Family-based model checking with a feature μ-calculus

Maurice ter Beek, Erik de Vink, and Tim Willemse

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

QUANTICOL Plenary Meeting
ISTI-CNR, Pisa, Italy
February 7th, 2017

Outline

1. Context: verification of behavioural SPL models
   - SPL: software product lines
   - Family-based modelling and analysis
   - FTS: featured transition systems
2. Towards family-based model checking with mCRL2
   - mCRL2: language and toolset
   - The μ-calculus μL over LTSs
   - A feature μ-calculus μL_f over FTSs
3. Main results of paper @ FMSPLE'16
   - From μL_f to μL
4. Main results of paper @ FASE'17
   - A μ-calculus with data μL_{FO} over parametrised LTSs
   - From μL_f to μL_{FO} (and back to μL)
   - Family-based partitioning algorithm for μL_f
   - Case study: minepump SPL benchmark
5. Conclusions and future work
   - The quest for an efficient partitioning strategy

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

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

⇒ 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

FTS: featured transition systems

Classen et al., Model Checking Lots of Systems @ ICSE'10

quanticol www.quanticol.eu

FTS $F = (S, \theta, s_*)$ over actions $\mathcal{A}$ and features $\mathcal{F}$ (typical element f)

- S a finite set of states
- $\theta : S \times \mathcal{A} \times S \to \mathbb{B}[\mathcal{F}]$ the transition constraint function
- $s_* \in S$ the initial state

LTS $L = (S, \to, s_*)$ over actions $\mathcal{A}$

- S a finite set of states
- $\to \subseteq S \times \mathcal{A} \times S$ the transition relation
- $s_* \in S$ the initial state

FTS: featured transition systems

Classen et al., Model Checking Lots of Systems @ ICSE'10

FTS $F = (S, \theta, s_*)$ over actions $\mathcal{A}$ and features $\mathcal{F}$ (typical element f)

- S a finite set of states
- $\theta : S \times \mathcal{A} \times S \to \mathbb{B}[\mathcal{F}]$ the transition constraint function
- $s_* \in S$ the initial state

LTS $F|p = (S, \to_{F|p}, s_*)$ projection of F with respect to product p

$\to_{F|p} \subseteq S \times \mathcal{A} \times S$ such that $s \xrightarrow{a}_{F|p} t$ iff $p \models \theta(s, a, t)$

$\mathcal{P} \subseteq \mathbf{2}^{\mathcal{F}}$ set of products p, q, ...

$P \subseteq \mathcal{P}$ product family, identified by feature expression $\gamma_P \in \mathbb{B}[\mathcal{F}]$

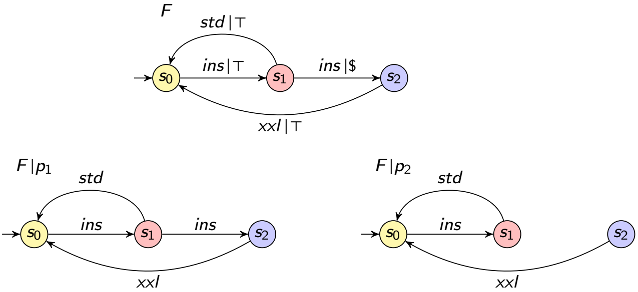
$\gamma \in \mathbb{B}[\mathcal{F}]$ interpreted as set of products Q_γ,

i.e. products p for which the induced truth assignment

(**true** for $\mathsf{f} \in p$, **false** for $\mathsf{f} \notin p$) validates γ

FTS of example SPL

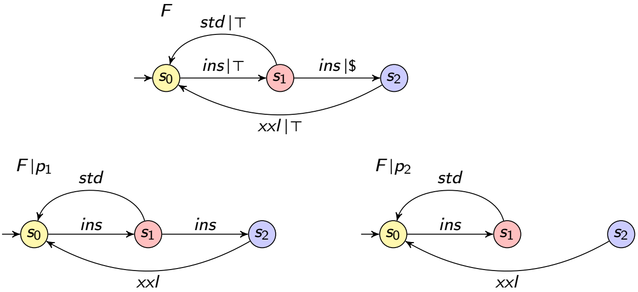
Product line of (four) coffee machines with independent features $\{\$, €\}$



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?

