

# 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

- 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

## 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

## 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

## 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

## 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
- $\theta : S \times \mathcal{A} \times S \rightarrow \mathbb{B}[\mathcal{F}]$  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  $p$

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

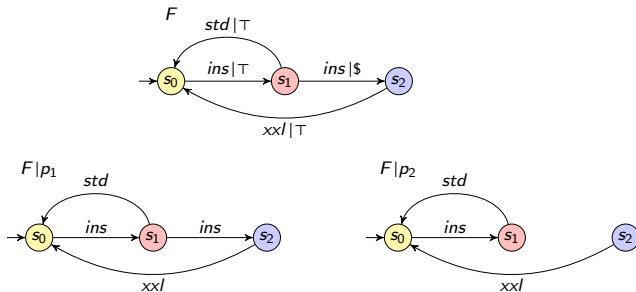
$\mathcal{P} \subseteq 2^{\mathcal{F}}$  set of products  $p, q, \dots$

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

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

i.e. products  $p$  for which the induced truth assignment  
(**true** for  $f \in p$ , **false** for  $f \notin p$ ) validates  $\gamma$

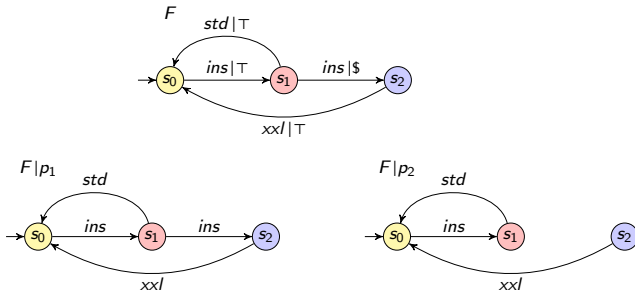
Product line of (four) coffee machines with independent features  $\{\$, \epsilon\}$



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  $\{\$, \epsilon\}$



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?

- 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  $\mu$ -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

- 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  $\mu$ -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

- 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

- 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

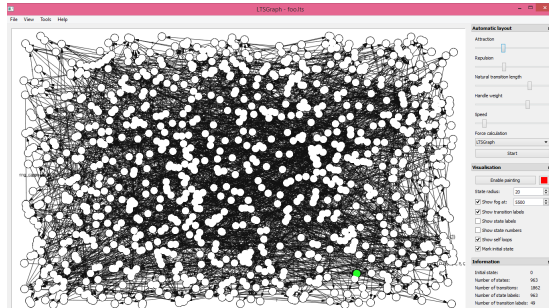
- Formal, process-algebraic specification of distributed and concurrent systems, associated **industrial-strength** toolset
- Explore  $10^6$  states/second, state spaces up to  $10^{12}$  states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, **parametrised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

- 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

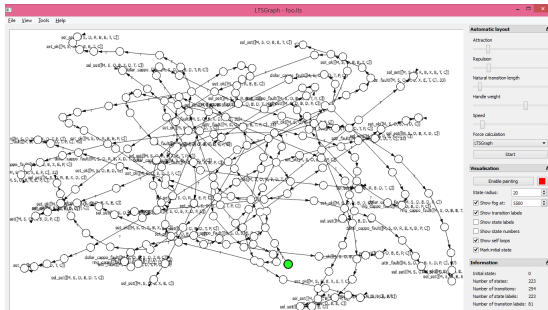


- Formal, process-algebraic specification of distributed and concurrent systems, associated **industrial-strength** toolset
- Explore  $10^6$  states/second, state spaces up to  $10^{12}$  states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, **parametrised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

model checking

- Highly optimised, actively maintained

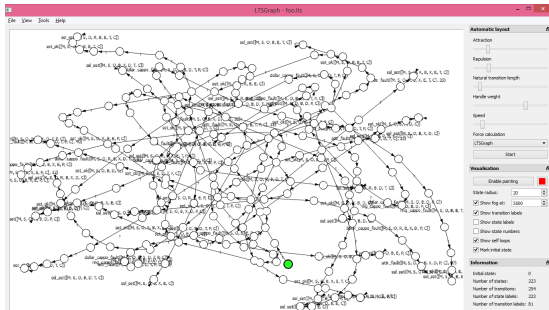
- Intermediate artifacts user-accessible



