Family-based model checking with a feature $\mu$-calculus

Maurice ter Beek, Erik de Vink, and Tim Willemse

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

VARIETE Closing Workshop
ITU, Copenhagen
March 28th, 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 $\mu$-calculus $\mu L$ over LTSs
   - A feature $\mu$-calculus $\mu L_f$ over FTSs

3. Main results of paper @ FMSPLE’16
   - From $\mu L_f$ to $\mu L$

4. Main results of paper @ FASE’17
   - A $\mu$-calculus with data $\mu L_{FO}$ over parametrised LTSs
   - From $\mu L_f$ to $\mu L_{FO}$ (and back to $\mu L$)
   - Family-based partitioning algorithm for $\mu L_f$
   - 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

⇒ 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

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

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

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

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

- $S$ a finite set of states
- $\rightarrow \subseteq S \times \mathcal{A} \times S$ the transition relation
- $s_* \in S$ the initial state
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 $F|p = (S, \rightarrow_{F|p}, s_*)$ projection of $F$ with respect to product $p$

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

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

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

$\gamma \in \mathbb{B}[\mathcal{F}]$ interpreted as set of products $Q_\gamma$, i.e. products $p$ for which the induced truth assignment ($true$ for $f \in p$, $false$ for $f \notin p$) validates $\gamma$
FTS of example SPL

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

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?
FTS of example SPL

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

![State transition diagram]

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?
Towards family-based model checking

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

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

- We extended branching bisimulation for LTSs to branching feature bisimulation for FTSs
  Belder, ter Beek & de Vink @ FMSPLE’15

- 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

- 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
Towards family-based model checking

- We showed how to use mCRL2 for product-based model checking of SPL models
ter Beek & de Vink @ FormalSE’14, SPLC’14

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

- We extended branching bisimulation for LTSs to branching feature bisimulation for FTSs
Belder, ter Beek & de Vink @ FMSPLE’15

- We defined feature-oriented modal $\mu$-calculi to reason on FTSs by explicitly incorporating feature expressions in the modal operators
ter Beek, de Vink & Willemse @ FMSPLE’16

- 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 $\mu$-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 $\mu$-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 $\mu$-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 $\mu$-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 $\mu$-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 $\mu$-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 $\mu$-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 $\mu$-calculus with data (subsuming LTL, CTL, etc.)
- Visualisation, behavioural reduction, model checking
- Highly optimised, actively maintained
- Intermediate artefacts user-accessible

www.mcrl2.org
The modal $\mu$-calculus $\mu L$

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

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

$$
\varphi :: = \bot \mid T \mid \\
\quad \neg \varphi \mid \varphi \lor \psi \mid \varphi \land \psi \\
\quad \langle a \rangle \varphi \mid [a] \varphi \\
\quad X \mid \mu X. \varphi \mid \nu X. \varphi
$$

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 $\varphi$ are in the scope of an even number of negations (guarantees well-definedness fixpoint formulae)
Examples of $\mu L$-formulae

- $\langle a \rangle (\, [b] \perp \land \langle c \rangle \top \,)$
  - “it is possible to execute action $a$, after which action $b$ cannot be executed whereas action $c$ can”

- $\mu X. (\langle a \rangle X \lor \langle b \rangle \top)$
  - “there exists a finite repetition of executing action $a$, followed by an execution of action $b$”

- $\nu X. (\mu Y. [a] Y \land [b] X)$
  - “action $b$ is executed infinitely often on all infinite executions containing actions $a$ and $b$”
Examples of $\mu L$-formulae

- $\langle a \rangle ( [b] \bot \land \langle c \rangle T )$
  
  “it is possible to execute action $a$, after which action $b$ cannot be executed whereas action $c$ can”

- $\mu X . ( \langle a \rangle X \lor \langle b \rangle T )$
  
  “there exists a finite repetition of executing action $a$, followed by an execution of action $b$”

- $\nu X . ( \mu Y . [a] Y \land [b] X )$
  
  “action $b$ is executed infinitely often on all infinite executions containing actions $a$ and $b$”

$\mu X$: finite looping
Examples of $\mu L$-formulae

- $\langle a \rangle (\lbrack b \rbrack \bot \land \langle c \rangle \top)$
  
  “it is possible to execute action $a$, after which action $b$ cannot be executed whereas action $c$ can”

