VMC: Recent Advances and Challenges Ahead

Maurice H. ter Beek and Franco Mazzanti

Formal Methods && Tools lab, ISTI–CNR, Pisa, Italy

SPLat 2014 @ SPLC 2014
Tuesday 16 September 2014
Florence, Italy

Disclaimer and acknowledgement

- My talk is about the variability model checker VMC, hence...
- normally I would now start by explaining what VMC is all about, but I assume that you stayed awake during Stefania's keynote ☺

- VMC is a product of the KandISTI family of model checkers that has been developed at ISTI–CNR over the last two decades
- Its main developer is Franco Mazzanti, but many other members of ISTI's Formal Methods && Tools lab have contributed to its development, in particular Stefania Gnesi & Alessandro Fantechi

- This family of model checkers has been developed during a series of EU projects (Agile, Sensoria, Quanticol, LearnPAd)

Outline

A family of model checkers

KandISTI 2014

CMC	FMC	VMC
	UMC	

http://fmt.isti.cnr.it/kandisti/

Main features

- Four different specification languages, one common verification engine
- Manually explore a system's evolutions and generate a summary of its behavior — powerful minimization techniques
- Investigate abstract system properties by using a branching-time temporal logic supported by an on-the-fly model checker
- The fragment of this logic without fixed-point operators allows verifications with a complexity which is linear with respect to the size of the model and the size of the formula
- Obtain a clear explanation of the model-checking results in terms of possible evolutions of the specific computational model
- Current limit for an exhaustive verification is a statespace of millions of states

FMC: origin of our on-the-fly model-checking approach

- A system is a hierarchical composition (net) of sequential automata (terms)
- Terms can be recursively defined using a simple process algebra which supports features coming from CCS, CSP and LOTOS
- Communication and synchronization among terms is achieved through synchronous operations over channels
- Term definitions can be parametrized, and communication operations allow (integer) value passing
- Bounded model-checking (also useful for explanations)
- Action-based branching-time logic inspired by ACTL and enriched with weak until operators, box/diamond operators and fixed-point operators

UMC: Support for state/event-based models and logics

ter Beek, Fantechi, Gnesi, Mazzanti @ *Sci. Comput. Program.*, 2011

- A system is a set of communicating UML-like state machines
- $\langle \mathit{Source} \rangle \rightarrow \langle \mathit{Target} \rangle \; \{ \, \langle \mathit{EventTrigger} \rangle \, [\langle \mathit{Guard} \rangle] \; / \; \langle \mathit{Actions} \rangle \, \}$
- UCTL: an action- and state-based logic
- Allows to express not only properties of evolution steps (i.e. related to the executed actions) but also internal properties of states (e.g. related to the values of object attributes)
- Actual system structure defined by a set of active object instantiations
- A full UMC model is defined by a sequence of Class and Objects declarations and by a final definition of a set of Abstraction rules
- The overall behavior of a system is formalized as an abstract L^2TS and abstraction rules allow to define what we want to see as labels of the states and edges of the L^2TS

CMC: Parametrized logic formulas for expressing data correlations among actions

Fantechi, Gnesi, Lapadula, Mazzanti, Pugliese, Tiezzi @ *ACM TOSEM*, 2012

- A system is specified in COWS: Calculus for Orchestration of Web Services
- Allows to model service publication and discovery, SLA negotiation, orchestration, service interactions and compositions, fault and compensation handling
- SocL: a logic allowing to express the correlation between dynamically generated values appearing inside actions at different times, representing the correlation values which allow, e.g., to relate the responses of a service to their specific request, or to handle the concept of a session involving a long sequence of interactions among the interacting partners
- $AG\ [\mathit{request}(p, \$id)]\ AF\ \mathit{response}(p, \%id)\ \mathit{true}$

VMC: Behavioral variability analysis for product families

ter Beek, Mazzanti, Sulova @ FM'12; ter Beek, Gnesi, Mazzanti @ SPLC'12

- A system is specified as a Modal Transition System in process-algebraic terms, together with an optional set of variability constraints
- VMC allows two kinds of behavioral variability analyses
  1. A logic property expressed in a variability-aware version of ACTL (v-ACTL) can directly be verified against the MTS modeling the family behavior, relying on the fact that under certain syntactic conditions the validity of the property over the MTS guarantees the validity of the same property for all products of the family
  2. The actual set of valid product behavior can explicitly be generated and the resulting LTSs can be verified against the same logic property (expressed in ACTL), which is surely less efficient than direct MTS verification but allows to precisely identify the set of features whose interactions may cause the original property to fail over the whole family

VMC: Behavioral variability analysis for product families

ter Beek, Mazzanti, Sulova @ FM'12; ter Beek, Gnesi, Mazzanti @ SPLC'12

- A system is specified as a Modal Transition System in process-algebraic terms, together with an optional set of variability constraints
- VMC allows two kinds of behavioral variability analyses
  1. A logic property expressed in a variability-aware version of ACTL (v-ACTL) can directly be verified against the MTS modeling the family behavior, relying on the fact that under certain syntactic conditions the validity of the property over the MTS guarantees the validity of the same property for all products of the family
  2. The actual set of valid product behavior can explicitly be generated and the resulting LTSs can be verified against the same logic property (expressed in ACTL), which is surely less efficient than direct MTS verification but allows to precisely identify the set of features whose interactions may cause the original property to fail over the whole family

VMC: Behavioral variability analysis for product families

ter Beek, Mazzanti, Sulova @ FM'12; ter Beek, Gnesi, Mazzanti @ SPLC'12

- A system is specified as a Modal Transition System in process-algebraic terms, together with an optional set of variability constraints
- VMC allows two kinds of behavioral variability analyses
  1. A logic property expressed in a variability-aware version of ACTL (v-ACTL) can directly be verified against the MTS modeling the family behavior, relying on the fact that under certain syntactic conditions the validity of the property over the MTS guarantees the validity of the same property for all products of the family
  2. The actual set of valid product behavior can explicitly be generated and the resulting LTSs can be verified against the same logic property (expressed in ACTL), which is surely less efficient than direct MTS verification but allows to precisely identify the set of features whose interactions may cause the original property to fail over the whole family

VMC (cont'd)

