Formal Description of Variability in Product Families

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

joint work with

P. Asirelli	A. Fantechi	S. Gnesi
ISTI–CNR	University of Florence	ISTI–CNR

SPLC 2011

München, Germany
24 August 2011

Outline

1. Background and aim of our research activity
2. Running example: Coffee machine family
3. Logic vACTL interpreted over MTSs
4. Advanced variability management
5. Model checking families and products
6. Conclusions and future work

Variability management/modeling

Variability management

Key difference between SPLE and 'conventional' software engineering

Variability modeling

How to explicitly define **optional**, **alternative**, **mandatory**, **required**, or **excluded** features of a product family as *variation points*

Managing variability with formal methods

Show that a certain product belongs to a product family or, instead, derive a product from a family by properly selecting features

Formally prove characteristics of products and families alike

Variability management/modeling

Variability management

Key difference between SPLE and ‘conventional’ software engineering

Variability modeling

How to explicitly define **optional**, **alternative**, **mandatory**, **required**, or **excluded** features of a product family as *variation points*

Managing variability with formal methods

Show that a certain product belongs to a product family or, instead, derive a product from a family by properly selecting features

Formally prove characteristics of products and families alike

Variability management/modeling

Variability management

Key difference between SPLE and ‘conventional’ software engineering

Variability modeling

How to explicitly define **optional**, **alternative**, **mandatory**, **required**, or **excluded** features of a product family as *variation points*

Managing variability with formal methods

Show that a certain product belongs to a product family or, instead, derive a product from a family by properly selecting features

Formally prove characteristics of products and families alike

Aim of our research activity at large

Aim

- One formal framework to express both feature-based constraints over the products of a family and constraints over their behavior
- Tool support for derivation of products and formal verification over products and families alike

Outcome IFM'10, ACOTA @ ASE'10, PLEASE @ ICSE'11, FMOODS'11, SEW-34 @ FM'11

- MTS: Modal Transition Systems (Larsen, Wasowski et alii)
- vACTL: variability and action-based CTL
- VMC: Variability Model Checker

Comparison

FTS: Featured Transition Systems, LTL/fCTL, and SNIP/NuSMV (Classen, Heymans et alii @ ICSE'10/'11)

Aim of our research activity at large

Aim

- One formal framework to express both feature-based constraints over the products of a family and constraints over their behavior
- Tool support for derivation of products and formal verification over products and families alike

Outcome IFM'10, ACOTA @ ASE'10, PLEASE @ ICSE'11, FMOODS'11, SEW-34 @ FM'11

- MTS: Modal Transition Systems (Larsen, Wasowski et alii)
- vACTL: variability and action-based CTL
- VMC: Variability Model Checker

Comparison

FTS: Featured Transition Systems, LTL/fCTL, and SNIP/NuSMV (Classen, Heymans et alii @ ICSE'10/'11)

Aim of our research activity at large

Aim

- One formal framework to express both feature-based constraints over the products of a family and constraints over their behavior
- Tool support for derivation of products and formal verification over products and families alike

Outcome IFM'10, ACOTA @ ASE'10, PLEASE @ ICSE'11, FMOODS'11, SEW-34 @ FM'11

- MTS: Modal Transition Systems (Larsen, Wasowski et alii)
- vACTL: variability and action-based CTL
- VMC: Variability Model Checker

Comparison

FTS: Featured Transition Systems, LTL/fCTL, and SNIP/NuSMV (Classen, Heymans et alii @ ICSE'10/'11)

Modal Transition Systems (MTSs)

Originally introduced by Larsen & Thomsen @ LICS 1988

MTSs are now an established model to formalize a product family's

- *underlying behavior*, shared among all products, and
- *variation points*, differentiating between products

MTS is an LTS that distinguishes between may and must transitions (modeling *optional* or *mandatory* features, resp.)

MTS cannot model variability constraints regarding *alternative* features nor those regarding *requires* and *excludes* inter-feature relations

We will model such advanced variability constraints by means of an associated set of logical formulae expressed in our variability and action-based branching-time temporal logic vACTL

Modal Transition Systems (MTSs)

Originally introduced by Larsen & Thomsen @ LICS 1988

MTSs are now an established model to formalize a product family's

- *underlying behavior*, shared among all products, and
- *variation points*, differentiating between products

MTS is an LTS that distinguishes between may and must transitions (modeling *optional* or *mandatory* features, resp.)

MTS cannot model variability constraints regarding *alternative* features nor those regarding *requires* and *excludes* inter-feature relations

We will model such advanced variability constraints by means of an associated set of logical formulae expressed in our variability and action-based branching-time temporal logic vACTL

Modal Transition Systems (MTSs)

Originally introduced by Larsen & Thomsen @ LICS 1988

MTSs are now an established model to formalize a product family's

- *underlying behavior*, shared among all products, and
- *variation points*, differentiating between products

MTS is an LTS that distinguishes between may and must transitions (modeling *optional* or *mandatory* features, resp.)

