comprehensible — the sources of requirements are documented, requirements are uniquely identifiable.
  - testable, objective — the final product can objectively be checked for satisfying a requirement.

- correctness and completeness are defined relative to something which is usually only in the customer's head.
- it is difficult to be sure of correctness and completeness.

- "Dear customer, please tell me what is in your head!" is in almost all cases not a solution!
  - It's not unusual that even the customer does not precisely know ...
  - For example, the customer may not be aware of contradictions due to technical limitations.
... so, off to "technological paradise where [...] everything happens according to the blueprints".

(2012b, 2011b; Lovins and Lovins, 2012a)
Completeness

Definition. [Completeness] A decision table \( T \) is called complete if and only if the disjunction of all rules' premises is a tautology, i.e. if

\[
\models \bigvee_{r \in T} \mathcal{F}_{pre}(r).
\]
Completeness: Example

<table>
<thead>
<tr>
<th>F: room ventilation</th>
<th>r_1</th>
<th>r_2</th>
<th>r_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>button pressed?</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>ventilation off?</td>
<td>×</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>ventilation on?</td>
<td>–</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>go</td>
<td>×</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>start ventilation</td>
<td>×</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

* Is \( T \) complete?

**No.** (Because there is no rule for, e.g., the case \( \sigma(b) = \text{true}, \sigma(on) = \text{false}, \sigma(off) = \text{false} \)).

Recall:

\[
\begin{align*}
F(r_1) &= c_1 \land c_2 \land \neg c_3 \land a_1 \land \neg a_2 \\
F(r_2) &= c_1 \land \neg c_2 \land c_3 \land \neg a_1 \land a_2 \\
F(r_3) &= \neg c_1 \land \text{true} \land \text{true} \land \neg a_1 \land \neg a_2
\end{align*}
\]

\[
F_{\text{pre}}(r_1) \lor F_{\text{pre}}(r_2) \lor F_{\text{pre}}(r_3) \\
= (c_1 \land c_2 \land \neg c_3) \lor (c_1 \land \neg c_2 \land c_3) \lor (\neg c_1 \land \text{true} \land \text{true})
\]

is not a tautology.

Requirements Analysis with Decision Tables

- Assure we have formalised requirements as decision table \( T \).
- **If \( T \) is (formally) incomplete,**
  - then there is probably a case not yet discussed with the customer, or some misunderstandings.
- **If \( T \) is (formally) complete,**
  - then there still may be misunderstandings.
  - If there are no misunderstandings, then we did discuss all cases.
- **Note:**
  - Whether \( T \) is (formally) complete is **decidable**.
  - Deciding whether \( T \) is complete reduces to plain SAT.
  - There are efficient tools which decide SAT.
  - In addition, decision tables are often much easier to understand than natural language text.
For Convenience: The ‘else’ Rule

- **Syntax:**

<table>
<thead>
<tr>
<th>T: decision table</th>
<th>( r_1 )</th>
<th>( r_n )</th>
<th>else</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( v_1,1 )</td>
<td>( v_1,n )</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>( c_m )</td>
<td>( v_m,1 )</td>
<td>( v_m,n )</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>( a_1 )</td>
<td>( w_1,1 )</td>
<td>( w_1,n )</td>
<td>( w_1,e )</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( a_k )</td>
<td>( w_k,1 )</td>
<td>( w_k,n )</td>
<td>( w_k,e )</td>
</tr>
</tbody>
</table>

- **Semantics:**

\[
\mathcal{F}(\text{else}) := \neg \left( \bigvee_{r \in T \setminus \{\text{else}\}} \mathcal{F}_{\text{pre}}(r) \right) \land \mathcal{F}(w_{1,e}, a_1) \land \cdots \land \mathcal{F}(w_{k,e}, a_k)
\]