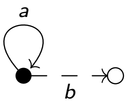
- The abstract model associated to this variability-oriented process algebra is an L^{2}TS in which edges are labelled with sets of labels, and where the additional 'may' label is added to the optional edges to specify variability — could easily add features as well
- v-ACTL is built over a subset of ACTL, but enriched with deontic operators $AF^{\Box}$, $EF^{\Box}$ and $\langle\,\rangle^{\Box}$, which are actually implemented in VMC by a translation into plain ACTL

$$T = a.T + b(may).nil$$

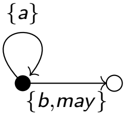
$\langle a\rangle^{\Box}\ true$ $\qquad$ $\langle a \wedge \neg\, may\rangle\ true$

VMC (cont'd)

- The abstract model associated to this variability-oriented process algebra is an L^{2}TS in which edges are labelled with sets of labels, and where the additional 'may' label is added to the optional edges to specify variability — could easily add features as well
- v-ACTL is built over a subset of ACTL, but enriched with deontic operators $AF^{\Box}$, $EF^{\Box}$ and $\langle\,\rangle^{\Box}$, which are actually implemented in VMC by a translation into plain ACTL



$$T = a.T + b(may).nil$$

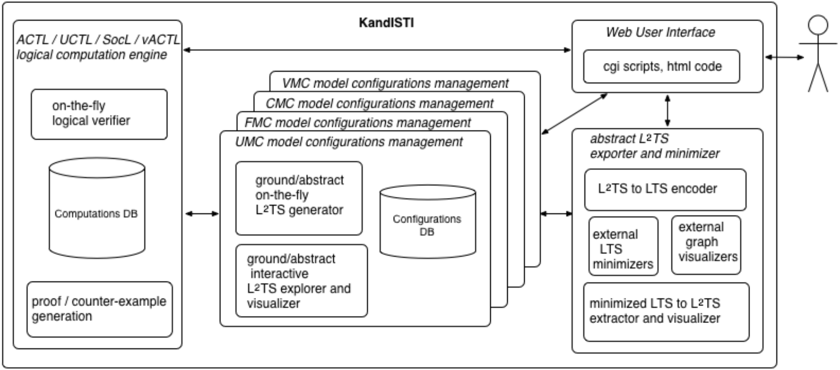


$\langle a\rangle^{\Box}$ *true* $\qquad$ $\langle a \wedge \neg\, may\rangle$ *true*

Overall structure of the model checkers

Statespace generation (depending on the underlying computational model) is separated from L^{2}TS analysis

Explicit abstraction mechanism allows to specify which details of the model become observable labels on the L^{2}TS' states and transitions



VMC: Variability Model Checker

Efficient bounded, on-the-fly model checking, providing explanations

Profits from optimizations of KandISTI

VMC now accepts as input a specification in a value-passing modal process algebra, possibly with additional *n*-ary 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 v-ACTL properties over MTSs/LTSs
- interactively explain why a property is (not) satisfied

VMC (v6.0, released in June 2014) is freely usable online:

`http://fmt.isti.cnr.it/vmc/`

VMC: Variability Model Checker

Efficient bounded, on-the-fly model checking, providing explanations

Profits from optimizations of KandISTI

VMC now accepts as input a specification in a value-passing modal process algebra, possibly with additional *n*-ary 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 v-ACTL properties over MTSs/LTSs
- interactively explain why a property is (not) satisfied

VMC (v6.0, released in June 2014) is freely usable online:

`http://fmt.isti.cnr.it/vmc/`

Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible may and necessary must transitions
  Larsen, Thomsen @ LICS'88 (each necessary transition is also admissible)
- Recognized as a useful model to describe in a compact way the possible behavior of all the products (LTS) of a product family
  Fischbein, Uchitel, Braberman @ ROSATEA'06; Fantechi, Gnesi @ ESEC/FSE'07, SPLC'08; Larsen, Nyman, Wąsowski @ ESOP'07; Lauenroth, Pohl, Töhning @ ASE'09
- MTS cannot model variability constraints regarding alternative features, nor regarding requires/excludes inter-feature relations
  Asirelli, ter Beek, Fantechi, Gnesi @ iFM'10
- Our solution: add a set of variability constraints to the MTS to be able to decide which derivable products (LTS) are valid ones
  Asirelli, ter Beek, Fantechi, Gnesi @ SPLC'11

Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible may and necessary must transitions
  Larsen, Thomsen @ LICS'88 (each necessary transition is also admissible)
- Recognized as a useful model to describe in a compact way the possible behavior of all the products (LTS) of a product family
  Fischbein, Uchitel, Braberman @ ROSATEA'06; Fantechi, Gnesi @ ESEC/FSE'07, SPLC'08; Larsen, Nyman, Wąsowski @ ESOP'07; Lauenroth, Pohl, Töhning @ ASE'09
- MTS cannot model variability constraints regarding alternative features, nor regarding requires/excludes inter-feature relations
  Asirelli, ter Beek, Fantechi, Gnesi @ iFM'10
- Our solution: add a set of variability constraints to the MTS to be able to decide which derivable products (LTS) are valid ones
  Asirelli, ter Beek, Fantechi, Gnesi @ SPLC'11

Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible may and necessary must transitions
  Larsen, Thomsen @ LICS'88 (each necessary transition is also admissible)
- Recognized as a useful model to describe in a compact way the possible behavior of all the products (LTS) of a product family
  Fischbein, Uchitel, Braberman @ ROSATEA'06; Fantechi, Gnesi @ ESEC/FSE'07, SPLC'08; Larsen, Nyman, Wąsowski @ ESOP'07; Lauenroth, Pohl, Töhning @ ASE'09
- MTS cannot model variability constraints regarding alternative features, nor regarding requires/excludes inter-feature relations
  Asirelli, ter Beek, Fantechi, Gnesi @ iFM'10
- Our solution: add a set of variability constraints to the MTS to be able to decide which derivable products (LTS) are valid ones
  Asirelli, ter Beek, Fantechi, Gnesi @ SPLC'11

Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible may and necessary must transitions
  Larsen, Thomsen @ LICS'88 (each necessary transition is also admissible)
- Recognized as a useful model to describe in a compact way the possible behavior of all the products (LTS) of a product family
  Fischbein, Uchitel, Braberman @ ROSATEA'06; Fantechi, Gnesi @ ESEC/FSE'07, SPLC'08; Larsen, Nyman, Wąsowski @ ESOP'07; Lauenroth, Pohl, Töhning @ ASE'09
- MTS cannot model variability constraints regarding alternative features, nor regarding requires/excludes inter-feature relations
  Asirelli, ter Beek, Fantechi, Gnesi @ iFM'10
- Our solution: add a set of variability constraints to the MTS to be able to decide which derivable products (LTS) are valid ones
  Asirelli, ter Beek, Fantechi, Gnesi @ SPLC'11