- $\mu X. (\langle a \rangle X \lor \langle b \rangle \top)$
  
  “there exists a finite repetition of executing action $a$, followed by an execution of action $b$”

- $\nu X. (\mu Y. [a] Y \land [b] X)$
  
  “action $b$ is executed infinitely often on all infinite executions containing actions $a$ and $b$”

$\mu X$: finite looping vs. $\nu X$: infinite looping
Semantics of $\mu L$ over LTSs

State sets $U \in sSet = 2^S$, state-based environments $\varepsilon \in sEnv = X \rightarrow sSet$

Semantics $[\cdot]_L : \mu L \rightarrow sEnv \rightarrow sSet$

- $[[\bot]]_L(\varepsilon) = \emptyset$
- $[[\top]]_L(\varepsilon) = S$
- $[[\neg \varphi]]_L(\varepsilon) = S \setminus [[\varphi]]_L(\varepsilon)$
- $[[\varphi \lor \psi]]_L(\varepsilon) = [[\varphi]]_L(\varepsilon) \cup [[\psi]]_L(\varepsilon)$
- $[[\varphi \land \psi]]_L(\varepsilon) = [[\varphi]]_L(\varepsilon) \cap [[\psi]]_L(\varepsilon)$

- $[[\langle a \rangle \varphi]]_L(\varepsilon) = \{ s \mid \exists t : s \xrightarrow{a} t \land t \in [[\varphi]]_L(\varepsilon) \}$
- $[[[a] \varphi]]_L(\varepsilon) = \{ s \mid \forall t : s \xrightarrow{a} t \Rightarrow t \in [[\varphi]]_L(\varepsilon) \}$
- $[[X]]_L(\varepsilon) = \varepsilon(X)$
- $[[\mu X. \varphi]]_L(\varepsilon) = \text{lfp}( U \mapsto [[\varphi]]_L(\varepsilon[U/X]))$
- $[[\nu X. \varphi]]_L(\varepsilon) = \text{gfp}( U \mapsto [[\varphi]]_L(\varepsilon[U/X]))$

Variant environment $\varepsilon[U/X]$ yields $\varepsilon(Y)$ for $Y \neq X$, the set $U$ for $X$
Semantics of $\mu L$ over LTSs

state sets $U \in sSet = 2^S$, state-based environments $\varepsilon \in sEnv = X \rightarrow sSet$

semantics $[\cdot]_L : \mu L \rightarrow sEnv \rightarrow sSet$

$$
\begin{align*}
\llbracket \bot \rrbracket_L(\varepsilon) &= \emptyset \\
\llbracket \top \rrbracket_L(\varepsilon) &= S \\
\llbracket \neg \varphi \rrbracket_L(\varepsilon) &= S \setminus \llbracket \varphi \rrbracket_L(\varepsilon) \\
\llbracket (\varphi \lor \psi) \rrbracket_L(\varepsilon) &= \llbracket \varphi \rrbracket_L(\varepsilon) \cup \llbracket \psi \rrbracket_L(\varepsilon) \\
\llbracket (\varphi \land \psi) \rrbracket_L(\varepsilon) &= \llbracket \varphi \rrbracket_L(\varepsilon) \cap \llbracket \psi \rrbracket_L(\varepsilon) \\
\llbracket \langle a \rangle \varphi \rrbracket_L(\varepsilon) &= \{ s : \exists t : s \xrightarrow{a} t \land t \in \llbracket \varphi \rrbracket_L(\varepsilon) \} \\
\llbracket [a] \varphi \rrbracket_L(\varepsilon) &= \{ s : \forall t : s \xrightarrow{a} t \Rightarrow t \in \llbracket \varphi \rrbracket_L(\varepsilon) \} \\
\llbracket X \rrbracket_L(\varepsilon) &= \varepsilon(X) \\
\llbracket \mu X. \varphi \rrbracket_L(\varepsilon) &= \text{lfp}(U \mapsto \llbracket \varphi \rrbracket_L(\varepsilon[U/X])) \\
\llbracket \nu X. \varphi \rrbracket_L(\varepsilon) &= \text{gfp}(U \mapsto \llbracket \varphi \rrbracket_L(\varepsilon[U/X]))
\end{align*}
$$

variant environment $\varepsilon[U/X]$ yields $\varepsilon(Y)$ for $Y \neq X$, the set $U$ for $X$
A feature $\mu$-calculus $\mu L_f$

