The Variability Model Checker VMC

Maurice H. ter Beek
ISTI–CNR, Pisa, Italy

1 Introduction

We demonstrate an experimental tool for modelling and analysing (behavioural) variability in product families modelled as Modal Transition Systems (MTSs) [8]. A product family is a compact way for describing different products through their commonalities and variabilities (often defined by features). An MTS is a Labelled Transition System (LTS) distinguishing optional (may) and mandatory (must) transitions [1]. In [6], MTSs were recognized as a formal method for describing the possible operational behaviour of a family’s products. The standard derivation methodology for obtaining a product from an MTS modelling a product family is as follows: include all (reachable) must transitions and a subset of the (reachable) may transitions. Each selection is a product (i.e., an LTS). Unfortunately, MTSs cannot model all common variability constraints. The solution adopted in [2, 3] is to enrich an MTS description with a set of constraints defining which of the standardly derivable products should be considered as acceptable valid products. In [2], an appropriate variability and action-based branching-time temporal logic to formalize these constraints is defined, while [3] contains an algorithm to derive all and only LTSs describing valid products. This methodology is implemented in VMC and in this paper we demonstrate how to use VMC by means of the following simple and intuitive case study from [2, 3].

2 Case Study: A Product Family

A family of coffee machines has the following initial list of informal requirements:

1. Initially, a coin must be inserted; either a euro, exclusively for European products, or a dollar, exclusively for Canadian products;
2. After inserting a coin, the user has to choose whether (s)he wants sugar, by pressing one of two buttons, after which (s)he may select a beverage;
3. The choice of beverage (coffee, tea, cappuccino) varies, but all products must offer coffee while only European products may offer cappuccino;
4. Optionally, a ringtone may be rung after delivering a beverage. However, a ringtone must be rung in all products offering cappuccino;

We model all valid product behaviour by the MTS of Fig. 1(l) and the additional constraints: (i) euro and dollar are alternative; (ii) dollar excludes cappuccino; (iii) cappuccino requires ring a tone. Note that constraint (iii) cannot invalidate a product ringing a tone before delivering cappuccino; the behavioural description of a product (family) as provided by an LTS (MTS) must impose such orderings.

3 Encoding and Analyzing Product Families with VMC

VMC [8] is part of a family of on-the-fly model checkers developed at ISTI–CNR over the last two decades for verifying logic formulae in an action- and state-based branching-time temporal logic derived from the family of logics based on classic CTL, including FMC [7], UMC [4], and CMC [5].

VMC in particular is derived from FMC. Beyond interactively exploring an MTS, model checking properties over an MTS, and visualizing the interactive explanations of a verification result, VMC furthermore allows the generation of
all valid products (according to the given constraints) of an MTS describing a product family and the verification of properties over each valid product.

VMC accepts as input a textual process-algebraic encoding of an MTS and a set of constraints of the form ALTernative, EXCludes, REQuires, and IFF (hiding their logic formalization given in [2]). The distinction among may and must transitions is encoded in the resulting LTS by typing action labels corresponding to may transitions as \textit{may}(). The MTS in Fig. 1(l) modelling the coffee machine family from the case study was generated by VMC from the following encoding (the associated set of constraints is only taken into account for product generation):

\begin{verbatim}
T1 = may(euro).T2 + may(dollar).T2
T2 = sugar.T3 + no_sugar.T4
T3 = coffee.T5 + may(cappuccino).T6 + may(tea).T7
T4 = coffee.T8 + may(cappuccino).T9 + may(tea).T10
T5 = pour_sugar.T6
T6 = pour_sugar.T9
T7 = pour_sugar.T10
T8 = pour_coffee.T11
T9 = pour_coffee.T11 + pour_milk.T12
T10 = pour_tea.T13
T11 = pour_milk.T13
T12 = pour_coffee.T13
T13 = may(ring_a_tone).T14 + may(no_ring).T14
T14 = take_cup.T1

Constraints { }

end
\end{verbatim}

The variability logic defined in [2] can be directly encoded in the logic accepted by VMC by considering the typed actions. This latter logic contains the classic box and diamond modal operators [], ⟨⟩, the classic existential and universal state operators \(E, A\) (quantifying over paths), and action-based versions of the CTL until operators \(W, U\) (resulting also in an action-based version of the ‘eventually’ operator \(F\)). Using VMC it is thus possible to specify and verify properties which are surely preserved in all products by checking them over the family MTS:

1. \textbf{The MTS guarantees that if a euro or dollar action occurs, afterwards for all standardly derivable products it is eventually possible to reach action coffee.}

\[ [\text{may(euro)} \text{ or may(dollar)}] E [\text{true} \{\text{not may(*)}\} U \{\text{coffee}\} \text{ true}] \]

This formula prohibits a path leading to coffee to contain any (i.e. *) may transition (beyond the initial one). Asked to model check it over the MTS of Fig. 1(l), VMC reports it holds and offers the possibility to explain this result (cf. Fig. 2).
4 Generating and Analyzing Valid Products with VMC

Beyond generating all valid products (LTSs), VMC thus allows browsing them, verifying whether they satisfy a certain property (logic formula), and understanding why a specific valid product does (not) satisfy the verified property. To do so, VMC allows to open for each product a new window with its textual encoding.