Refinement (classic)

- Implementations of an MTS are LTSs: refine a partial description (with dotted edges) into a more detailed one (only solid edges), reflecting increased knowledge on admissible but not necessary behavior
- Does fit SPLE notion that each product of a product family is a refinement of that family, based on the understanding that an LTS (product behavior) conforms to the MTS (family behavior)

- But may result in a possibly infinite set of, potentially not bisimilar, implementations (explosion due to the unbounded expansion of loops)

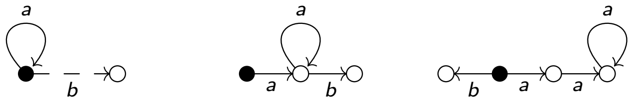
Refinement (classic)

- Implementations of an MTS are LTSs: refine a partial description (with dotted edges) into a more detailed one (only solid edges), reflecting increased knowledge on admissible but not necessary behavior
- Does fit SPLE notion that each product of a product family is a refinement of that family, based on the understanding that an LTS (product behavior) conforms to the MTS (family behavior)

- But may result in a possibly infinite set of, potentially not bisimilar, implementations (explosion due to the unbounded expansion of loops)

Refinement (classic)

- Implementations of an MTS are LTSs: refine a partial description (with dotted edges) into a more detailed one (only solid edges), reflecting increased knowledge on admissible but not necessary behavior
- Does fit SPLE notion that each product of a product family is a refinement of that family, based on the understanding that an LTS (product behavior) conforms to the MTS (family behavior)



- But may result in a possibly infinite set of, potentially not bisimilar, implementations (explosion due to the unbounded expansion of loops)

Refinement (implemented in VMC)

- VMC: a simpler notion of refinement preserves the exact original branching structure of the MTS in the products, corresponding to the modelers' intuition ("a feature either is present or is not")
- Implementations: "turn a dotted edge into a solid edge or remove it altogether"

- Immediate advantage: limited set of product behaviors

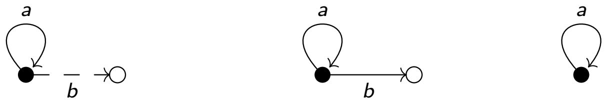
Refinement (implemented in VMC)

- VMC: a simpler notion of refinement preserves the exact original branching structure of the MTS in the products, corresponding to the modelers' intuition ("a feature either is present or is not")
- Implementations: "turn a dotted edge into a solid edge or remove it altogether"

- Immediate advantage: limited set of product behaviors

Refinement (implemented in VMC)

- VMC: a simpler notion of refinement preserves the exact original branching structure of the MTS in the products, corresponding to the modelers' intuition ("a feature either is present or is not")
- Implementations: "turn a dotted edge into a solid edge or remove it altogether"



- Immediate advantage: limited set of product behaviors

Assumptions (implemented in VMC)

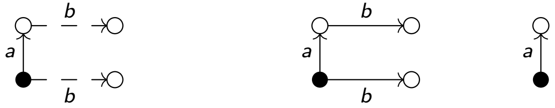
- An action models a piece of functionality (a feature if you like)
- Coherence: no solid edge is labeled with the same action as a (different) dotted edge ("a feature either is optional or is not")
- Consistency: a decision to 'implement' a may but not must transition in a product must be *consistent* throughout the product, reflecting the fact that functionality (or a feature) either is or is not present in a product, independent of its behavioral context

Assumptions (implemented in VMC)

- An action models a piece of functionality (a feature if you like)
- Coherence: no solid edge is labeled with the same action as a (different) dotted edge ("a feature either is optional or is not")
- Consistency: a decision to 'implement' a may but not must transition in a product must be *consistent* throughout the product, reflecting the fact that functionality (or a feature) either is or is not present in a product, independent of its behavioral context

Assumptions (implemented in VMC)

- An action models a piece of functionality (a feature if you like)
- Coherence: no solid edge is labeled with the same action as a (different) dotted edge ("a feature either is optional or is not")
- Consistency: a decision to 'implement' a may but not must transition in a product must be *consistent* throughout the product, reflecting the fact that functionality (or a feature) either is or is not present in a product, independent of its behavioral context



Variability constraints (implemented in VMC)

Variability constraints of the form ALTernative, EXCludes, REQuires, but also compositions like a REQ (b ALT c)

a ALT b

Semantics in ACTL:

a_1 ALT a_2 ALT $\cdots$ ALT a_n :

$$\bigvee_{1\le i\le n}((EF\,\{a_i\}\,true) \bigwedge_{1\le j\ne i\le n}(\neg\, EF\,\{a_j\}\,true))$$

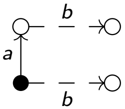
a_1 OR a_2 OR $\cdots$ OR a_n : $\bigvee_{1\le i\le n}(EF\,\{a_i\}\,true)$

a_j EXC a_k : $(EF\,\{a_j\}\,true) \implies (\neg\, EF\,\{a_k\}\,true) \wedge$
$(EF\,\{a_k\}\,true) \implies (\neg\, EF\,\{a_j\}\,true)$

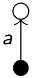
a_j REQ a_k : $(EF\,\{a_j\}\,true) \implies (EF\,\{a_k\}\,true)$

Variability constraints (implemented in VMC)

Variability constraints of the form ALTernative, EXCludes, REQuires, but also compositions like a REQ (b ALT c)



a ALT b



Semantics in ACTL:

a_1 ALT a_2 ALT $\cdots$ ALT a_n :

$$\bigvee_{1 \leq i \leq n}((EF\,\{a_i\}\,true) \bigwedge_{1 \leq j \neq i \leq n}(\neg\, EF\,\{a_j\}\,true))$$

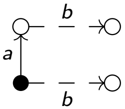
a_1 OR a_2 OR $\cdots$ OR a_n : $\bigvee_{1 \leq i \leq n}(EF\,\{a_i\}\,true)$

a_j EXC a_k : $(EF\,\{a_j\}\,true) \implies (\neg\, EF\,\{a_k\}\,true) \wedge$
$(EF\,\{a_k\}\,true) \implies (\neg\, EF\,\{a_j\}\,true)$