D3.2 We showed how to use mCRL2 for product-based model checking of SPL models
ter Beek & de Vink @ FormaliSE'14, SPLC'14

D3.2 We proposed a feature-oriented modular verification technique for SPL models with mCRL2
ter Beek & de Vink @ FMSPLE'14, ISoLA'14

D3.2 We extended branching bisimulation for LTSs to branching feature bisimulation for FTSs
Belder, ter Beek & de Vink @ FMSPLE'15

D3.3 We defined feature-oriented modal μ-calculi to reason on FTSs by explicitly incorporating feature expressions in the modal operators
ter Beek, de Vink & Willemse @ FMSPLE'16

D3.3 We showed how to use off-the-shelf tools like mCRL2 for family-based model checking of SPL models
ter Beek, de Vink & Willemse @ FASE'17

- D3.2 We showed how to use mCRL2 for product-based model checking of SPL models — ter Beek & de Vink @ FormaliSE'14, SPLC'14
- D3.2 We proposed a feature-oriented modular verification technique for SPL models with mCRL2 — ter Beek & de Vink @ FMSPLE'14, ISoLA'14
- D3.2 We extended branching bisimulation for LTSs to branching feature bisimulation for FTSs — Belder, ter Beek & de Vink @ FMSPLE'15
- D3.3 We defined feature-oriented modal μ-calculi to reason on FTSs by explicitly incorporating feature expressions in the modal operators — ter Beek, de Vink & Willemse @ FMSPLE'16
- D3.3 We showed how to use off-the-shelf tools like mCRL2 for family-based model checking of SPL models — ter Beek, de Vink & Willemse @ FASE'17

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

`www.mcrl2.org`

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

`www.mcrl2.org`

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,

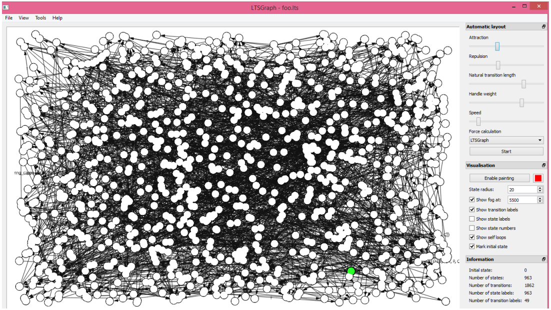
  behavioural reduction,

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

www.mcrl2.org

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



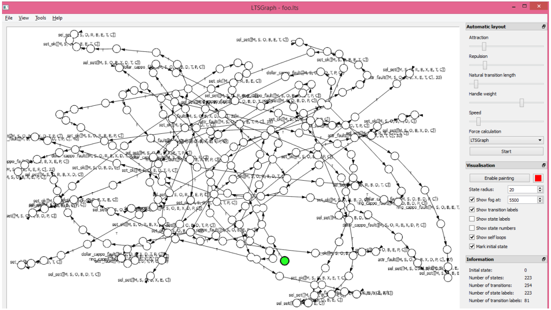
www.mcrl2.org

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



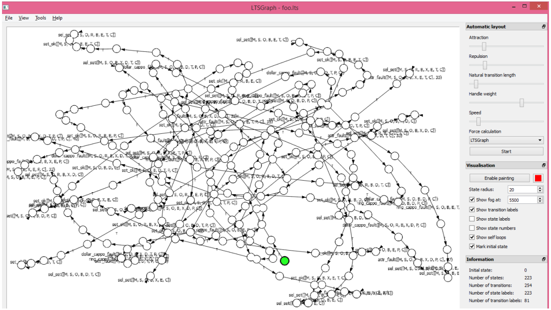
www.mcrl2.org

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



www.mcrl2.org

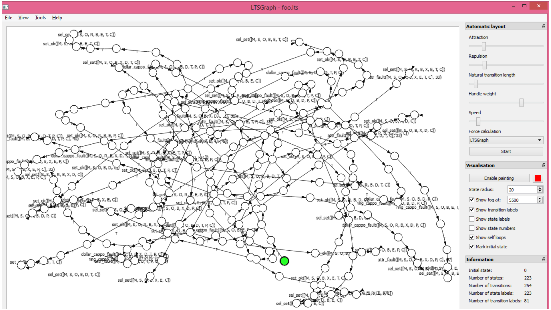
mCRL2: language and toolset

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,

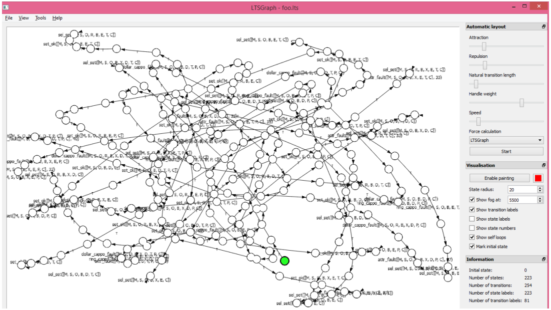
  model checking
- Highly optimised, actively maintained
- Intermediate artifacts user-accessible



www.mcrl2.org

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



www.mcrl2.org

The modal μ-calculus μL

quanticol www.quanticol.eu

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)

