Family-based model checking with mCRL2

Maurice ter Beek, Erik de Vink, and Tim Willemse

ISTI-CNR, Pisa, Italy & TU/e, Eindhoven, The Netherlands

VaMoS 2017

Eindhoven, The Netherlands
February 2nd, 2017

Family-based modelling and analysis

Software product line (SPL) or product family

- Configurable (software) system whose variants (products) differ by the provided features, i.e. the functionality that is relevant for an end-user
- Popular in embedded and critical systems domain: formal modelling and analysis techniques for proving SPL behaviour correct are widely studied
  Thüm et al., A classification and survey of analysis strategies for SPLs @ *ACM Comput. Surv.* (2014)
- Challenge existing formal methods and tools by potentially high number of different products, each giving rise to a large state space in general

⇒ Lift success stories from products to families exploiting variability

Dedicated family-based SPL behavioural models and model checkers (e.g. FTSs, Feature Nets, MTSs, PL-CCS, DeltaCCS, QFLan, SNIP, ProVeLines, VMC) but recently. . .

Dimovski et al., Family-based model checking without a family-based model checker @ SPIN'15
Chrszon et al., Family-based modeling and analysis for probabilistic systems – featuring ProFeat @ FASE'16
Dimovski et al., Variability-specific Abstraction Refinement for Family-based Model Checking @ FASE'17

Family-based modelling and analysis

quanticol
www.quanticol.eu

Software product line (SPL) or product family

- Configurable (software) system whose variants (products) differ by the provided features, i.e. the functionality that is relevant for an end-user
- Popular in embedded and critical systems domain: formal modelling and analysis techniques for proving SPL behaviour correct are widely studied
  Thüm et al., A classification and survey of analysis strategies for SPLs @ *ACM Comput. Surv.* (2014)
- Challenge existing formal methods and tools by potentially high number of different products, each giving rise to a large state space in general

$\Rightarrow$ Lift success stories from products to families exploiting variability

Dedicated family-based SPL behavioural models and model checkers (e.g. FTSs, Feature Nets, MTSs, PL-CCS, DeltaCCS, QFLan, SNIP, ProVeLines, VMC) but recently...

Dimovski et al., Family-based model checking without a family-based model checker @ SPIN'15
Chrszon et al., Family-based modeling and analysis for probabilistic systems – featuring ProFeat @ FASE'16
Dimovski et al., Variability-specific Abstraction Refinement for Family-based Model Checking @ FASE'17

Family-based modelling and analysis

Software product line (SPL) or product family

- Configurable (software) system whose variants (products) differ by the provided features, i.e. the functionality that is relevant for an end-user
- Popular in embedded and critical systems domain: formal modelling and analysis techniques for proving SPL behaviour correct are widely studied
  Thüm et al., A classification and survey of analysis strategies for SPLs @ *ACM Comput. Surv.* (2014)
- Challenge existing formal methods and tools by potentially high number of different products, each giving rise to a large state space in general

$\Rightarrow$ Lift success stories from products to families exploiting variability

Dedicated family-based SPL behavioural models and model checkers (e.g. FTSs, Feature Nets, MTSs, PL-CCS, DeltaCCS, QFLan, SNIP, ProVeLines, VMC) but recently...

Dimovski et al., Family-based model checking without a family-based model checker @ SPIN'15
Chrszon et al., Family-based modeling and analysis for probabilistic systems – featuring ProFeat @ FASE'16
Dimovski et al., Variability-specific Abstraction Refinement for Family-based Model Checking @ FASE'17

Family-based modelling and analysis

Software product line (SPL) or product family

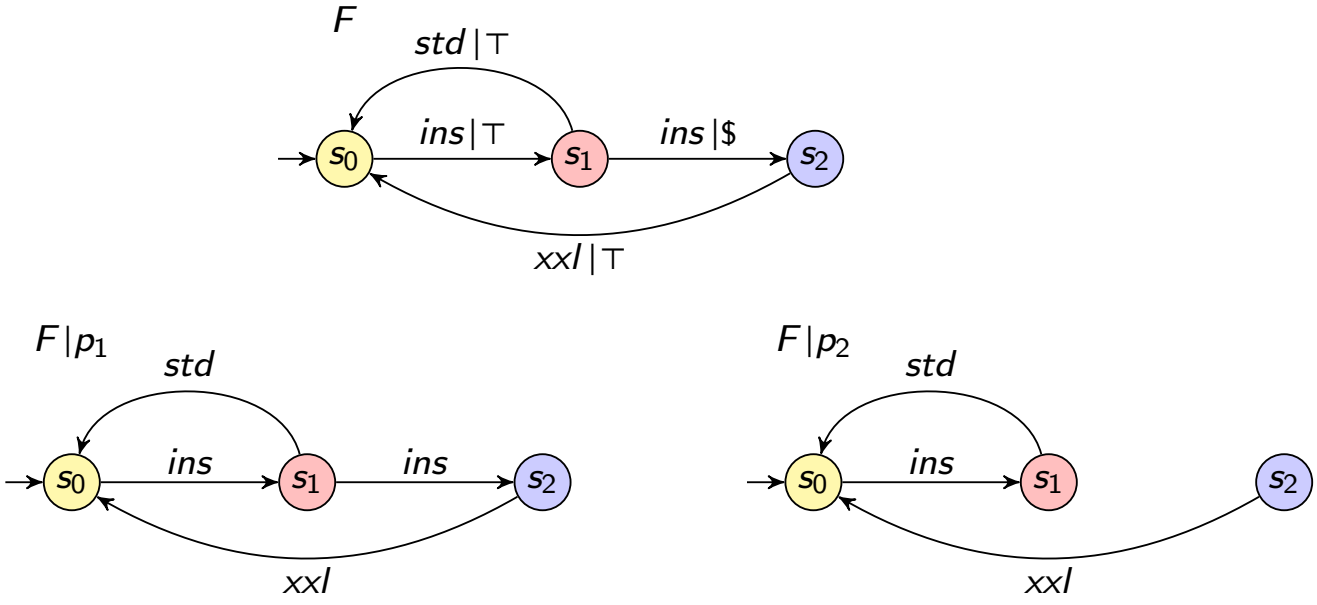
- Configurable (software) system whose variants (products) differ by the provided features, i.e. the functionality that is relevant for an end-user
- Popular in embedded and critical systems domain: formal modelling and analysis techniques for proving SPL behaviour correct are widely studied
  Thüm et al., A classification and survey of analysis strategies for SPLs @ *ACM Comput. Surv.* (2014)
- Challenge existing formal methods and tools by potentially high number of different products, each giving rise to a large state space in general

