Quantitative Analysis of Probabilistic Models of SPL with Statistical Model Checking

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

joint work with

Axel Legay
Inria Rennes, France

Alberto Lluch Lafuente
DTU, Lyngby, Denmark

Andrea Vandin
IMT, Lucca, Italy
~~U Southampton, UK~~

FMSPLE 2015

ETAPS, London, UK
11 April 2015

Outline

Formal methods in SPLE

Aim

- Use of formal methods to verify also the behaviour of SPL
- Smooth integration of SPL configuration and behaviour
- Process-algebraic model, automated analysis with Maude

Ongoing research presented at FMSPLE

'12 S. Gnesi & M. Petrocchi: *Towards an executable algebra for product lines*

'13 M.H. ter Beek, A. Lluch Lafuente & M. Petrocchi: *Combining declarative and procedural views in the specification and analysis of product families*

This year's novelty

1. Specify probabilistic configurations and behaviour
2. Semantics based on Discrete-Time Markov Chains
3. Quantitative analysis with statistical model checker

PFLAN: Probabilistic Feature-oriented Language

Concurrent constraint programming paradigm

- Probabilistic modelling: uncertainty, failure rates, randomisation
- Quantitative analysis: Quality of Service, reliability, performance

Combine declarative and procedural specification

- Constraint store allows to specify all ordinary feature constraints
- Rich set of process-algebraic operators allows to specify both the configuration and the behaviour of products

Declarative and procedural views closely related

1. process execution is constrained by the store to avoid inconsistencies
2. process can query the store to resolve configuration / behavioural option
3. process can update the store by adding new features (also at runtime)

PFLAN: Syntax

Rate $r \in \mathbb{R}^+$, actions $a \in \mathcal{A}$, propositions $p \in \mathcal{P}$, features $f, g \in \mathcal{F}$

(fragments)	F	$::=$	$[S \mid P]$
(constraints)	S, T	$::=$	$K \mid f \triangleright g \mid f \otimes g \mid S\ T \mid \top \mid \bot$
(processes)	P, Q	$::=$	$\emptyset \mid X \mid (A, r).P \mid P + Q \mid P; Q \mid P \parallel Q$
(actions)	A	$::=$	$a \mid \mathsf{install}(f) \mid \mathsf{ask}(K)$
(propositions)	K	$::=$	$p \mid \neg K \mid K \vee K$

Constraints

- Store: *consistent(S)*, inconsistent ($\bot$) or no constraint at all ($\top$)
- Universe $\mathcal{P}$ of propositions: predicates $has(f)$ and *in(context)*
- Action constraints $do(a) \rightarrow p$: guard to allow/forbid executing a

Specify probabilistic aspects of SPL models

- $(A, r).P$: perform action A with rate / weight r, then behave as P

$\Rightarrow$ Likelihood of user behaviour or of installation of a certain feature

PFLAN: Semantics

$\rightarrow\ \subseteq \mathbb{N}^{\mathbb{F}\times\mathbb{R}^{+}\times\mathbb{F}}$, with $\mathbb{F}$ set of all terms generated by F

$$(\textsc{Inst})\ \frac{consistent(S\ \ has(f))}{[S \mid (\mathrm{install}(f), r).P] \xrightarrow{r} [S\ \ has(f) \mid P]} \qquad (\textsc{Or})\ \frac{[S \mid P] \xrightarrow{r} [S' \mid P']}{[S \mid P + Q] \xrightarrow{r} [S' \mid P']}$$

$$(\textsc{Ask})\ \frac{S \vdash K}{[S \mid (\mathrm{ask}(K), r).P] \xrightarrow{r} [S \mid P]} \qquad (\textsc{Seq})\ \frac{[S \mid P] \xrightarrow{r} [S' \mid P']}{[S \mid P; Q] \xrightarrow{r} [S' \mid P'; Q]}$$

$$(\textsc{Act})\ \frac{S = (S'\ \ do(a) \rightarrow K) \qquad S \vdash K}{[S \mid (a, r).P] \xrightarrow{r} [S \mid P]} \qquad (\textsc{Par})\ \frac{[S \mid P] \xrightarrow{r} [S' \mid P']}{[S \mid P \parallel Q] \xrightarrow{r} [S' \mid P' \parallel Q]}$$

DTMC semantics

- normalising rates into [0..1] such that $\forall$ states s : $\sum s \xrightarrow{\ell} = 1$
- transition ℓabel corresponds to probability transition being executed

$\Rightarrow$ use SMC because in general the DTMC is too large to generate

1-1 correspondence with rewrite rules in Maude implementation

$\Rightarrow$ Implementation compact, easy to read and (efficiently) executable

Statistical Model Checking

Model Checking (MC)

- Automatically check whether a model satisfies a temporal logic property (LTL, CTL) and provide a counterexample if it doesn't
- Exhaustive, but suffers from state explosion problem
- BLAST, CADP, JPF, mCRL2, PRISM, (Nu)SMV, SPIN, UPPAAL, ...