Examples of μL-formulae

- $\langle a \rangle ([b] \bot \wedge \langle c \rangle \top)$

  "it is possible to execute action a, after which action b cannot be executed whereas action c can"

- $\mu X . (\langle a \rangle X \vee \langle b \rangle \top)$

  "there exists a finite repetition of executing action a, followed by an execution of action b"

- $\nu X . (\mu Y . [a] Y \wedge [b] X)$

  "action b is executed infinitely often on all infinite executions containing actions a and b"

Examples of μL-formulae

- $\langle a \rangle ([b] \bot \wedge \langle c \rangle \top)$

  "it is possible to execute action a, after which action b cannot be executed whereas action c can"

- $\mu X . (\langle a \rangle X \vee \langle b \rangle \top)$

  "there exists a finite repetition of executing action a, followed by an execution of action b"

- $\nu X . (\mu Y . [a] Y \wedge [b] X)$

  "action b is executed infinitely often on all infinite executions containing actions a and b"

μX: finite looping

Examples of μL-formulae

- $\langle a \rangle ([b] \bot \wedge \langle c \rangle \top)$

  "it is possible to execute action a, after which action b cannot be executed whereas action c can"

- $\mu X . (\langle a \rangle X \vee \langle b \rangle \top)$

  "there exists a finite repetition of executing action a, followed by an execution of action b"

- $\nu X . (\mu Y . [a] Y \wedge [b] X)$

  "action b is executed infinitely often on all infinite executions containing actions a and b"

μX: finite looping vs. νX: infinite looping

Semantics of μL over LTSs

quanticol
www.quanticol.eu

state sets $U \in sSet = 2^S$, state-based environments $\varepsilon \in sEnv = \mathcal{X} \to sSet$
semantics $[\![\cdot]\!]_L : \mu L \to sEnv \to sSet$

$$
\begin{aligned}
[\![\bot]\!]_L(\varepsilon) &= \emptyset \\
[\![\top]\!]_L(\varepsilon) &= S \\
[\![\neg\varphi]\!]_L(\varepsilon) &= S \setminus [\![\varphi]\!]_L(\varepsilon) \\
[\![(\varphi \vee \psi)]\!]_L(\varepsilon) &= [\![\varphi]\!]_L(\varepsilon) \cup [\![\psi]\!]_L(\varepsilon) \\
[\![(\varphi \wedge \psi)]\!]_L(\varepsilon) &= [\![\varphi]\!]_L(\varepsilon) \cap [\![\psi]\!]_L(\varepsilon) \\
[\![\langle a \rangle\varphi]\!]_L(\varepsilon) &= \{\, s \mid \exists t \colon s \xrightarrow{a} t \wedge t \in [\![\varphi]\!]_L(\varepsilon) \,\} \\
[\![[a]\varphi]\!]_L(\varepsilon) &= \{\, s \mid \forall t \colon s \xrightarrow{a} t \Rightarrow t \in [\![\varphi]\!]_L(\varepsilon) \,\} \\
[\![X]\!]_L(\varepsilon) &= \varepsilon(X) \\
[\![\mu X.\varphi]\!]_L(\varepsilon) &= \mathit{lfp}(\, U \mapsto [\![\varphi]\!]_L(\varepsilon[U/X])\,) \\
[\![\nu X.\varphi]\!]_L(\varepsilon) &= \mathit{gfp}(\, U \mapsto [\![\varphi]\!]_L(\varepsilon[U/X])\,)
\end{aligned}
$$

variant environment $\varepsilon[U/X]$ yields $\varepsilon(Y)$ for $Y \neq X$, the set U for X

Semantics of μL over LTSs

state sets $U \in sSet = 2^S$, state-based environments $\varepsilon \in sEnv = \mathcal{X} \to sSet$
semantics $[\![\cdot]\!]_L : \mu L \to sEnv \to sSet$

$$[\![\bot]\!]_L(\varepsilon) = \emptyset$$

$$[\![\top]\!]_L(\varepsilon) = S$$

$$[\![\neg\varphi]\!]_L(\varepsilon) = S \setminus [\![\varphi]\!]_L(\varepsilon)$$

$$[\![(\varphi \vee \psi)]\!]_L(\varepsilon) = [\![\varphi]\!]_L(\varepsilon) \cup [\![\psi]\!]_L(\varepsilon)$$

$$[\![(\varphi \wedge \psi)]\!]_L(\varepsilon) = [\![\varphi]\!]_L(\varepsilon) \cap [\![\psi]\!]_L(\varepsilon)$$

