COOPERATION

quanticol
www.quanticol.eu

Model Checking Value-Passing Modal Specifications

Maurice H. ter Beek, Stefania Gnesi, and Franco Mazzanti

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

PSI 2014
St. Petersburg
25-6-2014

Disclaimer and acknowledgement

- My talk is about (an application of) model checking, hence...
- normally I would briefly explain what model checking is about, but I assume that you stayed awake during the keynote ☺

- Quite an honor to talk right after the 2007 Turing award winner!
- A lot of what I know about model checking comes from his 1999 book "Model Checking" and from his course at the 2005 Lipari summer school on "Formal Methods: Theory and Practice"

- The rest comes from my lab's nearly 20 years experience with a family of model checkers developed during a series of EU projects

Outline

1. QUANTICOL: A brief introduction
2. Software Product Line Engineering: Behavioral variability analysis
3. From single product analysis to product family analysis
4. Variability Model Checker (VMC)
   - Modal Transition Systems with variability constraints
   - A value-passing modal process algebra
   - v-ACTL: A logic to express variability
5. An example family of Bike-Sharing Systems (BSS)
6. Conclusions and future work

QUANTICOL: A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

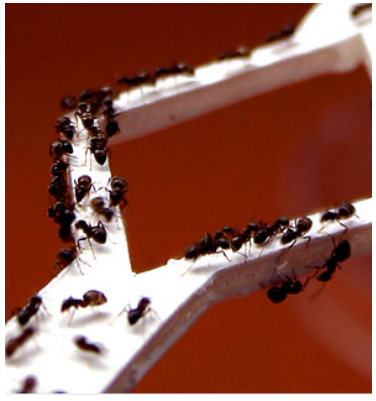
EU FP7-ICT FET-Proactive STREP: 1 April 2013 – 31 March 2017

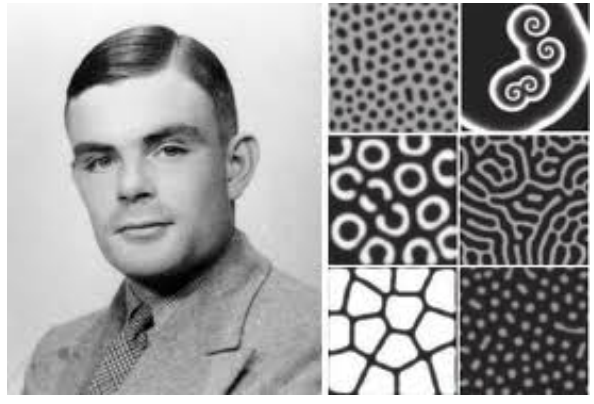
- University of Edinburgh, Scotland, Jane Hillston (Coordinator)
- CNR-ISTI, Pisa, Italy, Mieke Massink
- University of Southampton, England, Mirco Tribastone
- EPFL, Lausanne, Switzerland, Jean-Yves Le Boudec
- IMT Lucca, Italy, Rocco De Nicola


Collective Adaptive Systems (CAS)

quanticol
www.quanticol.eu

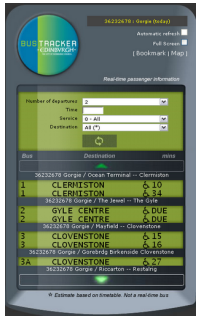
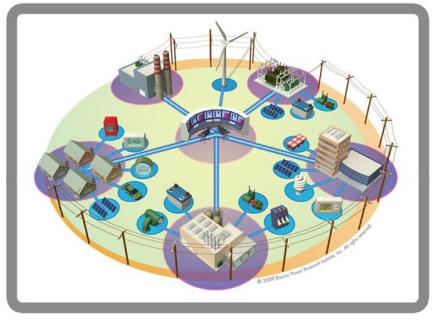
Many examples of decentralized collective adaptive behavior in nature





QUANTICOL: focus on applications arising in context of smart cities



Highly distributed systems with adaptive behavior relying on continuous feedback of vast numbers of consumers and producers