**Proposition.** If decision table \( T \) has an ’else’-rule, then \( T \) is complete.

---

**Uselessness**

**Definition.** [Uselessness] Let \( T \) be a decision table.

A rule \( r \in T \) is called useless (or: redundant) if and only if there is another (different) rule \( r' \in T \)

- whose premise is implied by the one of \( r \) and
- whose effect is the same as \( r \)’s.

i.e. if

\[
\exists r' \neq r \in T \models (\mathcal{F}_{\text{pre}}(r) \implies \mathcal{F}_{\text{pre}}(r')) \land (\mathcal{F}_{\text{eff}}(r) \iff \mathcal{F}_{\text{eff}}(r')).
\]

\( r \) is called **subsumed** by \( r' \).

- Again: uselessness is decidable; reduces to SAT.
Uselessness: Example

<table>
<thead>
<tr>
<th>T: room ventilation</th>
<th>r₁</th>
<th>r₂</th>
<th>r₃</th>
<th>r₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>b button pressed?</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>off ventilation off?</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>−</td>
</tr>
<tr>
<td>on ventilation on?</td>
<td>−</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>go start ventilation</td>
<td>×</td>
<td>−</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stop stop ventilation</td>
<td>−</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Rule r₄ is **subsumed** by r₃.
- Rule r₅ is **not subsumed** by r₄.

- Useless rules “do not hurt” as such.
- Yet useless rules should be removed to make the table more readable, yielding an easier usable specification.

---

**Useless Requirements on Requirements Specification Documents**

The representation and form of a requirements specification should be:

- **easily understandable** — all affected people should be able to understand the requirements specification.
- **precise** — the requirements specification should not introduce new uncertainties or room for interpretation (→ testable, objective).
- **easily maintainable** — creating and maintaining the requirements specification should be easy and should not need unnecessary effort.
- **easily usable** — storage of and access to the requirements specification should not need significant effort.

- Rule r₅ is **subsumed** by r₃.

**Note:** Once again, it’s about compromises.

- A very precise **objective** requirements specification may not be easily understandable by every affected person.
- → provide redundant explanations.
- It is not trivial to have both, low maintenance effort and low access effort.
- → value low access effort higher, a requirements specification document is much more often read than changed or written (and most changes require reading beforehand).

- Useless rules “do not hurt” as such.
- Yet useless rules should be removed to make the table more readable, yielding an easier usable specification.
**Definition.** [Determinism]

A decision table $T$ is called **deterministic** if and only if the premises of all rules are **pairwise disjoint**, i.e. if

$$\forall r_1 \neq r_2 \in T \implies \neg (F_{\text{pre}}(r_1) \land F_{\text{pre}}(r_2)).$$

Otherwise, $T$ is called **non-deterministic**.

- And again: determinism is **decidable**; reduces to SAT.

---

**Determinism: Example**

<table>
<thead>
<tr>
<th>$T$: room ventilation</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>button pressed?</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$-$</td>
</tr>
<tr>
<td>ventilation off?</td>
<td>$\times$</td>
<td>$-$</td>
<td>$*$</td>
</tr>
<tr>
<td>ventilation on?</td>
<td>$-$</td>
<td>$-$</td>
<td>$*$</td>
</tr>
<tr>
<td>start ventilation</td>
<td>$\times$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>stop ventilation</td>
<td>$-$</td>
<td>$\times$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

- Is $T$ deterministic? **Yes.**
Determinism: Another Example

<table>
<thead>
<tr>
<th>$T_{\text{absr}}$: room ventilation</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$ button pressed?</td>
<td>×</td>
<td>×</td>
<td>−</td>
</tr>
<tr>
<td>$go$ start ventilation</td>
<td>×</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>$stop$ stop ventilation</td>
<td>−</td>
<td>×</td>
<td>−</td>
</tr>
</tbody>
</table>

- Is $T_{\text{absr}}$ deterministic?  **No.**

By the way...

- Is non-determinism a bad thing in general?
  - Just the opposite: non-determinism is a very, very powerful **modelling tool**.