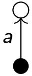
a_j REQ a_k : $(EF\,\{a_j\}\,true) \implies (EF\,\{a_k\}\,true)$

Variability constraints (implemented in VMC)

Variability constraints of the form ALTernative, EXCludes, REQuires, but also compositions like a REQ (b ALT c)



a ALT b



Semantics in ACTL:

a_1 ALT a_2 ALT $\cdots$ ALT a_n :

$$\bigvee_{1 \leq i \leq n} ((EF\,\{a_i\}\,true) \bigwedge_{1 \leq j \neq i \leq n} (\neg\, EF\,\{a_j\}\,true))$$

a_1 OR a_2 OR $\cdots$ OR a_n : $\bigvee_{1 \leq i \leq n} (EF\,\{a_i\}\,true)$

a_j EXC a_k : $(EF\,\{a_j\}\,true) \implies (\neg\, EF\,\{a_k\}\,true) \wedge$

$$(EF\,\{a_k\}\,true) \implies (\neg\, EF\,\{a_j\}\,true)$$

a_j REQ a_k : $(EF\,\{a_j\}\,true) \implies (EF\,\{a_k\}\,true)$

Product derivation (implemented in VMC)

A product LTS is obtained from a coherent family MTS as follows:

1. include all (reachable) must transitions (solid edges)
2. include subset of the (reachable) may but not must transitions (dotted edges), remove others
3. satisfy consistency assumption
4. satisfy all variability constraints

Each selection gives rise to a different variant

Modeling MTS/LTS as L^2TS makes VMC a product of the KandISTI family of model checkers

Product derivation (implemented in VMC)

A product LTS is obtained from a coherent family MTS as follows:

1. include all (reachable) must transitions (solid edges)
2. include subset of the (reachable) may but not must transitions (dotted edges), remove others
3. satisfy consistency assumption
4. satisfy all variability constraints

Each selection gives rise to a different variant

Modeling MTS/LTS as L^2TS makes VMC a product of the KandISTI family of model checkers

A value-passing modal process algebra (input to VMC)

Let $\mathcal{A}$ be a set of actions, let $a \in \mathcal{A}$ and let $L \subseteq \mathcal{A}$
Processes are built from terms and actions according to the syntax

$$
\begin{array}{lcl}
N & ::= & [P] \\
P & ::= & K(e) \;\mid\; P \,/L/\, P
\end{array}
$$

$[P]$ denotes a closed system, i.e. it cannot evolve on input actions $a(?v)$
$K(e)$ is process identifier from set of process definitions of form $K(v) \stackrel{\text{def}}{=} T$

$$
\begin{array}{lcl}
T & ::= & nil \;\mid\; K(e) \;\mid\; A.T \;\mid\; T+T \;\mid\; [e \bowtie e]\,T \\
A & ::= & a(e) \;\mid\; a(may, e) \;\mid\; a(?v) \;\mid\; a(may, ?v) \\
e & ::= & v \;\mid\; \textbf{int} \;\mid\; e \pm e
\end{array}
$$

$\bowtie \in \{<, \leq, =, \neq, \geq, >\}$, v is a variable, **int** is an integer, $\pm \in \{+, -, \times, \div\}$

A value-passing modal process algebra (input to VMC)

Let $\mathcal{A}$ be a set of actions, let $a \in \mathcal{A}$ and let $L \subseteq \mathcal{A}$
Processes are built from terms and actions according to the syntax

$$\begin{array}{lcl} N & ::= & [P] \\ P & ::= & K(e) \mid P\,/L/\,P \end{array}$$

$[P]$ denotes a closed system, i.e. it cannot evolve on input actions $a(?v)$
$K(e)$ is process identifier from set of process definitions of form $K(v) \stackrel{\text{def}}{=} T$

$$\begin{array}{lcl} T & ::= & nil \mid K(e) \mid A.T \mid T + T \mid [e \bowtie e]\, T \\ A & ::= & a(e) \mid a(may, e) \mid a(?v) \mid a(may, ?v) \\ e & ::= & v \mid \textbf{int} \mid e \pm e \end{array}$$

$\bowtie \in \{<, \leq, =, \neq, \geq, >\}$, v is a variable, **int** is an integer, $\pm \in \{+, -, \times, \div\}$

Processes

nil	terminated process that has finished execution
K	process identifier that is used for modelling recursive sequential processes
$A.P$	process that can execute action A and then behave as P
$P + Q$	process that can non-deterministically choose to behave as P or as Q
$P\,/L/\,Q$	process formed by the parallel composition of P and Q (may synchronize on actions in L and interleave others)

Distinguish must actions a and may but not must actions $a\,(may)$
Each action type is treated differently in the SOS semantics

Processes

nil	terminated process that has finished execution
K	process identifier that is used for modelling recursive sequential processes
$A.P$	process that can execute action A and then behave as P
$P + Q$	process that can non-deterministically choose to behave as P or as Q
$P\,/L/\,Q$	process formed by the parallel composition of P and Q (may synchronize on actions in L and interleave others)

Distinguish must actions a and may but not must actions $a\,(may)$
Each action type is treated differently in the SOS semantics

Semantics in SOS style over MTS

$$(\text{sys})\ \frac{P \xrightarrow{a(e)} P'}{[P] \xrightarrow{a(e)} [P']} \qquad\qquad (\text{act}_{\Box})\ \frac{}{\alpha.P \xrightarrow{\alpha} P}\ {\scriptstyle \alpha\in\{a(e),a(?v)\}}$$

$$(\text{or}_{\Box})\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}\ {\scriptstyle \alpha\in\{a(e),a(?v)\}} \qquad\qquad (\text{int}_{\Box})\ \frac{P \xrightarrow{\ell} P'}{P \,/L/\, Q \xrightarrow{\ell} P' \,/L/\, Q}\ {\scriptstyle \ell\notin L}$$

$$(\text{par}_{\Box})\ \frac{P \xrightarrow{a(e_1)} P' \qquad Q \xrightarrow{a(e_2)} Q'}{P \,/L/\, Q \xrightarrow{a} P' \,/L/\, Q'}\ {\scriptstyle a\in L,\, e_1=e_2} \qquad (\text{par}_{\Box})\ \frac{P \xrightarrow{a(?v)} P' \qquad Q \xrightarrow{a(e)} Q'}{P \,/L/\, Q \xrightarrow{a} P'[{}^{e}/_{v}] \,/L/\, Q'}\ {\scriptstyle a\in L}$$

$$(\text{guard})\ \frac{}{[e_1 \bowtie e_2]\, P(e_3) \rightarrow P(e_3)}\ {\scriptstyle e_1\bowtie e_2}$$