MTS cannot model variability constraints regarding *alternative* features nor those regarding *requires* and *excludes* inter-feature relations

We will model such advanced variability constraints by means of an associated set of logical formulae expressed in our variability and action-based branching-time temporal logic vACTL

Definition of MTS

$(Q, A, \overline{q}, \rightarrow_{\Box}, \rightarrow_{\Diamond})$ is an MTS with

- *underlying* LTS $(Q, A, \overline{q}, \rightarrow_{\Box} \cup \rightarrow_{\Diamond})$
- *may* transition relation $\rightarrow_{\Diamond} \subseteq Q \times A \times Q$ (*possible* transitions)
- *must* transition relation $\rightarrow_{\Box} \subseteq Q \times A \times Q$ (*mandatory* transitions)

By definition, mandatory transitions must also be possible: $\rightarrow_{\Box} \subseteq \rightarrow_{\Diamond}$

The set of all must paths from q_1 is denoted by $\Box$-*path*(q_1)

$\sigma^{\Box} = q_1 a_1 q_2 a_2 q_3 \cdots$ is a *must path* (from q_1) if $q_i \xrightarrow{a_i}_{\Box} q_{i+1}\ \forall\, i > 0$

Subfamilies/products are obtained by preserving **all** must transitions, turning **some** may transitions into must transitions, and removing **some**/**all** remaining ones (resulting in MTS/LTS)

Definition of MTS

$(Q, A, \overline{q}, \rightarrow_\Box, \rightarrow_\Diamond)$ is an MTS with

- *underlying* LTS $(Q, A, \overline{q}, \rightarrow_\Box \cup \rightarrow_\Diamond)$
- *may* transition relation $\rightarrow_\Diamond \subseteq Q \times A \times Q$ (*possible* transitions)
- *must* transition relation $\rightarrow_\Box \subseteq Q \times A \times Q$ (*mandatory* transitions)

By definition, mandatory transitions must also be possible: $\rightarrow_\Box \subseteq \rightarrow_\Diamond$

The set of all must paths from q_1 is denoted by $\Box$-*path*(q_1)

$\sigma^\Box = q_1 a_1 q_2 a_2 q_3 \cdots$ is a *must path* (from q_1) if $q_i \xrightarrow{a_i}_\Box q_{i+1}\ \forall\, i > 0$

Subfamilies/products are obtained by preserving **all** must transitions, turning **some** may transitions into must transitions, and removing **some**/**all** remaining ones (resulting in MTS/LTS)

Definition of MTS

$(Q, A, \overline{q}, \rightarrow_\Box, \rightarrow_\Diamond)$ is an MTS with

- *underlying* LTS $(Q, A, \overline{q}, \rightarrow_\Box \cup \rightarrow_\Diamond)$
- *may* transition relation $\rightarrow_\Diamond \subseteq Q \times A \times Q$ (*possible* transitions)
- *must* transition relation $\rightarrow_\Box \subseteq Q \times A \times Q$ (*mandatory* transitions)

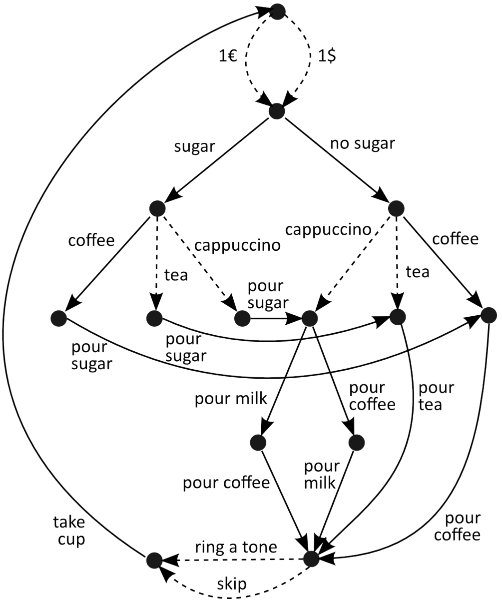
By definition, mandatory transitions must also be possible: $\rightarrow_\Box \subseteq \rightarrow_\Diamond$

The set of all must paths from q_1 is denoted by $\Box$-*path*(q_1)

$\sigma^\Box = q_1 a_1 q_2 a_2 q_3 \cdots$ is a *must path* (from q_1) if $q_i \xrightarrow{a_i}_\Box q_{i+1}\ \forall\, i > 0$

Subfamilies/products are obtained by preserving **all** must transitions, turning **some** may transitions into must transitions, and removing **some**/**all** remaining ones (resulting in MTS/LTS)

Running example: Coffee machine family



Static & behavioral requirements of product families

Static requirements: features constituting different products

- Only accepted coins are 1€, exclusively for European products, and 1$, exclusively for Canadian products (**alternative** features)
- All products offer coffee (**mandatory** feature); cappuccino only offered by European products (**excludes** relation among features)
- A ringtone is rung in products that offer cappuccino (**requires** relation among features)