Probabilistic Model Checking (PMC)

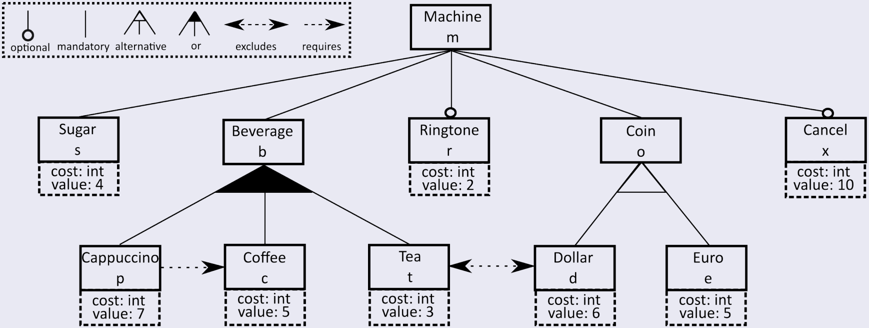
- Model check whether a stochastic model satisfies a temporal logic property (PCTL, CSL) with a probability greater than a certain threshold
- Model uncertainty/performance; do quantitative analysis (QoS, ...)
- CADP, LiQuor, MRMC, PARAM, PRISM, UPPAAL-PRO, ...

Statistical Model Checking (SMC)

- Simulation-based technique to statistically approximate (P)MC
- Highly parallelisable and automatable; tunable preciseness via CI
- APMC, PLASMA, PRISM, UPPAAL, (P)VeStA, MultiVeStA, YMER, ...

Running example: family of coffee machines

Structural constraints



Behavioural constraints

- **Initially** a coin must be inserted, **after** which the user must choose whether s/he wants sugar, **after** which s/he may select a beverage
- A ringtone must be rung **after** serving cappuccino, otherwise it may
- The coffee machine returns idle **after** the beverage has been taken

A specification

$$
\begin{aligned}
F &\doteq [S \mid Q] \\
S &\doteq S_1 \;\; S_2 \\
S_1 &\doteq \mathit{has}(\mathit{euro}) \vee \mathit{has}(\mathit{dollar}) \qquad \mathit{has}(\mathit{euro}) \rightarrow \mathit{in}(\mathit{Europe}) \qquad \mathit{has}(\mathit{dollar}) \rightarrow \mathit{in}(\mathit{Canada}) \\
&\quad\; \mathit{has}(\mathit{coffee}) \vee \mathit{has}(\mathit{tea}) \vee \mathit{has}(\mathit{cappuccino}) \qquad \mathit{has}(\mathit{tea}) \rightarrow \mathit{in}(\mathit{Europe}) \qquad \mathit{dollar} \otimes \mathit{euro} \\
&\quad\; \mathit{cappuccino} \triangleright \mathit{coffee} \qquad \mathit{do}(\mathit{euro}) \rightarrow \mathit{has}(\mathit{euro}) \;\cdots\; \mathit{do}(\mathit{out\_of\_milk}) \rightarrow \mathit{has}(\mathit{cappuccino}) \\
S_2 &\doteq \mathit{in}(\mathit{Europe}) \\
Q &\doteq D + (\mathrm{install}(\mathit{pre}-\mathit{conf}), 10).R \\
D &\doteq (\mathrm{install}(\mathit{euro}),10).Q + (\mathrm{install}(\mathit{dollar}),10).Q + (\mathrm{install}(\mathit{sugar}),10).Q + (\mathrm{install}(\mathit{cancel}), 7).Q \\
&\quad\; + (\mathrm{install}(\mathit{coffee}), 9).Q + (\mathrm{install}(\mathit{tea}), 6).Q + (\mathrm{install}(\mathit{cappuccino}), 3).Q \\
R &\doteq ((\mathit{euro}, 25).\emptyset + (\mathit{dollar}, 25).\emptyset); P_1 \\
P_0 &\doteq (\mathit{return\_coin},10).R + (\mathit{no\_return},1).R \\
P_1 &\doteq (\mathit{cancel}, 5).P_0 + P_2 + P_3 \\
P_2 &\doteq (\mathit{sugar},15).\emptyset; ((\mathit{pour\_sugar},10).P_3 + (\mathit{out\_of\_sugar}, 2).P_1) \\
P_3 &\doteq (\mathit{coffee}, 20).P_4 + (\mathit{tea},12).P_5 + (\mathit{cappuccino}, 8).P_6 \\
P_4 &\doteq (\mathit{pour\_coffee},10).P_8 + (\mathit{out\_of\_coffee}, 2).P_3 \\
P_5 &\doteq (\mathit{pour\_tea},10).P_8 + (\mathit{out\_of\_tea}, 2).P_3 \\
P_6 &\doteq (\mathit{pour\_milk},10).\emptyset; ((\mathit{pour\_coffee},10).P_8 + (\mathit{out\_of\_coffee}, 2).R) + (\mathit{out\_of\_milk}, 2).P_3 \\
P_8 &\doteq P_9 + (\mathrm{install}(\mathit{ringtone}), 8).(\mathit{ringtone}, 18).P_9 \\
P_9 &\doteq (\mathit{take\_drink},10).R + (\mathit{no\_cup},1).R
\end{aligned}
$$