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

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

$$
\varphi_f ::= \bot \mid \top \mid \\
\neg \varphi_f \mid \varphi_f \lor \psi_f \mid \varphi_f \land \psi_f \mid \\
\langle a \mid \chi \rangle \varphi_f \mid [a \mid \chi] \varphi_f \mid \\
X \mid \mu X. \varphi_f \mid \nu X. \varphi_f
$$

for $\mu X. \varphi_f$ and $\nu X. \varphi_f$ an even number of negations as before
A semantics of $\mu L_f$ over FTSs

state-family pairs $(s, P) \in sPSet = 2^{S \times 2^P}$
state-family environments $\zeta \in sPEnv = X \to sPSet$
semantics $[\cdot]_F : \mu L_f \to sPEnv \to sPSet$

\[
\begin{align*}
[\bot]_F(\zeta) &= \emptyset \\
[\top]_F(\zeta) &= S \times 2^P \\
[\neg \varphi_f]_F(\zeta) &= (S \times 2^P) \setminus [\varphi_f]_F(\zeta) \\
[(\varphi_f \lor \psi_f)]_F(\zeta) &= [\varphi_f]_F(\zeta) \cup [\psi_f]_F(\zeta) \\
[(\varphi_f \land \psi_f)]_F(\zeta) &= [\varphi_f]_F(\zeta) \cap [\psi_f]_F(\zeta) \\
\langle a \mid \chi \rangle \varphi_f}_F(\zeta) &= \ldots \\
\langle[ a \mid \chi \rangle \varphi_f}_F(\zeta) &= \ldots \\
[X]_F(\zeta) &= \zeta(X) \\
[\mu X. \varphi_f]_F(\zeta) &= \text{lfp}(W \mapsto [\varphi_f]_F(\zeta[W/X])) \\
[\nu X. \varphi_f]_F(\zeta) &= \text{gfp}(W \mapsto [\varphi_f]_F(\zeta[W/X]))
\end{align*}
\]
A semantics of $\mu L_f$ over FTSs

$$\llbracket \langle a \mid \chi \rangle \varphi_f \rrbracket_F(\zeta) = \{ (s, P) \mid P \subseteq Q_\chi \land \exists \gamma, t : s \xrightarrow{a\mid \gamma} t \land P \subseteq Q_\gamma \land (t, P \cap Q_\chi \cap Q_\gamma) \in \llbracket \varphi_f \rrbracket_F(\zeta) \}$$

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

$$\llbracket [a \mid \chi] \varphi_f \rrbracket_F(\zeta) = \{ (s, P) \mid \forall \gamma, t : s \xrightarrow{a\mid \gamma} t \land P \cap Q_\chi \cap Q_\gamma \neq \emptyset \Rightarrow (t, P \cap Q_\chi \cap Q_\gamma) \in \llbracket \varphi_f \rrbracket_F(\zeta) \}$$

$[a \mid \chi] \varphi_f$ 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, $\varphi_f$ holds for $P'$ in the target state of that $a$-transition
A semantics of $\mu L_f$ over FTSs

\[
\llangle a \mid \chi \rrangle \varphi_f \big|_F(\zeta) = \{ (s, P) \mid P \subseteq Q_\chi \land \exists \gamma, t : s \xrightarrow{a \mid \gamma} F \; t \land P \subseteq Q_\gamma \land \left( t, P \cap Q_\chi \cap Q_\gamma \right) \in \llbracket \varphi_f \rrbracket_F(\zeta) \}
\]

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

\[
\llbracket [a \mid \chi] \varphi_f \rrbracket_F(\zeta) = \{ (s, P) \mid \forall \gamma, t : s \xrightarrow{a \mid \gamma} F \; t \land P \cap Q_\chi \cap Q_\gamma \neq \emptyset \Rightarrow \left( t, P \cap Q_\chi \cap Q_\gamma \right) \in \llbracket \varphi_f \rrbracket_F(\zeta) \}
\]

$[a \mid \chi] \varphi_f$ 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, $\varphi_f$ holds for $P'$ in the target state of that $a$-transition.
Example $\mu L_f$ formula: duality lost