Behavioral requirements: admitted sequences of operations

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

Static & behavioral requirements of product families

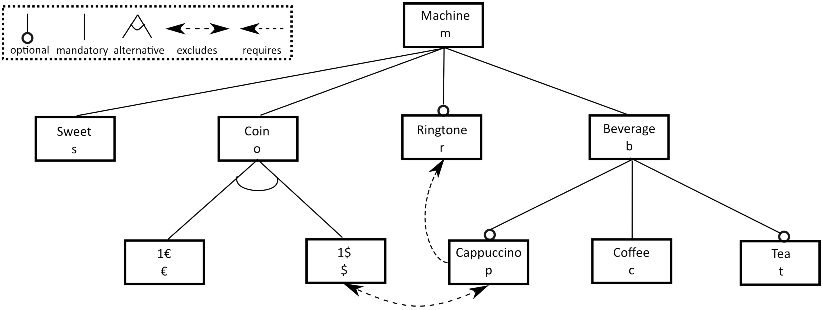
Static requirements: features constituting different products

- Only accepted coins are 1€, exclusively for European products, and 1$, exclusively for Canadian products (**alternative** features)
- All products offer coffee (**mandatory** feature); cappuccino only offered by European products (**excludes** relation among features)
- A ringtone is rung in products that offer cappuccino (**requires** relation among features)

Behavioral requirements: admitted sequences of operations

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

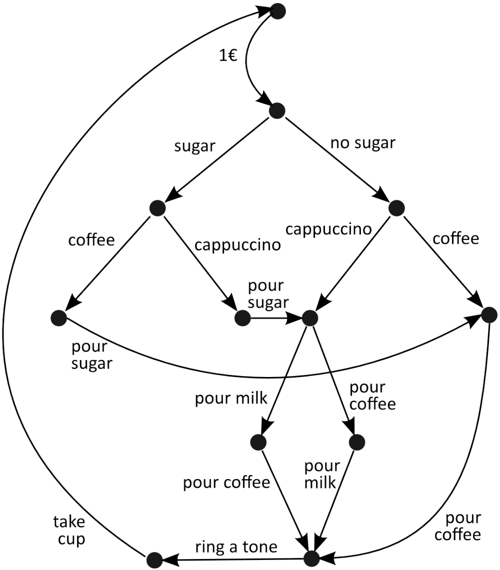
Coffee machine family: Feature model



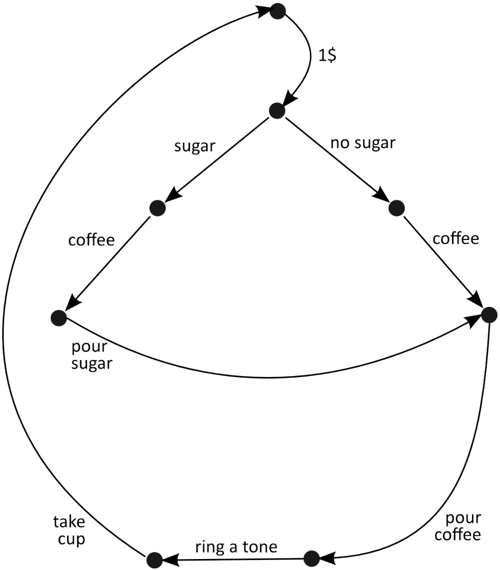
10 different valid products (coffee machines defined by features)

$$\{\{m, s, o, b, c, \text{€}\}, \{m, s, o, b, c, \text{€}, r\}, \{m, s, o, b, c, \text{€}, t\},$$
$$\{m, s, o, b, c, \$\}, \{m, s, o, b, c, \$, r\}, \{m, s, o, b, c, \$, t\},$$
$$\{m, s, o, b, c, \text{€}, t, r\}, \{m, s, o, b, c, \text{€}, p, r\},$$
$$\{m, s, o, b, c, \$, t, r\}, \{m, s, o, b, c, \text{€}, p, r, t\}\}$$

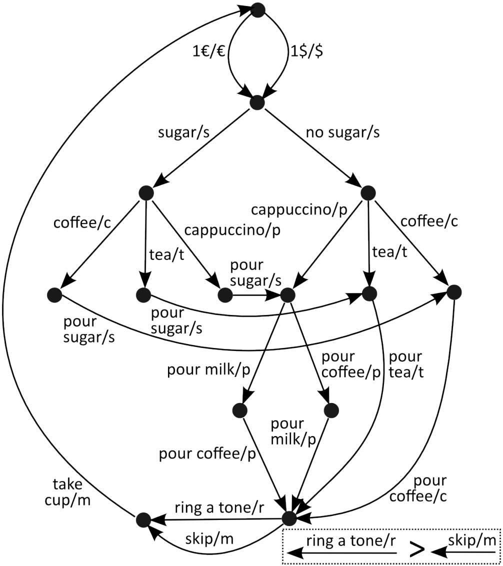
A European coffee machine ($\{m, s, o, b, c, €, p, r\}$)



