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

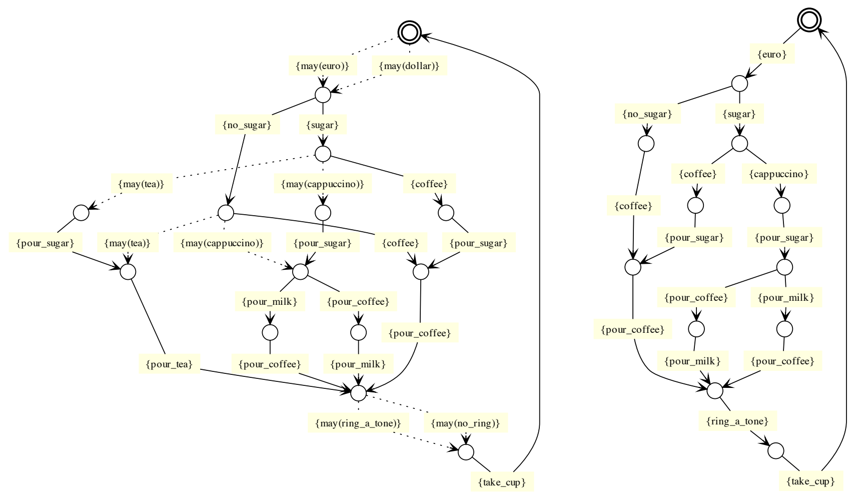


Fig. 1. MTS of coffee machine family (l) and an apparently valid European product (r) as generated by VMC; dashed edges labelled may(·) are may transitions, the others must

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 `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):

```
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.T8
T6 = pour_sugar.T9
T7 = pour_sugar.T10
T8 = pour_coffee.T13
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
```

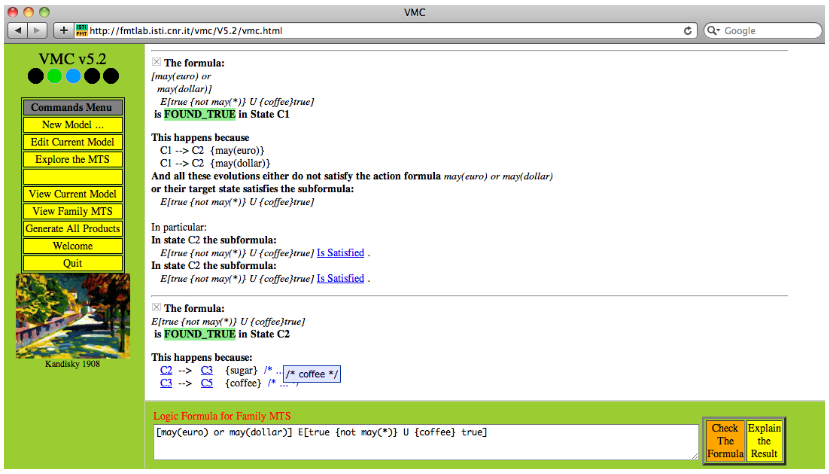
```
net SYS = T1

Constraints {
  euro ALT dollar
  dollar EXC cappuccino
  cappuccino REQ ring_a_tone
  ring_a_tone ALT no_ring
}
```

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 $[\,], \langle\,\rangle$, 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) *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.*

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

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).



VMC

http://fmtlab.isti.cnr.it/vmc/V5.2/vmc.html

VMC v5.2

The formula:
[may(euro) or
may(dollar)]
E[true {not may(*)} U {coffee}true]
is FOUND_TRUE in State C1

This happens because
C1 --> C2 {may(euro)}
C1 --> C2 {may(dollar)}
And all these evolutions either do not satisfy the action formula *may(euro) or may(dollar)*
or their target state satisfies the subformula:
E[true {not may(*)} U {coffee}true]

In particular:
In state C2 the subformula:
E[true {not may(*)} U {coffee}true] Is Satisfied .
In state C2 the subformula:
E[true {not may(*)} U {coffee}true] Is Satisfied .

The formula:
E[true {not may(*)} U {coffee}true]
is FOUND_TRUE in State C2

This happens because:
C2 --> C3 {sugar} /* ... /* coffee */
C3 --> C5 {coffee} /* ... */

Logic Formula for Family MTS

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

Fig. 2. Explanation of model checking Property 1 over MTS of Fig. 1(l) in VMC

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.*

$$(EF\ \langle sugar\rangle\ \langle cappuccino\rangle\ true)\ implies\ EF\ \langle no\_sugar\rangle\ \langle cappuccino\rangle\ true$$

