• 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
(i) $\sigma \vdash \phi$, if we have (i.e., true)

$$\frac{T}{F}$$

(ii) $\sigma \vdash \phi$, if we have (i.e., false)

$$\frac{F}{T}$$

(iii) $\sigma \vdash \phi$, if we have (i.e., true)

$$\frac{F}{T}$$

Let $\Phi \vdash \Phi'$.

\[
\begin{array}{c}
\text{Description of action} \\
\text{Description of condition} \\
\text{Decision Table Semantics}
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Decision Table} & \text{Example} \\
\hline
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Decision Table} & \text{Example} \\
\hline
\hline
\end{array}
\]
If decision table \( T \) has an 'else' rule, then \( T \) is complete.

Uselessness: Example

\[ \sigma \rightarrow \begin{cases} b & \text{ventilation on?} \\ \text{ventilation off?} \end{cases} \]

Again: uselessness is called \( T \) \( \in \) \( r \) 'k,e' \( 1 \), \( e \) 1 \( 1 \), \( r \) \( F \) \( \wedge \cdot \cdot \cdot \wedge \) \( r \) \( F \) \( \wedge \) \( \sigma \rightarrow \begin{cases} a & \text{stop ventilation} \\ \text{start ventilation} \end{cases} \]

Useless rules "do not hurt" as such.

Uselessness: Example

\[ \sigma \rightarrow \begin{cases} b & \text{ventilation on?} \\ \text{ventilation off?} \end{cases} \]

Again: uselessness is called \( T \) \( \in \) \( r \) 'k,e' \( 1 \), \( e \) 1 \( 1 \), \( r \) \( F \) \( \wedge \cdot \cdot \cdot \wedge \) \( r \) \( F \) \( \wedge \) \( \sigma \rightarrow \begin{cases} a & \text{stop ventilation} \\ \text{start ventilation} \end{cases} \]

Useless rules "do not hurt" as such.
Definition.

A decision table $T$ is called deterministic if and only if the premises of all rules are pairwise disjoint, i.e. if $\forall r_1 \neq r_2 \in T \cdot |F_{\text{pre}}(r_1) \land \neg F_{\text{pre}}(r_2)| = 0$. Otherwise, $T$ is called non-deterministic.

Determinism: Example

$T$: room ventilation

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

• Is $T$ deterministic? Yes.

Determinism: Another Example

$T_{\text{abstr}}$: room ventilation

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

• Is $T_{\text{abstr}}$ 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.

Domain Modelling for Decision Tables

Example: $T$: room ventilation

<table>
<thead>
<tr>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>button pressed?</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>ventilation off?</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>ventilation on?</td>
<td>–</td>
<td>×</td>
</tr>
<tr>
<td>go</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>start ventilation</td>
<td>–</td>
<td>×</td>
</tr>
<tr>
<td>stop ventilation</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 $\sigma | = \text{on} \land \text{off}$ and $\sigma | = \neg \text{on} \land \neg \text{off}$ 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/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)).
Implementing vacuous rules is a waste of effort!

• Makes using the table less easy!

May hint on inconsistencies on customer's side.

\[ \text{confl} \text{ doesn't hurt as such} \]

: Like uselessness, vacuity \( \varphi \) wrt. Vacuity

on landing gear too little weight

anti-crosswind manoeuvre puts

14 Sep. 1993

off \( \land \neg \)
on

\( \neg \times \)

stop ventilation

\( \neg \times \)

start ventilation

go, then aircraft is in the air.

not satisfied

and \( \lgsw \):

if conditions

Idea

\[ \neg \times \]

\( \times \)

321

r

r

r

: room ventilation

\( T \)

\( \times \)

\( \lgsw \):

if conditions

\[ \neg \times \]

\( \times \)

\( \times \)

\( \times \)

enable thrust-reverse

\( \times \)

\( \times \)

\( \times \)

thrust-reverse requested

\( \times \)

\( \times \)

\( \times \)

spoilers requested

\[ \neg \times \]

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)

\( \times \)
Whenever the button is pressed, let it blink (in addition to go/stop action).

$$\text{off} \land \neg \text{on} \lor \neg \text{off} \rightarrow \text{blnk}$$

- start ventilation
- ventilation on?
- ventilation off?
- button pressed?

That is, if exactly one action is allowed in the collecting semantics.

Let $$A \in \mathcal{a}$$ be a decision table over $$\{\text{r, c} \}$$, and let $$\text{C}$$ be the transitive, symmetric closure of $$\text{coll}$$.

$$(\mathcal{a}, \text{coll}) := \text{T}^cz(\mathcal{a}, \text{coll})$$ if and only if there are no conflicting actions as in the toy examples given here.

Again: consistency is not always as obvious as in the toy examples given here!

• Decision Tables: an example for a formal requirements specification language with
  • formal syntax,
  • formal semantics.
• 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.

References