$\Rightarrow$ Lift success stories from products to families exploiting variability

Dedicated family-based SPL behavioural models and model checkers (e.g. FTSs, Feature Nets, MTSs, PL-CCS, DeltaCCS, QFLan, SNIP, ProVeLines, VMC) but recently...

Dimovski et al., Family-based model checking without a family-based model checker @ SPIN'15
Chrszon et al., Family-based modeling and analysis for probabilistic systems – featuring ProFeat @ FASE'16
Dimovski et al., Variability-specific Abstraction Refinement for Family-based Model Checking @ FASE'17

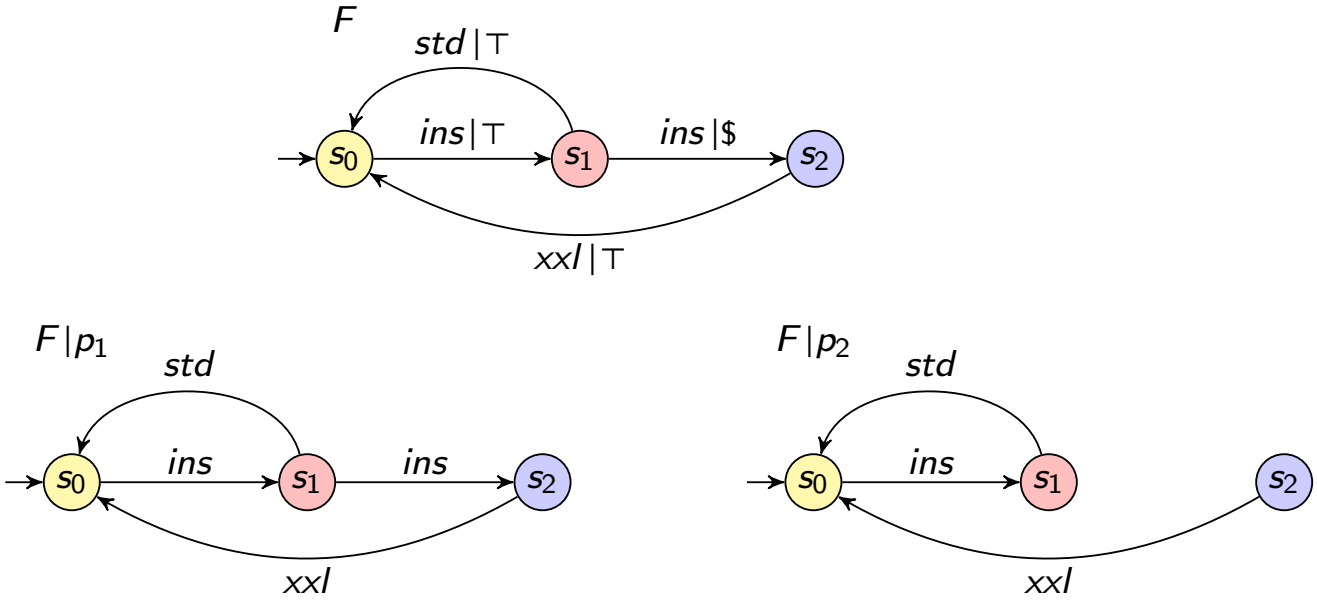
Product line of (four) coffee machines with independent features $\{\$, €\}$



Products with feature $ can obtain an *xxl* coffee upon coin insertion, but products without cannot

how to express this? and how to model check this efficiently?