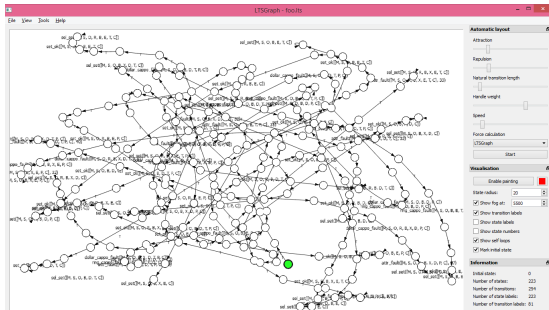
- Formal, process-algebraic specification of distributed and concurrent systems, associated **industrial-strength** toolset
- Explore  $10^6$  states/second, state spaces up to  $10^{12}$  states
- Built-in datatypes (e.g. Bool, Int, Real, Sets, Functions) and user-defined abstract datatypes, **parametrised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

■ Highly optimised,  
actively maintained

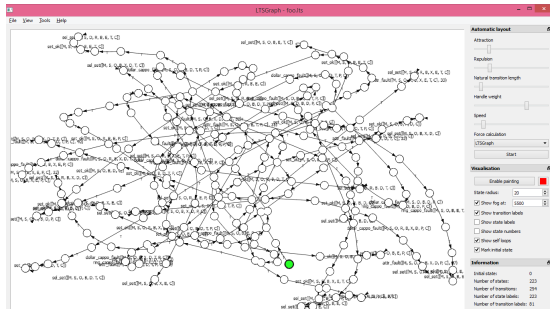
■ Intermediate artifacts  
user-accessible



- 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



- 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



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

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

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

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

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

“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  $b$ ”

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

“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  $b$ ”

$\mu X$ : finite looping

- $\langle a \rangle ([b] \perp \wedge \langle c \rangle \top)$   
“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  $b$ ”

$\mu X$ : finite looping vs.  $\nu X$ : infinite looping

state sets  $U \in sSet = 2^S$ , state-based environments  $\varepsilon \in sEnv = \mathcal{X} \rightarrow sSet$   
semantics  $\llbracket \cdot \rrbracket_L : \mu L \rightarrow sEnv \rightarrow sSet$

$$\llbracket \perp \rrbracket_L(\varepsilon) = \emptyset$$

$$\llbracket \top \rrbracket_L(\varepsilon) = S$$

$$\llbracket \neg\varphi \rrbracket_L(\varepsilon) = S \setminus \llbracket \varphi \rrbracket_L(\varepsilon)$$

$$\llbracket (\varphi \vee \psi) \rrbracket_L(\varepsilon) = \llbracket \varphi \rrbracket_L(\varepsilon) \cup \llbracket \psi \rrbracket_L(\varepsilon)$$

$$\llbracket (\varphi \wedge \psi) \rrbracket_L(\varepsilon) = \llbracket \varphi \rrbracket_L(\varepsilon) \cap \llbracket \psi \rrbracket_L(\varepsilon)$$

$$\llbracket \langle a \rangle \varphi \rrbracket_L(\varepsilon) = \{s \mid \exists t: s \xrightarrow{a} t \wedge t \in \llbracket \varphi \rrbracket_L(\varepsilon)\}$$

$$\llbracket [a] \varphi \rrbracket_L(\varepsilon) = \{s \mid \forall t: s \xrightarrow{a} t \Rightarrow t \in \llbracket \varphi \rrbracket_L(\varepsilon)\}$$

$$\llbracket X \rrbracket_L(\varepsilon) = \varepsilon(X)$$

$$\llbracket \mu X. \varphi \rrbracket_L(\varepsilon) = lfp(U \mapsto \llbracket \varphi \rrbracket_L(\varepsilon[U/X]))$$

$$\llbracket \nu X. \varphi \rrbracket_L(\varepsilon) = gfp(U \mapsto \llbracket \varphi \rrbracket_L(\varepsilon[U/X]))$$