$s, P \models F \varphi_f$ iff $(s, P) \in \llbracket \varphi_f \rrbracket_F$

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

Clearly: $\{f, g\} \models F_{|p_1} \langle a \rangle_T$

$\{g\} \models F_{|p_2} \langle a \rangle_T$

but... $\{p_1, p_2\} \not\models F \langle a|\top\rangle_T$

Hence, since neither $\{p_1, p_2\} \models F \langle a|\top\rangle_T$ nor $\{p_1, p_2\} \models F \langle a|\bot\rangle_T$, $\langle a|\chi\rangle$ and $[a|\chi]$ are not each other's dual
Example \( \mu L_f \) formula: duality lost

\[ s, P \models_F \varphi_f \text{ iff } (s, P) \in \llbracket \varphi_f \rrbracket_F \]

Products \( p_1 = \{f, g\} \) and \( p_2 = \{g\} \)

Clearly: \( \{f, g\} \models_F p_1 \langle a \rangle_T \) \( \{g\} \models_F p_2 \langle a \rangle_T \)

but... \( \{p_1, p_2\} \not\models_F \langle a | T \rangle_T \)

Hence, since neither \( \{p_1, p_2\} \models_F \langle a | T \rangle_T \) nor \( \{p_1, p_2\} \models_F [a | T] \perp \), \( \langle a | \chi \rangle \) and \( [a | \chi] \) are not each other's dual
Example $\mu L_f$ formula: duality lost

$s, P \models_F \varphi_f$ iff $(s, P) \in \vDash \varphi_f$ $F$

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

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

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

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

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

$\langle a|\chi\rangle$ and $[a|\chi]$ are not each other’s dual
Examples of $\mu L_f$-formulae

- $\langle \text{ins} | T \rangle ( [\text{ins} | \epsilon] \perp \land \langle \text{std} | T \rangle \top )$
  
  “the family of products $P$ that can execute $\text{ins}$, after which $\text{ins}$ cannot be executed by products satisfying $\epsilon$, while $\text{std}$ can be executed by all products of $P$”

- $\nu X. \mu Y. ([\text{ins} | \epsilon] Y \land [\text{xxl} | \epsilon] Y) \land [\text{std} | \epsilon] X$
  
  “for the (sub)family of products with feature $\epsilon$, action $\text{std}$ occurs infinitely often on all infinite runs over $\{ \text{ins}, \text{xxl}, \text{std} \}$”

- $[\text{true}^* | T ] ( [\text{ins} | $] (\text{true}^* \cdot \text{xxl} | T \top \top ) \land [\text{xxl} | \neg$ ] \perp )$
  
  “products with feature $\$ $ can obtain an $\text{xxl}$ coffee upon coin insertion, but products without cannot”
Examples of $\mu L_f$-formulae

- $\langle ins|\top \rangle (\langle ins|\epsilon \rangle \bot \land \langle std|\top \rangle \top )$
  
  “the family of products $P$ that can execute $ins$, after which $ins$ cannot be executed by products satisfying $\epsilon$, while $std$ can be executed by all products of $P$”

- $\nu X.\mu Y.((\langle ins|\epsilon \rangle Y \land \langle xxl|\epsilon \rangle Y) \land \langle std|\epsilon \rangle X)$

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

- $\langle true^*|\top \rangle ((\langle ins|\bot \rangle $ true$^*$.xxl|\top \rangle \top ) \land \langle xxl|\neg\bot \rangle \bot)$

  “products with feature $\bot$ can obtain an xxl coffee upon coin insertion, but products without cannot”
Examples of $\mu L_f$-formulae

1. $\langle \text{ins} \mid \top \rangle \left( \left[ \text{ins} \mid \epsilon \right] \bot \land \langle \text{std} \mid \top \rangle \top \right)$
   “the family of products $P$ that can execute $\text{ins}$, after which $\text{ins}$ cannot be executed by products satisfying $\epsilon$, while $\text{std}$ can be executed by all products of $P$”

2. $\nu X . \mu Y . (\left[ \text{ins} \mid \epsilon \right] Y \land \left[ \text{xxl} \mid \epsilon \right] Y) \land \left[ \text{std} \mid \epsilon \right] X$
   “for the (sub)family of products with feature $\epsilon$, action $\text{std}$ occurs infinitely often on all infinite runs over $\{\text{ins, xxl, std}\}$”

3. $[\text{true}^* \mid \top] \left( \left[ \text{ins} \mid \$ \right] \langle \text{true}^*. \text{xxl} \mid \top \rangle \top \right) \land \left[ \text{xxl} \mid \neg \$ \right] \bot$
   “products with feature $\$ \$ can obtain an $\text{xxl}$ coffee upon coin insertion, but products without cannot”

multi-feature $\mu L_f$ formula, novel also w.r.t. fLTL and fCTL
From $\mu L_f$ to $\mu L$

Model checking a $\mu L_f$-formula over an FTS for an individual product reduces to model checking a $\mu L$-formula over the corresponding LTS.

