

Modelling, analysing and verifying variability by means of Modal Transition Systems

M.H. ter Beek, A. Fantechi, S. Gnesi
and other FMT members

Formal Methods and Tools lab
ISTI-CNR, Pisa, Italy

Dagstuhl Seminar 13091
24 February - 1 March 2013

- 1 Aim and achievements of our research on variability
- 2 Running example: a family of coffee machines
- 3 Our Variability Model Checker VMC
- 4 Discussion and future work

Aim

- To develop a logical framework capable of dealing with variability (behavioural)
- To provide tools to support this framework with formal verification (model checking)

Main ingredient: Modal Transition Systems (Larsen et al.)

- Labelled Transition Systems distinguishing **may** (admissible) and **must** (necessary) transitions

ESEC/FSE'07, SPLC'08, VaMoS'09, VaMoS'10, iFM'10, ACoTA @ ASE'10, PLEASE @ ICSE'11, FMOODS/FORTE'11, SEW-34 @ FM'11, SPLC'11, demo @ iFM'12, tool @ FM'12, FMSPLE'12, demo @ SPLC'12, ISoLA'12,...

- v-ACTL: ACTL-like branching-time temporal logic (a.k.a. MHML)
- VMC: Variability Model Checker

Aim and achievements

Aim

- To develop a logical framework capable of dealing with variability (behavioural)
- To provide tools to support this framework with formal verification (model checking)

Main ingredient: Modal Transition Systems (Larsen et al.)

- Labelled Transition Systems distinguishing **may** (admissible) and **must** (necessary) transitions

ESEC/FSE'07, SPLC'08, VaMoS'09, VaMoS'10, iFM'10, ACoTA @ ASE'10, PLEASE @ ICSE'11, FMOODS/FORTE'11, SEW-34 @ FM'11, SPLC'11, demo @ iFM'12, tool @ FM'12, FMSPLC'12, demo @ SPLC'12, ISoLA'12,...

- v-ACTL: ACTL-like branching-time temporal logic (a.k.a. MHML)
- VMC: Variability Model Checker

Aim

- To develop a logical framework capable of dealing with variability (behavioural)
- To provide tools to support this framework with formal verification (model checking)

Main ingredient: Modal Transition Systems (Larsen et al.)

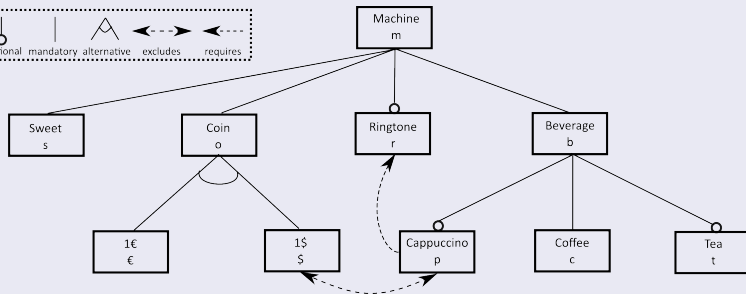
- Labelled Transition Systems distinguishing **may** (admissible) and **must** (necessary) transitions

ESEC/FSE'07, SPLC'08, VaMoS'09, VaMoS'10, iFM'10, ACoTA @ ASE'10, PLEASE @ ICSE'11, FMOODS/FORTE'11, SEW-34 @ FM'11, SPLC'11, demo @ iFM'12, tool @ FM'12, FMSPLE'12, demo @ SPLC'12, ISoLA'12,...

- v-ACTL: ACTL-like branching-time temporal logic (a.k.a. MHML)
- VMC: Variability Model Checker

Running example: family of coffee machines

Feature-based constraints

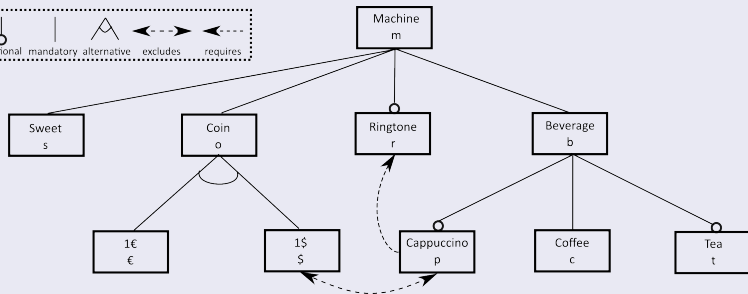


Behavioural constraints

- **After** coin insertion, user must press a button to choose whether (s)he wants sugar, **after** which (s)he may select a beverage
- The optional ringtone is rung **after** delivering a beverage
- The machine returns to its idle state **after** the beverage is taken