  - Read table $T_{\text{absr}}$ as:
    - the button may switch the ventilation **on** under certain conditions (which I will specify later), and
    - the button may switch the ventilation **off** under certain conditions (which I will specify later).

  We in particular state that we do not (under any condition) want to see *on* and *off* executed together, and that we do not (under any condition) see *go* or *stop* without button pressed.

- On the other hand: non-determinism may not be intended by the customer.

---

**Content**

- (Basic) Decision Tables
  - Syntax, Semantics
- ...for Requirements Specification
- ...for Requirements Analysis
  - Completeness,
  - Useless Rules,
  - Determinism
- Domain Modelling
  - Conflict Axiom,
  - Relative Completeness,
  - Vacuous Rules,
  - Conflict Relation
- Collecting Semantics
- Discussion
Domain Modelling for Decision Tables

**Example:**

<table>
<thead>
<tr>
<th>$T$: room ventilation</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$ button pressed?</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>off ventilation off?</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>on ventilation on?</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>go start ventilation</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>stop stop ventilation</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

* If $on$ and $off$ model opposite output values of one and the same sensor for “room ventilation on/off”, then $\sigma \models on \land off$, and $\sigma \models \neg on \land \neg off$ never happen in reality for any observation $\sigma$.

* Decision table $T$ is incomplete for exactly these cases.
  ($T$ “does not know” that $on$ and $off$ can be opposites in the real-world).

* We should be able to “tell” $T$ that $on$ and $off$ are opposites (if they are).
  Then $T$ would be relative complete (relative to the domain knowledge that $on/\neg off$ are opposites).

**Bottom-line:**

* Conditions and actions are abstract entities without inherent connection to the real world.

* When modelling real-world aspects by conditions and actions, we may also want to represent relations between actions/conditions in the real-world ($\rightarrow$ domain model [Bjørner, 2006]).
A conflict axiom over conditions $C$ is a propositional formula $\phi_{confI}$ over $C$.

**Intuition:** a conflict axiom characterises all those cases, i.e. all those combinations of condition values which ‘cannot happen’—according to our understanding of the domain.

**Note:** the decision table semantics remains unchanged!

**Example:**

Let $\phi_{confI} = (on \land off) \lor (\neg on \land \neg off)$. "on models an opposite of off, neither can both be satisfied nor both non-satisfied at a time’

**Notation:**

<table>
<thead>
<tr>
<th>$T$: room ventilation</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$: button pressed?</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>off: ventilation off?</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>on: ventilation on?</td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$go$: start ventilation</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stop: stop ventilation</td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
</tbody>
</table>

\[
\phi_{confI} = (on \land off) \lor (\neg on \land \neg off)
\]

**Pitfalls in Domain Modelling** *(Wikipedia, 2015)*


- To stop a plane after touchdown, there are spoilers and thrust-reverse systems.
- Enabling one of those while in the air, can have fatal consequences.
- Design decision: the software should block activation of spoilers or thrust-revers while in the air.
- Simplified decision table of blocking procedure:

<table>
<thead>
<tr>
<th>$P$:</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>spo: spoilers requested</td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
<tr>
<td>thr: thrust-reverse requested</td>
<td></td>
<td></td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>at least 6.3 tons weight on each landing gear strut</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spd: wheels turning faster than 333 km/h</td>
<td></td>
<td>$\times$</td>
<td></td>
<td>$\times$</td>
</tr>
<tr>
<td>spl: enable spoilers</td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
<tr>
<td>thr: enable thrust-reverse</td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
</tr>
</tbody>
</table>

**Idea:** if conditions $lgsw$ and $spd$ not satisfied, then aircraft is in the air.

**14 Sep. 1993:**

- wind conditions not as announced from tower, tail- and crosswinds,
- anti-crosswind manoeuvre puts too little weight on landing gear
- wheels didn’t turn fast due to hydroplaning.
Relative Completeness