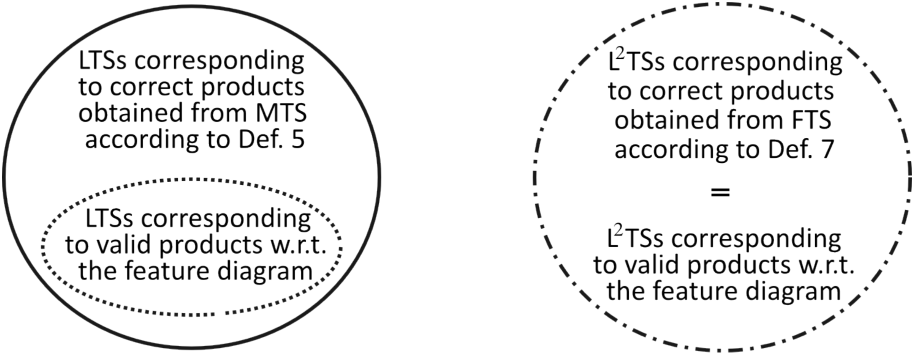
A Canadian coffee machine ($\{m, s, o, b, c, \$, r\}$)



FTS associated to feature model of family



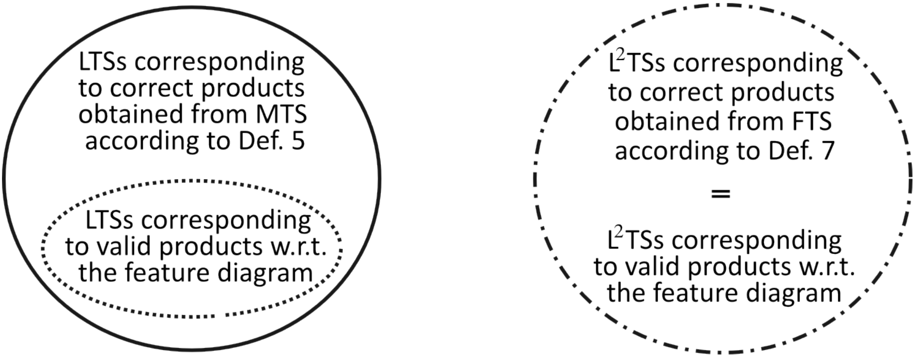
Product derivation: MTS versus FTS



FTS All and only products that are correct w.r.t. the requirements are derived (price: include a feature diagram in each FTS)

MTS Also correctly derived products may violate constraints of the type that MTSs cannot model (cf. LTS on next slide)

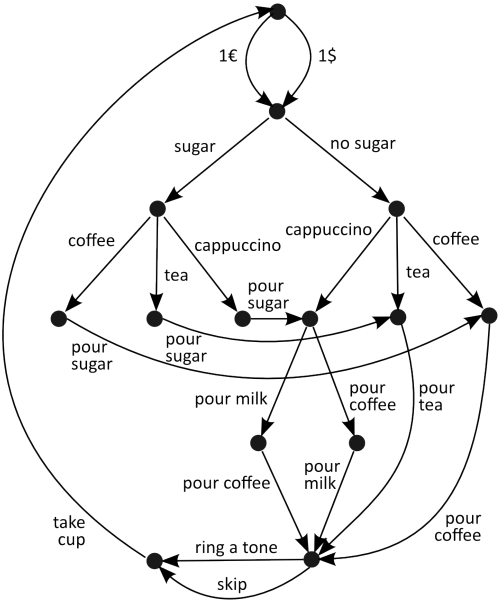
Product derivation: MTS versus FTS



FTS All and only products that are correct w.r.t. the requirements are derived (price: include a feature diagram in each FTS)

MTS Also correctly derived products may violate constraints of the type that MTSs cannot model (cf. LTS on next slide)

A correct but not valid product LTS of MTS



Definition of vACTL (slight revision of MHML from paper)

Variability and action-based branching-time temporal logic

A temporal logic based on the "Hennessy-Milner logic with until", but augmented with deontic O (*obligatory*) and P (*permitted*) operators, CTL's path operators E and A and ACTL's action-based Until operator, both with and without a deontic interpretation

Syntax of vACTL

$$\phi ::= \mathit{true} \mid \neg\,\phi \mid \phi \wedge \phi' \mid \langle a \rangle\, \phi \mid [a]\, \phi \mid \langle a \rangle^{\Box}\, \phi \mid [a]^{\Box}\, \phi \mid E\, \pi \mid A\, \pi$$

$$\pi ::= \phi\, \{\varphi\}\, U\, \{\varphi'\}\, \phi' \mid \phi\, \{\varphi\}\, U^{\Box}\, \{\varphi'\}\, \phi'$$

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

$\langle a \rangle^{\Box}$ and $[a]^{\Box}$ represent the classic deontic modalities O and P, resp.

