Challenges in modelling and analyzing quantitative aspects of a bike-sharing product line

Stefania Gnesi

(joint work with Maurice H. ter Beek and Alessandro Fantechi and Franco Mazzanti)

ISTI–CNR, Pisa, Italy

Universita' di Genova, 23-07-2014

Outline

Aim of our research activity in

- Develop a framework able to deal with 'quantitative variability'
- Provide tools to support this framework with formal verification

Preliminary results

- We established a tool chain of (academic) tools with different functionalities regarding the analysis of SPL, from feature modelling to product derivation, to the quantitative evaluation of the attributes of products (structural variability)
- We defined a value-passing modal process algebra interpreted over Modal Transition Systems (MTS) (behavioural variability)
- We defined a deontic (variability-aware) extension of an action-based branching-time modal temporal logic able to handle data
- We extended our Variability Model Checker (VMC) to support this setup

(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

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

Bike-sharing systems (BSS)

$\Rightarrow$ 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

- 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 behaviour of users
  - efficient (dynamic) redistribution of bikes between stations

Quantitative Business models and Product Line Families

Finding the right BSS for a particular city poses many questions

- How many and what kind of bikes?
- How many and what kind of stations and where to place them?
- Which features (antitheft, maintenance, smart services, etc.)?
- With or without (dynamic) redistribution?
- Incentives for users to return bikes to less popular stations?
- Costs and charging policy (credit card, keycard, etc.)?

How to evaluate the various options, costs/benefits, improvements and changes in a systematic way?

A BSS Product Line

In the requirements engineering phase we performed text mining on a set of documents describing current BSS to extract a set of features: NLP approach based on so-called contrastive analysis to identify commonalities and variabilities from natural language documents

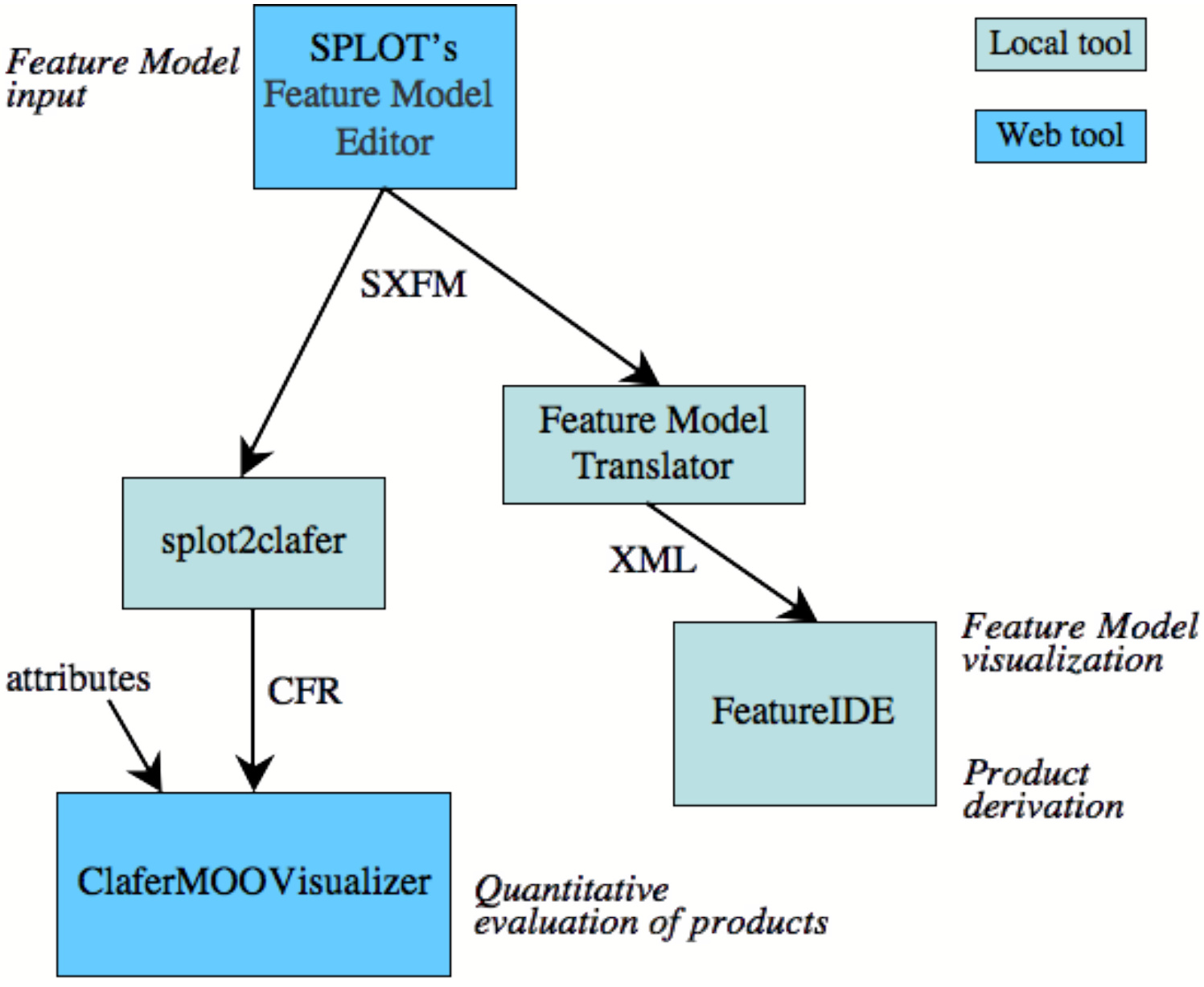
Ferrari, Spagnolo, dell'Orletta @ SPLC'13