A specification: initial configuration

$$F \doteq [S \mid Q]$$

$$S \doteq S_1\ S_2$$

$$S_1 \doteq has(euro) \vee has(dollar) \qquad has(euro) \rightarrow in(Europe) \qquad has(dollar) \rightarrow in(Canada)$$
$$has(coffee) \vee has(tea) \vee has(cappuccino) \qquad has(tea) \rightarrow in(Europe) \qquad dollar \otimes euro$$
$$cappuccino \triangleright coffee \qquad do(euro) \rightarrow has(euro) \ \cdots \ do(out\_of\_milk) \rightarrow has(cappuccino)$$

$$S_2 \doteq in(Europe)$$

$$\boldsymbol{Q} \doteq \boldsymbol{D} + (\mathbf{install}(\boldsymbol{pre-conf}), \mathbf{10}).R$$

$$\boldsymbol{D} \doteq (\mathbf{install}(\boldsymbol{euro}),\mathbf{10}).\boldsymbol{Q} + \mathbf{install}(\boldsymbol{dollar}),\mathbf{10}).\boldsymbol{Q} + (\mathbf{install}(\boldsymbol{sugar}),\mathbf{10}).\boldsymbol{Q} + (\mathbf{install}(\boldsymbol{cancel}),\mathbf{7}).\boldsymbol{Q}$$
$$+ (\mathbf{install}(\boldsymbol{coffee}), \mathbf{9}).\boldsymbol{Q} + (\mathbf{install}(\boldsymbol{tea}), \mathbf{6}).\boldsymbol{Q} + (\mathbf{install}(\boldsymbol{cappuccino}), \mathbf{3}).\boldsymbol{Q}$$

$$R \doteq \ldots \ runtime\ behaviour\ \ldots$$

Declarative part

- Some features can be installed during initial configuration phase D
- Rates determine order of installation (e.g. independent designers ‘competing’ to install the features for which they are responsible)
- Probability to install *sugar* in first step given *pre-conf*, *euro*, *cancel*, *coffee*, *tea*, *cappuccino* can be installed: $\frac{10}{10+10+10+7+9+6+3} = \frac{2}{11}$
- *pre-conf* signals that a product (coffee machine) is pre-configured

A specification: runtime behaviour

Procedural part

- When a user cancels, the machine turns idle and returns the coin

... but with a probability of $1/11$, the machine does not return the coin

- Coffee has a higher probability ($\frac{1}{2}$) of being chosen than tea (0.3) and cappuccino ($\frac{1}{5}$)
- Features may also be installed (bound) at runtime (e.g. *ringtone*)

$$
\begin{aligned}
D &\doteq \ \ldots \ \textit{initial configuration} \ \ldots \\
R &\doteq ((euro, 25).\emptyset + (dollar, 25).\emptyset); P_1 \\
\boldsymbol{P_0} &\doteq (\boldsymbol{return\_coin}, \mathbf{10}).\boldsymbol{R} + (\boldsymbol{no\_return}, \mathbf{1}).\boldsymbol{R} \\
P_1 &\doteq (\boldsymbol{cancel}, \mathbf{5}).\boldsymbol{P_0} + P_2 + P_3 \\
P_2 &\doteq (sugar, 15).\emptyset; ((pour\_sugar, 10).P_3 + (out\_of\_sugar, 2).P_1) \\
\boldsymbol{P_3} &\doteq (\boldsymbol{coffee}, \mathbf{20}).P_4 + (\boldsymbol{tea}, \mathbf{12}).P_5 + (\boldsymbol{cappuccino}, \mathbf{8}).P_6 \\
P_4 &\doteq (pour\_coffee, 10).P_8 + (out\_of\_coffee, 2).P_3 \\
P_5 &\doteq (pour\_tea, 10).P_8 + (out\_of\_tea, 2).P_3 \\
P_6 &\doteq (pour\_milk, 10).\emptyset; ((pour\_coffee, 10).P_8 + (out\_of\_coffee, 2).R) + (out\_of\_milk, 2).P_3 \\
P_8 &\doteq P_9 + (\mathbf{install}(\boldsymbol{ringtone}), \mathbf{8}).(ringtone, 18).P_9 \\
P_9 &\doteq (take\_drink, 10).R + (no\_cup, 1).R
\end{aligned}
$$