$$[\![\langle a \rangle\varphi]\!]_L(\varepsilon) = \{\, s \mid \exists t \colon s \xrightarrow{a} t \wedge t \in [\![\varphi]\!]_L(\varepsilon) \,\}$$

$$[\![\,[a]\varphi]\!]_L(\varepsilon) = \{\, s \mid \forall t \colon s \xrightarrow{a} t \Rightarrow t \in [\![\varphi]\!]_L(\varepsilon) \,\}$$

$$[\![X]\!]_L(\varepsilon) = \varepsilon(X)$$

$$[\![\mu X.\varphi]\!]_L(\varepsilon) = \mathit{lfp}(\, U \mapsto [\![\varphi]\!]_L(\varepsilon[U/X])\,)$$

$$[\![\nu X.\varphi]\!]_L(\varepsilon) = \mathit{gfp}(\, U \mapsto [\![\varphi]\!]_L(\varepsilon[U/X])\,)$$

variant environment $\varepsilon[U/X]$ yields $\varepsilon(Y)$ for $Y \neq X$, the set U for X

A feature μ-calculus μL_f

quanticol www.quanticol.eu

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

A semantics of μL_f over FTSs

state-family pairs $(s, P) \in sPSet = 2^{S \times 2^{\mathcal{P}}}$

state-family environments $\zeta \in sPEnv = \mathcal{X} \to sPSet$

semantics $[\![\cdot]\!]_F : \mu L_f \to sPEnv \to sPSet$

$$
\begin{aligned}
[\![\bot]\!]_F(\zeta) &= \emptyset \\
[\![\top]\!]_F(\zeta) &= S \times 2^{\mathcal{P}} \\
[\![\neg\varphi_f]\!]_F(\zeta) &= (S \times 2^{\mathcal{P}}) \setminus [\![\varphi_f]\!]_F(\zeta) \\
[\![(\varphi_f \vee \psi_f)]\!]_F(\zeta) &= [\![\varphi_f]\!]_F(\zeta) \cup [\![\psi_f]\!]_F(\zeta) \\
[\![(\varphi_f \wedge \psi_f)]\!]_F(\zeta) &= [\![\varphi_f]\!]_F(\zeta) \cap [\![\psi_f]\!]_F(\zeta) \\
[\![\langle a|\chi\rangle\varphi_f]\!]_F(\zeta) &= \ldots \\
[\![[a|\chi]\varphi_f]\!]_F(\zeta) &= \ldots \\
[\![X]\!]_F(\zeta) &= \zeta(X) \\
[\![\mu X.\varphi_f]\!]_F(\zeta) &= lfp(\, W \mapsto [\![\varphi_f]\!]_F(\zeta[W/X])\,) \\
[\![\nu X.\varphi_f]\!]_F(\zeta) &= gfp(\, W \mapsto [\![\varphi_f]\!]_F(\zeta[W/X])\,)
\end{aligned}
$$

A semantics of μL_f over FTSs

$$[\![\langle a|\chi\rangle\varphi_f]\!]_F(\zeta) = \{\,(s,P) \mid P \subseteq Q_\chi \wedge \exists \gamma, t\colon s \xrightarrow{a|\gamma}_F t \wedge P \subseteq Q_\gamma \wedge (t, P \cap Q_\chi \cap Q_\gamma) \in [\![\varphi_f]\!]_F(\zeta)\,\}$$

$\langle a|\chi\rangle\varphi_f$ holds (for a family P with respect to an FTS F in a state s)
if all products in P satisfy the feature expression χ and there is an a-transition, shared among all products in P, that leads to a state where φ_f holds for P

$$[\![[a|\chi]\varphi_f]\!]_F(\zeta) = \{\,(s,P) \mid \forall \gamma, t\colon s \xrightarrow{a|\gamma}_F t \wedge P \cap Q_\chi \cap Q_\gamma \neq \emptyset \Rightarrow (t, P \cap Q_\chi \cap Q_\gamma) \in [\![\varphi_f]\!]_F(\zeta)\,\}$$

$[a|\chi]\varphi_f$ holds (for a family P with respect to an FTS F in a state s)
if for each subset P' of P for which an a-transition is possible, φ_f holds for P' in the target state of that a-transition

A semantics of μL_f over FTSs

quanticol
www.quanticol.eu

$$[\![\langle a|\chi\rangle \varphi_f]\!]_F(\zeta) = \{ (s,P) \mid P \subseteq Q_\chi \wedge \exists \gamma, t \colon s \xrightarrow{a|\gamma}_F t \wedge P \subseteq Q_\gamma \wedge (t, P \cap Q_\chi \cap Q_\gamma) \in [\![\varphi_f]\!]_F(\zeta) \}$$