Definition of vACTL (slight revision of MHML from paper)

Variability and action-based branching-time temporal logic

A temporal logic based on the "Hennessy-Milner logic with until", but augmented with deontic O (*obligatory*) and P (*permitted*) operators, CTL's path operators E and A and ACTL's action-based Until operator, both with and without a deontic interpretation

Syntax of vACTL

$$\phi ::= true \mid \neg\,\phi \mid \phi \wedge \phi' \mid \langle a \rangle\, \phi \mid [a]\, \phi \mid \langle a \rangle^{\Box}\, \phi \mid [a]^{\Box}\, \phi \mid E\,\pi \mid A\,\pi$$

$$\pi ::= \phi\ \{\varphi\}\ U\ \{\varphi'\}\ \phi' \mid \phi\ \{\varphi\}\ U^{\Box}\ \{\varphi'\}\ \phi'$$

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

$\langle a \rangle^{\Box}$ and $[a]^{\Box}$ represent the classic deontic modalities O and P, resp.

Definition of vACTL (slight revision of MHML from paper)

Variability and action-based branching-time temporal logic

A temporal logic based on the "Hennessy-Milner logic with until", but augmented with deontic O (*obligatory*) and P (*permitted*) operators, CTL's path operators E and A and ACTL's action-based Until operator, both with and without a deontic interpretation

Syntax of vACTL

$$\phi ::= true \mid \neg\,\phi \mid \phi \wedge \phi' \mid \langle a \rangle\, \phi \mid [a]\, \phi \mid \langle a \rangle^{\Box}\, \phi \mid [a]^{\Box}\, \phi \mid E\,\pi \mid A\,\pi$$

$$\pi ::= \phi\ \{\varphi\}\ U\ \{\varphi'\}\ \phi' \mid \phi\ \{\varphi\}\ U^{\Box}\ \{\varphi'\}\ \phi'$$

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

$\langle a \rangle^{\Box}$ and $[a]^{\Box}$ represent the classic deontic modalities O and P, resp.

vACTL: Semantics over MTS

- $q \models true$ always holds
- $q \models \neg\,\phi$ iff not $q \models \phi$
- $q \models \phi \wedge \phi'$ iff $q \models \phi$ and $q \models \phi'$
- $q \models \langle a\rangle\,\phi$ iff $\exists\, q' \in Q$ such that $q \xrightarrow{a}_{\Diamond} q'$, and $q' \models \phi$
- $q \models [a]\,\phi$ iff $\forall\, q' \in Q$ such that $q \xrightarrow{a}_{\Diamond} q'$, we have $q' \models \phi$
- $q \models \langle a\rangle^{\Box}\,\phi$ iff $\exists\, q' \in Q$ such that $q \xrightarrow{a}_{\Box} q'$, and $q' \models \phi$
- $q \models [a]^{\Box}\,\phi$ iff $\forall\, q' \in Q$ such that $q \xrightarrow{a}_{\Box} q'$, we have $q' \models \phi$
- $q \models E\,\pi$ iff $\exists\,\sigma' \in path(q)$ such that $\sigma' \models \pi$
- $q \models A\,\pi$ iff $\forall\,\sigma' \in path(q)$ such that $\sigma' \models \pi$
- $\sigma \models \phi\,\{\varphi\}\,U\,\{\varphi'\}\,\phi'$ iff $\exists\, j \geq 1 : \sigma(j) \models \phi', \sigma\{j\} \models \varphi'$, and $\sigma(j+1) \models \phi'$, and $\forall 1 \leq i < j : \sigma(i) \models \phi$ and $\sigma\{i\} \models \varphi$
- $\sigma \models \phi\,\{\varphi\}\,U^{\Box}\,\{\varphi'\}\,\phi'$ iff σ is a must path $\sigma^{\Box}$ and $\sigma^{\Box} \models \phi\,\{\varphi\}\,U\,\{\varphi'\}\,\phi'$

Abbreviations: $EF\phi = E(true\,\{true\}\,U\,\{true\}\,\phi)$; $EF^{\Box}\phi = E(true\,\{true\}\,U^{\Box}\,\{true\}\,\phi)$; $EF\,\{\varphi\}\,true = E(true\,\{true\}\,U\,\{\varphi\}\,true)$; $EF^{\Box}\{\varphi\}\,true = E(true\,\{true\}\,U^{\Box}\,\{\varphi\}\,true)$; $AG\,\phi = \neg EF\,\neg\phi$; $AG^{\Box}\,\phi = \neg EF^{\Box}\,\neg\phi$; etc.