Subsequently we derived an initial feature model from the features that we considered most interesting for developing 'good quality' BSS

We believe they constitute a sufficient starting point for our study

Finally, we sought support in available tools for the possibility of adding attributes and quantitative constraints to our BSS model

Tool chain


Feature Model input
SPLOT's Feature Model Editor
Local tool
Web tool
SXFM
Feature Model Translator
splot2clafer
XML
attributes
CFR
FeatureIDE
Feature Model visualization
Product derivation
ClaferMOOVisualizer
Quantitative evaluation of products

Feature model (with S.P.L.O.T.)

Mendonça, Branco, Cowan @ OOPSLA'09

Feature Diagram

- Bikesharing
  - Status
    - [1..*]
      - RTInfoWeb
      - AllBikesNow
  - Bike
    - Localization
      - [1..*]
        - GPS
        - RFID
    - Antithieves
  - DockingStation
    - [1..1]
      - Fixed
      - FixedPortable
      - Flexible
    - Maintenance
  - Redistribution
    - Reward
  - Users

Cross-Tree Constraints

- (¬AllBikesNow ∨ GPS)
- (GPS ∨ ¬Antithieves)
- (Keycard ∨ ¬KeycardDispenser)
- (¬Keycard ∨ KeycardReader)
- (¬Keycard ∨ KeycardDispenser)

Click to create a constraint

Feature Information Table

Id:	
Name:	
Description:	
Type:	
#Children:	
Tree level:	