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_tea.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 EXC sugared_cappuccino
   sugared_cappuccino REQ ring_a_tone
   ring_a_tone ALT no_ring }
```

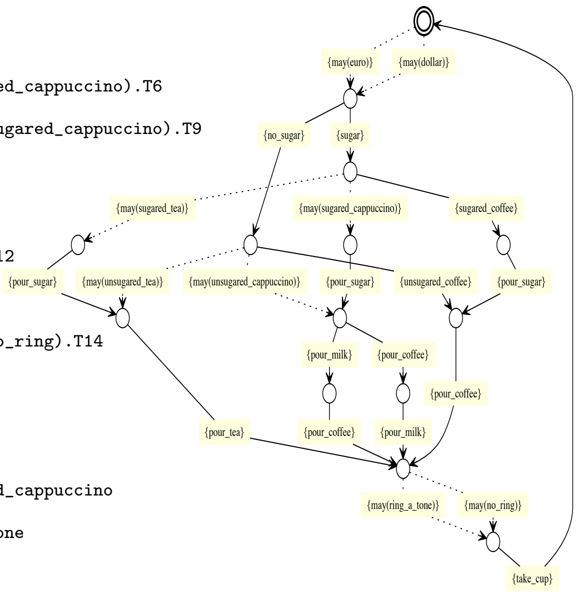
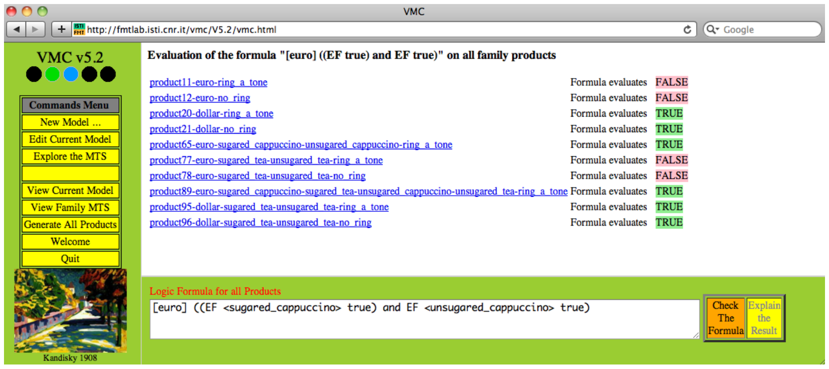


Fig. 3. MTS of a refined coffee machine family: input (l) and output (r) of VMC

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).



Evaluation of the formula "[euro] ((EF true) and EF true)" on all family products

```
product11-euro-ring_a_tone                                                    Formula evaluates FALSE
product12-euro-no_ring                                                        Formula evaluates FALSE
product20-dollar-ring_a_tone                                                  Formula evaluates TRUE
product21-dollar-no_ring                                                      Formula evaluates TRUE
product65-euro-sugared_cappuccino-unsugared_cappuccino-ring_a_tone            Formula evaluates TRUE
product77-euro-sugared_tea-unsugared_tea-ring_a_tone                          Formula evaluates FALSE
product78-euro-sugared_tea-unsugared_tea-no_ring                              Formula evaluates FALSE
product89-euro-sugared_cappuccino-sugared_tea-unsugared_cappuccino-unsugared_tea-ring_a_tone Formula evaluates TRUE
product95-dollar-sugared_tea-unsugared_tea-ring_a_tone                        Formula evaluates TRUE
product96-dollar-sugared_tea-unsugared_tea-no_ring                            Formula evaluates TRUE
```

Logic Formula for all Products

```
[euro] ((EF <sugared_cappuccino> true) and EF <unsugared_cappuccino> true)
```

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:

$$[euro]\,((EF\,\langle sugared\_coffee\rangle\; true)\; and\; EF\,\langle unsugared\_coffee\rangle\; true)$$

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

1. A. Antonik, M. Huth, K.G. Larsen, U. Nyman, A. Wąsowski. 20 Years of modal and mixed specifications. B. EATCS 95 (2008), 94–129.
2. P. Asirelli, M.H. ter Beek, A. Fantechi, S. Gnesi. A Logical Framework to Deal with Variability. In *IFM'10*, LNCS 6396, Springer, 2010, 43–58.
3. P. Asirelli, M.H. ter Beek, A. Fantechi, S. Gnesi. Formal Description of Variability in Product Families. In *SPLC'11*, IEEE, 2011, 130–139.
4. M.H. ter Beek, A. Fantechi, S. Gnesi, F. Mazzanti. A state/event-based model-checking approach for the analysis of abstract system properties. *Sci. Comput. Program.* 76, 2 (2011), 119–135.
5. A. Fantechi, A. Lapadula, R. Pugliese, F. Tiezzi, S. Gnesi, F. Mazzanti. A Logical Verification Methodology for Service-Oriented Computing. *ACM TOSEM* 21, 3 (2012).
6. D. Fischbein, S. Uchitel, V.A. Braberman. A foundation for behavioural conformance in software product line architectures. In *ROSATEA'06*, ACM, 2006, 39–48.
7. S. Gnesi, F. Mazzanti. On the Fly Verification of Networks of Automata. In *PDPTA'99*, CSREA Press, 1999, 1040–1046.
8. F. Mazzanti, A. Sulova. VMC v5.2. `http://fmtlab.isti.cnr.it/vmc/V5.2/`