Product line of (four) coffee machines with independent features $\{\$, €\}$



Products with feature \$ can obtain an *xxl* coffee upon coin insertion, but products without cannot

how to express this? and how to model check this efficiently?

mCRL2: language and toolset

quanticol
www.quanticol.eu

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation,
  behavioural reduction,
  model checking
- Highly optimised,
  actively maintained
- Intermediate artifacts
  user-accessible

mCRL2: language and toolset

quanticol
www.quanticol.eu

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation,
  behavioural reduction,
  model checking
- Highly optimised,
  actively maintained
- Intermediate artifacts
  user-accessible

mCRL2: language and toolset

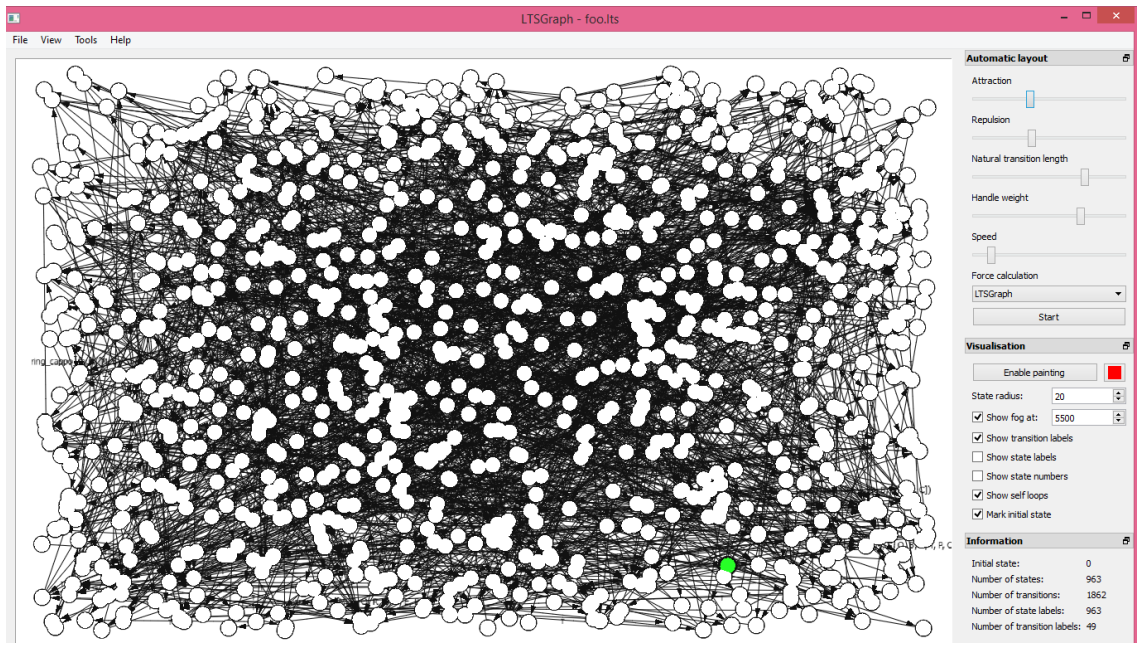
quanticol
www.quanticol.eu

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation, behavioural reduction, model checking
- Highly optimised, actively maintained
- Intermediate artifacts user-accessible

mCRL2: language and toolset

quanticol
www.quanticol.eu

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation,
- behavioural reduction,
- model checking
- Highly optimised,
- actively maintained
- Intermediate artifacts
- user-accessible

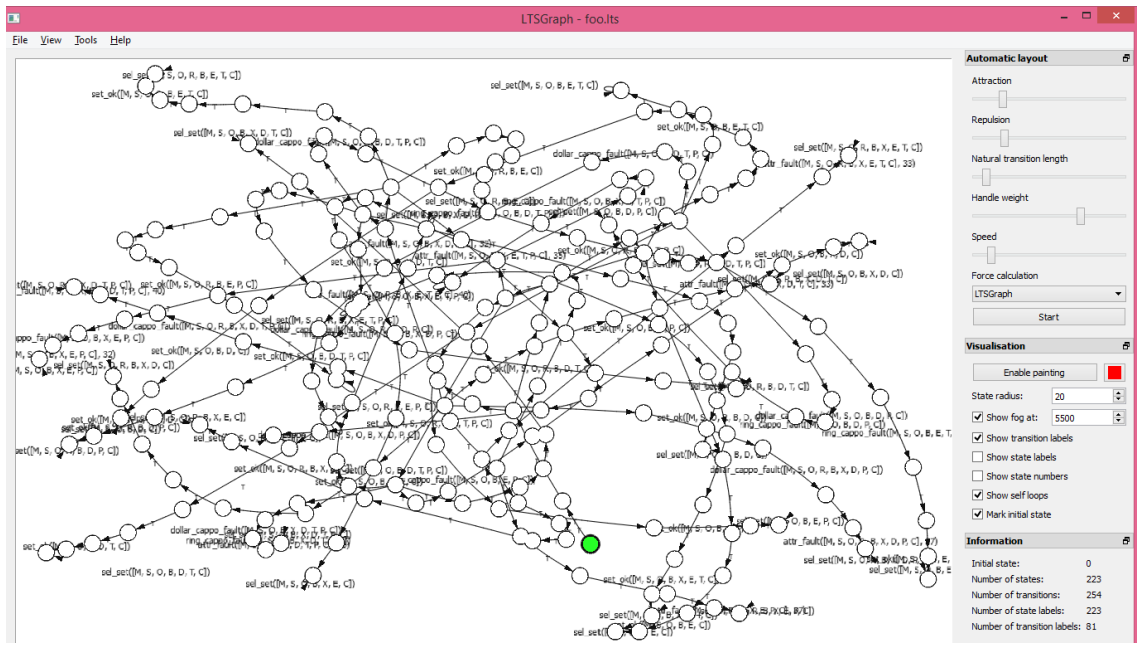


mCRL2: language and toolset

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation,

  behavioural reduction,

  model checking
- Highly optimised, actively maintained
- Intermediate artifacts user-accessible