Update Feature Model

Feature Model Statistics

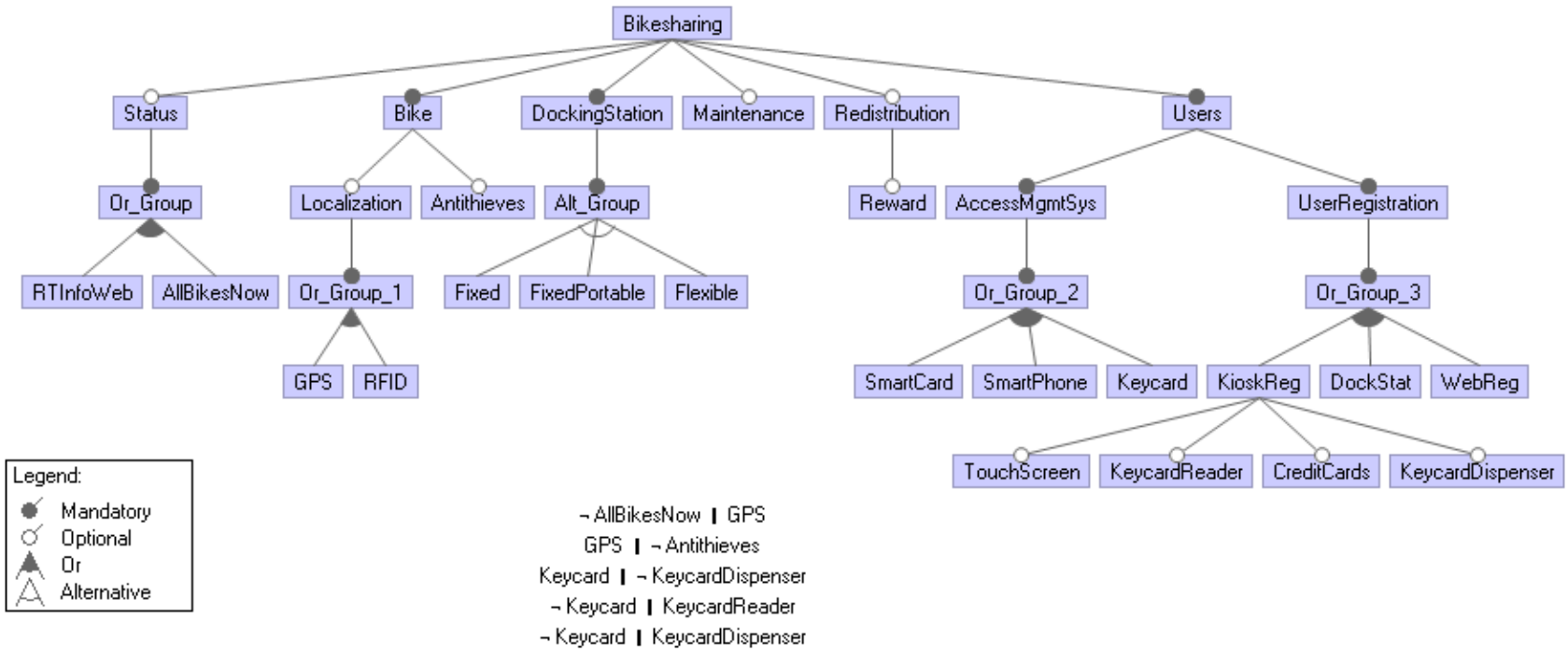
#Features	29
#Mandatory	5
#Optional	10
#XOR groups	1
#OR groups	4
#Grouped	13
#Cross-Tree Constraints (CTC)	5
CTCR (%)	0.21
#CTC distinct vars	6
CTC clause density	0.83

Feature Model Analysis

✔ Consistency	Consistent
✔ Dead Features	None
✔ Core Features	6 feature(s)
✔ Valid Configurations	60,840

Feature model (with FeatureIDE)