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

$$
pr(\bot, p) = \bot \\
pr(\top, p) = \top \\
pr(\neg \varphi_f, p) = \neg pr(\varphi_f, p) \\
pr(\varphi_f \lor \psi_f, p) = pr(\varphi_f) \lor pr(\psi_f) \\
pr(\varphi_f \land \psi_f, p) = pr(\varphi_f) \land pr(\psi_f) \\
pr(\langle a \mid \chi \rangle \varphi_f, p) = \text{if } p \in Q_x \text{ then } \langle a \rangle pr(\varphi_f, p) \text{ else } \bot \text{ end} \\
pr([a \mid \chi] \varphi_f, p) = \text{if } p \in Q_x \text{ then } [a] pr(\varphi_f, p) \text{ else } \top \text{ end} \\
pr(X, p) = X \\
pr(\mu X \cdot \varphi_f, p) = \mu X . pr(\varphi_f, p) \\
pr(\nu X \cdot \varphi_f, p) = \nu X . pr(\varphi_f, p)
$$
From $\mu L_f$ to $\mu L$

Model checking a $\mu L_f$-formula over an FTS for an individual product reduces to model checking a $\mu L$-formula over the corresponding LTS projection function $pr : \mu L_f \times \mathcal{P} \to \mu L$

$$
pr(\bot, p) = \bot
$$

$$
pr(\top, p) = \top
$$

$$
pr(\neg \varphi_f, p) = \neg pr(\varphi_f, p)
$$

$$
pr(\varphi_f \lor \psi_f, p) = pr(\varphi_f) \lor pr(\psi_f)
$$

$$
pr(\varphi_f \land \psi_f, p) = pr(\varphi_f) \land pr(\psi_f)
$$

$$
pr(\langle a | \chi \rangle \varphi_f, p) = \text{if } p \in Q_\chi \text{ then } \langle a \rangle pr(\varphi_f, p) \text{ else } \bot \text{ end}
$$

$$
pr([a | \chi] \varphi_f, p) = \text{if } p \in Q_\chi \text{ then } [a] pr(\varphi_f, p) \text{ else } \top \text{ end}
$$

$$
pr(X, p) = X
$$

$$
pr(\mu X . \varphi_f, p) = \mu X . pr(\varphi_f, p)
$$

$$
pr(\nu X . \varphi_f, p) = \nu X . pr(\varphi_f, p)
$$
Main results of paper @ FMSPLE’16

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

**Theorem 1**  
$s, \{p\} \models_F \varphi_f \iff s \models_{F|p} \text{pr}(\varphi_f, p)$  
closed $\varphi_f \in \mu L_f$, $s \in S$, product $p \in \mathcal{P}$

**Theorem 2**  
$s, P \models_F \varphi_f \implies \forall p \in P: s \models_{F|p} \text{pr}(\varphi_f, p)$  
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 $s \not\models_{F|p} \text{pr}(\varphi_f, p)$ 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**

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

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 $P$
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** \[ s, \mathcal{P} \models_F \varphi_f \implies \forall p \in \mathcal{P}: s \models_{F|p} pr(\varphi_f, p) \]

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

Note: in general $s, \mathcal{P} \not\models_F \varphi_f$ does not imply $s \not\models_{F|p} pr(\varphi_f, p)$ for all products in family $\mathcal{P}$
Given an FTS $F$ and a set of products $\mathcal{P}$

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

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

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

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

Note: in general $s, \mathcal{P} \not\models_F \varphi_f$ does not imply $s \not\models_{F|p} \text{pr}(\varphi_f, p)$ for all products in family $\mathcal{P}$
A first-order $\mu$-calculus with data $\mu L_{FO}$