vACTL: Semantics over MTS

- $q \models true$ always holds
- $q \models \neg\phi$ iff not $q \models \phi$
- $q \models \phi \wedge \phi'$ iff $q \models \phi$ and $q \models \phi'$
- $q \models \langle a \rangle\, \phi$ iff $\exists\, q' \in Q$ such that $q \xrightarrow{a}_{\Diamond} q'$, and $q' \models \phi$
- $q \models [a]\, \phi$ iff $\forall\, q' \in Q$ such that $q \xrightarrow{a}_{\Diamond} q'$, we have $q' \models \phi$
- $q \models \langle a \rangle^{\Box}\, \phi$ iff $\exists\, q' \in Q$ such that $q \xrightarrow{a}_{\Box} q'$, and $q' \models \phi$
- $q \models [a]^{\Box}\, \phi$ iff $\forall\, q' \in Q$ such that $q \xrightarrow{a}_{\Box} q'$, we have $q' \models \phi$
- $q \models E\,\pi$ iff $\exists\,\sigma' \in path(q)$ such that $\sigma' \models \pi$
- $q \models A\,\pi$ iff $\forall\,\sigma' \in path(q)$ such that $\sigma' \models \pi$
- $\sigma \models \phi\, \{\varphi\}\, U\, \{\varphi'\}\, \phi'$ iff $\exists\, j \geq 1 : \sigma(j) \models \phi'$, $\sigma\{j\} \models \varphi'$, and $\sigma(j+1) \models \phi'$, and $\forall 1 \leq i < j : \sigma(i) \models \phi$ and $\sigma\{i\} \models \varphi$
- $\sigma \models \phi\, \{\varphi\}\, U^{\Box}\, \{\varphi'\}\, \phi'$ iff σ is a must path $\sigma^{\Box}$ and $\sigma^{\Box} \models \phi\, \{\varphi\}\, U\, \{\varphi'\}\, \phi'$

Abbreviations: $EF\phi = E(true\,\{true\}\, U\,\{true\}\, \phi)$; $EF^{\Box}\phi = E(true\,\{true\}\, U^{\Box}\,\{true\}\,\phi)$; $EF\,\{\varphi\}\; true = E(true\,\{true\}\, U\,\{\varphi\}\; true)$; $EF^{\Box}\{\varphi\}\; true = E(true\,\{true\}\, U^{\Box}\,\{\varphi\}\; true)$; $AG\,\phi = \neg EF\,\neg\phi$; $AG^{\Box}\,\phi = \neg EF^{\Box}\,\neg\phi$; etc.

Advanced variability management

vaCTL can complement behavioral description of MTS by expressing constraints over possible products of a family that MTS cannot model

Template ALT: Features F1 and F2 are *alternative*

$$(EF^{\Box}\,\{F1\}\ true \vee EF^{\Box}\,\{F2\}\ true) \wedge \neg(EF\,\{F1\}\ true \wedge EF\,\{F2\}\ true)$$

Template EXC: Feature F1 *excludes* feature F2

$$((EF\,\{F1\}\ true) \Longrightarrow (AG\,\neg\langle F2\rangle\ true)) \wedge ((EF\,\{F2\}\ true) \Longrightarrow (AG\,\neg\langle F1\rangle\ true))$$

Template REQ: Feature F1 *requires* feature F2

$$(EF\,\{F1\}\ true) \Longrightarrow (EF^{\Box}\,\{F2\}\ true)$$

Define no full temporal ordering among the related features

Duty of behavioral LTS/MTS model, to be verified by vaCTL formulae

Advanced variability management

VACTL can complement behavioral description of MTS by expressing constraints over possible products of a family that MTS cannot model

Template ALT: Features F1 and F2 are *alternative*

$$(EF^{\square}\,\{F1\}\,true \vee EF^{\square}\,\{F2\}\,true) \wedge \neg(EF\,\{F1\}\,true \wedge EF\,\{F2\}\,true)$$

Template EXC: Feature F1 *excludes* feature F2

$$((EF\,\{F1\}\,true) \Longrightarrow (AG\,\neg\langle F2\rangle\,true)) \wedge ((EF\,\{F2\}\,true) \Longrightarrow (AG\,\neg\langle F1\rangle\,true))$$

Template REQ: Feature F1 *requires* feature F2

$$(EF\,\{F1\}\,true) \Longrightarrow (EF^{\square}\,\{F2\}\,true)$$

Define no full temporal ordering among the related features

Duty of behavioral LTS/MTS model, to be verified by VACTL formulae

Advanced variability management

vACTL can complement behavioral description of MTS by expressing constraints over possible products of a family that MTS cannot model

Template ALT: Features F1 and F2 are *alternative*

$$(EF^{\Box}\,\{F1\}\,true \vee EF^{\Box}\,\{F2\}\,true) \wedge \neg(EF\,\{F1\}\,true \wedge EF\,\{F2\}\,true)$$

Template EXC: Feature F1 *excludes* feature F2

$$((EF\,\{F1\}\,true) \Longrightarrow (AG\,\neg\langle F2\rangle\,true)) \wedge ((EF\,\{F2\}\,true) \Longrightarrow (AG\,\neg\langle F1\rangle\,true))$$

Template REQ: Feature F1 *requires* feature F2

$$(EF\,\{F1\}\,true) \;\Longrightarrow\; (EF^{\Box}\,\{F2\}\,true)$$

Define no full temporal ordering among the related features

Duty of behavioral LTS/MTS model, to be verified by vACTL formulae

Advanced variability management

VACTL can complement behavioral description of MTS by expressing constraints over possible products of a family that MTS cannot model

Template ALT: Features F1 and F2 are *alternative*

$$(EF^{\Box}\,\{F1\}\,true \vee EF^{\Box}\,\{F2\}\,true) \wedge \neg(EF\,\{F1\}\,true \wedge EF\,\{F2\}\,true)$$