Thüm, Kästner, Benduhn, Meinicke, Saake, Leich



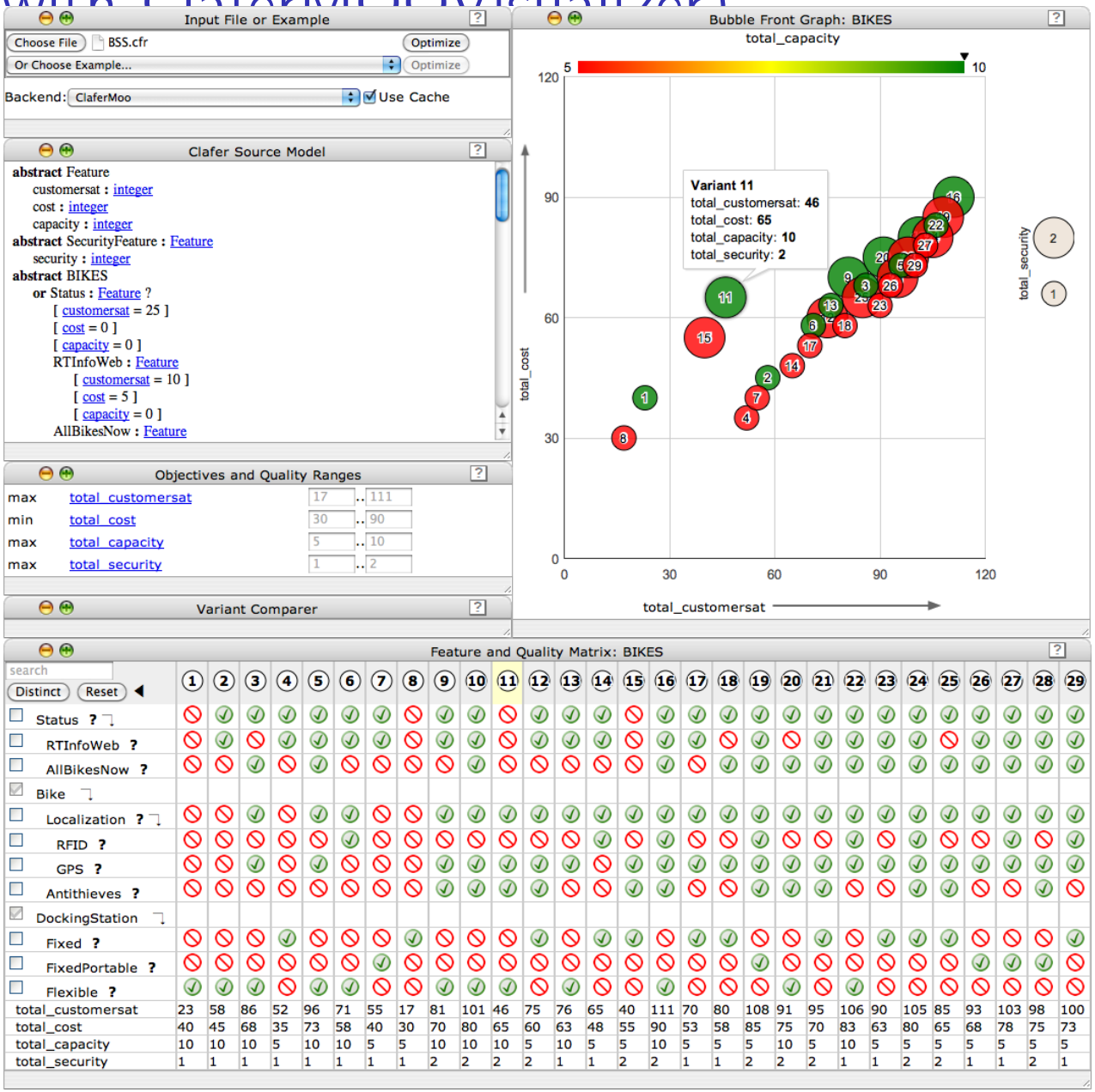
We may turn this into an attributed feature model, e.g. to measure the "quality" of different products