Running example: family of coffee machines

Feature-based constraints



Behavioural constraints

- **After** coin insertion, user must press a button to choose whether (s)he wants sugar, **after** which (s)he may select a beverage
- The optional ringtone is rung **after** delivering a beverage
- The machine returns to its idle state **after** the beverage is taken

Modal Transition Systems (MTSs)

Their use for behavioural modelling of SPLs due to Uchitel et al.

A behavioural model, amenable to model checking, able to formalize

- *shared behaviour*, common among all variants, and
- *variation points*, differentiating between variants

An LTS is obtained by including all (reachable) must transitions and a subset of the (reachable) may transitions—each selection is a variant (a.k.a. an implementation)

Limitation

MTSs cannot model **alternative** features nor **requires/excludes** CTC

Our solution

Additional variability constraints to single out the valid variants (and an algorithm to derive only—and possibly all—valid ones)

Modal Transition Systems (MTSs)

Their use for behavioural modelling of SPLs due to Uchitel et al.

A behavioural model, amenable to model checking, able to formalize

- *shared behaviour*, common among all variants, and
- *variation points*, differentiating between variants

An LTS is obtained by including all (reachable) must transitions and a subset of the (reachable) may transitions—each selection is a variant (a.k.a. an implementation)

Limitation

MTSs cannot model **alternative** features nor **requires/excludes** CTC

Our solution

Additional variability constraints to single out the valid variants (and an algorithm to derive only—and possibly all—valid ones)

Modal Transition Systems (MTSs)

Their use for behavioural modelling of SPLs due to Uchitel et al.

A behavioural model, amenable to model checking, able to formalize

- *shared behaviour*, common among all variants, and
- *variation points*, differentiating between variants

An LTS is obtained by including all (reachable) must transitions and a subset of the (reachable) may transitions—each selection is a variant (a.k.a. an implementation)

Limitation

MTSs cannot model **alternative** features nor **requires/excludes** CTC

Our solution

Additional variability constraints to single out the valid variants (and an algorithm to derive only—and possibly all—valid ones)


Coffee machine family: a possible MTS model

VMC

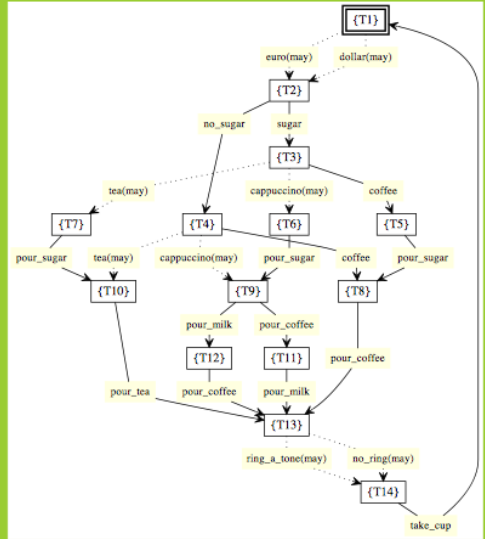
http://fmtlab.isti.cnr.it/vmc/V5.4/vmc.html

VMC v5.4

- Commands Menu
- New Model ...
- Edit Current Model
- Explore the MTS
- Modelcheck MTS ...
- View Current Model
- Draw Family MTS
- Generate Products
- Welcome
- Quit