Declarative and procedural feature configuration

Select feature f in an *explicit* and *declarative* way

- Include the proposition $has(f)$ in the initial store
- For core features that are mandatory for all products

Select feature f in an *implicit* and *declarative* way

- Derive f as a consequence of other constraints
- For features that apparently are not mandatory, but that are enforced by the constraints (e.g. in a store with constraints $g \triangleright f$ and $has(g)$, the presence of f can be inferred)

Install feature f dynamically in a *procedural* way

- Install f during the runtime execution of a process
- Allows designers to delay feature configuration decisions to runtime
- Useful to model staged configuration (what about dynamic SPL?)

MultiVeStA and its property language MultiQuaTEx

MultiVeStA extends (P)VeStA

- Developed and maintained by Stefano Sebastio & Andrea Vandin
  `https://code.google.com/p/multivesta/`
- Used so far to analyse transportation systems, volunteer clouds, crowd-steering, and swarm robotic scenarios, now product lines

MultiQuaTEx extends QuaTEx

- Statistical estimations of quantitative properties in MultiQuaTEx
- Expected values of properties that can take on any value from $\mathbb{R}$
  "Average cost of products generated from an SPL specification?"
- Computed as mean value of n samples taken from n simulations, with n large enough such that $(1-\alpha) \times 100\%$ CI is bounded by δ
  (i.e. if an expression is estimated as x, then with probability $(1-\alpha)$ its actual expected value belongs to the interval $[\overline{x} - {}^{\delta}/_{2}, \overline{x} + {}^{\delta}/_{2}]$)

Quantitative analyses of running example

Example properties (removed context restriction)

- P_1 Probability to run into deadlock before completing pre-configuration
- P_2 Probability to have a feature installed at pre-configuration (or step x)
- P_3 Average cost of (intermediate) product at pre-configuration (or step x)

MultiQuaTEx expressions corresponding to P_2 and P_3

```
ProductCostAfterPreconf() =
 if {s.rval("pre-conf") == 1.0} then s.rval("cost")
                                else #ProductCostAfterPreconf() fi ;
IsInstalledAfterPreconf(feature) =
 if {s.rval("pre-conf") == 1.0} then s.rval(feature)
                                else #IsInstalledAfterPreconf({feature}) fi ;
eval E[ProductCostAfterPreconf()];
eval E[IsInstalledAfterPreconf("sugar")]; eval E[IsInstalledAfterPreconf("ringtone")];
eval E[IsInstalledAfterPreconf("cancel")]; eval E[IsInstalledAfterPreconf("cappuccino")];
eval E[IsInstalledAfterPreconf("coffee")]; eval E[IsInstalledAfterPreconf("tea")];
eval E[IsInstalledAfterPreconf("dollar")]; eval E[IsInstalledAfterPreconf("euro")];
```

Probability installing features (P_2) and average cost products (P_3)

Rate	*sugar*	*ringtone*	*cancel*	*cappuccino*	*coffee*	*tea*	*dollar*	*euro*	Cost
10	0.49	0.0	0.45	0.13	0.50	0.40	0.33	0.38	14.07
50	0.17	0.0	0.11	0.0	0.14	0.10	0.12	0.13	4.46

Parametric quantitative analyses

Modelling and analysis with PFLan and MultiVeStA

1. Probabilistic behaviour of users
2. Likelihood of installing features
3. Average cost of products
4. Probabilistic quantifications of ordinary temporal properties