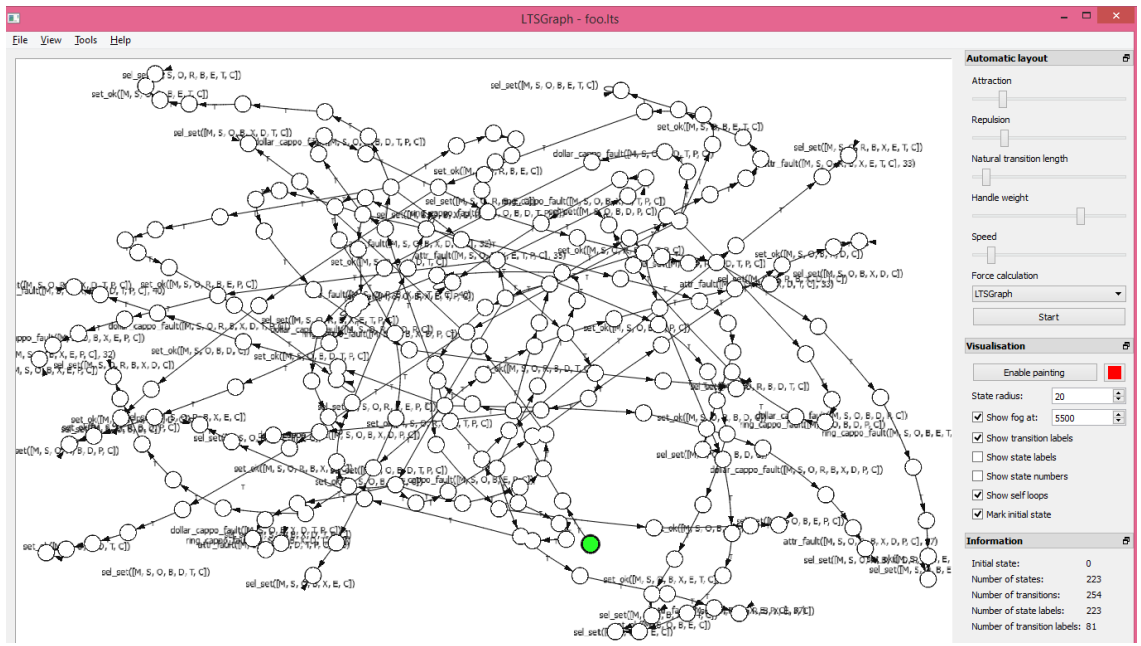


mCRL2: language and toolset

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation,

  behavioural reduction,

  model checking
- Highly optimised, actively maintained
- Intermediate artifacts user-accessible



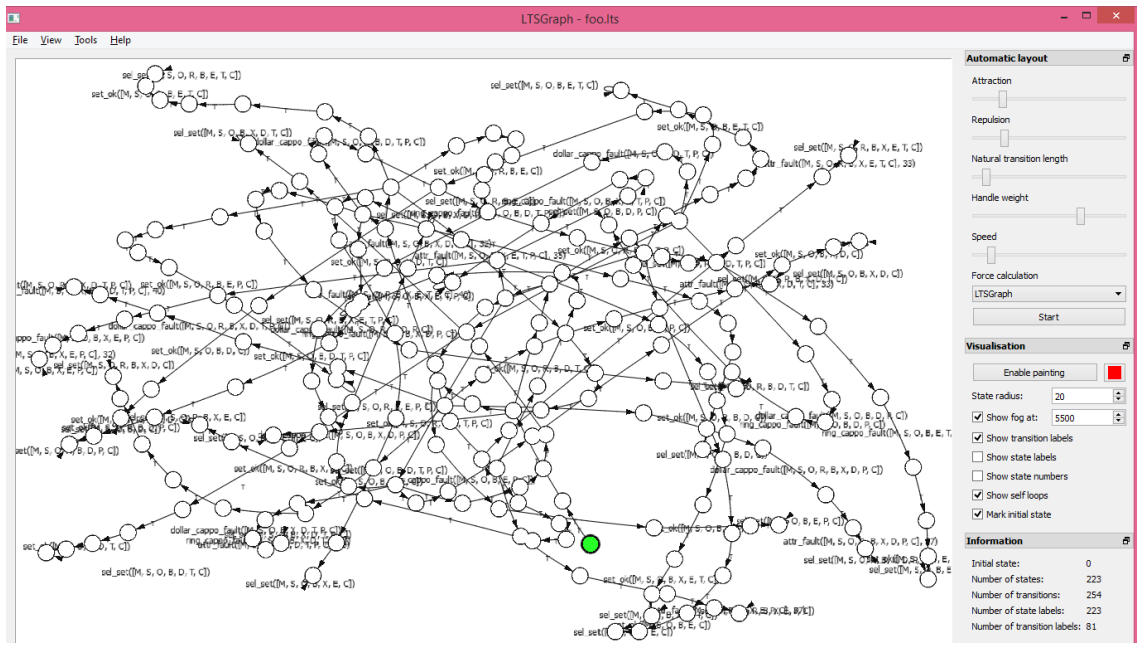
mCRL2: language and toolset

quanticol
www.quanticol.eu

- Formal, process-algebraic specification of distributed and concurrent systems, associated industrial-strength toolset
- Explore 10^6 states/second, state spaces up to 10^{12} states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, parametrised actions
- Modal μ-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation,

  behavioural reduction,

  model checking
- Highly optimised, actively maintained
- Intermediate artifacts user-accessible