- Scalable verification approaches (model checkers)
- Quantitative variability analysis and product families

Concrete case study on bike-sharing systems (BSS)

- Popular sustainable means of transportation in urban environment
- Challenging case study offering interesting runtime optimization problems and exhibiting variability in the kind of features and in their quantitative characteristics

T3.3 Relating local and global system views with variability analysis

- Study relations between (representations of) small populations and compact (family) representation of large population 'built' from them by indicating the commonalities and variabilities of single entities in their overall environment

(Software) Product Line Engineering

Develop a family of products (product line) using a shared platform or architecture (commonalities) and mass customization (variabilities)

Aim: maximize commonalities whilst minimizing cost of variations (i.e., of individual products), thus specifically facilitating (software) reuse in a predictive manner

Variability in terms of features:

- End-user visible pieces of functionality that represent both commonalities (e.g., mandatory, required) and variabilities (e.g., optional, alternative)
- Only specific combinations of features concern valid products

Complex: *"We always have 126,000,000 different bicycles in store! But only the parts for 1,000..."*

Formal methods in SPLE

quantiCol
www.quanticol.eu

Computer-aided analysis of variability models

- Traditionally: focus on modeling/analyzing structural constraints
- But: software systems often embedded/distributed/safety-critical
- Important: model/analyze also behavior (e.g., quality assurance)

Goal: rigorously establish critical requirements of (software) systems
$\Rightarrow$ lift success stories from single product/system engineering to SPLE

Recent approaches to formally model behavioral variability:

- Variants of UML diagrams (Haugen et al., Jézéquel et al.)
- Extensions of Petri nets (Clarke et al.)
- Numerous models with LTS-like semantics: MTS (Fischbein et al., Fantechi et al.), I/O automata (Larsen et al., Lauenroth et al.), CCS/CSP (Gruler et al., Gnesi et al., ter Beek et al., Tribastone), FTS (Classen et al.), FSM (Millo et al.)

Scalability is a major issue!

(slide by C. Kästner, CMU)

with 33 optional, independent features

a unique product for every

person on this planet

The framework so far

Aim: develop a framework able to handle behavioral variability and provide tools to support it with formal verification (model checking)

Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible may and necessary must transitions

  Larsen, Thomsen @ LICS'88

- 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

- 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

Modal Transition Systems (MTS)

Larsen, Thomsen @ LICS'88

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

1. shared behavior: common among all variants
2. variation points: differentiate between variants

A Labelled Transition System (LTS) is a quadruple $(Q, A, \overline{q}, \rightarrow)$ where Q is a set of states, A is a set of actions, $\overline{q} \in Q$ is the initial state and $\rightarrow \subseteq Q \times A \times Q$ is the transition relation

A Modal Transition System (MTS) is a quintuple $(Q, A, \overline{q}, \rightarrow_{\Box}, \rightarrow_{\Diamond})$ such that $(Q, A, \overline{q}, \rightarrow_{\Box} \cup \rightarrow_{\Diamond})$ is an LTS, called its underlying LTS

An MTS has two distinct transition relations

1. may transition relation $\rightarrow_{\Diamond} \subseteq Q \times A \times Q$: possible transitions
2. must transition relation $\rightarrow_{\Box} \subseteq Q \times A \times Q$: required transitions

By definition, any required transition is also possible: $\rightarrow_{\Box} \subseteq \rightarrow_{\Diamond}$
$(\dashrightarrow \equiv \rightarrow_{\Diamond} \setminus \rightarrow_{\Box})$

Derive products (implemented in VMC)

quanticol
www.quanticol.eu

A product LTS is obtained from a family MTS in the following way

1. include all (reachable) must transitions and
2. include subset of the (reachable) may transitions, remove others
3. satisfy assumptions of coherence and consistency
4. satisfy variability constraints

Each selection gives rise to a different variant

Let $(Q, A, \overline{q}, \delta^{\Diamond}, \delta^{\Box})$ be a coherent MTS, i.e. $\exists \xrightarrow{a}_{\Diamond} \implies \nexists \xrightarrow{a}_{\Box}$
The set $\{\mathcal{P}_i = (Q_i, A, \overline{q}, \delta_i) \mid i > 0\}$ of product LTSs is obtained by considering each pair of $Q_i \subseteq Q$ and $\delta_i \subseteq \delta^{\Diamond} \cup \delta^{\Box}$ to be defined s.t.