Similarly in case of may actions and for the remaining operators

Semantics in SOS style over MTS

$$(\text{sys})\ \frac{P \stackrel{a(e)}{\rightarrow} P'}{[P] \stackrel{a(e)}{\rightarrow} [P']} \qquad (\text{act}_\Box)\ \frac{}{\alpha.P \stackrel{\alpha}{\rightarrow}_\Box P}\ {\scriptstyle \alpha\in\{a(e),a(?v)\}}$$

$$(\text{or}_\Box)\ \frac{P \stackrel{\alpha}{\rightarrow}_\Box P'}{P + Q \stackrel{\alpha}{\rightarrow}_\Box P'}\ {\scriptstyle \alpha\in\{a(e),a(?v)\}} \qquad (\text{int}_\Box)\ \frac{P \stackrel{\ell}{\rightarrow}_\Box P'}{P\,/L/\,Q \stackrel{\ell}{\rightarrow}_\Box P'\,/L/\,Q}\ {\scriptstyle \ell\notin L}$$

$$(\text{par}_\Box)\ \frac{P \stackrel{a(e_1)}{\rightarrow}_\Box P' \quad Q \stackrel{a(e_2)}{\rightarrow}_\Box Q'}{P\,/L/\,Q \stackrel{a}{\rightarrow}_\Box P'\,/L/\,Q'}\ {\scriptstyle a\in L,\, e_1=e_2} \qquad (\text{par}_\Box)\ \frac{P \stackrel{a(?v)}{\rightarrow}_\Box P' \quad Q \stackrel{a(e)}{\rightarrow}_\Box Q'}{P\,/L/\,Q \stackrel{a}{\rightarrow}_\Box P'[^e/_v]\,/L/\,Q'}\ {\scriptstyle a\in L}$$

$$(\text{guard})\ \frac{}{[e_1 \bowtie e_2]\, P(e_3) \rightarrow P(e_3)}\ {\scriptstyle e_1 \bowtie e_2}$$

Similarly in case of may actions and for the remaining operators

ACTL: action-based CTL

Syntax over action formulas (boolean compositions of actions, denoted by ψ), state formulas (ϕ) and path formulas (π)

$$\begin{aligned}
\phi &::= \mathit{true} \mid \neg\phi \mid \phi \wedge \phi \mid [\psi]\,\phi \mid \langle\psi\rangle\,\phi \mid E\,\pi \mid A\,\pi \mid \\
&\quad\;\; \mu Y.\phi(Y) \mid \nu Y.\phi(Y) \\
\pi &::= X\,\{\psi\}\,\phi \mid [\phi\,\{\psi\}\,U\,\{\psi'\}\,\phi'] \mid [\phi\,\{\psi\}\,W\,\{\psi'\}\,\phi'] \mid \\
&\quad\;\; [\phi\,\{\psi\}\,U\,\phi'] \mid [\phi\,\{\psi\}\,W\,\phi'] \mid F\,\phi \mid F\,\{\psi\}\,\phi \mid G\,\phi
\end{aligned}$$

(Y is a propositional variable, $\phi(Y)$ is syntactically monotone in Y)

μ and ν: recursion ("finite looping"/"liveness" and "looping"/"safety")

X, U, W, F: action-based neXt, Until, Weak until, Future ("eventually")

ACTL: action-based CTL

Syntax over action formulas (boolean compositions of actions, denoted by ψ), state formulas (ϕ) and path formulas (π)

$$\begin{aligned}
\phi \; &::= \; \mathit{true} \;\mid\; \neg\,\phi \;\mid\; \phi \wedge \phi \;\mid\; [\psi]\,\phi \;\mid\; \langle\psi\rangle\,\phi \;\mid\; E\,\pi \;\mid\; A\,\pi \;\mid \\
&\quad\;\; \mu\, Y.\phi(Y) \;\mid\; \nu\, Y.\phi(Y) \\
\pi \; &::= \; X\,\{\psi\}\;\phi \;\mid\; [\phi\;\{\psi\}\,U\,\{\psi'\}\;\phi'] \;\mid\; [\phi\;\{\psi\}\,W\,\{\psi'\}\;\phi'] \;\mid \\
&\quad\;\; [\phi\;\{\psi\}\,U\;\phi'] \mid [\phi\;\{\psi\}\,W\;\phi'] \mid F\,\phi \;\mid\; F\,\{\psi\}\;\phi \;\mid\; G\,\phi
\end{aligned}$$

(Y is a propositional variable, $\phi(Y)$ is syntactically monotone in Y)

μ and ν: recursion ("finite looping"/"liveness" and "looping"/"safety")

X, U, W, F: action-based neXt, Until, Weak until, Future ("eventually")

ACTL: action-based CTL

Syntax over action formulas (boolean compositions of actions, denoted by ψ), state formulas (ϕ) and path formulas (π)

$$
\begin{aligned}
\phi \;::=\;& \mathit{true} \;\mid\; \neg\,\phi \;\mid\; \phi \wedge \phi \;\mid\; [\psi]\,\phi \;\mid\; \langle\psi\rangle\,\phi \;\mid\; E\,\pi \;\mid\; A\,\pi \;\mid\\
& \mu\,Y.\phi(Y) \;\mid\; \nu\,Y.\phi(Y)\\
\pi \;::=\;& X\,\{\psi\}\;\phi \;\mid\; [\phi\;\{\psi\}\,U\,\{\psi'\}\;\phi'] \;\mid\; [\phi\;\{\psi\}\,W\,\{\psi'\}\;\phi'] \;\mid\\
& [\phi\;\{\psi\}\,U\;\phi'] \mid [\phi\;\{\psi\}\,W\;\phi'] \mid F\,\phi \;\mid\; F\,\{\psi\}\;\phi \;\mid\; G\,\phi
\end{aligned}
$$

(Y is a propositional variable, $\phi(Y)$ is syntactically monotone in Y)

μ and ν: recursion (“finite looping”/“liveness” and “looping”/“safety”)