The modal μ-calculus μL (over LTSs)

set of actions $\mathcal{A}$, set of variables $\mathcal{X}$

μ-calculus μL over $\mathcal{A}$ and $\mathcal{X}$, formula $\varphi \in \mu L$ given by

$$\begin{aligned}\varphi ::= {} & \bot \mid \top \mid \\ & \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \\ & \langle a \rangle \varphi \mid [a]\varphi \mid \\ & X \mid \mu X.\varphi \mid \nu X.\varphi\end{aligned}$$

duality $\langle a \rangle \varphi \equiv \neg[a]\neg\varphi$, a positive normal form avoids negations

---

for $\mu X.\varphi$ and $\nu X.\varphi$, all free occurrences of X in φ are in the scope of an even number of negations (guarantees well-definedness fixpoint formulae)

A feature μ-calculus μL_f (over FTSs)

set of actions $\mathcal{A}$, set of features $\mathcal{F}$, set of variables $\mathcal{X}$

feature μ-calculus μL_f over $\mathcal{A}$, $\mathcal{F}$, and $\mathcal{X}$, formula $\varphi_f \in \mu L_f$ given by

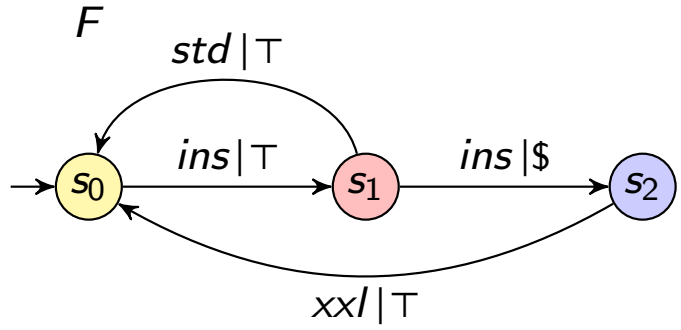
$$\begin{aligned}\varphi_f ::= \; & \bot \mid \top \mid \\ & \neg\varphi_f \mid \varphi_f \vee \psi_f \mid \varphi_f \wedge \psi_f \mid \\ & \langle a|\chi\rangle\varphi_f \mid [a|\chi]\varphi_f \mid \\ & X \mid \mu X.\varphi_f \mid \nu X.\varphi_f\end{aligned}$$

for $\mu X.\varphi_f$ and $\nu X.\varphi_f$ an even number of negations as before

Examples of μL_f-formulae

- $\langle ins|\top\rangle\,(\,[ins|\in]\bot \wedge \langle std|\top\rangle\top\,)$

  "the family of products P that can execute *ins*, after which *ins* cannot be executed by products satisfying $\in$, while *std* can be executed by all products of P"



- $\nu X.\mu Y.((\,[ins|\in]\,Y \wedge [xxl|\in]\,Y\,) \wedge [std|\in]\,X\,)$

  "for the (sub)family of products with feature $\in$, action *std* occurs infinitely often on all infinite runs over $\{ins, xxl, std\}$"

- $[\,\mathtt{true}^*\,|\,\top\,]\;(\,(\,[\,ins\,|\,\$\,]\,\langle\,\mathtt{true}^*.xxl\,|\,\top\,\rangle\,\top\,) \wedge [\,xxl\,|\,\neg\$\,]\,\bot\,)$

  "products with feature \$ can obtain an *xxl* coffee upon coin insertion, but products without cannot"

multi-feature μL_f formula, novel also w.r.t. fLTL and fCTL

Main results of paper @ FMSPLE'16

Model checking a μL_f-formula over an FTS for an individual product reduces to model checking a μL-formula over the corresponding LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 1 $s, \{p\} \models_F \varphi_f \iff s \models_{F|p} pr(\varphi_f, p)$

closed $\varphi_f \in \mu L_f$, $s \in S$, product $p \in \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Note: in general $s, P \not\models_F \varphi_f$ does not imply $s \not\models_{F|p} pr(\varphi_f, p)$ for all products in family P

Main results of paper @ FMSPLE'16

Model checking a μL_f-formula over an FTS for an individual product reduces to model checking a μL-formula over the corresponding LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 1 $s, \{p\} \models_F \varphi_f \iff s \models_{F|p} pr(\varphi_f, p)$

closed $\varphi_f \in \mu L_f$, $s \in S$, product $p \in \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Note: in general $s, P \not\models_F \varphi_f$ does not imply $s \not\models_{F|p} pr(\varphi_f, p)$ for all products in family P

Main results of paper @ FMSPLE'16

Model checking a μL_f-formula over an FTS for an individual product reduces to model checking a μL-formula over the corresponding LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 1 $s, \{p\} \models_F \varphi_f \iff s \models_{F|p} pr(\varphi_f, p)$

closed $\varphi_f \in \mu L_f$, $s \in S$, product $p \in \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Note: in general $s, P \not\models_F \varphi_f$ does not imply $s \not\models_{F|p} pr(\varphi_f, p)$ for all products in family P