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

state sets  $U \in sSet = 2^S$ , state-based environments  $\varepsilon \in sEnv = \mathcal{X} \rightarrow sSet$   
semantics  $\llbracket \cdot \rrbracket_L : \mu L \rightarrow sEnv \rightarrow sSet$

$$\llbracket \perp \rrbracket_L(\varepsilon) = \emptyset$$

$$\llbracket \top \rrbracket_L(\varepsilon) = S$$

$$\llbracket \neg\varphi \rrbracket_L(\varepsilon) = S \setminus \llbracket \varphi \rrbracket_L(\varepsilon)$$

$$\llbracket (\varphi \vee \psi) \rrbracket_L(\varepsilon) = \llbracket \varphi \rrbracket_L(\varepsilon) \cup \llbracket \psi \rrbracket_L(\varepsilon)$$

$$\llbracket (\varphi \wedge \psi) \rrbracket_L(\varepsilon) = \llbracket \varphi \rrbracket_L(\varepsilon) \cap \llbracket \psi \rrbracket_L(\varepsilon)$$

$$\llbracket \langle a \rangle \varphi \rrbracket_L(\varepsilon) = \{s \mid \exists t: s \xrightarrow{a} t \wedge t \in \llbracket \varphi \rrbracket_L(\varepsilon)\}$$

$$\llbracket [a] \varphi \rrbracket_L(\varepsilon) = \{s \mid \forall t: s \xrightarrow{a} t \Rightarrow t \in \llbracket \varphi \rrbracket_L(\varepsilon)\}$$

$$\llbracket X \rrbracket_L(\varepsilon) = \varepsilon(X)$$

$$\llbracket \mu X. \varphi \rrbracket_L(\varepsilon) = \text{lfp}(U \mapsto \llbracket \varphi \rrbracket_L(\varepsilon[U/X]))$$

$$\llbracket \nu X. \varphi \rrbracket_L(\varepsilon) = \text{gfp}(U \mapsto \llbracket \varphi \rrbracket_L(\varepsilon[U/X]))$$

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

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

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

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

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

semantics  $\llbracket \cdot \rrbracket_F : \mu L_f \rightarrow sPEnv \rightarrow sPSet$

$$\llbracket \perp \rrbracket_F(\zeta) = \emptyset$$

$$\llbracket \top \rrbracket_F(\zeta) = S \times 2^P$$

$$\llbracket \neg \varphi_f \rrbracket_F(\zeta) = (S \times 2^P) \setminus \llbracket \varphi_f \rrbracket_F(\zeta)$$

$$\llbracket (\varphi_f \vee \psi_f) \rrbracket_F(\zeta) = \llbracket \varphi_f \rrbracket_F(\zeta) \cup \llbracket \psi_f \rrbracket_F(\zeta)$$

$$\llbracket (\varphi_f \wedge \psi_f) \rrbracket_F(\zeta) = \llbracket \varphi_f \rrbracket_F(\zeta) \cap \llbracket \psi_f \rrbracket_F(\zeta)$$

$$\llbracket \langle a | \mathcal{X} \rangle \varphi_f \rrbracket_F(\zeta) = \dots$$

$$\llbracket [a | \mathcal{X}] \varphi_f \rrbracket_F(\zeta) = \dots$$

$$\llbracket X \rrbracket_F(\zeta) = \zeta(X)$$

$$\llbracket \mu X. \varphi_f \rrbracket_F(\zeta) = \text{lfp}(W \mapsto \llbracket \varphi_f \rrbracket_F(\zeta[W/X]))$$

$$\llbracket \nu X. \varphi_f \rrbracket_F(\zeta) = \text{gfp}(W \mapsto \llbracket \varphi_f \rrbracket_F(\zeta[W/X]))$$

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

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

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

$[a | \chi] \varphi_f$  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,  $\varphi_f$  holds for  $P'$  in the target state of that  $a$ -transition