X, U, W, F: action-based neXt, Until, Weak until, Future (“eventually”)

v-ACTL: variability-aware ACTL (implemented in VMC)

v-ACTL$^\Box$:

$$\phi ::= \textit{false} \mid \textit{true} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\psi]\,\phi \mid \langle\psi\rangle^\Box\,\phi \mid$$
$$\textit{EF}^\Box\,\phi \mid \textit{EF}^\Box\,\{\psi\}\,\phi \mid \textit{AF}^\Box\,\phi \mid \textit{AF}^\Box\,\{\psi\}\,\phi \mid \textit{AG}\,\phi$$

any formula that is true for MTS, is also true for all products (LTSs)

v-ACTL$^-$:

$$\chi ::= \textit{false} \mid \textit{true} \mid \chi \wedge \chi \mid \chi \vee \chi \mid \langle\psi\rangle\,\chi \mid$$
$$\textit{EF}\,\chi \mid \textit{EF}\{\psi\}\,\chi$$

any formula that is false for MTS, is also false for all products (LTSs)

The latest version of VMC notifies the user whenever preservation of a verification result is applicable

v-ACTL: variability-aware ACTL (implemented in VMC)

v-ACTL$^{\Box}$:

$$\phi ::= \mathit{false} \mid \mathit{true} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\psi]\,\phi \mid \langle\psi\rangle^{\Box}\,\phi \mid$$
$$EF^{\Box}\,\phi \mid EF^{\Box}\,\{\psi\}\,\phi \mid AF^{\Box}\,\phi \mid AF^{\Box}\,\{\psi\}\,\phi \mid AG\,\phi$$

any formula that is true for MTS, is also true for all products (LTSs)

v-ACTL^{-}:

$$\chi ::= \mathit{false} \mid \mathit{true} \mid \chi \wedge \chi \mid \chi \vee \chi \mid \langle\psi\rangle\,\chi \mid$$
$$EF\,\chi \mid EF\{\psi\}\,\chi$$

any formula that is false for MTS, is also false for all products (LTSs)

The latest version of VMC notifies the user whenever preservation of a verification result is applicable

v-ACTL: variability-aware ACTL (implemented in VMC)

v-ACTL$^\square$:

$$\phi ::= \textit{false} \mid \textit{true} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\psi]\,\phi \mid \langle\psi\rangle^\square\,\phi \mid$$
$$\textit{EF}^\square\,\phi \mid \textit{EF}^\square\,\{\psi\}\,\phi \mid \textit{AF}^\square\,\phi \mid \textit{AF}^\square\,\{\psi\}\,\phi \mid \textit{AG}\,\phi$$

any formula that is true for MTS, is also true for all products (LTSs)

v-ACTL$^-$:

$$\chi ::= \textit{false} \mid \textit{true} \mid \chi \wedge \chi \mid \chi \vee \chi \mid \langle\psi\rangle\,\chi \mid$$
$$\textit{EF}\,\chi \mid \textit{EF}\{\psi\}\,\chi$$

any formula that is false for MTS, is also false for all products (LTSs)

The latest version of VMC notifies the user whenever preservation of a verification result is applicable

Informal semantics of v-ACTL

$[\psi]\,\phi$ in all next states reachable by a may transition executing an action satisfying ψ, ϕ holds

$[\psi]^{\Box}\,\phi$ in all next states reachable by a must transition executing an action satisfying ψ, ϕ holds

$\langle\psi\rangle\,\phi \equiv \neg[\psi]\,\neg\phi$ a next state exists, reachable by a may transition executing an action satisfying ψ, in which ϕ holds

$\langle\psi\rangle^{\Box}\phi \equiv \neg[\psi]^{\Box}\neg\phi$ a next state exists, reachable by a must transition executing an action satisfying ψ, in which ϕ holds

($\langle\psi\rangle^{\Box}$ and $[\psi]^{\Box}$ represent the classic deontic modalities O and P)

Informal semantics of v-ACTL

$[\psi]\,\phi$ in all next states reachable by a may transition executing an action satisfying ψ, ϕ holds

$[\psi]^{\Box}\,\phi$ in all next states reachable by a must transition executing an action satisfying ψ, ϕ holds

$\langle\psi\rangle\,\phi \equiv \neg[\psi]\,\neg\phi$ a next state exists, reachable by a may transition executing an action satisfying ψ, in which ϕ holds

$\langle\psi\rangle^{\Box}\phi \equiv \neg[\psi]^{\Box}\neg\phi$ a next state exists, reachable by a must transition executing an action satisfying ψ, in which ϕ holds

($\langle\psi\rangle^{\Box}$ and $[\psi]^{\Box}$ represent the classic deontic modalities O and P)

Informal semantics of v-ACTL

$[\psi]\,\phi$ in all next states reachable by a may transition executing an action satisfying ψ, ϕ holds

$[\psi]^{\Box}\,\phi$ in all next states reachable by a must transition executing an action satisfying ψ, ϕ holds

$\langle\psi\rangle\,\phi \equiv \neg[\psi]\,\neg\phi$ a next state exists, reachable by a may transition executing an action satisfying ψ, in which ϕ holds

$\langle\psi\rangle^{\Box}\phi \equiv \neg[\psi]^{\Box}\neg\phi$ a next state exists, reachable by a must transition executing an action satisfying ψ, in which ϕ holds

($\langle\psi\rangle^{\Box}$ and $[\psi]^{\Box}$ represent the classic deontic modalities O and P)

Informal semantics of v-ACTL (cont'd)

$E\,\pi$ there exists a full path on which π holds

$A\,\pi$ on all possible full paths, π holds

$F\,\phi$ there exists a future state in which ϕ holds

$F^{\square}\,\phi$ ... and all transitions until that state are must transitions

$F\,\{\psi\}\,\phi$ there exists a future state, reached by an action satisfying ψ, in which ϕ holds

$F^{\square}\,\{\psi\}\,\phi$ ... and all transitions until that state are must transitions

$G\,\phi \equiv \neg F\,\neg\phi$ the path is a full path on which ϕ holds in all states

$AG\,\phi \equiv \neg EF\,\neg\phi$ in all states on all paths, ϕ holds

A full path is a path that cannot be extended further ($q \cdots$ or $q \not\rightarrow$)

Informal semantics of v-ACTL (cont'd)

$E\,\pi$ there exists a full path on which π holds

$A\,\pi$ on all possible full paths, π holds

$F\,\phi$ there exists a future state in which ϕ holds

$F^{\Box}\,\phi$ . . . and all transitions until that state are must transitions

$F\,\{\psi\}\,\phi$ there exists a future state, reached by an action satisfying ψ, in which ϕ holds

