A Set-Based Logical Language for Specification of Combinatorial Models

Anna Zamansky, Maria Spichkova

Contact: maria.spichkova@rmit.edu.au
Combinatorial Model

A **combinatorial model** is defined as tuple consisting of
- a set of parameters,
- their respective possible values, and
- a set of logical restrictions on the value combinations

**Combinatorial Test Design (CTD)** - methodology for test design of complex software systems, in which a system is modelled using a combinatorial model
Combinatorial Model

In CTD, a **system** is modelled using a finite set of system parameters together with their corresponding associated values

\[
\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}
\]

A **scenario (test)** is an assignment of a value from \( V(\mathcal{A}_i) \) to each \( \mathcal{A}_i \)

A **combinatorial model** for a system is defined as a set of scenarios.
Combinatorial Test Design (CTD)

• **Aim:** To systematically optimise the number of test cases, while ensuring the coverage of given conditions.

• **Issues:**
  
  • What should be the process of constructing combinatorial models?
  
  • Does a manual process for modelling and maintaining the test space fit the industry needs? 😊
What we would like to have?

• Formalization of the visual notation
• Reducing the cognitive load of the modeller and tester when specifying the logical restrictions
• Reducing the chances for human errors
Example

Parameters:
- Item Status (denoted by IS)
- Order Shipping (denoted by OS)
- Delivery Timeframe (denoted by DT)

Values:

\[ \mathcal{A} = \{IS, OS, DT\} \]

\[ IS = \{InStock, OutOfStock, NoSuchProduct\} \]
\[ OS = \{Air, Ground\} \]
\[ DT = \{Immediate, 3Days, 1Month\} \]

Combinatorial model of the system is a set of scenarios (assignments of values to parameters), such as:

\[ s_1 : (IS = InStock, OS = Air, DT = Immediate) \]
\[ s_2 : (IS = InStock, OS = Ground, DT = Immediate) \]

How many possible scenarios could we have for this example? 😊

But in praxis not all scenarios are executable.

**Challenge:** Separating the valid (executable) scenarios from the invalid ones
More challenges!

The same?

\[ R_1 : DT = \text{Immediate} \rightarrow OS \neq \text{Ground} \]
\[ R_2 : DT = \text{Immediate} \rightarrow OS = \text{Air} \]

But what if we add new value for OS?

\[ s_3 = (IS = \text{InStock}, OS = \text{Sea}, DT = \text{Immediate}) \]

Challenge:
Inconsistent interpretation of test validity in case a new parameter value is added
Let $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be a set of variables, and $M_\mathcal{A}$ be a set $V = \{V(\mathcal{A}_1), \ldots, V(\mathcal{A}_n)\}$ of value sets associated with those variables.

Atomic formula:

$\mathcal{A}_i \in V(\mathcal{A}_i) \setminus X,$

where $X \subseteq V(\mathcal{A}_i)$, $1 \leq i \leq n$, or a more explicit form

$\mathcal{A}_i \in Y \land Y \subseteq V(\mathcal{A}_i).$

might be shorter for small data sets

more explicit and provides more intuitive representation
A valuation is an assignment of values to variables so that
\[ v(s_i) \in V(A_i) \]

We say that \((S, v)\) is a model of an atomic formula denoted by

\[ (S, v) \models A_i \in V(A_i) \setminus X \text{ for some } X \]

if
\[ v(A_i) \in V(A_i) \text{ and } v(A_i) \notin X \]
Running Example

\[ M_A = \{D_1, D_2, D_3\} \]

\[ D_1 = \{InStock, OutOfStock, NoSuchProduct\} \]
\[ D_2 = \{Air, Ground\} \]
\[ D_3 = \{Immediate, 3Days, 1Month\} \]

\[ DT = Immediate \rightarrow OS \neq Ground \]

can be written in our language in the following ways

\[ \psi_1 = DT \in D_3 \setminus \{3Days, 1Month\} \]
\[ \rightarrow OS \in D_2 \setminus \{Ground\} \]