1. every $q \in Q_i$ is reachable in $\mathcal{P}_i$ from $\overline{q}$ via transitions from δ_i
2. there exists no $(q, a, q') \in \delta^{\Box} \setminus \delta_i$ such that $q \in Q_i$
3. LTS is consistent: both $\xrightarrow{a}_{\Diamond} \rightsquigarrow \xrightarrow{a}_{\Box}$ and $\not\xrightarrow{a}_{\Box}$ not allowed

Variability constraints (implemented in VMC)

Variability constraints of the form ALTernative, EXCludes, REQuires

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

But also compositions like a REQ $(b$ ALT $c)$

VMC: Variability Model Checker

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

quanticol
www.quanticol.eu

VMC builds on optimization of UMC (input: UML state machines)

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

VMC: bounded, on-the-fly model checking, providing explanations

↘ Biere, Cimatti, Clarke, Zhu @ TACAS'99

VMC accepts as input a specification in (value-passing) modal process algebra, possibly with additional 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/

A value-passing modal process algebra

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 a process identifier from set of process definitions of form $K(v) \stackrel{\text{def}}{=} T$

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

$\bowtie \in \{<, \leq, =, \neq, \geq, >\}$, 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 (it can synchronize on actions in L and interleave others)

We distinguish must actions $a \in \delta^{\Box}$ and may but not must actions $a(may) \in \delta^{\Diamond} \setminus \delta^{\Box}$ (each action type is treated differently in the SOS semantics)

Semantics in SOS style over MTS

$$(\mathrm{SYS})\ \frac{P \xrightarrow{a(e)} P'}{[P] \xrightarrow{a(e)} [P']} \qquad (\mathrm{ACT}_\Box)\ \frac{}{\alpha.P \xrightarrow{\alpha} P}\ \alpha \in \{a(e), a(?v)\}$$

$$(\mathrm{OR}_\Box)\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}\ \alpha \in \{a(e), a(?v)\} \qquad (\mathrm{INT}_\Box)\ \frac{P \xrightarrow{\ell} P'}{P\ /L/\ Q \xrightarrow{\ell} P'\ /L/\ Q}\ \ell \notin L$$

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

$$(\mathrm{GUARD})\ \frac{}{[e_1 \bowtie e_2]\, P(e_3) \to P(e_3)}\ e_1 \bowtie e_2$$

(similarly in case of may actions and for the remaining operators)

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")

Variability-aware ACTL

v-ACTL$^{\Box}$:

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

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

v-ACTL^{-}:

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

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

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

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

$E\,\pi$ 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

$F^{\Box}\,\phi$ there exists a future state in which ϕ holds 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

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

Semantics of v-ACTL over MTS $(Q, A, \overline{q}, \delta^{\Diamond}, \delta^{\Box})$

Let $q \in Q$ and σ a full path (from q) with ith state $\sigma(i)$ and ith action $\sigma\{i\}$

$q \models \neg\phi$ iff $q \not\models \phi$

$q \models \phi \wedge \phi'$ iff $q \models \phi$ and $q \models \phi'$

$q \models [\psi]\,\phi$ iff $\forall\, q' \in Q$ s.t. $q \xrightarrow{a(e)}_{\Diamond} q'$ and $a(e) \models \psi$, we have $q' \models \phi$

$q \models [\psi]^{\Box}\,\phi$ iff $\forall\, q' \in Q$ s.t. $q \xrightarrow{a(e)}_{\Box} q'$ and $a(e) \models \psi$, we have $q' \models \phi$