Kandisky 1919



```
graph TD
    T1["{T1}"] -- "euro(may)" --> T2["{T2}"]
    T1 -- "dollar(may)" --> T2
    T2 -- "no_sugar" --> T4["{T4}"]
    T2 -- "sugar" --> T3["{T3}"]
    T3 -- "tea(may)" --> T7["{T7}"]
    T3 -- "cappuccino(may)" --> T6["{T6}"]
    T3 -- "coffee" --> T5["{T5}"]
    T4 -- "pour_sugar" --> T10["{T10}"]
    T4 -- "tea(may)" --> T10
    T4 -- "cappuccino(may)" --> T9["{T9}"]
    T4 -- "pour_sugar" --> T8["{T8}"]
    T4 -- "coffee" --> T8
    T5 -- "pour_sugar" --> T8
    T9 -- "pour_milk" --> T12["{T12}"]
    T9 -- "pour_coffee" --> T11["{T11}"]
    T10 -- "pour_tea" --> T13["{T13}"]
    T12 -- "pour_coffee" --> T13
    T11 -- "pour_milk" --> T13
    T8 -- "pour_coffee" --> T13
    T13 -- "ring_a_tone(may)" --> T14["{T14}"]
    T13 -- "no_ring(may)" --> T14
    T14 -- "take_cup" --> T1
```

A valid European coffee machine

The screenshot shows a web browser window titled "VMCP" with the URL <http://fmtlab.isti.cnr.it/vmcp/V5.2/vmcp.html?remotemodel>. The interface is split into two main sections:

- Left Panel:** Contains the "VMCproduct (on product)" logo with five colored circles (black, green, black, black, black). Below it is a "Commands Menu" with buttons for "New Model ...", "Edit Current Model", "Explore the LTS", "View Current Model", "View the LTS Graph", and "Quit". At the bottom of this panel is a reproduction of the painting "The Fighting Crows" by Paul Gauguin, captioned "Kandisky 1919".
- Right Panel:** Displays a state transition graph for a coffee machine model. The graph starts at state $\{T1\}$ (highlighted with a red border). Transitions are labeled with actions: "euro" leads to $\{T2\}$; "no_sugar" leads to $\{T4\}$; "sugar" leads to $\{T3\}$. From $\{T3\}$, "coffee" leads to $\{T5\}$ and "cappuccino" leads to $\{T6\}$. From $\{T4\}$, "coffee" leads to $\{T8\}$ and "cappuccino" leads to $\{T9\}$. From $\{T5\}$, "pour_sugar" leads to $\{T9\}$. From $\{T6\}$, "pour_sugar" leads to $\{T9\}$. From $\{T8\}$, "pour_coffee" leads to $\{T11\}$. From $\{T9\}$, "pour_coffee" leads to $\{T12\}$ and "pour_milk" leads to $\{T13\}$. From $\{T12\}$, "pour_milk" leads to $\{T11\}$. From $\{T13\}$, "pour_coffee" leads to $\{T11\}$. From $\{T11\}$, "ring_a_tone" leads to $\{T14\}$. Finally, "take_cup" leads back to $\{T1\}$.

A valid Canadian coffee machine

The screenshot shows a web browser window titled "VMCP" with the URL <http://fmtlab.isti.cnr.it/vmcp/V5.2/vmcp.html?remotemodel>. The interface is divided into two main sections on a green background.

Left Section: VMCproduct (on product)

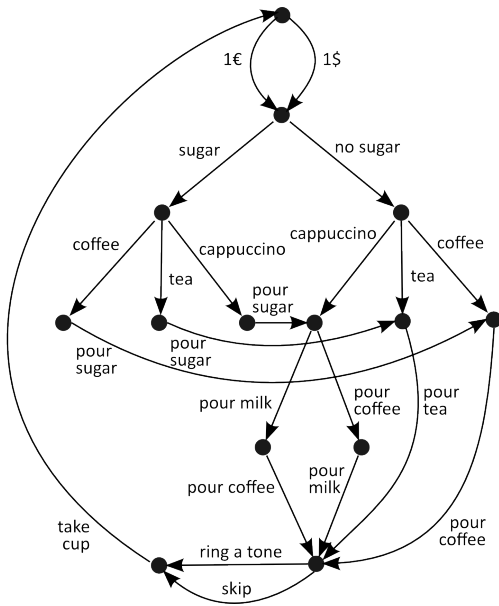
- Five colored circles: black, green, black, black, black.
- Commands Menu** (grey header):
 - New Model ...
 - Edit Current Model
 - Explore the LTS
 - View Current Model
 - View the LTS Graph
 - Quit
- Image of a painting by Kandinsky (1919).