\[ \psi_2 = (DT \in \{Immediate\} \land \{Immediate\} \subseteq D_3) \]
\[ \rightarrow OS \in D_2 \setminus \{Ground\}, \]

\[ \psi_3 = (DT \in \{Immediate\} \land \{Immediate\} \subseteq D_3) \]
\[ \rightarrow OS \in \{Air\} \land \{Air\} \subseteq D_2. \]
Many possible options for correction:

\[ v_1(DT) = 3\text{Days}, \quad v_1(OS) = \text{Sea}, \quad v_1(IS) = \text{Instock} \]
\[ v_2(DT) = \text{Immediate}, \quad v_2(OS) = \text{Sea}, \quad v_2(IS) = \text{Instock} \]

\[ D'_2 = D_2 \cup \{\text{Sea}\} \]

\[ M'_A = \{D_1, D'_2, D_3\} \]

\[ \psi'_1 = \quad \text{DT} \in D_3 \setminus \{3\text{Days, 1Month}\} \]
\[ \quad \rightarrow \text{OS} \in D'_2 \setminus \{\text{Ground}\} \]

\[ \psi'_2 = \quad \text{DT} \in \{\text{Immediate}\} \land \{\text{Immediate}\} \subseteq D_3 \]
\[ \quad \rightarrow \text{OS} \in D'_2 \setminus \{\text{Ground}\} \]

\[ \psi'_3 = \quad \text{DT} \in \{\text{Immediate}\} \land \{\text{Immediate}\} \subseteq D_3 \]
\[ \quad \rightarrow \text{OS} \in \{\text{Air}\} \land \{\text{Air}\} \subseteq D'_2 \]

\[ \psi''_1 = \quad \text{DT} \in D_3 \setminus \{3\text{Days, 1Month}\} \]
\[ \quad \rightarrow \text{OS} \in D'_2 \setminus \{\text{Ground, Sea}\} \]

\[ \psi''_2 = \quad \text{DT} \in \{\text{Immediate}\} \land \{\text{Immediate}\} \subseteq D_3 \]
\[ \quad \rightarrow \text{OS} \in D'_2 \setminus \{\text{Ground, Sea}\} \]

\[ \psi''_3 = \quad \text{DT} \in \{\text{Immediate}\} \land \{\text{Immediate}\} \subseteq D_3 \]
\[ \quad \rightarrow \text{OS} \in \{\text{Air}\} \land \{\text{Air}\} \subseteq D'_2 \setminus \{\text{Sea}\} \]

\[ \psi'''_3 = \quad \text{DT} \in \{\text{Immediate}\} \land \{\text{Immediate}\} \subseteq D_3 \]
\[ \quad \rightarrow \text{OS} \in \{\text{Air, Sea}\} \land \{\text{Air}\} \subseteq D'_2 \]
Proposed Visualisation

$D_2 = \{\text{Air, Ground}\}$

Validity of the new value is decided for $D_2 = D_2 \cup \{\text{Sea}\}$
What next?

- Tool support
- Scalability analysis
- Application in Learning & Teaching
Thank you!
IS10: Collaboration in Software and System Engineering

Areas of interest include but are not limited to:

• Collaborative modelling and analysis of sustainable software
• Collaborative aspects of global requirements engineering
• Collaborative aspects of formal methods in conceptual modelling, specification, and design
• Collaborative aspects of testing, verification and validation of systems
• New best practices for software and system engineering education to support team-based learning
• Innovative curriculum, assessment or course formats to support team-based learning of software and system engineering
• Diversity in software and systems engineering teams
• Intercultural aspects in software and systems engineering
• Usability aspects in software and systems engineering (including formal methods)
• Successful case studies on application of formal methods in collaborative projects
• Comprehensibility and readability of formal methods in software engineering
• Teaching of formal methods and collaborative aspects thereof
• Cross-disciplinary software and systems engineering (including application of formal methods)
• Industrial challenges, experience reports and case studies