$q \models E\,\pi$ iff $\exists$ full path σ' from q: $\sigma' \models \pi$

$q \models A\,\pi$ iff $\forall$ full path σ' from q: $\sigma' \models \pi$

$q \models \mu\, Y.\phi(Y)$ iff $\bigvee_{i\geq 0} \phi^i(\mathit{false})$

$q \models \nu\, Y.\phi(Y)$ iff $\bigwedge_{i\geq 0} \phi^i(\mathit{true})$

$q \models F\,\phi$ iff $\exists j \geq 1 : \sigma(j) \models \phi$

$q \models F^{\Box}\,\phi$ iff $\exists j \geq 1 : \sigma(j) \models \phi$ and $\forall\, 1 \leq i < j : (\sigma(i), \sigma\{i\}, \sigma(i+1)) \in \delta^{\Box}$

$q \models F\,\{\psi\}\,\phi$ iff $\exists j \geq 1 : \sigma\{j\} \models \psi$ and $\sigma(j+1) \models \phi$

$q \models F^{\Box}\,\{\psi\}\,\phi$ iff $\exists j \geq 1 : \sigma\{j\} \models \psi$ and $\sigma(j+1) \models \phi$, and $\forall\, 1 \leq i \leq j$: $(\sigma(i), \sigma\{i\}, \sigma(i+1)) \in \delta^{\Box}$

Bike-sharing systems (BSS)

- Simple concept: a user arrives at a docking station, pays for a bike, uses it for a while and returns it to a station
- Multiple benefits: reduction of vehicular traffic (congestion), pollution, energy consumption, etc.

- Docking stations distributed over a city, typically close to other public transportation hubs (e.g. subway and tram stations)
- (Subscribed) users may rent an available bike and drop it off at any station in the city
- To improve the efficiency and the user satisfaction of BSS, the load between the different stations may be balanced
  - incentive schemes (rewards) to change the behavior of users
  - efficient (dynamic) redistribution of bikes between stations

History of BSS

DeMaio @ *Journal of Public Transportation*, 2009

- 1st generation free BSS introduced in Amsterdam (*witte fietsen*)
- 2nd generation born in Denmark, first large-scale BSS launched in Copenhagen (*Bycyklen*)
- 3rd generation technology-based BSS in $>$ 500 cities worldwide (*Vélib'* in Paris: over $20,000$ bikes and $1,800$ stations; largest in Hangzhou: $\pm 50,000$ bikes and $2,000$ stations, one every 100m)
- 4th generation BSS are already being developed, incl. movable and solar-powered stations, electric bikes and mobile (i)phone real-time availability applications

We started to collaborate with *PisaMo*, an in-house public mobility company of Pisa's administration that recently introduced the public BSS *CicloPi* in Pisa (currently only some 150 bikes and 15 stations)

Behavioral requirements of a BSS

We consider a BSS with N stations and a fleet of M bikes;
Each station i has a capacity K_i; Redistribution is optional

1. Users arrive at station i
2. If a user arrives at a station with no available bike, (s)he leaves
3. Otherwise, (s)he takes a bike and chooses station j to return it
4. If there are less than K_j bikes at station j when (s)he arrives, (s)he returns the bike and leaves
5. If the station is full she chooses another station k and goes there
6. Redistribution of bikes may be asked for and may possibly occur
7. The user rides like this again until (s)he can return the bike