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

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

\[
\varphi_f ::= \bot \mid T \mid \\
\neg \varphi \mid \varphi \lor \psi \mid \varphi \land \psi \mid \\
\gamma_1 \Rightarrow \gamma_2 \mid \\
\exists v. \varphi \mid \forall v. \varphi \mid \\
\langle a(v) \rangle \varphi \mid [a(v)] \varphi \mid \\
\bar{X}(\gamma) \mid \mu \bar{X}(v_{\bar{X}} := \gamma). \varphi \mid \nu \bar{X}(v_{\bar{X}} := \gamma). \varphi \\
\]

with the usual restrictions on free variables and the number of negations
Semantics $\mu L_{FO}$ over parametrised LTSs

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

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

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

- $S$ a finite set of states
- $\rightarrow \subseteq S \times \mathcal{A} \times S$ the transition relation
- $s_\star \in S$ the initial state
Semantics $\mu L_{FO}$ over parametrised LTSs

FTS $F = (S, \theta, s_\ast)$ over actions $A$ and features $F$

- $S$ a finite set of states
- $\theta : S \times A \times S \rightarrow B[F]$ the transition constraint function
- $s_\ast \in S$ the initial state

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

- $\rightarrow$ is defined by $s \xrightarrow{a(\gamma)} t$ iff $\theta(s, a, t) = \gamma$ and $\gamma \neq \bot$

$\mu L_{FO}$ is a fragment of the logic from:

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

where its full semantics can be found
FTS $F = (S, \theta, s_*)$ over actions $\mathcal{A}$ and features $\mathcal{F}$

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

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

- $\rightarrow$ is defined by $s \xrightarrow{a(\gamma)} t$ iff $\theta(s, a, t) = \gamma$ and $\gamma \neq \bot$

e.g.
$\llbracket \langle a(v) \rangle \varphi \rrbracket_{FO(\xi)(\theta)} = \{ s \mid \exists \gamma, t : s \xrightarrow{a(\gamma)} t \land \theta(v) = Q_\gamma \land t \in \llbracket \varphi \rrbracket_{FO(\xi)(\theta)} \}$
i.e.
$\langle a(v) \rangle \varphi$ holds if there is a transition $s \xrightarrow{a(\gamma)} t$ such that family $\theta(v)$ equals family $Q_\gamma$ (associated with transition’s feature expression $\gamma$) and $t$ satisfies $\varphi$
From $\mu L_f$ to $\mu L_{FO}$

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

$$tr(\gamma, \bot) = \bot$$
$$tr(\gamma, \top) = \top$$
$$tr(\gamma, \neg \varphi_f) = \neg tr(\gamma, \varphi_f)$$
$$tr(\gamma, \varphi_f \lor \psi_f) = tr(\gamma, \varphi_f) \lor tr(\gamma, \psi_f)$$
$$tr(\gamma, \varphi_f \land \psi_f) = tr(\gamma, \varphi_f) \land tr(\gamma, \psi_f)$$
$$tr(\gamma, \langle a|\chi \rangle \varphi_f) = (\gamma \Rightarrow \chi) \land \exists v. \langle a(v) \rangle((\gamma \Rightarrow v) \land tr(\gamma \land \chi \land v, \varphi_f))$$
$$tr(\gamma, [a|\chi] \varphi_f) = \forall v. [a(v)]((\gamma \land \chi \land v \Rightarrow \bot) \lor tr(\gamma \land \chi \land v, \varphi_f))$$

$$tr(\gamma, X) = \tilde{X}(\gamma)$$
$$tr(\gamma, \mu X. \varphi_f) = \mu \tilde{X}(v := \gamma). tr(v, \varphi_f)$$
$$tr(\gamma, \nu X. \varphi_f) = \nu \tilde{X}(v := \gamma). tr(v, \varphi_f)$$
A main result of paper @ FASE’17

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

**Theorem 3**

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

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

**Theorem 2**

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

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

**Lemma 1**

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

closed, negation-free, $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$
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** \[ s, P \models_F \varphi_f \Rightarrow \forall p \in P: s \models_{F|p} pr(\varphi_f, p) \]
closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

**Lemma 1** \[ s, P \models_F \varphi_f^c \Rightarrow \forall p \in P: s \not\models_{F|p} pr(\varphi_f, p) \]
closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$
Recall a result of paper @ FMSPLE’16

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** \[ s, P \models_F \varphi_f \Rightarrow \forall p \in P: s \models_{F|p} pr(\varphi_f, p) \]
closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$