Definition. [Completeness wrt. Conflict Axiom]
A decision table $T$ is called complete wrt. conflict axiom $\varphi_{conf}$ if and only if the disjunction of all rules' premises and the conflict axiom is a tautology, i.e. if

$$\models \varphi_{conf} \lor \bigvee_{r \in T} F_{pre}(r).$$

- **Intuition:** a relative complete decision table explicitly cares for all cases which 'may happen'.
- **Note:** with $\varphi_{conf} = false$, we obtain the previous definitions as a special case.
  Fits intuition: $\varphi_{conf} = false$ means we don’t exclude any states from consideration.

Example

$$\begin{array}{|c|c|c|c|}
\hline
\text{Trigger} & F_1 & F_2 & F_3 \\
\hline
\text{button pressed?} & 0 & 0 & 1 \\
\text{ventilation off?} & 0 & 1 & 1 \\
\text{ventilation on?} & 1 & 0 & 1 \\
\text{start ventilation} & 1 & 1 & 1 \\
\text{stop ventilation} & 1 & 1 & 1 \\
\hline
\end{array}$$

- $T$ is complete wrt. its conflict axiom.
- **Pitfall:** if on and off are outputs of two different, independent sensors, then $\sigma \models on \land off$ is possible in reality (e.g. due to sensor failures). Decision table $T$ does not tell us what to do in that case!
Definition. [Vacuity wrt. Conflict Axiom]

A rule \( r \in T \) is called vacuous wrt. conflict axiom \( \varphi_{\text{conf}} \) if and only if the premise of \( r \) implies the conflict axiom, i.e. if \( \models \mathcal{F}_{\text{pre}}(r) \rightarrow \varphi_{\text{conf}} \).

- **Intuition**: a vacuous rule would only be enabled in states which 'cannot happen'.

**Example**:

<table>
<thead>
<tr>
<th>( r )</th>
<th>( r_1 )</th>
<th>( r_2 )</th>
<th>( r_3 )</th>
<th>( r_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>button pressed?</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
</tr>
<tr>
<td>ventilation off?</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
</tr>
<tr>
<td>ventilation on?</td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>start ventilation</td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>stop ventilation</td>
<td></td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>

\[ \neg((\text{on} \land \text{off}) \lor (\neg\text{on} \land \neg\text{off})) \]

- **Vacuity wrt. \( \varphi_{\text{conf}} \)**: Like uselessness, vacuity doesn't hurt as such but
  - May hint on inconsistencies on customer's side. (Misunderstandings with conflict axiom?)
  - Makes using the table less easy! (Due to more rules.)
  - Implementing vacuous rules is a waste of effort!
Conflicting Actions

Definition. [Conflict Relation] A conflict relation on actions $A$ is a transitive and symmetric relation $\trianglerighteq \subseteq (A \times A)$.

Definition. [Consistency] Let $r$ be a rule of decision table $T$ over $C$ and $A$.

(i) Rule $r$ is called consistent with conflict relation $\trianglerighteq$ if and only if there are no conflicting actions in its effect, i.e. if
\[ \models \mathcal{F}_{\text{eff}}(r) \rightarrow \bigwedge_{(a_1, a_2) \in \trianglerighteq} \neg (a_1 \land a_2). \]

(ii) $T$ is called consistent with $\trianglerighteq$ iff all rules $r \in T$ are consistent with $\trianglerighteq$.

- Again: consistency is decidable; reduces to SAT.
Example: Conflicting Actions

<table>
<thead>
<tr>
<th>T: room ventilation</th>
<th>r₁</th>
<th>r₂</th>
<th>r₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>off</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>on</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>go</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>button pressed?</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ventilation off?</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>ventilation on?</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>start ventilation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stop ventilation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>¬((on ∧ off) ∨ (¬on ∧ ¬off))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Let $‡$ be the transitive, symmetric closure of $\{(stop, go)\}$.
  "actions stop and go are not supposed to be executed at the same time"
- Then rule $r₁$ is inconsistent with $‡$.