Attributed feature model in Clafer

Bąk, Czarnecki, Wąsowski @ SLE'10, Murashkin, Antkiewicz, Rayside, Czarnecki @ SPLC'13

```
abstract Feature
  customersat : integer
  cost : integer
  capacity : integer

abstract SecFeature : Feature
  security : integer

abstract BIKES
  or Status : Feature ?
    [ customersat = 25 ]
    [ cost = 0 ]
    [ capacity = 0 ]
    RTInfoWeb : Feature
      [ customersat = 10 ]
      [ cost = 5 ]
      [ capacity = 0 ]
    AllBikesNow : Feature
      [ customersat = 20 ]
      [ cost = 10 ]
      [ capacity = 0 ]
  Bike : Feature
      [ customersat = 0 ]
      [ cost = 0 ]
      [ capacity = 0 ]
  or Localization : Feature ?
     [ customersat = 3 ]
     [ cost = 3 ]
     [ capacity = 0 ]
     RFID : Feature
       [ customersat = 10 ]
       [ cost = 10 ]
       [ capacity = 0 ]
     GPS : Feature
       [ customersat = 15 ]
       [ cost = 15 ]
       [ capacity = 0 ]
   Antithieves : SecFeature ?
     [ customersat = 5 ]
     [ cost = 7 ]
     [ capacity = 0 ]
     [ security = 1 ]
xor DockingStation : SecFeature
  [ customersat = 0 ]
  [ cost = 0 ]
  [ capacity = 0 ]
  [ security = 1 ]
  Fixed : Feature
    [ customersat = 17 ]
    [ cost = 30 ]
    [ capacity = 5 ]
    FixedPortable: Feature
      [ customersat = 20 ]
      [ cost = 35 ]
      [ capacity = 5 ]
    Flexible: Feature
      [ customersat = 23 ]
      [ cost = 40 ]
      [ capacity = 10 ]
  [ Antithieves => GPS ]
  [ AllBikesNow => GPS ]

  total_customersat : integer =
    sum Feature.customersat
  total_cost : integer =
    sum Feature.cost
  total_capacity : integer =
    sum Feature.capacity
  total_security : integer =
    sum SecFeature.security

Mybike : BIKES
<< max Mybike.total_customersat >>
<< min Mybike.total_cost >>
<< max Mybike.total_capacity >>
<< max Mybike.total_security >>
```

Multi-objective optimization (with ClaferMOOVisualizer)



Behavioural requirements of a PL

Structural requirements identify the features of the different products

Behavioural requirements define the admitted sequences of operations

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. A 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

The framework so far

Aim: develop a framework able to handle behavioural 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 behaviour 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 behavioural model, amenable to model checking, able to formalize

1. shared behaviour: 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)

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

1. include all (reachable) must transitions and
2. include a subset of the (reachable) may transitions

Each selection gives rise to a different variant

Let $(Q, A, \overline{q}, \delta^{\Diamond}, \delta^{\Box})$ be an MTS
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 and
2. there exists no $(q, a, q') \in \delta^{\Box} \setminus \delta_i$ such that $q \in Q_i$

v-ACTL is used to complement a behavioural description by an MTS by expressing those constraints that MTS cannot model

v-ACTL: A logic to express variability

v-ACTL defines action formulas (boolean compositions of actions, denoted by ψ), state formulas (ϕ) and path formulas (π)

Let $a, b \in \mathcal{A}$. Action formulas are built over a set $\mathcal{A}$ of actions

$$\psi \quad ::= \quad \mathit{true} \mid a \mid \neg\psi \mid \psi \wedge \psi$$

($\mathit{false} \equiv \neg\mathit{true}$, $\psi \vee \psi' \equiv \neg(\neg\psi \wedge \neg\psi')$ and $\psi \implies \psi' \equiv \neg\psi \vee \psi'$)

The satisfaction relation $a \models \psi$ of a formula ψ by a is

$a \models \mathit{true}$ always holds

$a \models b$ iff $a = b$

$a \models \neg\psi$ iff $a \not\models \psi$

$a \models \psi \wedge \psi'$ iff $a \models \psi$ and $a \models \psi'$

Syntax of v-ACTL

$$\begin{aligned}\phi \quad &::= \quad \mathit{true} \mid \neg\,\phi \mid \phi \wedge \phi \mid [\psi]\,\phi \mid [\psi]^{\Box}\,\phi \mid \\ &\qquad\quad E\,\pi \mid A\,\pi \mid \mu\,Y.\phi(Y) \mid \nu\,Y.\phi(Y) \\ \pi \quad &::= \quad F\,\phi \mid F^{\Box}\,\phi \mid F\,\{\psi\}\,\phi \mid F^{\Box}\,\{\psi\}\,\phi\end{aligned}$$

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

($\mathit{false} \equiv \neg\mathit{true}$, $\psi \vee \psi' \equiv \neg(\neg\psi \wedge \neg\psi')$ and $\psi \Longrightarrow \psi' \equiv \neg\psi \vee \psi'$)

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

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

Model checking: $T \models \psi$?