$\langle a|\chi\rangle\varphi_f$ holds (for a family P with respect to an FTS F in a state s)
if all products in P satisfy the feature expression χ and
there is an a-transition, shared among all products in P,
that leads to a state where φ_f holds for P

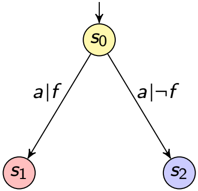
$$[\![[a|\chi] \varphi_f]\!]_F(\zeta) = \{ (s,P) \mid \forall \gamma, t \colon s \xrightarrow{a|\gamma}_F t \wedge P \cap Q_\chi \cap Q_\gamma \neq \emptyset \Rightarrow (t, P \cap Q_\chi \cap Q_\gamma) \in [\![\varphi_f]\!]_F(\zeta) \}$$

$[a|\chi]\varphi_f$ holds (for a family P with respect to an FTS F in a state s)
if for each subset P' of P for which an a-transition is possible,
φ_f holds for P' in the target state of that a-transition

Example μL_f formula: duality lost

quanticol
www.quanticol.eu

$s,P \models_F \varphi_f$ iff $(s,P) \in [\![\varphi_f]\!]_F$



Products $p_1 = \{f, g\}$ and $p_2 = \{g\}$

Clearly: $\{f, g\} \models_{F|p_1} \langle a \rangle \top$

$\{g\} \models_{F|p_2} \langle a \rangle \top$

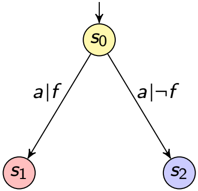
but... $\{p_1, p_2\} \not\models_F \langle a|\top \rangle \top$

Hence, since neither $\{p_1, p_2\} \models_F \langle a|\top \rangle \top$ nor $\{p_1, p_2\} \models_F [a|\top]\bot$,

$\langle a|\chi \rangle$ and $[a|\chi]$ are not each other's dual

Example μL_f formula: duality lost

$s,P \models_F \varphi_f$ iff $(s,P) \in [\![\varphi_f]\!]_F$



Products $p_1 = \{f, g\}$ and $p_2 = \{g\}$

Clearly: $\{f, g\} \models_{F|p_1} \langle a \rangle \top$

$\{g\} \models_{F|p_2} \langle a \rangle \top$

but... $\{p_1, p_2\} \not\models_F \langle a|\top \rangle \top$

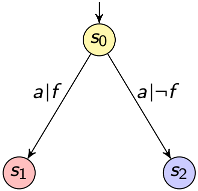
Hence, since neither $\{p_1, p_2\} \models_F \langle a|\top \rangle \top$ nor $\{p_1, p_2\} \models_F [a|\top]\bot$,

$\langle a|\chi \rangle$ and $[a|\chi]$ are not each other's dual

Example μL_f formula: duality lost

quanticol
www.quanticol.eu

$s, P \models_F \varphi_f$ iff $(s, P) \in [\![\varphi_f]\!]_F$



Products $p_1 = \{f, g\}$ and $p_2 = \{g\}$

Clearly: $\{f, g\} \models_{F|p_1} \langle a \rangle \top$

$\{g\} \models_{F|p_2} \langle a \rangle \top$

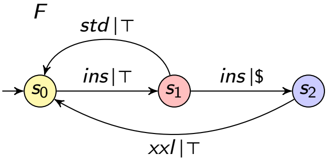
but... $\{p_1, p_2\} \not\models_F \langle a|\top \rangle \top$

Hence, since neither $\{p_1, p_2\} \models_F \langle a|\top \rangle \top$ nor $\{p_1, p_2\} \models_F [a|\top] \bot$, $\langle a|\chi \rangle$ and $[a|\chi]$ are not each other's dual

Examples of μL_f-formulae

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

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



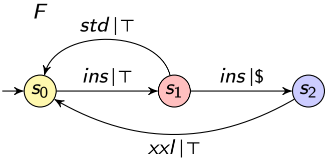
- $\nu X.\mu Y.(([ins|€]\,Y \wedge [xxl|€]\,Y) \wedge [std|€]X)$

  "for the (sub)family of products with feature €, 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"

Examples of μL_f-formulae



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

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

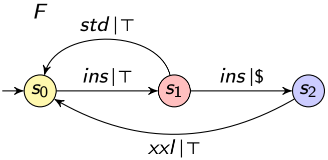
- $\nu X.\mu Y.(([ins|\epsilon]\,Y \wedge [xxl|\epsilon]\,Y) \wedge [std|\epsilon]X)$

  "for the (sub)family of products with feature ϵ, 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"

Examples of μL_f-formulae



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

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

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

  "for the (sub)family of products with feature €, 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