Suppose we generate all valid products in VMC and then check for each one:
(2) If it is possible to obtain a sugared cappuccino, then it is also possible to obtain an unsugared cappuccino.

$$\left( \text{EF} \langle \text{sugar} \rangle \langle \text{cappuccino} \rangle \text{true} \right) \text{ implies } \text{EF} \langle \text{no_sugar} \rangle \langle \text{cappuccino} \rangle \text{true}$$

Property 2 does not hold for all valid products, revealing ambiguous constraints: the one of Fig. 1(r) satisfies all constraints but offers cappuccino only with sugar. A way to solve such ambiguity is to refine actions by explicitly distinguishing sugared from unsugared ones and to extend the constraints accordingly (cf. Fig. 3).

T1 = may(euro).T2 + may(dollar).T2
T2 = sugar.T3 + no_sugar.T4
T3 = sugared_coffee.T5 + may(sugared_cappuccino).T6 + may(sugared_tea).T7
T4 = unsugared_coffee.T8 + may(unsugared_cappuccino).T9 + may(unsugared_tea).T10
T5 = pour_sugar.T8
T6 = pour_sugar.T9
T7 = pour_sugar.T10
T8 = pour_coffee.T13
T9 = pour_coffee.T11 + pour_milk.T12
T10 = pour_milk.T13
T11 = pour_milk.T13
T12 = pour_coffee.T13
T13 = may(ring_a_tone).T14 + may(no_ring).T14
T14 = take_cup.T1
net SYS = T1

Constraints
{ euro ALT dollar
  sugared_tea IFF unsugared_tea
  sugared_cappuccino IFF unsugared_cappuccino
  dollar E0 sugared_cappuccino
  sugared_cappuccino REQ ring_a_tone
  ring_a_tone ALT no_ring }
From this refined family/MTS, VMC generates the 10 valid products/LTSs depicted in Fig. 4 (listing moreover which of the optional actions they contain), over which VMC can subsequently model check whether all European products offer both sugared and unsugared cappuccino by means of the formula written in Fig. 4 (recall that cappuccino is an optional feature, even for European products).

![VMC interface](image)

**Fig. 4.** All valid products/LTSs of the family/MTS of Fig. 3 as generated by VMC and the result of model checking a variant of Property 2 over all valid coffee machines

By clicking on a product, VMC displays it in a new window for further analysis.

Likewise, VMC can verify whether all valid products offer both sugared and unsugared coffee; indeed, it states that the following formula holds for all 10 products:

\[ \text{[euro]} \ ((EF \langle \text{sugared\_coffee} \rangle \text{true}) \text{ and } EF \langle \text{unsugared\_coffee} \rangle \text{true}) \]

The reader is invited to use VMC: the case study is available from the examples.

## 5 Getting Acquainted with VMC

VMC’s core contains a command-line version of the model checker and a product generation procedure, both stand-alone executables written in Ada (easy to compile for Windows/Linux/Solaris/MacOSX) and wrapped with a set of CGI scripts handled by a web server, facilitating a graphical html-oriented GUI and integration with other tools for LTS minimization and graph drawing. Its development is ongoing, but a prototype version is used at ISTI–CNR for academic purposes. VMC is publicly usable online [8] and its executables are available upon request.

**Acknowledgements.** To P. Asirelli, A. Fantechi, and S. Gnesi for joint research that led to VMC and to F. Mazzanti and A. Sulova for the development of VMC.

## References

8. F. Mazzanti, A. Sulova. VMC v5.2. [http://fmtlab.isti.cnr.it/vmc/V5.2/](http://fmtlab.isti.cnr.it/vmc/V5.2/)
VMC is an experimental tool for modelling and analysing (behavioural) variability in product families modelled as Modal Transition Systems (MTSs), written by F. Mazzanti and A. Sulova.

**VMC** accepts as input a textual process-algebraic encoding of an MTS and a set of constraints of the form ALTerative, EXCludes, REQuires, and IFF.

**VMC** can also visualize a product family/MTS (cf. 5).

**VMC** can derive all valid products (LTSs) of a family (MTS + constraints) for further analyses (cf. 4).

**VMC** can help understand why a valid product does (not) satisfy a specific verified property by opening a new window for inspecting its encoding (cf. 1) and its visualization

**VMC** thus allows to interactively explore an MTS of a product family, model check properties (logic formulae) over an MTS, visualize the (interactive) explanations of a verification result, generate all the family’s valid products (w.r.t. the given constraints) represented as LTSs, browse and explore these, model check whether they satisfy certain properties, and help understand why a certain valid product does (not) satisfy the verified properties by inspecting it individually.

**VMC** is part of a family of on-the-fly model checkers developed at ISTI-CNR for verifying logic formulae in a CTL-like action- and state-based branching-time temporal logic.

**VMC**’s core holds a command-line version of the model checker and a product generation procedure, both stand-alone executables in Ada (easy to compile for MacOSX/Windows/Linux/Solaris), and is wrapped with a set of CGI scripts handled by a web server, to help graphical html-oriented GUI and integration with LTS minimization / graph drawing tools.

**VMC** is a prototype, publicly usable online; its executables are available upon request.