Verify a property expressed as a logical formula ψ over a model T

- 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 model checking

- Only a fragment of the overall state space might need to be generated and analyzed to be able to produce the correct result
- Improves performance and allows to handle infinite-state systems

Bounded model checking:

- Start evaluation by assuming a certain value as maximum depth
  - If the evaluation reaches a result within the requested depth, then the result holds for the whole system
  - Otherwise the maximum depth is increased and the evaluation is retried (preserving all useful partial results)
- Setting a small initial maximum depth and a small automatic increment of this bound at each (re-)evaluation failure leads to a reasonable (almost minimal) explanation

VMC: Variability Model Checker

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

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

ter Beek, Fantechi, Gnesi & Mazzanti @ *Science of Computer Programming*, 2011

VMC accepts as input a model specified in the modal process algebra (+ variability constraints of form ALTernative, EXCludes, REQuires?)

- 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

VMS is freely usable online: `http://fmtlab.isti.cnr.it/vmc/`

Model checking of v-ACTL formulae over MTS can be achieved in a complexity that is linear w.r.t. the state space size

Extending the framework with data

Critical point in the formalization by MTS: lack of a possibility to model an adequate representation of the data that may need to be described when considering real systems (even for family of BSS)

Other possible approaches:

- Parametric MTS (Kretinsky, Larsen et al)
- Parametric modelling with the formal process-algebraic specification language mCRL2 and its industry-strength toolset (research in progress by ter Beek & de Vink)

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 $K(v) \stackrel{\text{def}}{=} T$

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

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

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

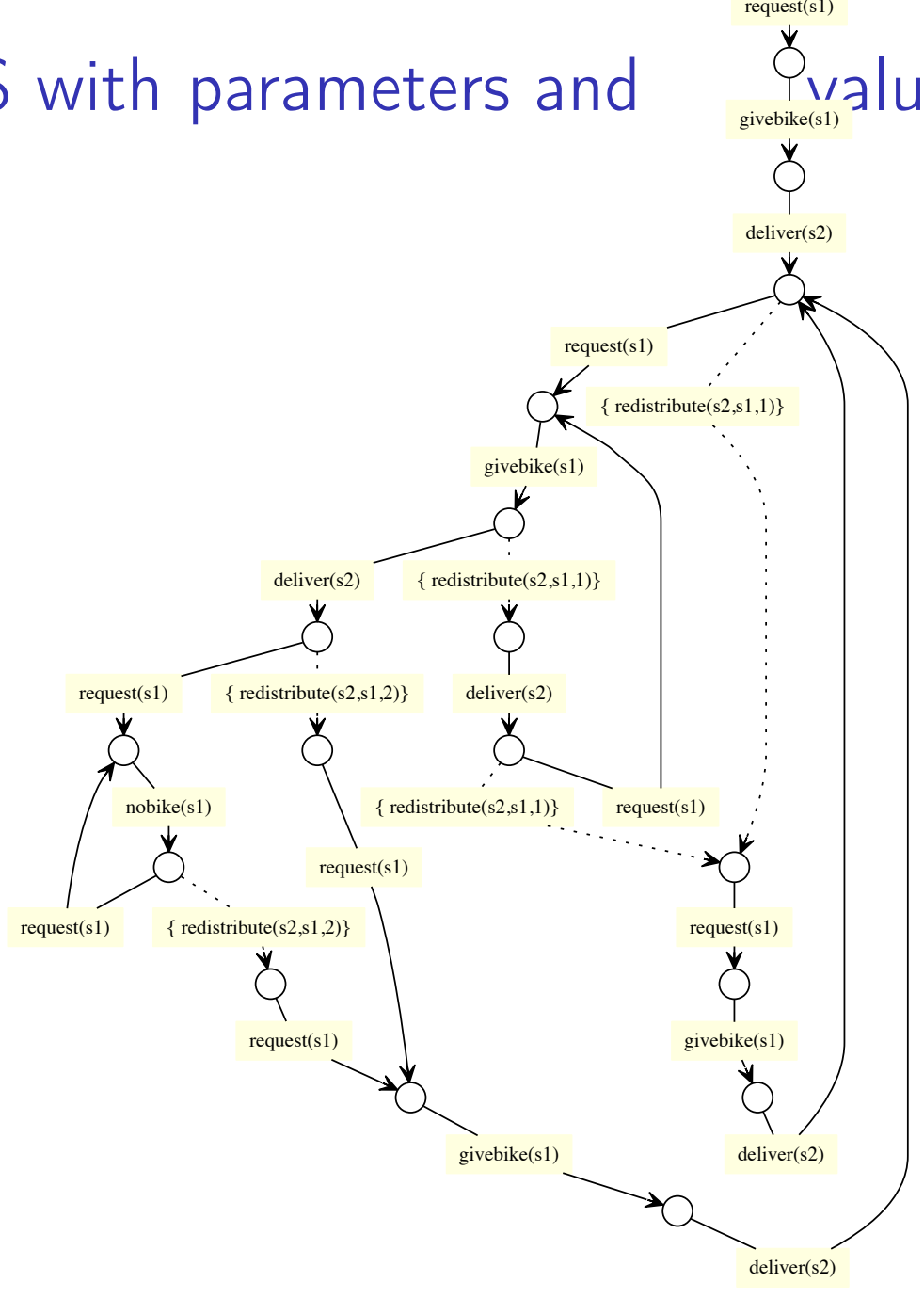
nil	terminated process that has finished execution
K	process identifier that is used for modelling recursive sequential processes
$A.P$	process that can execute action A and then behave as P
$P + Q$	process that can non-deterministically choose to behave as P or as Q
$P \,/L/\, Q$	process formed by the 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)