From μL_f to μL

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

projection function $pr : \mu L_f \times \mathcal{P} \to \mu L$

$$
\begin{aligned}
pr(\bot, p) &= \bot \\
pr(\top, p) &= \top \\
pr(\neg\varphi_f, p) &= \neg pr(\varphi_f, p) \\
pr(\varphi_f \vee \psi_f, p) &= pr(\varphi_f) \vee pr(\psi_f) \\
pr(\varphi_f \wedge \psi_f, p) &= pr(\varphi_f) \wedge pr(\psi_f) \\
pr(\langle a|\chi\rangle\varphi_f, p) &= \textbf{if } p \in Q_\chi \textbf{ then } \langle a\rangle pr(\varphi_f, p) \textbf{ else } \bot \textbf{ end} \\
pr([a|\chi]\varphi_f, p) &= \textbf{if } p \in Q_\chi \textbf{ then } [a] pr(\varphi_f, p) \textbf{ else } \top \textbf{ end} \\
pr(X, p) &= X \\
pr(\mu X.\varphi_f, p) &= \mu X.pr(\varphi_f, p) \\
pr(\nu X.\varphi_f, p) &= \nu X.pr(\varphi_f, p)
\end{aligned}
$$

From μL_f to μL

quanticol
www.quanticol.eu

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

projection function $pr : \mu L_f \times \mathcal{P} \rightarrow \mu L$

$$
\begin{aligned}
pr(\bot, p) &= \bot \\
pr(\top, p) &= \top \\
pr(\neg\varphi_f, p) &= \neg pr(\varphi_f, p) \\
pr(\varphi_f \vee \psi_f, p) &= pr(\varphi_f) \vee pr(\psi_f) \\
pr(\varphi_f \wedge \psi_f, p) &= pr(\varphi_f) \wedge pr(\psi_f) \\
pr(\langle a|\chi\rangle\varphi_f, p) &= \textbf{if } p \in Q_\chi \textbf{ then } \langle a\rangle pr(\varphi_f, p) \textbf{ else } \bot \textbf{ end} \\
pr([a|\chi]\varphi_f, p) &= \textbf{if } p \in Q_\chi \textbf{ then } [a]\, pr(\varphi_f, p) \textbf{ else } \top \textbf{ end} \\
pr(X, p) &= X \\
pr(\mu X.\varphi_f, p) &= \mu X.pr(\varphi_f, p) \\
pr(\nu X.\varphi_f, p) &= \nu X.pr(\varphi_f, p)
\end{aligned}
$$

Main results of paper @ FMSPLE'16

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

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

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

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

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

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

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

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

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

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

quanticol

www.quanticol.eu

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

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

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

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

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

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

---

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

Semantics μL_{FO} over parametrised LTSs

FTS $F = (S, \theta, s_*)$ over actions $\mathcal{A}$ and features $\mathcal{F}$

- S a finite set of states
- $\theta : S \times \mathcal{A} \times S \to \mathbb{B}[\mathcal{F}]$ the transition constraint function
- $s_* \in S$ the initial state

LTS $L = (S, \to, s_*)$ over actions $\mathcal{A}$

- S a finite set of states
- $\to \subseteq S \times \mathcal{A} \times S$ the transition relation
- $s_* \in S$ the initial state

Semantics μL_{FO} over parametrised LTSs

FTS $F = (S, \theta, s_*)$ over actions $\mathcal{A}$ and features $\mathcal{F}$

- S a finite set of states
- $\theta : S \times \mathcal{A} \times S \to \mathbb{B}[\mathcal{F}]$ the transition constraint function
- $s_* \in S$ the initial state

parametrised LTS $L(F) = (S, \to, s_*)$ for F over actions
$\mathcal{A}[\mathcal{F}] = \{ a(\gamma) \mid a \in \mathcal{A},\ \gamma \in \mathbb{B}[\mathcal{F}] \}$

- $\to$ is defined by $s \xrightarrow{a(\gamma)} t$ iff $\theta(s, a, t) = \gamma$ and $\gamma \not\equiv \bot$

μL_{FO} is a fragment of the logic from:

Groote & Mateescu, Verification of temporal properties of processes in a setting with data @ AMAST'99

Groote & Willemse, Model-checking processes with data @ *Sci. Comput. Program.* (2005)

where its full semantics can be found

Semantics μL_{FO} over parametrised LTSs

FTS $F = (S, \theta, s_*)$ over actions $\mathcal{A}$ and features $\mathcal{F}$

- S a finite set of states
- $\theta : S \times \mathcal{A} \times S \to \mathbb{B}[\mathcal{F}]$ the transition constraint function
- $s_* \in S$ the initial state

parametrised LTS $L(F) = (S, \to, s_*)$ for F over actions $\mathcal{A}[\mathcal{F}] = \{\, a(\gamma) \mid a \in \mathcal{A},\ \gamma \in \mathbb{B}[\mathcal{F}]\,\}$