Template EXC: Feature F1 *excludes* feature F2

$$((EF\,\{F1\}\,true) \Longrightarrow (AG\,\neg\langle F2\rangle\,true)) \wedge ((EF\,\{F2\}\,true) \Longrightarrow (AG\,\neg\langle F1\rangle\,true))$$

Template REQ: Feature F1 *requires* feature F2

$$(EF\,\{F1\}\,true) \implies (EF^{\Box}\,\{F2\}\,true)$$

Define no full temporal ordering among the related features

Duty of behavioral LTS/MTS model, to be verified by VACTL formulae

Advanced variability management

vaCTL can complement behavioral description of MTS by expressing constraints over possible products of a family that MTS cannot model

Template ALT: Features F1 and F2 are *alternative*

$$(EF^{\Box}\,\{F1\}\,true \vee EF^{\Box}\,\{F2\}\,true) \wedge \neg(EF\,\{F1\}\,true \wedge EF\,\{F2\}\,true)$$

Template EXC: Feature F1 *excludes* feature F2

$$((EF\,\{F1\}\,true) \Longrightarrow (AG\,\neg\langle F2\rangle\,true)) \wedge ((EF\,\{F2\}\,true) \Longrightarrow (AG\,\neg\langle F1\rangle\,true))$$

Template REQ: Feature F1 *requires* feature F2

$$(EF\,\{F1\}\,true) \Longrightarrow (EF^{\Box}\,\{F2\}\,true)$$

Define no full temporal ordering among the related features

Duty of behavioral LTS/MTS model, to be verified by vaCTL formulae

Model checking families and products

Verify property expressed as logical formula ψ over model T

- Decide whether $T \models \psi$, where $\models$ is the logic's satisfaction relation
- If $T \not\models \psi$, then it is usually easy to generate a counterexample
- If T is finite, model checking thus reduces to a graph search

On the fly: Only a fragment of the overall state space might need to be generated and analysed to be able to produce the correct result

VMC: `http://fmtlab.isti.cnr.it/vmc/`

An on-the-fly model-checker for vACTL is defined as particularization of the FMC model checker for ACTL over a CCS-like input language

Recently implemented also our algorithm for product derivation

- Explore and verify product families (MTS)
- Generate all valid products, explore and verify products (LTS)

Model checking families and products

Verify property expressed as logical formula ψ over model T

- Decide whether $T \models \psi$, where $\models$ is the logic's satisfaction relation
- If $T \not\models \psi$, then it is usually easy to generate a counterexample
- If T is finite, model checking thus reduces to a graph search

On the fly: Only a fragment of the overall state space might need to be generated and analysed to be able to produce the correct result

VMC: `http://fmtlab.isti.cnr.it/vmc/`

An on-the-fly model-checker for vACTL is defined as particularization of the FMC model checker for ACTL over a CCS-like input language

Recently implemented also our algorithm for product derivation

- Explore and verify product families (MTS)
- Generate all valid products, explore and verify products (LTS)

Model checking families and products

Verify property expressed as logical formula ψ over model T

- Decide whether $T \models \psi$, where $\models$ is the logic's satisfaction relation
- If $T \not\models \psi$, then it is usually easy to generate a counterexample
- If T is finite, model checking thus reduces to a graph search

On the fly: Only a fragment of the overall state space might need to be generated and analysed to be able to produce the correct result

VMC: `http://fmtlab.isti.cnr.it/vmc/`

An on-the-fly model-checker for vACTL is defined as particularization of the FMC model checker for ACTL over a CCS-like input language

Recently implemented also our algorithm for product derivation

- Explore and verify product families (MTS)
- Generate all valid products, explore and verify products (LTS)

Model checking temporal orderings

Property D: Always once a coffee has been selected, a coffee is eventually delivered

$$AG\ [coffee]\ AF^{\square}\ \{pour\ coffee\}\ true$$

Property E: A coffee machine may never deliver a coffee before a coin has been inserted

$$A\ [true\ \{\neg\ pour\ coffee\}\ U^{\square}\ \{1\$ \vee 1€\}\ true]$$

Property	MTS family	European	Canadian	LTS product
A: ALT	*false*	*true*	*true*	*false*
B: EXC	*false*	*true*	*true*	*false*
C: REQ	*false*	*true*	*true*	*true*
D	*true*	*true*	*true*	*true*
E	*true*	*true*	*true*	*true*

Model checking temporal orderings

Property D: Always once a coffee has been selected, a coffee is eventually delivered

$$AG\ [coffee]\ AF^{\Box}\ \{pour\ coffee\}\ true$$

Property E: A coffee machine may never deliver a coffee before a coin has been inserted

$$A\ [true\ \{\neg\ pour\ coffee\}\ U^{\Box}\ \{1\$ \vee 1€\}\ true]$$

Property	MTS family	European	Canadian	LTS product
A: ALT	*false*	*true*	*true*	*false*
B: EXC	*false*	*true*	*true*	*false*
C: REQ	*false*	*true*	*true*	*true*
D	*true*	*true*	*true*	*true*
E	*true*	*true*	*true*	*true*