**Lemma 1** \[ s, P \models_F \varphi_f^c \Rightarrow \forall p \in P: s \not\models_{F|p} pr(\varphi_f, p) \]
closed, negation-free $\varphi_f \in \mu L_f$, $s \in S$, family $P \subseteq \mathcal{P}$
Another result of paper @ FASE’17

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 $P \subseteq \mathcal{P}$
Family-based partitioning for $\mu L_f$

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

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

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

**Algorithm 1** Family-Based Partitioning

1: function $\text{FBP}(P, \varphi_f)$
2:     if $s_*, P \models_{F} \varphi_f$ then return $(P, \emptyset)$
3:     else
4:         if $s_*, P \models_{F} \varphi^c_f$ then return $(\emptyset, P)$
5:         else partition $P$ into $(P_1, P_2)$
6:             $(P_1^+, P_1^-) \leftarrow \text{FBP}(P_1, \varphi_f)$
7:             $(P_2^+, P_2^-) \leftarrow \text{FBP}(P_2, \varphi_f)$
8:             return $(P_1^+ \cup P_2^+, P_1^- \cup P_2^-)$
9:     end if
10: end if
11: end function
Theorem 4 \( \text{FBP}(P, \varphi_f) \) terminates and returns a partitioning \( (P_\oplus, P_\ominus) \) of \( P \) satisfying

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

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

Algorithm 1 Family-Based Partitioning

1: \textbf{function} \text{FBP}(P, \varphi_f)
2: \hspace{1em} \textbf{if} \; s_*, P \models_{F} \varphi_f \; \textbf{then return} \; (P, \emptyset)
3: \hspace{1em} \textbf{else}
4: \hspace{2em} \textbf{if} \; s_*, P \models_{F} \varphi'_f \; \textbf{then return} \; (\emptyset, P)
5: \hspace{2em} \textbf{else partition} \; P \; \text{into} \; (P_1, P_2)
6: \hspace{3em} (P_1^+, P_1^-) \leftarrow \text{FBP}(P_1, \varphi_f)
7: \hspace{3em} (P_2^+, P_2^-) \leftarrow \text{FBP}(P_2, \varphi_f)
8: \hspace{1em} \textbf{return} \; (P_1^+ \cup P_2^+, P_1^- \cup P_2^-)
9: \hspace{1em} \textbf{end if}
10: \hspace{1em} \textbf{end if}
11: \textbf{end function}
### Minepump SPL benchmark ($|\mathcal{P}| = 2^7$)


