Combinatorial Testing Methods for Modelling Composed Systems

Dimitris E. Simos, Ludwig Kampel, Bernhard Garn, Murat Ozcan

**Combinatorial Methods for Modelling Composed Systems**

Complex systems composed of multiple components
Emerging in vast numbers in modern information society:
- Contemporary software design relies on modular software architecture, making it better understandable and maintainable
- A modern vehicle is a composed complex system in itself
- Communicating autonomous vehicles are even more complex
- Smart buildings like hospitals, shopping malls, etc.

- Composed SUT with components $C_1$, $C_2$, $C_3$, and $C_4$, e.g., a car:
  - $C_1$: Wheels modelled via producer, type
  - $C_2$: Engine modelled via fuel, drive-mode, filter
  - $C_3$: Infotainment modelled via streaming, audio, remote
  - $C_4$: Communication modelled via ip-version, connection, speed

**Methodology**
Different in nature and in their application domains, these systems under test (SUTs) are systems composed of sub-systems:
- Devise a combinatorial model and a test suite for each sub-system
- Devise a combinatorial model for the unifying meta-system
- Apply a combinatorial construction to merge test suites of the components to a test suite for the whole SUT
- From theory we know: The coverage of all $t$-way interactions is inherited to the overall test suite

**BACnet (Building Automation and Control networking) Protocol**

BACnet enables devices access via the network.
- Interoperability among different vendors’ equipment.
- One operator interface to handle any device in the network.
- US, EU and ISO standard

**Combinatorial Methods for Testing BACnet**

- BACnet models devices using an object oriented structure.
- Event Enrollment Objects (EEOs) provide an interface to communicate with devices.
- $5$ million ways to configure an EEO.

- $\text{Harmonization Tool}$
  - Create EEO
  - Translate EEO
  - APOGEE Insight$^\text{®}$

- $\text{FieldPanel}$
  - Check EEO
  - Signal

- $\text{Thermostat}$
  - $\text{not}$ $\text{reject}$

**A Plug-in Construction for CAs Reflecting Composed Systems**

- Goal: Construct CAs with more factors from CAs with less factors.
- Idea: Adapt plug-in construction from classic design theory for CAs.
- Methodology: Make use of coverage inheritance.
- Application: Combinatorial Testing for composed (Software) Systems.

**Theorem.**
Given an $\text{MCA } M = \text{MCA}(N, t, k, \{w_1, \ldots, w_n\})$ and two families $T_1 = \text{MCA}(N, t, k, \{w_1, \ldots, w_n\})$ and $S_1 = \text{MCA}(N, t, k, \{w_1, \ldots, w_n\})$ of MCA$s$, for $i = 1, \ldots, k$. Then a $\text{MCA}(N, t, k, \{w_1, \ldots, w_n\})$ can be constructed, where $M = N \times \text{max}(S_1, T_1)$ and $\tau = \text{min}(S_1, T_1)$.