- $\to$ is defined by $s \xrightarrow{a(\gamma)} t$ iff $\theta(s, a, t) = \gamma$ and $\gamma \not\equiv \bot$

e.g.

$$[\![\langle a(\upsilon)\rangle\varphi]\!]_{FO}(\xi)(\theta) = \{\, s \mid \exists\gamma,t\colon s \xrightarrow{a(\gamma)} t \wedge \theta(\upsilon) = Q_\gamma \wedge t \in [\![\varphi]\!]_{FO}(\xi)(\theta)\,\}$$

i.e.

$\langle a(\upsilon)\rangle\varphi$ holds if there is a transition $s \xrightarrow{a(\gamma)} t$ such that family $\theta(\upsilon)$ equals family Q_γ (associated with transition's feature expression γ) and t satisfies φ

From μL_f to μL_{FO}

quanticol
www.quanticol.eu

translation function $tr : \mathbb{B}[\mathcal{F}] \times \mu L_f \to \mu L_{\mathrm{FO}}$

$$
\begin{aligned}
tr(\gamma, \bot) &= \bot \\
tr(\gamma, \top) &= \top \\
tr(\gamma, \neg\varphi_f) &= \neg tr(\gamma, \varphi_f) \\
tr(\gamma, \varphi_f \vee \psi_f) &= tr(\gamma, \varphi_f) \vee tr(\gamma, \psi_f) \\
tr(\gamma, \varphi_f \wedge \psi_f) &= tr(\gamma, \varphi_f) \wedge tr(\gamma, \psi_f) \\
tr(\gamma, \langle a|\chi\rangle\varphi_f) &= (\gamma \Rightarrow \chi) \wedge \exists \upsilon.\langle a(\upsilon)\rangle((\gamma \Rightarrow \upsilon) \wedge tr(\gamma\wedge\chi\wedge\upsilon, \varphi_f)) \\
tr(\gamma, [a|\chi]\varphi_f) &= \forall \upsilon.[a(\upsilon)]((\gamma\wedge\chi\wedge\upsilon \Rightarrow \bot) \vee tr(\gamma\wedge\chi\wedge\upsilon, \varphi_f)) \\
tr(\gamma, X) &= \tilde{X}(\gamma) \\
tr(\gamma, \mu X.\varphi_f) &= \mu\tilde{X}(\upsilon{:=}\gamma).tr(\upsilon, \varphi_f) \\
tr(\gamma, \nu X.\varphi_f) &= \nu\tilde{X}(\upsilon{:=}\gamma).tr(\upsilon, \varphi_f)
\end{aligned}
$$

A main result of paper @ FASE'17

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

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

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

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

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

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

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

A main result of paper @ FASE'17

quanticol
www.quanticol.eu

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

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

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

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

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

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

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

Recall a result of paper @ FMSPLE'16

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

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

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

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

Another result of paper @ FASE'17

quanticol www.quanticol.eu

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

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

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

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

```
function FBP(P, φ_f)
    if s_*, P ⊨_F φ_f then return (P, ∅)
    else
        if s_*, P ⊨_F φ_f^c then return (∅, P)
        else partition P into (P_1, P_2)
            (P_1^+, P_1^-) ← FBP(P_1, φ_f)
            (P_2^+, P_2^-) ← FBP(P_2, φ_f)
            return (P_1^+ ∪ P_2^+, P_1^- ∪ P_2^-)
        end if
    end if
end function
```

Family-based partitioning algorithm

Theorem 4 $\text{FBP}(P, \varphi_f)$ 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

function $\text{FBP}(P, \varphi_f)$
  if $s_*, P \models_F \varphi_f$ **then return** $(P, \emptyset)$
  else
    if $s_*, P \models_F \varphi_f^c$ **then return** $(\emptyset, P)$
    else partition P into (P_1, P_2)
      $(P_1^+, P_1^-) \leftarrow \text{FBP}(P_1, \varphi_f)$
      $(P_2^+, P_2^-) \leftarrow \text{FBP}(P_2, \varphi_f)$
      return $(P_1^+ \cup P_2^+, P_1^- \cup P_2^-)$
    end if
  end if
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)