A μ-calculus with data μL_{FO} (over parametrised LTSs $L(F)$)

quanticol
www.quanticol.eu

set of 'sorted' actions $\mathcal{A}$, set of features $\mathcal{F}$, set of data variables $\mathcal{V}$, set of recursion variables $\tilde{\mathcal{X}}$

μ-calculus with data μL_{FO} over $\mathcal{A}, \mathcal{F}, \mathcal{V}$ and $\tilde{\mathcal{X}}$, formula $\varphi \in \mu L_{\mathrm{FO}}$ given by

$$
\begin{aligned}
\varphi_f ::= \; & \bot \mid \top \mid \\
& \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \\
& \gamma_1 \Rightarrow \gamma_2 \mid && (Q_{\gamma_1} \subseteq Q_{\gamma_2}) \\
& \exists v.\varphi \mid \forall v.\varphi \mid \\
& \langle a(v) \rangle \varphi \mid [a(v)]\varphi \mid \\
& \tilde{X}(\gamma) \mid \mu\tilde{X}(v_{\tilde{X}} := \gamma).\varphi \mid \nu\tilde{X}(v_{\tilde{X}} := \gamma).\varphi
\end{aligned}
$$

with the usual restrictions on free variables and the number of negations

A main result of paper @ FASE'17

Do not model check a μL_f-formula over an FTS, but model check the corresponding μL_{FO}-formula over the corresponding parametrised LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 3 $s, P \models_F \varphi_f \iff s \models_{L(F)} tr(\gamma_P, \varphi_f)$

closed $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Lemma 1 $s, P \models_F \varphi_f^c \implies \forall p \in P : s \not\models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

A main result of paper @ FASE'17

Do not model check a μL_f-formula over an FTS, but model check the corresponding μL_{FO}-formula over the corresponding parametrised LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 3 $s, P \models_F \varphi_f \iff s \models_{L(F)} tr(\gamma_{\mathrm{P}}, \varphi_f)$

closed $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Lemma 1 $s, P \models_F \varphi_f^c \implies \forall p \in P : s \not\models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Recall a result of paper @ FMSPLE'16

Do not model check a μL_f-formula over an FTS, but model check the corresponding μL_{FO}-formula over the corresponding parametrised LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 3 $s, P \models_F \varphi_f \iff s \models_{L(F)} tr(\gamma_{\mathrm{P}}, \varphi_f)$

closed $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Lemma 1 $s, P \models_F \varphi_f^c \implies \forall p \in P : s \not\models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Another result of paper @ FASE'17

Do not model check a μL_f-formula over an FTS, but model check the corresponding μL_{FO}-formula over the corresponding parametrised LTS

Given an FTS F and a set of products $\mathcal{P}$

Theorem 3 $s, P \models_F \varphi_f \iff s \models_{L(F)} tr(\gamma_{\mathrm{P}}, \varphi_f)$

closed $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Theorem 2 $s, P \models_F \varphi_f \implies \forall p \in P : s \models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Lemma 1 $s, P \models_F \varphi_f^c \implies \forall p \in P : s \not\models_{F|p} pr(\varphi_f, p)$

closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

Family-based partitioning for μL_f

Given a negation-free φ_f and a family P, compute a partitioning $(P_\oplus, P_\ominus)$ of P satisfying

$$\forall p \in P_\oplus : s_*, p \models_{F|p} pr(\varphi_f, p) \text{ and } \forall p \in P_\ominus : s_*, p \not\models_{F|p} pr(\varphi_f, p)$$

closed, negation-free $\varphi_f \in \mu L_f$, family $P \subseteq \mathcal{P}$

Algorithm 1 Family-Based Partitioning

```
1:  function FBP(P, φ_f)
2:      if s_*, P ⊨_F φ_f then return (P, ∅)
3:      else
4:          if s_*, P ⊨_F φ_f^c then return (∅, P)
5:          else partition P into (P_1, P_2)
6:              (P_1^+, P_1^-) ← FBP(P_1, φ_f)
7:              (P_2^+, P_2^-) ← FBP(P_2, φ_f)
8:              return (P_1^+ ∪ P_2^+, P_1^- ∪ P_2^-)
9:          end if
10:     end if
11: end function
```

Family-based partitioning for μL_f

Theorem 4 $\mathrm{FBP}(P, \varphi_f)$ terminates and returns a partitioning $(P_\oplus, P_\ominus)$ of P satisfying

$$\forall p \in P_\oplus : s_*, p \models_{F|p} pr(\varphi_f, p) \text{ and } \forall p \in P_\ominus : s_*, p \not\models_{F|p} pr(\varphi_f, p)$$

closed, negation-free $\varphi_f \in \mu L_f$, family $P \subseteq \mathcal{P}$

Algorithm 1 Family-Based Partitioning

```
1:  function FBP(P, φ_f)
2:      if s_*, P ⊨_F φ_f then return (P, ∅)
3:      else
4:          if s_*, P ⊨_F φ_f^c then return (∅, P)
5:          else partition P into (P_1, P_2)
6:              (P_1^+, P_1^-) ← FBP(P_1, φ_f)
7:              (P_2^+, P_2^-) ← FBP(P_2, φ_f)
8:              return (P_1^+ ∪ P_2^+, P_1^- ∪ P_2^-)
9:          end if
10:     end if
11: end function
```

Minepump SPL benchmark ($|\mathcal{P}| = 2^7$)