$F^{\Box}\,\{\psi\}\,\phi$ . . . and all transitions until that state are must transitions

$G\,\phi \equiv \neg F\,\neg\phi$ the path is a full path on which ϕ holds in all states

$AG\,\phi \equiv \neg EF\,\neg\phi$ in all states on all paths, ϕ holds

A full path is a path that cannot be extended further ($q \cdots$ or $q \not\rightarrow$)

Informal semantics of v-ACTL (cont'd)

$E\,\pi$	there exists a full path on which π holds
$A\,\pi$	on all possible full paths, π holds
$F\,\phi$	there exists a future state in which ϕ holds
$F^{\square}\,\phi$	... and all transitions until that state are must transitions
$F\,\{\psi\}\,\phi$	there exists a future state, reached by an action satisfying ψ, in which ϕ holds
$F^{\square}\,\{\psi\}\,\phi$	... and all transitions until that state are must transitions

$G\,\phi \equiv \neg F\,\neg\phi$ the path is a full path on which ϕ holds in all states

$AG\,\phi \equiv \neg EF\,\neg\phi$ in all states on all paths, ϕ holds

A full path is a path that cannot be extended further ($q \cdots$ or $q \not\rightarrow$)

Informal semantics of v-ACTL (cont'd)

$E\,\pi$	there exists a full path on which π holds
$A\,\pi$	on all possible full paths, π holds
$F\,\phi$	there exists a future state in which ϕ holds
$F^{\Box}\,\phi$	... and all transitions until that state are must transitions
$F\,\{\psi\}\,\phi$	there exists a future state, reached by an action satisfying ψ, in which ϕ holds
$F^{\Box}\,\{\psi\}\,\phi$	... and all transitions until that state are must transitions

$G\,\phi \equiv \neg F\,\neg\phi$ the path is a full path on which ϕ holds in all states

$AG\,\phi \equiv \neg EF\,\neg\phi$ in all states on all paths, ϕ holds

A full path is a path that cannot be extended further ($q \cdots$ or $q \not\rightarrow$)

Bike-sharing systems (BSS)

In Quanticol we collaborate with *PisaMo* (responsible for the public BSS *CicloPi* in Pisa, currently only some 150 bikes and 15 stations)

We recall the small case study presented by Stefania in her keynote

Inspired by Fricker, Gast @ arXiv, September 2013

We model 2 user groups that can take a bike from docking station I, ride it for a while (not modeled), and deliver it to docking station J

Initially, docking station I has 1 bike, which it gives (when available) to a requesting user or accepts from a returning user

Redistribution is optional: If docking station I receives more than 1 bike, the exceeding bikes are distributed to docking station J

Bike-sharing systems (BSS)

In Quanticol we collaborate with *PisaMo* (responsible for the public BSS *CicloPi* in Pisa, currently only some 150 bikes and 15 stations)

We recall the small case study presented by Stefania in her keynote

Inspired by Fricker, Gast @ arXiv, September 2013

We model 2 user groups that can take a bike from docking station I, ride it for a while (not modeled), and deliver it to docking station J

Initially, docking station I has 1 bike, which it gives (when available) to a requesting user or accepts from a returning user

Redistribution is optional: If docking station I receives more than 1 bike, the exceeding bikes are distributed to docking station J

Bike-sharing systems (BSS)

In Quanticol we collaborate with *PisaMo* (responsible for the public BSS *CicloPi* in Pisa, currently only some 150 bikes and 15 stations)

We recall the small case study presented by Stefania in her keynote

Inspired by Fricker, Gast @ arXiv, September 2013

We model 2 user groups that can take a bike from docking station I, ride it for a while (not modeled), and deliver it to docking station J

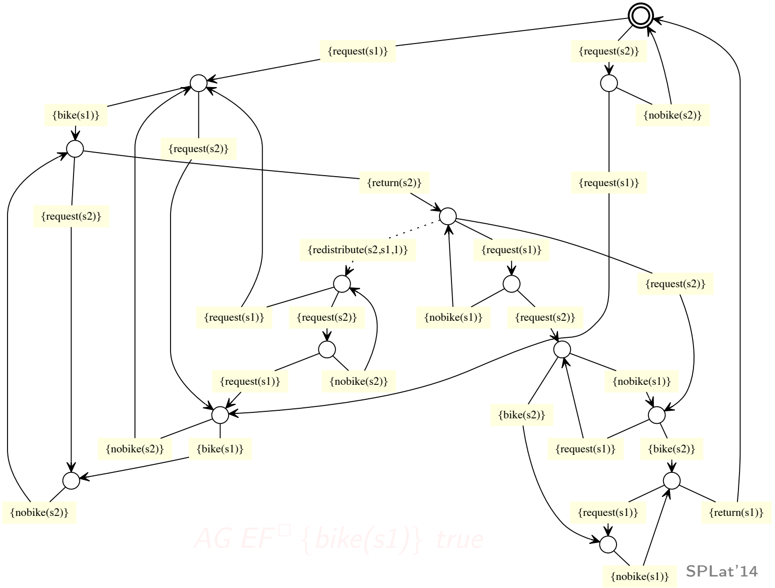
Initially, docking station I has 1 bike, which it gives (when available) to a requesting user or accepts from a returning user

Redistribution is optional: If docking station I receives more than 1 bike, the exceeding bikes are distributed to docking station J