Right Section: State Transition Graph

The graph shows states {T1} through {T14} connected by transitions labeled with actions:

- {T1} (highlighted) → {T2} (dollar)
- {T2} → {T3} (sugar) and {T4} (no_sugar)
- {T3} → {T5} (coffee)
- {T4} → {T8} (coffee)
- {T5} → {T8} (pour_sugar)
- {T8} → {T11} (pour_coffee)
- {T11} → {T14} (ring_a_tone)
- {T14} → {T1} (take_cup)

A variant LTS of above MTS that is not valid



v-ACTL: A logic interpreted over MTSs

Variability and action-based branching-time temporal logic

Based on action-based logic ACTL (De Nicola et al.), state-based logic CTL (Clarke et al.) and Hennessy-Milner Logic with Until (Larsen et al.)
Plus *deontic* interpretations of operators based on *must paths* in MTSs

Syntax of v-ACTL

$$\begin{aligned}\phi &::= \text{true} \mid \neg\phi \mid \phi \wedge \phi \mid \langle\varphi\rangle\phi \mid \langle\varphi\rangle^{\square}\phi \mid [\varphi]\phi \mid [\varphi]^{\square}\phi \mid E\pi \mid A\pi \\ \pi &::= \phi_{\varphi}U_{\varphi}\phi \mid \phi_{\varphi}U_{\varphi}^{\square}\phi \mid \phi_{\varphi}W_{\varphi}\phi \mid \phi_{\varphi}W_{\varphi}^{\square}\phi\end{aligned}$$

Thus defines state formulae ϕ , path formulae π and action formulae φ (boolean compositions of actions) over set of atomic actions $\{a, b, \dots\}$

Further operators

$$F\phi, F_{\varphi}\text{true}, F_{\varphi}^{\square}\text{true}, F^{\square}\phi, AG\phi, AG^{\square}\phi, \dots$$

v-ACTL: A logic interpreted over MTSs

Variability and action-based branching-time temporal logic

Based on action-based logic ACTL (De Nicola et al.), state-based logic CTL (Clarke et al.) and Hennessy-Milner Logic with Until (Larsen et al.)
Plus *deontic* interpretations of operators based on *must paths* in MTSs

Syntax of v-ACTL

$$\begin{aligned}\phi &::= \mathit{true} \mid \neg\phi \mid \phi \wedge \phi \mid \langle\varphi\rangle\phi \mid \langle\varphi\rangle^{\square}\phi \mid [\varphi]\phi \mid [\varphi]^{\square}\phi \mid E\pi \mid A\pi \\ \pi &::= \phi_{\varphi}U_{\varphi}\phi \mid \phi_{\varphi}U_{\varphi}^{\square}\phi \mid \phi_{\varphi}W_{\varphi}\phi \mid \phi_{\varphi}W_{\varphi}^{\square}\phi\end{aligned}$$

Thus defines state formulae ϕ , path formulae π and action formulae φ (boolean compositions of actions) over set of atomic actions $\{a, b, \dots\}$

Further operators

$$F\phi, F_{\varphi}\mathit{true}, F_{\varphi}^{\square}\mathit{true}, F^{\square}\phi, AG\phi, AG^{\square}\phi, \dots$$

v-ACTL: A logic interpreted over MTSs

Variability and action-based branching-time temporal logic

Based on action-based logic ACTL (De Nicola et al.), state-based logic CTL (Clarke et al.) and Hennessy-Milner Logic with Until (Larsen et al.)
Plus *deontic* interpretations of operators based on *must paths* in MTSs

Syntax of v-ACTL

$$\begin{aligned}\phi &::= \text{true} \mid \neg\phi \mid \phi \wedge \phi \mid \langle\varphi\rangle\phi \mid \langle\varphi\rangle^{\square}\phi \mid [\varphi]\phi \mid [\varphi]^{\square}\phi \mid E\pi \mid A\pi \\ \pi &::= \phi_{\varphi}U_{\varphi}\phi \mid \phi_{\varphi}U_{\varphi}^{\square}\phi \mid \phi_{\varphi}W_{\varphi}\phi \mid \phi_{\varphi}W_{\varphi}^{\square}\phi\end{aligned}$$