Value-passing BSS specification

```
   Station(I,N,J,M) = request(I).
       ( [N=0] nobike(I).Station(I,N,J,M) +
         [N>0] givebike(I).Station(I,N-1,J,M) ) +
     deliver(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(s1,2,s2,2) /redistribute/ Station(s2,2,s1,2)

   Users(I,J) = request(I).
       ( givebike(I).deliver(J).Users(I,J) +
         nobike(I).Users(I,J) )

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

net BSS = STATIONS /request,givebike,nobike,deliver/ USERS
```

MTS with parameters and values



Extending v-ACTL with action values

We only need to extend the definition of action formulas and their satisfaction relation

Let $a, b \in \mathcal{A}$. Action formulas are built over a set $\mathcal{A}$ of *actions*

$$\psi \quad ::= \quad \mathit{true} \mid a \mid a(e) \mid \neg\psi \mid \psi \wedge \psi$$

Now add to the satisfaction relation $a \models \psi$ of a formula ψ by a

$a(e) \models \mathit{true}$ always holds

$a(e) \models b$ iff $a = b$

$a(e) \models b(*)$ iff $a = b$

$a(e) \models b(e')$ iff $a = b$ and $e = e'$

$a(e) \models \neg\psi$ iff $a(e) \not\models \psi$

$a(e) \models \psi \wedge \psi'$ iff $a(e) \models \psi$ and $a(e) \models \psi'$

Extending VMC

The extended modelling and verification environment described so far has been implemented in VMC v6.0 (November 2013)