<table>
<thead>
<tr>
<th>( \Phi )</th>
<th>property in ( \mu L_f )</th>
<th>result</th>
<th>one-by-one</th>
<th>all-in-one</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varphi_1 )</td>
<td>Absence of deadlock ( [\text{true}^<em>] (\text{true}^</em>) \top )</td>
<td>128/0</td>
<td>10.02</td>
<td>2.07</td>
</tr>
<tr>
<td>( \varphi_2 )</td>
<td>The controller cannot infinitely often receive water level readings ( \mu X.([\neg \text{levelMsg}])^* \cdot \text{levelMsg} ) ( X )</td>
<td>0/128</td>
<td>10.18</td>
<td>0.16</td>
</tr>
<tr>
<td>( \varphi_3 )</td>
<td>The controller cannot fairly receive each of the three message types ( \mu X.([\text{true}^* \cdot \text{commandMsg}]) X \lor ([\text{true}^* \cdot \text{alarmMsg}]) X \lor ([\text{true}^* \cdot \text{levelMsg}]) X )</td>
<td>0/128</td>
<td>24.33</td>
<td>0.25</td>
</tr>
<tr>
<td>( \varphi_4 )</td>
<td>The pump cannot be switched on infinitely often ( (\mu X \cdot v Y.([\text{pumpStart} \cdot (\neg \text{pumpStop})^* \cdot \text{pumpStop}]) X \land [\neg \text{pumpStart}]) Y ) ) \land ([\text{true}^* \cdot \text{pumpStart}] \cdot \mu Z.([\neg \text{pumpStop}]) Z)</td>
<td>96/32</td>
<td>21.09</td>
<td>0.89</td>
</tr>
<tr>
<td>( \varphi_5 )</td>
<td>The system cannot be in a situation in which the pump runs indefinitely in the presence of methane ( [\text{true}^<em>] (([\text{pumpStart} \cdot (\neg \text{pumpStop})^</em> \cdot \text{methaneRise}]) \cdot \mu X.[][\text{R}]X ) \land ([\text{methaneRise} \cdot (\neg \text{methaneLower})^* \cdot \text{pumpStart}]) \cdot \mu X.[][\text{R}]X ) )</td>
<td>96/32</td>
<td>17.26</td>
<td>0.86</td>
</tr>
<tr>
<td>&amp; for ( R = \neg((\text{pumpStop} + \text{methaneLower}) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \varphi_6 )</td>
<td>Assuming fairness (( \varphi_3 )), the system cannot be in a situation in which the pump runs indefinitely in the presence of methane (( \varphi_5 )) ( [\text{true}^<em>] (([\text{pumpStart} \cdot (\neg \text{pumpStop})^</em> \cdot \text{methaneRise}]) \cdot \mu X.[][\text{R}]X ) \land ([\text{methaneRise} \cdot (\neg \text{methaneLower})^* \cdot \text{pumpStart}]) \cdot \mu X.[][\text{R}]X ) )</td>
<td>112/16</td>
<td>27.32</td>
<td>3.67</td>
</tr>
<tr>
<td>&amp; for ( \Psi = \mu X.([\text{R}^* \cdot \text{commandMsg}]) X \lor ([\text{R}^* \cdot \text{alarmMsg}]) X \lor ([\text{R}^* \cdot \text{levelMsg}]) X ) and ( R ) as before</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \varphi_7 )</td>
<td>The controller can always eventually receive/read a message, i.e. return to its initial state from any state ( [\text{true}^<em>] (\text{true}^</em>) \text{receiveMsg} ) ( \top )</td>
<td>128/0</td>
<td>18.36</td>
<td>2.40</td>
</tr>
<tr>
<td>( \varphi_8 )</td>
<td>Invariantly the pump is not started when the low water level signal fires ( [\text{true}^* \cdot \text{lowLevel} \cdot (\neg(\text{normalLevel} + \text{highLevel})))^* \cdot \text{pumpStart}] \perp )</td>
<td>128/0</td>
<td>5.67</td>
<td>3.05</td>
</tr>
<tr>
<td>( \varphi_9 )</td>
<td>Invariantly, when the level of methane rises, it inevitably decreases ( [\text{true}^* \cdot \text{methaneRise}] \cdot \mu X.[\neg \text{methaneLower}] X \land (\text{true}^*) \top )</td>
<td>0/128</td>
<td>20.47</td>
<td>0.21</td>
</tr>
<tr>
<td>( \varphi_{10} )</td>
<td>Products with feature ( \text{Ct} ) can switch on the pump ( (\text{true}^* \cdot \text{pumpStart} \mid \text{Ct}) \top )</td>
<td>32/96</td>
<td>6.49</td>
<td>0.31</td>
</tr>
<tr>
<td>( \varphi_{11} )</td>
<td>Products with feature ( \text{Ct} ) can always switch on the pump ( [\text{true}^* \mid \text{Ct}] (\text{true}^* \cdot \text{pumpStart} \mid \text{Ct}) \top )</td>
<td>28/100</td>
<td>21.11</td>
<td>2.32</td>
</tr>
<tr>
<td>( \varphi_{12} )</td>
<td>Products with features ( \text{Ct, Ma, Lh} ) can start the pump upon a high water level, but products without feature ( \text{Lh} ) cannot ( [\text{true}^<em>] \top ) (([\text{highLevel} \cdot \text{Ct} \land \text{Ma} \land \text{Lh}] (\text{true}^</em> \cdot \text{pumpStart} \mid \top \top ) \land [\text{pumpStart} \mid \neg \text{Lh}] \perp )</td>
<td>128/0</td>
<td>13.35</td>
<td>3.36</td>
</tr>
</tbody>
</table>

mCRL2 code distributed with mCRL2 toolset (svn revision 14493)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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

3. (difficult in particular for $\mu$-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-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 $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-calculus, since easily-interpretable feedback from its model checkers is generally missing so far)
Conclusions and future work

Introduced and compared feature-oriented $\mu$-calculi with FTS semantics

Resembles fLTL and fCTL by Classen et al., but $\mu L_f$ is more expressive

Translation to $\mu L_{FO}$ 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 $\mu L_f$; 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 $\mu$-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 ($\varphi_1$) 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 ($+60\%$)