Thus defines state formulae ϕ , path formulae π and action formulae φ (boolean compositions of actions) over set of atomic actions $\{a, b, \dots\}$

Further operators

$$F\phi, F_{\varphi}\text{true}, F_{\varphi}^{\square}\text{true}, F^{\square}\phi, AG\phi, AG^{\square}\phi, \dots$$

Tool: Variability Model Checker (VMC)

Modelling, analysing and verifying behavioural variability

Given a textual encoding of an MTS and a set of variability constraints:

- interactively explore the model (MTS)
- derive and explore (all) the model's valid variants (LTSs)
- visualize the model/variants graphically as MTS/LTSs
- verify branching-time temporal logic properties over MTSs/LTSs
- interactively explain why a property is (not) satisfied

Freely usable online: <http://fmtlab.isti.cnr.it/vmc/>

Core is a command-line version of the model checker and of a variant generation procedure, both stand-alone executables written in Ada, so:

- easy to compile for Windows | Linux | Solaris | Mac OS X
- wrapped with a set of CGI scripts handled by a web server:
 - easy to build graphical html-oriented GUI
 - easy to integrate with graph drawing tools

Tool: Variability Model Checker (VMC)

Modelling, analysing and verifying behavioural variability

Given a textual encoding of an MTS and a set of variability constraints:

- interactively explore the model (MTS)
- derive and explore (all) the model's valid variants (LTSS)
- visualize the model/variants graphically as MTS/LTSS
- verify branching-time temporal logic properties over MTSs/LTSS
- interactively explain why a property is (not) satisfied

Freely usable online: <http://fmtlab.isti.cnr.it/vmc/>

Core is a command-line version of the model checker and of a variant generation procedure, both stand-alone executables written in Ada, so:

- easy to compile for Windows | Linux | Solaris | Mac OS X
- wrapped with a set of CGI scripts handled by a web server:
 - easy to build graphical html-oriented GUI
 - easy to integrate with graph drawing tools

A property verified over variants with VMC

VMC v5.4

Commands Menu

- New Model ...
- Edit Current Model
- Explore the MTS
- View Current Model
- Draw Family MTS
- Generate Products
- Welcome
- Quit

Kandinsky 1908

Evaluation of the formula "[dollar] EF true" on all family products

product007-euro-ring_a_tone	Formula evaluates	TRUE
product008-euro-no_ring	Formula evaluates	TRUE
product010-dollar-ring_a_tone	Formula evaluates	FALSE
product011-dollar-no_ring	Formula evaluates	FALSE
product016-euro-cappuccino-ring_a_tone	Formula evaluates	TRUE
product018-euro-tea-ring_a_tone	Formula evaluates	TRUE
product019-euro-tea-no_ring	Formula evaluates	TRUE
product020-euro-cappuccino-tea-ring_a_tone	Formula evaluates	TRUE
product022-dollar-tea-ring_a_tone	Formula evaluates	FALSE
product023-dollar-tea-no_ring	Formula evaluates	FALSE

Logic Formula for all Products

[dollar] EF <cappuccino> true

Check The Formula Explain the Result

(As required, no valid variant (i.e. coffee machine) can deliver a cappuccino upon the insertion of a dollar!)

VMC in relation to other model checkers

VMC itself is a product of a family of model checkers

- developed and highly optimized at ISTI–CNR during last decades
- on-the-fly: in general no need to generate/explore full state space
- capable of verifying properties specified in a CTL-like action- and state-based branching-time temporal logic

Products: FMC (input: CCS/CSP/LOTOS-like process algebras), UMC (input: UML state machines) & CMC (input: WS orchestration calculus)

Detailed comparison with MTSA, SNIP & SPLVERIFIER: SPLC'12

MTSA (D'Ippolito et al.): built on top of LTSA, i.e. also LTS-based, but no specific features for analysing variability

SNIP (Classen et al.): no GUI, but built-in support for feature diagrams and model-checking algorithms specifically tailored for SPLE

SPLVERIFIER (Apel et al.): does focus on features, but addressing the detection of feature interactions rather than variability