Model checking temporal orderings

Property D: Always once a coffee has been selected, a coffee is eventually delivered

$$AG\ [coffee]\ AF^{\square}\ \{pour\ coffee\}\ true$$

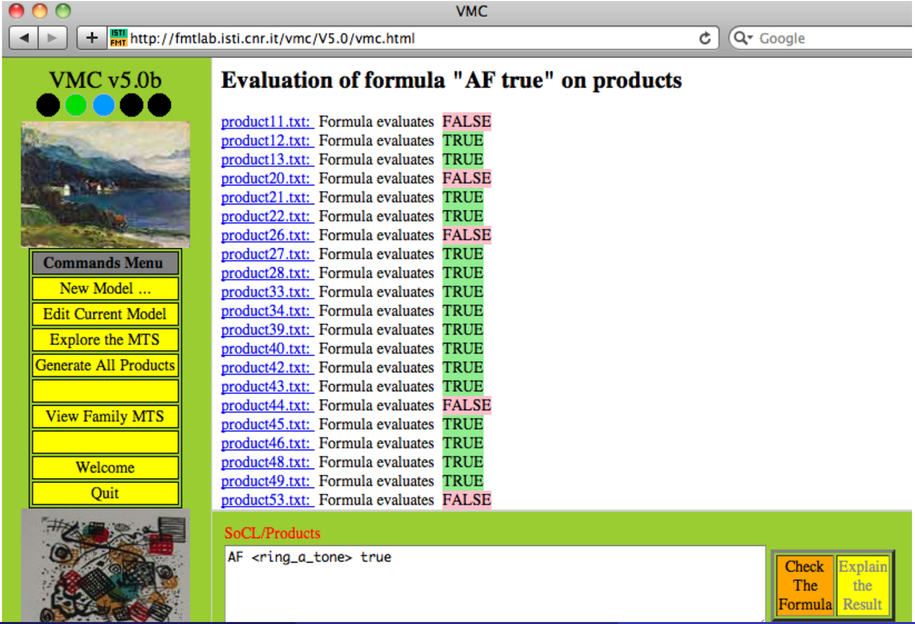
Property E: A coffee machine may never deliver a coffee before a coin has been inserted

$$A\ [true\ \{\neg\ pour\ coffee\}\ U^{\square}\ \{1\$ \vee 1€\}\ true]$$

Property	MTS family	European	Canadian	LTS product
A: ALT	*false*	*true*	*true*	*false*
B: EXC	*false*	*true*	*true*	*false*
C: REQ	*false*	*true*	*true*	*true*
D	*true*	*true*	*true*	*true*
E	*true*	*true*	*true*	*true*

MTS specification of coffee machine family in VMC

Result of ‘a coffee machine always eventually rings a tone’



VMC: Variability Model Checker

VMC can be used to automatically derive, from an MTS description of a product family **and** an associated set of vACTL formulae expressing further constraints for this family, **all** its valid products (i.e. a set of LTS descriptions of products, each one correct w.r.t. all vACTL constraints)

VMC can also be used to experiment with products, using vACTL to verify further (temporal) constraints, etc.

Future work required before possible application in industry

- A high-level language hiding all semantic details
  (investigate relation between features and actions)
- A predefined taxonomy for properties specified in vACTL
  (like specification patterns repository for LTL, (A)CTL, etc.)
- Scale to large, industrial-size product families
  (with many variation points and many features)

VMC: Variability Model Checker

VMC can be used to automatically derive, from an MTS description of a product family **and** an associated set of vACTL formulae expressing further constraints for this family, **all** its valid products (i.e. a set of LTS descriptions of products, each one correct w.r.t. all vACTL constraints)

VMC can also be used to experiment with products, using vACTL to verify further (temporal) constraints, etc.

Future work required before possible application in industry

- A high-level language hiding all semantic details
  (investigate relation between features and actions)
- A predefined taxonomy for properties specified in vACTL
  (like specification patterns repository for LTL, (A)CTL, etc.)
- Scale to large, industrial-size product families
  (with many variation points and many features)

VMC: Variability Model Checker

VMC can be used to automatically derive, from an MTS description of a product family **and** an associated set of vACTL formulae expressing further constraints for this family, **all** its valid products (i.e. a set of LTS descriptions of products, each one correct w.r.t. all vACTL constraints)

VMC can also be used to experiment with products, using vACTL to verify further (temporal) constraints, etc.

Future work required before possible application in industry

- A high-level language hiding all semantic details
  (investigate relation between features and actions)
- A predefined taxonomy for properties specified in vACTL
  (like specification patterns repository for LTL, (A)CTL, etc.)
- Scale to large, industrial-size product families
  (with many variation points and many features)

Publicity

Variability Modelling of Software-Intensive Systems (VaMoS'12)
6th International Workshop

Leipzig, Germany, January 25–27, 2012

⇒ `http://www.vamos-workshop.net`

Submission: October 30, 2011
Notification: November 27, 2011

with keynotes by:

- Paulo Borba, Centro de Informatica, Universidade Federal de Pernambuco, BR
- Charles Krueger, BigLever Software, Inc., USA