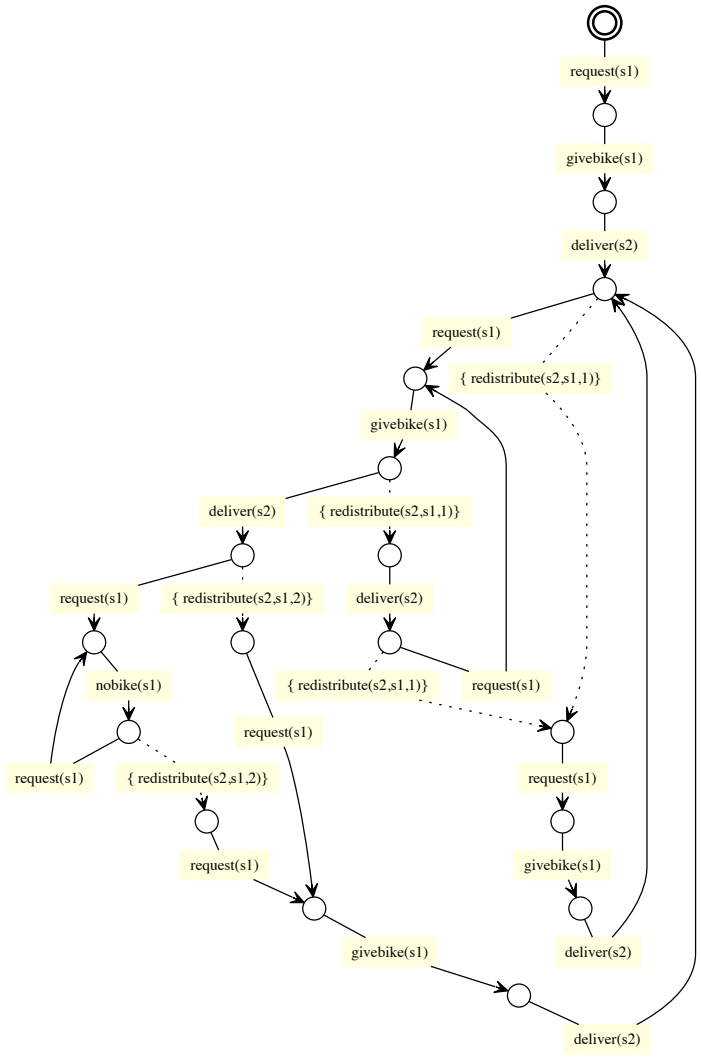
`http://fmt.isti.cnr.it/vmc/v6.0`

Accepts models specified in the value-passing modal process algebra

Allows model checking properties expressed in value-passing v-ACTL

Model checking modal specifications

- Eventually it must occur that station 1 has no bikes: $EF^{\Box}\ \{nobike(s1)\}\ true$
- Eventually it may occur that station 2 has no more bikes: $EF\ \{nobike(s2)\}\ true$
- It is always the case that eventually station 1 must give a bike, possibly after it has first received bikes after redistribution:
  $AG(\ (EF^{\Box}\ \{givebike(s1)\}\ true) \vee$
  $(EF^{\Box}\ [redistribute(*,s1,*)]\ EF^{\Box}\ \{givebike(s1)\}\ true)\ )$

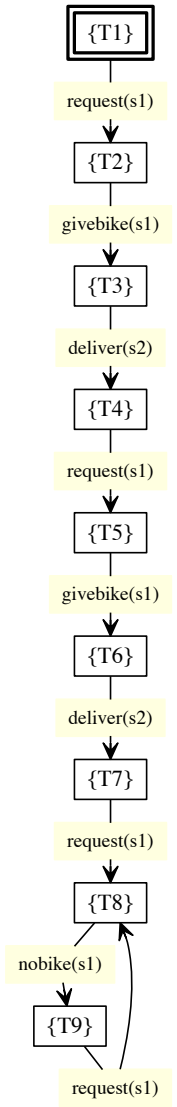


From modal specifications of families to products

Products may be derived according to an extension of the algorithms already present in VMC, for example:

Verification results of v-ACTL properties over an MTS are inherited by the entire family of derived product according to the following rules:

- Formulas without negation and only composed from *false*, *true* and the operators $\wedge$, $\vee$, $\langle\rangle^{\square}$, $[]$, μ, ν, $EF^{\square}$, $EF^{\square}\{\}$, $AF^{\square}$, $AF^{\square}\{\}$ and AG that are valid for a family MTS are also valid for all its product LTSs
- Formulas without negation and only composed from *false*, *true* and the operators $\wedge$, $\vee$, $\langle\rangle$, μ, ν, EF and $EF\{\}$ that are false for a family MTS are false for all its product LTSs



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 (attributed feature models)

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

Scalability ?