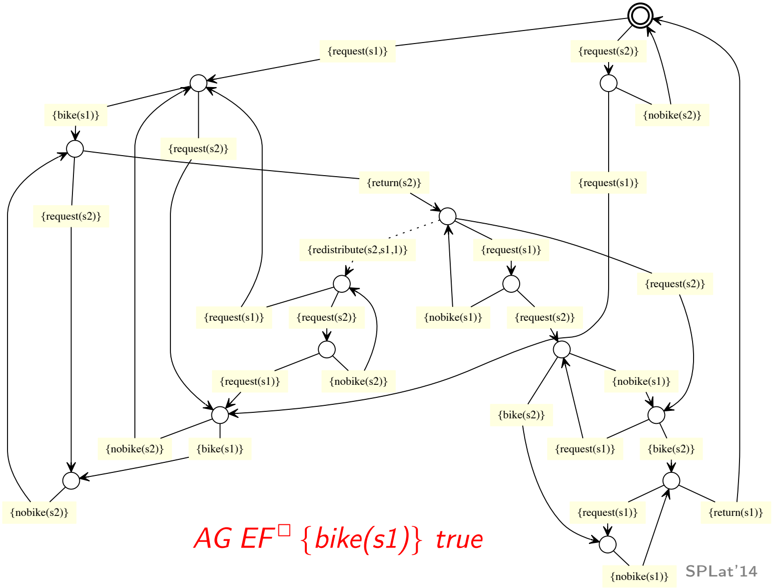
Value-passing BSS specification

```
    Station(I,N,J,M) = request(I).
        ( [N=0] nobike(I).Station(I,N,J,M) +
          [N>0] bike(I).Station(I,N-1,J,M) ) +
      return(I).Station(I,N+1,J,M) +
      redistribute(may,?FROM,?TO,?K).
        ( [TO = I] Station(I,N+K,J,M) +
          [TO /= I] Station(I,N,J,M) ) +
      [N > M] redistribute(may,I,J,N-M).Station(I,M,J,M)

net STATIONS = Station(s1,1,s2,1) /redistribute/ Station(s2,0,s1,0)

    Users(I,J) = request(I).
        ( bike(I).return(J).Users(I,J) +
          nobike(I).Users(I,J) )

net USERS = Users(s1,s2) // Users(s2,s1)

net BSS = STATIONS /request,bike,nobike,return/ USERS
```

MTS with parameters and values



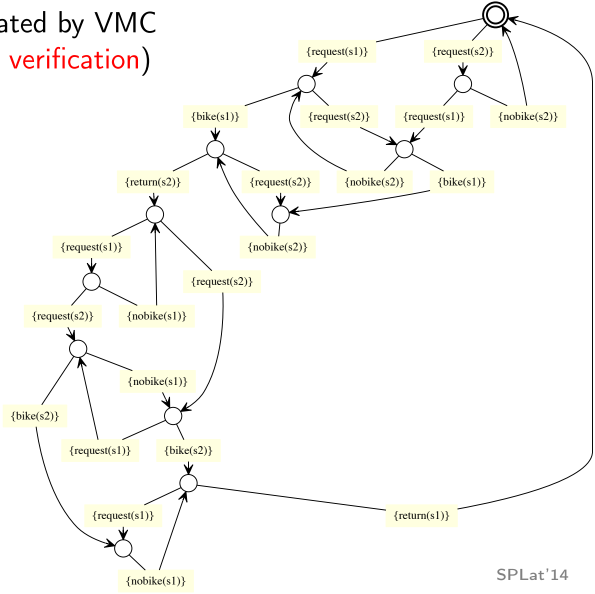
MTS with parameters and values



$AG\ EF^{\Box}\ \{bike(s1)\}\ true$

v-ACTL$^\Box$ formula: true ∀ products

Products generated by VMC
(not needed for verification)



SPL Analysis Strategies

Thüm, Apel, Kästner, Schaefer, Saake @ ACM Comput. Surv., 2014

Product-based analyses: verification on individually derived products, or at most a subset

Family-based analyses: verification on entire product family at once, using available variability knowledge about valid feature configurations to deduce the results for all its products

Possible trade-off (e.g. time vs. memory complexity)?

- Brute-force product-based analysis with model checkers highly optimized for single system engineering (e.g. SPIN, mCRL2)

  spinroot.com www.mcrl2.org (next talk!)

- Highly innovative family-based analysis with model checkers developed specifically for SPL (e.g. SNIP, NuSMV extension)

  Classen et al. @ STTT, 2012; Classen et al. @ IEEE TSE, 2013; Classen et al. @ Sci. Comput. Program., 2014

⇒ Implement some SPL-specific features in VMC?

SPL Analysis Strategies

Thüm, Apel, Kästner, Schaefer, Saake @ ACM Comput. Surv., 2014

Product-based analyses: verification on individually derived products, or at most a subset

Family-based analyses: verification on entire product family at once, using available variability knowledge about valid feature configurations to deduce the results for all its products

Possible trade-off (e.g. time vs. memory complexity) ?

- Brute-force product-based analysis with model checkers highly optimized for single system engineering (e.g. SPIN, mCRL2)

  `spinroot.com` `www.mcrl2.org` (next talk!)

- Highly innovative family-based analysis with model checkers developed specifically for SPL (e.g. SNIP, NuSMV extension)

  Classen et al. @ STTT, 2012; Classen et al. @ IEEE TSE, 2013; Classen et al. @ Sci. Comput. Program., 2014

⇒ Implement some SPL-specific features in VMC ?

SPL Analysis Strategies

Thüm, Apel, Kästner, Schaefer, Saake @ ACM Comput. Surv., 2014

Product-based analyses: verification on individually derived products, or at most a subset

Family-based analyses: verification on entire product family at once, using available variability knowledge about valid feature configurations to deduce the results for all its products

Possible trade-off (e.g. time vs. memory complexity) ?

- Brute-force product-based analysis with model checkers highly optimized for single system engineering (e.g. SPIN, mCRL2)

  `spinroot.com` `www.mcrl2.org` (next talk!)

- Highly innovative family-based analysis with model checkers developed specifically for SPL (e.g. SNIP, NuSMV extension)

  Classen et al. @ STTT, 2012; Classen et al. @ IEEE TSE, 2013; Classen et al. @ Sci. Comput. Program., 2014

⇒ Implement some SPL-specific features in VMC ?

Challenges ahead

Study and implement the derivation of products in the presence of both structural constraints (ALT, EXC, REQ from feature models) and quantitative constraints (so-called attributed feature models) ?

Weighted MTS ?

Parameter values in constraints?

Explicit behavioral variability constraints like $X\{a\}$ ALT $X\{b\}$? (consistency assumption!)

Study the inheritence of the result of verifying a v-ACTL formula over an MTS by its product LTS in the presence of such types of constraints

Scalability ?

Comparison with other tools (e.g. ProVeLines)

Publicity: start preparing for FMSPLE'15 in London, UK

Formal Methods and Analysis in Software Product Line Engineering (FMSPLE'15)

6th International Workshop

London, UK, April 11, 2015
(co-located with ETAPS'15)

⇒ `http://fmsple15.isti.cnr.it/` ⇐

Submission deadline: January 30, 2015

PC chairs:

- Joanne Atlee (University of Waterloo, Canada)
- Stefania Gnesi (ISTI–CNR, Pisa, Italy)