Classen et al., Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking. *IEEE Trans. Softw. Eng.* (2013)

quanticol
www.quanticol.eu

Φ	property in μL_f	result	one-by-one	all-in-one
φ_1	*Absence of deadlock* $[\,\mathtt{true}^*\,]\,\langle\,\mathtt{true}\,\rangle\,\top$	128/0	10.02	2.07
φ_2	*The controller cannot infinitely often receive water level readings* $\mu X\,.\,[\,(\,\neg \mathit{levelMsg}\,)^*\,.\,\mathit{levelMsg}\,]\,X$	0/128	10.18	0.16
φ_3	*The controller cannot fairly receive each of the three message types* $\mu X\,.\,(\,[\,\mathtt{true}^*\,.\,\mathit{commandMsg}\,]\,X \vee [\,\mathtt{true}^*\,.\,\mathit{alarmMsg}\,]\,X \vee [\,\mathtt{true}^*\,.\,\mathit{levelMsg}\,]\,X\,)$	0/128	24.33	0.25
φ_4	*The pump cannot be switched on infinitely often* $(\,\mu X\,.\,\nu Y\,.\,(\,[\,\mathit{pumpStart}\,.\,(\,\neg \mathit{pumpStop}\,)^*\,.\,\mathit{pumpStop}\,]\,X \wedge [\,\neg \mathit{pumpStart}\,]\,Y\,)\,)$ $\wedge\,(\,[\,\mathtt{true}^*\,.\,\mathit{pumpStart}\,]\,\mu Z\,.\,[\,\neg \mathit{pumpStop}\,]\,Z\,)$	96/32	21.09	0.89
φ_5	*The system cannot be in a situation in which the pump runs indefinitely in the presence of methane* $[\,\mathtt{true}^*\,]\;(\,([\mathit{pumpStart}\,.\,(\,\neg \mathit{pumpStop}\,)^*\,.\,\mathit{methaneRise}]\;\mu X.\,[R]X\,)$ $\wedge\,([\mathit{methaneRise}\,.\,(\,\neg \mathit{methaneLower}\,)^*\,.\,\mathit{pumpStart}]\;\mu X.\,[R]X\,)\,)$ for $R = \neg(\,\mathit{pumpStop} + \mathit{methaneLower}\,)$	96/32	17.26	0.86
φ_6	*Assuming fairness (φ_3), the system cannot be in a situation in which the pump runs indefinitely in the presence of methane (φ_5)* $[\,\mathtt{true}^*\,]\;(\,([\,\mathit{pumpStart}\,.\,(\,\neg \mathit{pumpStop}\,)^*\,.\,\mathit{methaneRise}\,]\,\Psi\,)$ $\wedge\,([\,\mathit{methaneRise}\,.\,(\,\neg \mathit{methaneLower}\,)^*\,.\,\mathit{pumpStart}\,]\,\Psi\,)\,)$ for $\Psi = \mu X\,.\,(\,[\,R^*\,.\,\mathit{commandMsg}\,]\,X \vee [\,R^*\,.\,\mathit{alarmMsg}\,]\,X \vee [\,R^*\,.\,\mathit{levelMsg}\,]\,X\,)$ and R as before	112/16	27.32	3.67
φ_7	*The controller can always eventually receive/read a message, i.e. return to its initial state from any state* $[\,\mathtt{true}^*\,]\,\langle\,\mathtt{true}^*\,.\,\mathit{receiveMsg}\,\rangle\,\top$	128/0	18.36	2.40
φ_8	*Invariantly the pump is not started when the low water level signal fires* $[\,\mathtt{true}^*\,.\,\mathit{lowLevel}\,.\,(\neg(\mathit{normalLevel} + \mathit{highLevel}))^*\,.\,\mathit{pumpStart}.]\,\bot$	128/0	5.67	3.05
φ_9	*Invariantly, when the level of methane rises, it inevitably decreases* $[\,\mathtt{true}^*\,.\,\mathit{methaneRise}\,]\;\mu X.\,[\neg \mathit{methaneLower}]\,X \wedge \langle\,\mathtt{true}\,\rangle\,\top$	0/128	20.47	0.21
φ_{10}	*Products with feature Ct can switch on the pump* $\langle\,\mathtt{true}^*\,.\,\mathit{pumpStart}\mid \mathit{Ct}\,\rangle\,\top$	32/96	6.49	0.31
φ_{11}	*Products with feature Ct can always switch on the pump* $[\,\mathtt{true}^*\mid \mathit{Ct}\,]\,\langle\,\mathtt{true}^*\,.\,\mathit{pumpStart}\mid \mathit{Ct}\,\rangle\,\top$	28/100	21.11	2.32
φ_{12}	*Products with features $\{Ct, Ma, Lh\}$ can start the pump upon a high water level, but products without feature Lh cannot* $[\mathtt{true}^*\mid\top]\,(([\mathit{highLevel}\mid \mathit{Ct} \wedge \mathit{Ma} \wedge \mathit{Lh}]\langle\mathtt{true}^*.\mathit{pumpStart}\mid\top\rangle\top) \wedge [\mathit{pumpStart}\mid\neg \mathit{Lh}]\,\bot)$	128/0	13.35	3.36

mCRL2 code distributed with mCRL2 toolset (svn revision 14493)

Conclusions and future work

Introduced and compared feature-oriented μ-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but μL_f is more expressive