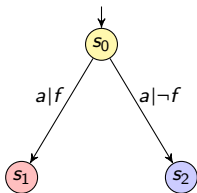
$$\llbracket \langle a | \chi \rangle \varphi_f \rrbracket_F(\zeta) = \{ (s, P) \mid P \subseteq Q_\chi \wedge \exists \gamma, t: s \xrightarrow{a\gamma}_F t \wedge P \subseteq Q_\gamma \wedge (t, P \cap Q_\chi \cap Q_\gamma) \in \llbracket \varphi_f \rrbracket_F(\zeta) \}$$

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

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

$[a | \chi] \varphi_f$  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,  $\varphi_f$  holds for  $P'$  in the target state of that  $a$ -transition

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



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

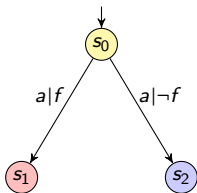
Clearly:  $\langle f, g \rangle \models_{F|p_1} \langle a \rangle T$

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

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

Hence, since neither  $\langle p_1, p_2 \rangle \models_F \langle a \rangle T$  nor  $\langle p_1, p_2 \rangle \models_F \langle a \rangle \perp$ ,  
 $\langle a | \chi \rangle$  and  $\langle a | \chi \rangle$  are not each other's dual

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



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

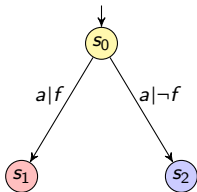
Clearly:  $\{f, g\} \models_{F|p_1} \langle a \rangle T$

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

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

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

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



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

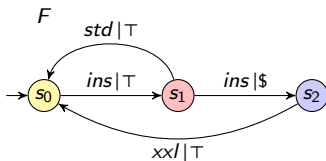
Clearly:  $\{f, g\} \models_{F|p_1} \langle a \rangle \top$

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

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

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

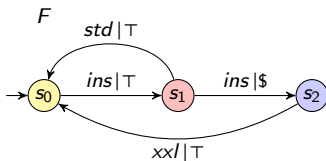
- $\langle ins | \top \rangle ( [ins | \epsilon] \perp \wedge \langle std | \top \rangle \top )$   
 “the family of products  $P$  that can execute  $ins$ , after which  $ins$  cannot be executed by products satisfying  $\epsilon$ , while  $std$  can be executed by all products of  $P$ ”



- $\nu X. \mu Y. ( ([ins | \epsilon] Y \wedge [xxl | \epsilon] Y ) \wedge [std | \epsilon] X )$   
 “for the (sub)family of products with feature  $\epsilon$ , action  $std$  occurs infinitely often on all infinite runs over  $\{ins, xxl, std\}$ ”
- $[true^* | \top] ( ([ins | \$] \langle true^*. xxl | \top \rangle \top ) \wedge [xxl | \neg \$] \perp )$   
 “products with feature  $\$$  can obtain an  $xxl$  coffee upon coin insertion, but products without cannot”

- $\langle ins | T \rangle ( [ins | \epsilon] \perp \wedge \langle std | T \rangle T )$

“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  $P$ ”



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

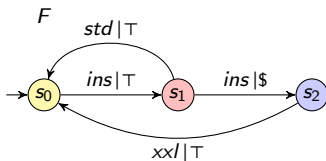
“for the (sub)family of products with feature  $\epsilon$ , action  $std$  occurs infinitely often on all infinite runs over  $\{ins, xxl, std\}$ ”

- $[true^* | T] ( ([ins | \$] \langle true^*. xxl | T \rangle T ) \wedge [xxl | \neg \$] \perp )$

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

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

“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  $P$ ”



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

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

- $[true^* | \top] (([ins | \$] \langle true^*. xxl | \top \rangle \top) \wedge [xxl | \neg \$] \perp)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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  $P$

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

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

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

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

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  $\mathcal{F}$

- $S$  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  $\gamma \neq \perp$

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

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

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

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  $\gamma \neq \perp$

e.g.