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* $\mu X\,.\,[\,(\,\neg levelMsg\,)^*.\,levelMsg\,]\,X$	0/128	10.18	0.16
φ_3	*The controller cannot fairly receive each of the three message types* $\mu X\,.\,(\,[\,\mathtt{true}^*.\,commandMsg\,]\,X \vee [\,\mathtt{true}^*.\,alarmMsg\,]\,X \vee [\,\mathtt{true}^*.\,levelMsg\,]\,X\,)$	0/128	24.33	0.25
φ_4	*The pump cannot be switched on infinitely often* $(\,\mu X\,.\,\nu Y\,.\,(\,[\,pumpStart\,.\,(\,\neg pumpStop\,)^*.\,pumpStop\,]\,X \wedge [\,\neg pumpStart\,]\,Y\,)\,)$ $\wedge\,(\,[\,\mathtt{true}^*.\,pumpStart\,]\,\mu Z\,.\,[\,\neg pumpStop\,]\,Z\,)$	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* $[\,\mathtt{true}^*\,]\;(\,(\,[pumpStart\,.\,(\,\neg pumpStop\,)^*.\,methaneRise]\;\mu X.\,[R]X\,)$ $\wedge\,(\,[methaneRise\,.\,(\,\neg methaneLower\,)^*.\,pumpStart]\;\mu X.\,[R]X\,)\,)$ for $R = \neg(\,pumpStop + methaneLower\,)$	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)* $[\,\mathtt{true}^*\,]\;(\,(\,[\,pumpStart\,.\,(\,\neg pumpStop\,)^*.\,methaneRise\,]\,\Psi\,)$ $\wedge\,(\,[\,methaneRise\,.\,(\,\neg methaneLower\,)^*.\,pumpStart\,]\,\Psi\,)\,)$ for $\Psi = \mu X\,.\,(\,[\,R^*.\,commandMsg\,]\,X \vee [\,R^*.\,alarmMsg\,]\,X \vee [\,R^*.\,levelMsg\,]\,X\,)$ 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* $[\,\mathtt{true}^*\,]\;\langle\,\mathtt{true}^*.\,receiveMsg\,\rangle\,\top$	128/0	18.36	2.40
φ_8	*Invariantly the pump is not started when the low water level signal fires* $[\,\mathtt{true}^*\,.\,lowLevel\,.\,(\neg(normalLevel + highLevel))^*\,.\,pumpStart.]\,\bot$	128/0	5.67	3.05
φ_9	*Invariantly, when the level of methane rises, it inevitably decreases* $[\,\mathtt{true}^*\,.\,methaneRise\,]\;\mu X.\,[\,\neg methaneLower\,]\,X \wedge \langle\,\mathtt{true}\,\rangle\,\top$	0/128	20.47	0.21
φ_{10}	*Products with feature Ct can switch on the pump* $\langle\,\mathtt{true}^*.\,pumpStart\,\vert\,Ct\,\rangle\,\top$	32/96	6.49	0.31
φ_{11}	*Products with feature Ct can always switch on the pump* $[\,\mathtt{true}^*\,\vert\,Ct\,]\,\langle\,\mathtt{true}^*.\,pumpStart\,\vert\,Ct\,\rangle\,\top$	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* $[\mathtt{true}^*\vert\top]\,((\,[highLevel\,\vert\,Ct \wedge Ma \wedge Lh]\langle\mathtt{true}^*.\,pumpStart\,\vert\top\rangle\top\,) \wedge [\,pumpStart\,\vert\neg Lh\,]\,\bot\,)$	128/0	13.35	3.36

mCRL2 code distributed with mCRL2 toolset (svn revision 14493)

Conclusions and future work

quanticol
www.quanticol.eu

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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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

Conclusions and future work

quanticol
www.quanticol.eu

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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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

quanticol
www.quanticol.eu

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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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 μL_{FO} allows family-based model checking multi-feature properties of *configurable systems* 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

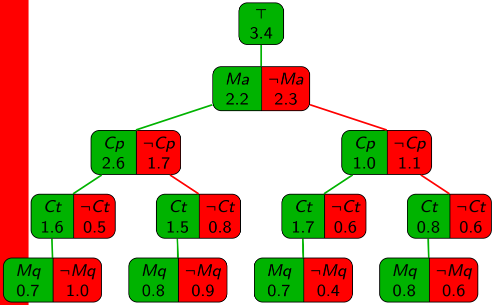
Future work: improve partitioning strategy

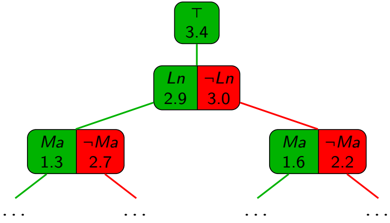
1. Determine heuristics for finding a good initial partitioning of P
2. Extract information from failed model-checking problems to find a good split-up of the family of products in line 5 of Algorithm 1

? (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

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)





(time in seconds)

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%)