Translation to μ-calculus with data allows family-based model checking multi-feature properties of SPL models with off-the-shelf tools (e.g. mCRL2)

Defined a first (naive) family-based partitioning procedure for μL_f; its efficiency depends on initial partitioning of P and quality of refinements

Future work: improve partitioning strategy

1. Determine heuristics for finding a good initial partitioning of P
2. Extract information from failed model-checking problems to find a good split-up of the family of products in line 5 of Algorithm 1

? (difficult in particular for μ-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)

Conclusions and future work

Introduced and compared feature-oriented μ-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but μL_f is more expressive

Translation to μ-calculus with data allows family-based model checking multi-feature properties of SPL models with off-the-shelf tools (e.g. mCRL2)

Defined a first (naive) family-based partitioning procedure for μL_f; its efficiency depends on initial partitioning of P and quality of refinements

Future work: improve partitioning strategy

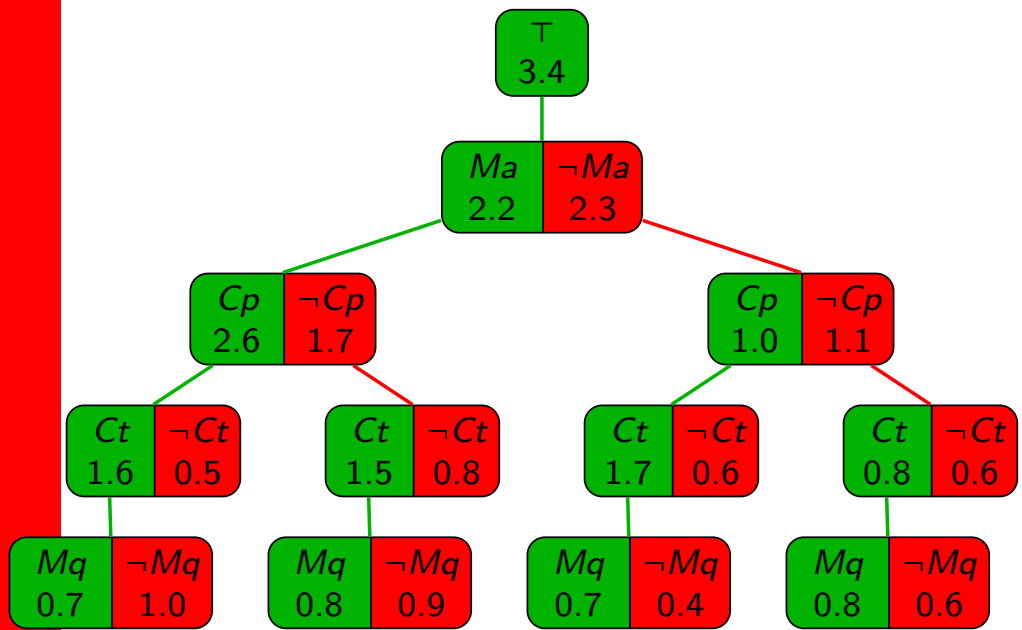
1. Determine heuristics for finding a good initial partitioning of P
2. Extract information from failed model-checking problems to find a good split-up of the family of products in line 5 of Algorithm 1

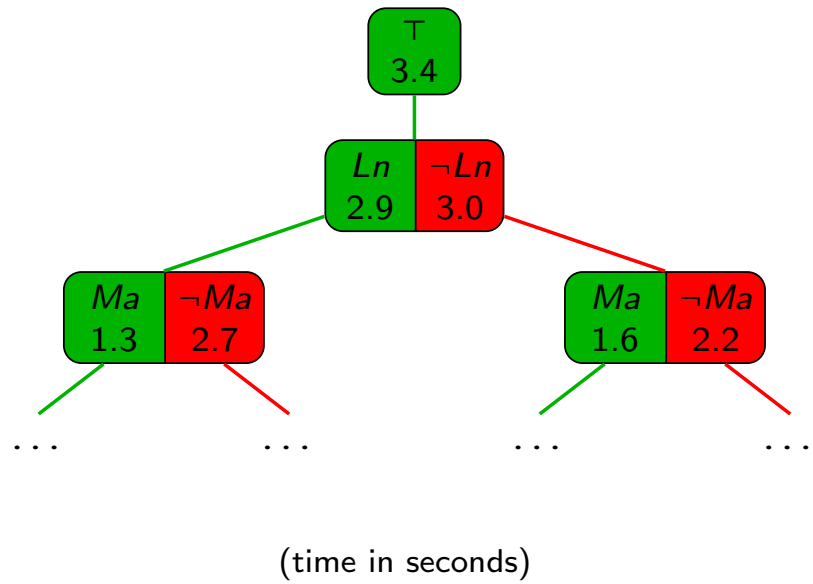
? (difficult in particular for μ-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)

The quest for an efficient strategy

quanticol
www.quanticol.eu

Execution of Algorithm 1 for deadlock freedom (φ_1) and with initial family $\top$
(family characterised at node is conjunction of features along path from root)





Optimal partitioning strategy

Total computation time: 27.9
Computation time leaves: 8.4
(i.e. Mq, $\neg Mq$, and $\neg Ct$ nodes)
At once $\forall$ possible families: 2.07

Non-optimal partitioning strategy
(splitting Ln and $\neg Ln$, then optimal)
Total computation time: 45.0 (+60%)