Inspired by Fricker, Gast @ arXiv, September 2013

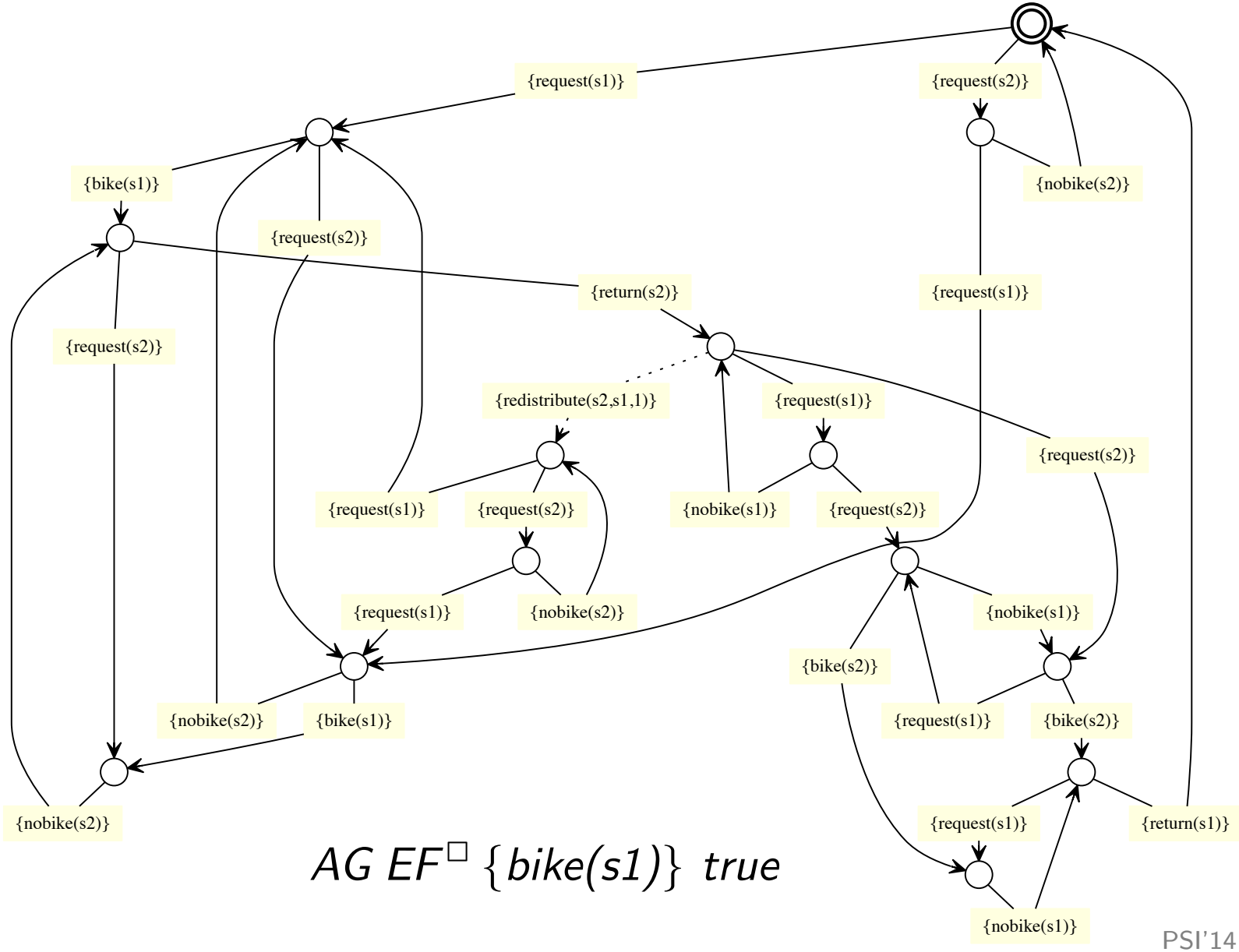
Value-passing BSS specification

quantiCol
www.quanticol.eu

```
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

quanticol
www.quanticol.eu

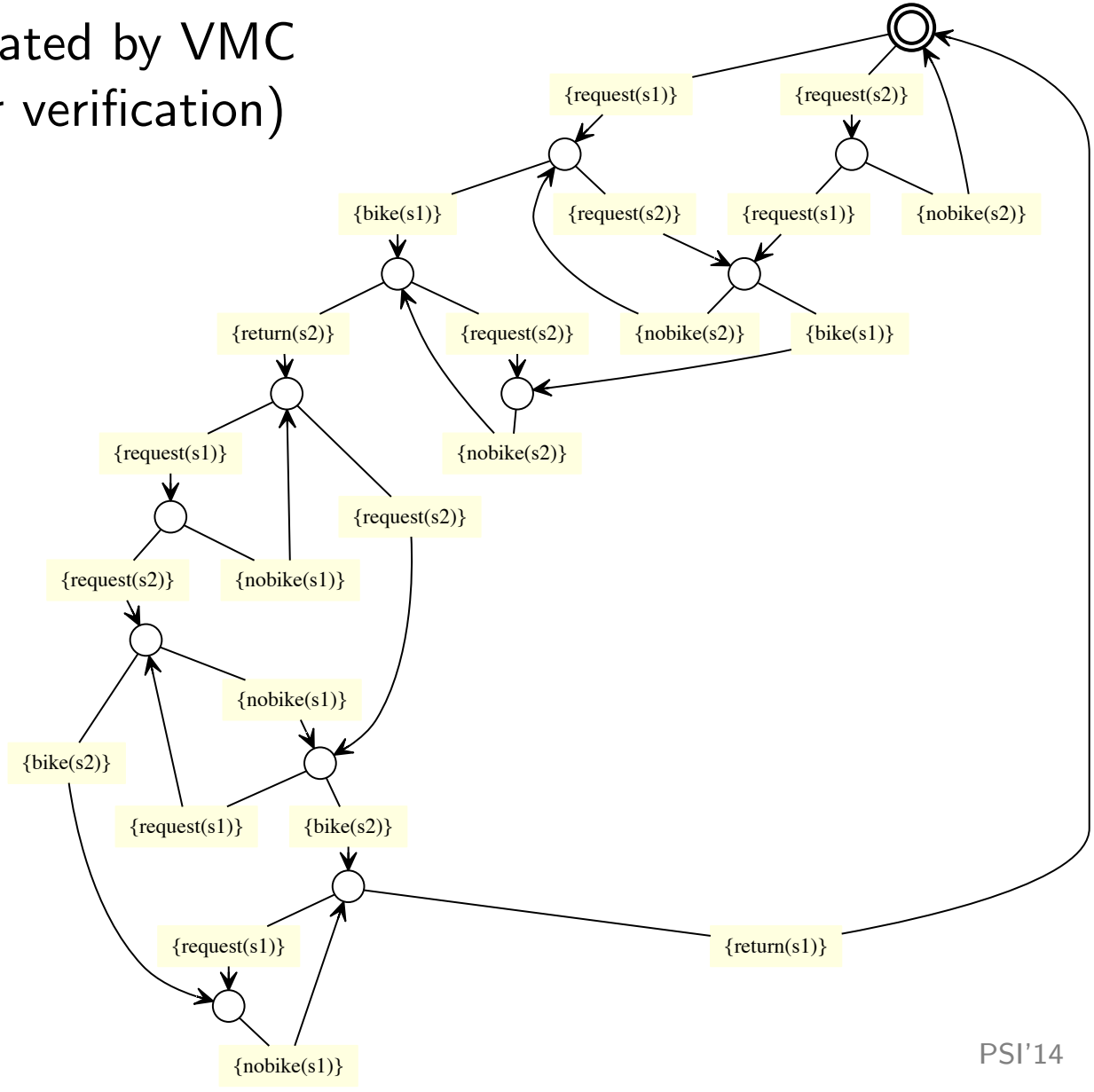


$AG\ EF^{\square}\ \{bike(s1)\}\ true$

v-ACTL$^{\square}$ formula: true $\forall$ products

quanticol
www.quanticol.eu

Products generated by VMC
(not needed for verification)



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?

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

Future work

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 ?

Publicity: SPLC'14 in Florence, Italy

http://www.splc2014.net/

- Research track
- Industry track
- Demo/tool track
- 8 Tutorials
- 7 Workshops
- Doctoral symposium
- Hall of fame
- Panels
- ...

Organized by our Formal Methods and Tools lab of ISTI-CNR, Pisa