VMC in relation to other model checkers

VMC itself is a product of a family of model checkers

- developed and highly optimized at ISTI–CNR during last decades
- on-the-fly: in general no need to generate/explore full state space
- capable of verifying properties specified in a CTL-like action- and state-based branching-time temporal logic

Products: FMC (input: CCS/CSP/LOTOS-like process algebras), UMC (input: UML state machines) & CMC (input: WS orchestration calculus)

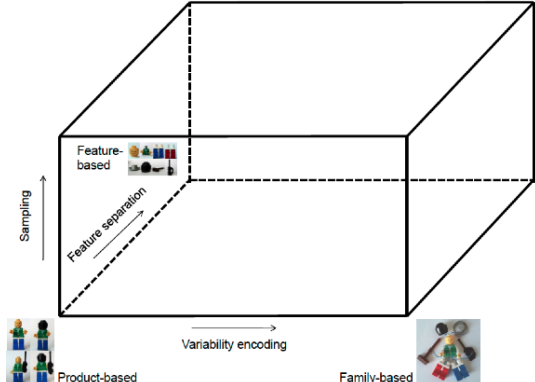
Detailed comparison with MTSA, SNIP & SPLVERIFIER: SPLC'12

MTSA (D'Ippolito et al.): built on top of LTSA, i.e. also LTS-based, but no specific features for analysing variability

SNIP (Classen et al.): no GUI, but built-in support for feature diagrams and model-checking algorithms specifically tailored for SPLE

SPLVERIFIER (Apel et al.): does focus on features, but addressing the detection of feature interactions rather than variability

VMC's position in the PLA cube (Apel et al.)



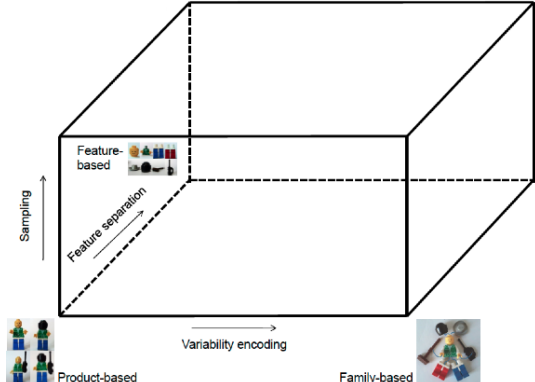
From family-based to product-based analysis

VMC can also be used to model and analyse a subfamily or a specific subset of a family's valid variants by restrictions applied via constraints

Add constraint `coffee EXC dollar` to the family specification

Only European coffee machines will be derived as valid variants, which can then be analyzed both as a subset and individually

VMC's position in the PLA cube (Apel et al.)



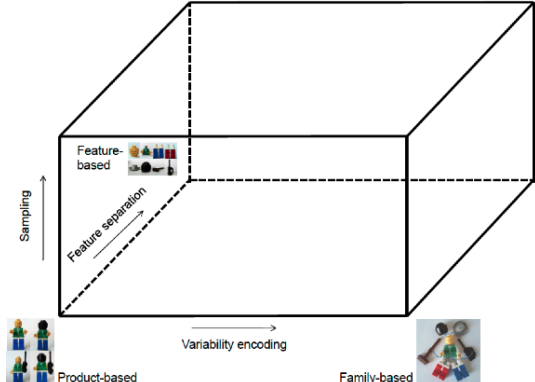
From family-based to product-based analysis

VMC can also be used to model and analyse a subfamily or a specific subset of a family's valid variants by restrictions applied via constraints

Add constraint `coffee EXC dollar` to the family specification

Only European coffee machines will be derived as valid variants, which can then be analyzed both as a subset and individually

VMC's position in the PLA cube (Apel et al.)



From family-based to product-based analysis

VMC can also be used to model and analyse a subfamily or a specific subset of a family's valid variants by restrictions applied via constraints

Add constraint `coffee EXC dollar` to the family specification

Only European coffee machines will be derived as valid variants, which can then be analyzed both as a subset and individually

Future work

- 1 Consider more expressive behavioural models (e.g. our (G)EMTSs)
- 2 Feature-based analysis, building on feature model composition (Acher et al.)
- 3 Confront with / learn from dynamic adaptation / reconfiguration, where all possible variants are often not known upfront