- A decision table with inconsistent rules may do harm in operation!
- Detecting an inconsistency only late during a project can incur significant cost!
- Inconsistencies – in particular in (multiple) decision tables, created and edited by multiple people, as well as in requirements in general – are not always as obvious as in the toy examples given here! (would be too easy...)
- And is even less obvious with the collecting semantics (→ in a minute).

Content

- (Basic) Decision Tables
  - Syntax, Semantics
- …for Requirements Specification
- …for Requirements Analysis
  - Completeness,
  - Useless Rules,
  - Determinism
- Domain Modelling
  - Conflict Axiom,
  - Relative Completeness,
  - Vacuous Rules,
  - Conflict Relation
- Collecting Semantics
- Discussion
Collecting Semantics

Let $T$ be a decision table over $C$ and $A$ and $\sigma$ be a model of an observation of $C$ and $A$. Then

$$F_{\text{coll}}(T) := \bigwedge_{a \in A} a \leftrightarrow \bigvee_{r \in T, r(a) = x} F_{\text{pre}}(r)$$

is called the collecting semantics of $T$. 
Collecting Semantics

Let $T$ be a decision table over $C$ and $A$ and $\sigma$ be a model of an observation of $C$ and $A$. Then

$$F_{\text{coll}}(T) := \bigwedge_{a \in A} a \leftrightarrow \bigvee_{r \in T, r(a) = x} F_{\text{pre}}(r)$$

is called the collecting semantics of $T$.

We say, $\sigma$ is allowed by $T$ in the collecting semantics if and only if $\sigma \models F_{\text{coll}}(T)$. That is, if exactly all actions of all enabled rules are planned/executed.

Example:

<table>
<thead>
<tr>
<th>$T$: room ventilation</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>button pressed?</td>
<td>$\frac{\square}{\top}$</td>
<td>$\frac{x}{\bot}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ventilation off?</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{\square}{\top}$</td>
<td></td>
</tr>
<tr>
<td>ventilation on?</td>
<td>$\frac{\square}{\top}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td></td>
</tr>
<tr>
<td>go</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{\square}{\top}$</td>
</tr>
<tr>
<td>start ventilation</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{\square}{\top}$</td>
</tr>
<tr>
<td>stop ventilation</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td></td>
<td>$\frac{\square}{\top}$</td>
</tr>
<tr>
<td>blink button</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td></td>
<td>$\frac{\square}{\top}$</td>
</tr>
<tr>
<td>blink button $\neg$go</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{x}{\bot}$</td>
<td>$\frac{\square}{\top}$</td>
</tr>
</tbody>
</table>

"Whenever the button is pressed, let it blink (in addition to go/stop action)."
Definition. [Consistency in the Collecting Semantics]
Decision table $T$ is called consistent with conflict relation $\triangleright$ in the collecting semantics (under conflict axiom $\varphi_{\text{conf}}$) if and only if there are no conflicting actions in the effect of jointly enabled transitions, i.e. if

$$\models F_{\text{coll}}(T) \land \varphi_{\text{conf}} \rightarrow \bigwedge_{(a_1, a_2) \in \triangleright} (\neg a_1 \land \neg a_2).$$

Discussion
**Decision Tables:** one example for a formal requirements specification language with

- formal syntax,
- formal semantics.

Requirements analysts can use DTs to
- formally (objectively, precisely)
  describe their understanding of requirements. Customers may need translations/explanation!

**DT** properties like
- (relative) completeness, determinism,
- uselessness,

... can be used to analyse requirements.
The discussed DT properties are decidable, there can be automatic analysis tools.

**Domain modelling** formalises assumptions on the context of software: for DTs:
- conflict axioms, conflict relation,

Note: wrong assumptions can have serious consequences.