$$\llbracket \langle a(v) \rangle \varphi \rrbracket_{FO}(\xi)(\theta) = \{ s \mid \exists \gamma, t: s \xrightarrow{a(\gamma)} t \wedge \theta(v) = Q_\gamma \wedge t \in \llbracket \varphi \rrbracket_{FO}(\xi)(\theta) \}$$

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  $\varphi$

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

$$tr(\gamma, \perp) = \perp$$

$$tr(\gamma, \top) = \top$$

$$tr(\gamma, \neg\varphi_f) = \neg tr(\gamma, \varphi_f)$$

$$tr(\gamma, \varphi_f \vee \psi_f) = tr(\gamma, \varphi_f) \vee tr(\gamma, \psi_f)$$

$$tr(\gamma, \varphi_f \wedge \psi_f) = tr(\gamma, \varphi_f) \wedge tr(\gamma, \psi_f)$$

$$tr(\gamma, \langle a | \chi \rangle \varphi_f) = (\gamma \Rightarrow \chi) \wedge \exists v. \langle a(v) \rangle ((\gamma \Rightarrow v) \wedge tr(\gamma \wedge \chi \wedge v, \varphi_f))$$

$$tr(\gamma, [a | \chi] \varphi_f) = \forall v. [a(v)] ((\gamma \wedge \chi \wedge v \Rightarrow \perp) \vee tr(\gamma \wedge \chi \wedge v, \varphi_f))$$

$$tr(\gamma, X) = \tilde{X}(\gamma)$$

$$tr(\gamma, \mu X. \varphi_f) = \mu \tilde{X}(v := \gamma). tr(v, \varphi_f)$$

$$tr(\gamma, \nu X. \varphi_f) = \nu \tilde{X}(v := \gamma). tr(v, \varphi_f)$$

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  $P \subseteq \mathcal{P}$

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  $P \subseteq \mathcal{P}$

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  $P \subseteq \mathcal{P}$

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  $P \subseteq \mathcal{P}$

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

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

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

---

## Algorithm 1 Family-Based Partitioning

---

```
1: function FBP( $P, \varphi_f$ )
2:   if  $s_*, P \models_F \varphi_f$  then return  $(P, \emptyset)$ 
3:   else
4:     if  $s_*, P \models_F \varphi_f^c$  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

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

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

---

### Algorithm 1 Family-Based Partitioning

---

```

1: function  $\text{FBP}(P, \varphi_f)$ 
2:   if  $s_*, P \models_F \varphi_f$  then return  $(P, \emptyset)$ 
3:   else
4:     if  $s_*, P \models_F \varphi_f^c$  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

```

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

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

www.quanticol.eu

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

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

Extend the framework for finding a **small model** for a given formula

Extract information from failed model-checking attempts to help the partitioning of the family of products by use of **heuristics**

Extend  $\mu L_f$  to include **feature-level** state transitions

Develop **configurable model checkers** for the proposed framework

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

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

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)

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)

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)

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)

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**

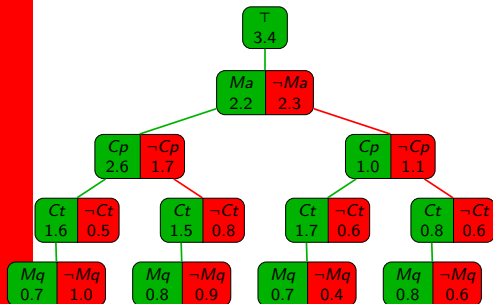
Translation to  $\mu L_{FO}$  allows **family-based** model checking **multi-feature** properties of *configurable systems* with **off-the-shelf** tools (e.g. mCRL2)

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)

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)



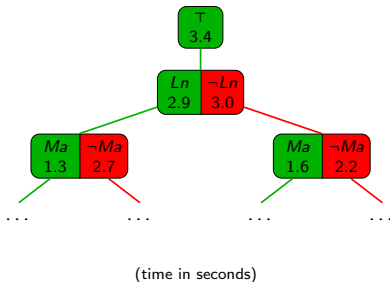
Optimal partitioning strategy

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%)