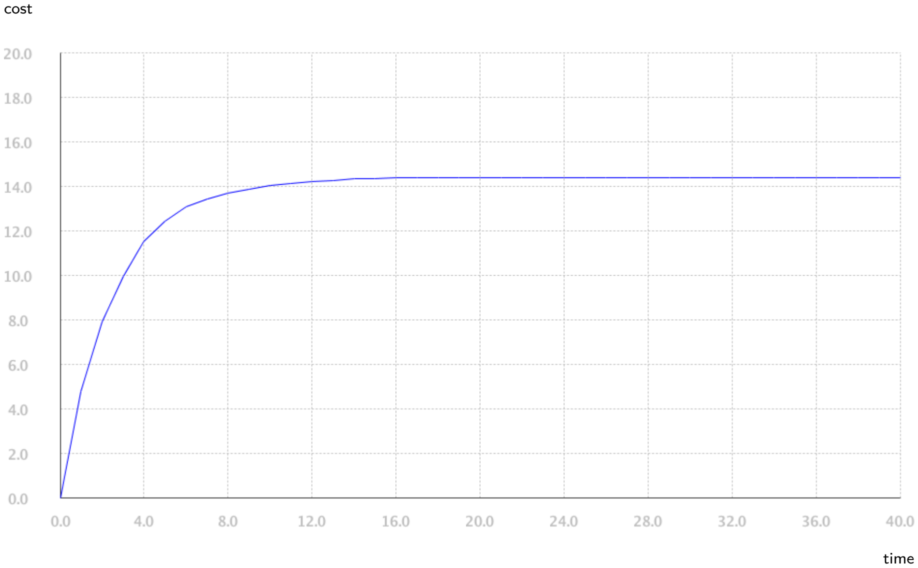
   Probability that coffee is poured while no cup was available
5. Future work: variable rates that depend on the constraint store

Probability of a user choosing a beverage depends on product's location

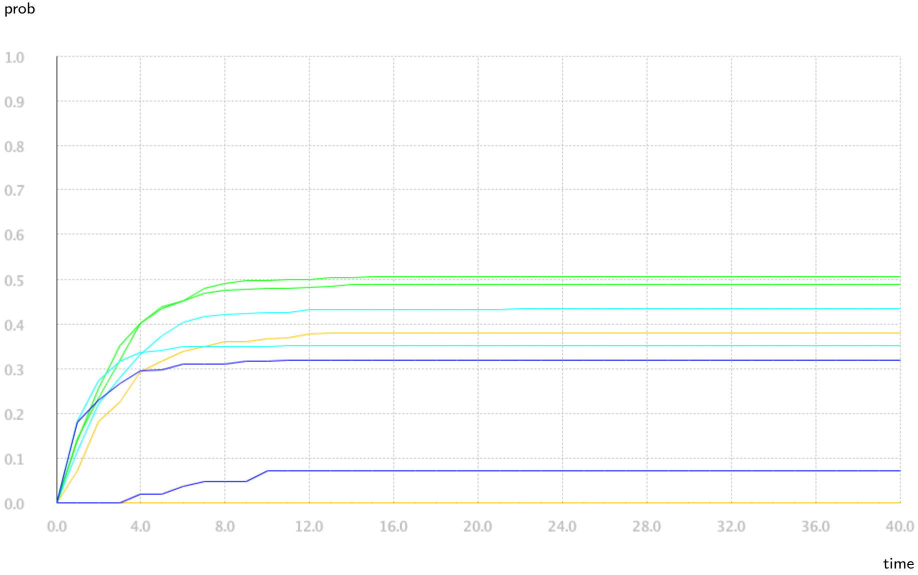
Parametric expressions for P_2 and P_3 (w.r.t. simulation steps)

```
ProductCostAtStep(step) =
 if {s.rval("steps") == step} then s.rval("cost")
                              else #ProductCostAfterPreconf(step)
 fi ;
IsInstalledAtStep(step,feature) = ...
eval parametric(E[ ProductCostAtStep(step) ], E[ IsInstalledAtStep(step,"sugar") ], ... ,
                E[ IsInstalledAtStep(step,"euro") ], step, 0.0, 1.0, 40.0) ;
```

Average product cost as time progresses



Average feature installation probability as time progresses



Conclusions and future work

Probabilistic Feature-oriented Language PFLan

+ Specify and verify both declarative and procedural aspects of SPL
+ Semantics neatly unifies static and dynamic feature configuration
+ Model with quantitative information, allowing quantitative analyses

Implementation in Maude/MultiVeStA for SAT solving and SMC

! Combine simplicity of testing with formality and precision of (P)MC

Ongoing future work (see you at SPLC?)

√ Explicit uninstallation of a feature, e.g. due to its malfunctioning or due to the need to replace it with a better (version of the) feature

√ Modelling quantitative (non-Boolean) constraints among features

√ Integration with Microsoft's highly efficient SMT solver Z3

? Test the scalability of our approach