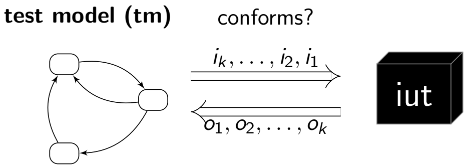
Model-Based Conformance Testing of Software Product Lines

23rd CREST Open Workshop

Malte Lochau, Real-Time Systems Lab, TU Darmstadt

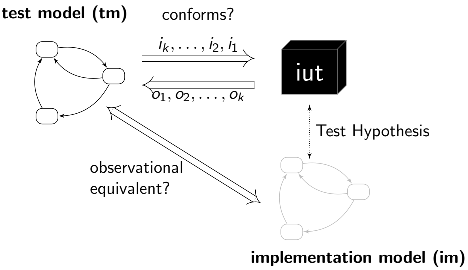
Nov. 20th, 2012

Model-Based Testing [UL07]



- Black-box assumption for implementation under test (iut)
- Automated derivation and application of test cases from a behavioral specification (test model)

Model-Based Conformance Testing [Tre99]



- Test Hypothesis for test result confidence and reproducibility [Ber91]
- Partial verification of the observable behavioral conformance [NH84]

Testing Preorder Relations

Implementation relation – equivalent behaviors:

$$\textbf{impl} \equiv \textbf{spec}$$

Testing Preorder Relations

Implementation relation – equivalent behaviors:

$$\textbf{impl} \equiv \textbf{spec}$$

Preorder relation – implementation conforms specification:

$$\textbf{impl} \sqsubseteq \textbf{spec}$$

Testing Preorder Relations

Implementation relation – equivalent behaviors:

$$\textbf{impl} \equiv \textbf{spec}$$

Preorder relation – implementation conforms specification:

$$\textbf{impl} \sqsubseteq \textbf{spec}$$

Model-based testing – test model as behavioral specification:

$$\textbf{impl} \sqsubseteq \textbf{tm}$$

Testing Preorder Relations

Implementation relation – equivalent behaviors:

$$\textbf{impl} \equiv \textbf{spec}$$

Preorder relation – implementation conforms specification:

$$\textbf{impl} \sqsubseteq \textbf{spec}$$

Model-based testing – test model as behavioral specification:

$$\textbf{impl} \sqsubseteq \textbf{tm}$$

Black-box assumption – imaginary implementation model:

$$\textbf{im} \sqsubseteq \textbf{tm}$$

Testing Preorder Relations

Implementation relation – equivalent behaviors:

$$\textbf{impl} \equiv \textbf{spec}$$

Preorder relation – implementation conforms specification:

$$\textbf{impl} \sqsubseteq \textbf{spec}$$

Model-based testing – test model as behavioral specification:

$$\textbf{impl} \sqsubseteq \textbf{tm}$$

Black-box assumption – imaginary implementation model:

$$\textbf{im} \sqsubseteq \textbf{tm}$$

Weakened implementation relation – testing equivalence:

$$\textbf{im} \sqsubseteq_{te} \textbf{tm}$$

Testing Preorder Relations

Implementation relation – equivalent behaviors:

$$\textbf{impl} \equiv \textbf{spec}$$

Preorder relation – implementation conforms specification:

$$\textbf{impl} \sqsubseteq \textbf{spec}$$

Model-based testing – test model as behavioral specification:

$$\textbf{impl} \sqsubseteq \textbf{tm}$$

Black-box assumption – imaginary implementation model:

$$\textbf{im} \sqsubseteq \textbf{tm}$$

Weakened implementation relation – testing equivalence:

$$\textbf{im} \sqsubseteq_{te} \textbf{tm}$$

Parameterized implementation relation – finite set of behaviors:

$$\textbf{im} \sqsubseteq_{te}^{TC} \textbf{tm}$$

Labeled Transition Systems (LTS)

- Labeled State-Transition Graph $(Proc, Act, \rightarrow)$
- LTS trace semantics $tr = (a_1, a_2, \ldots, a_n) \in Tr(s_0, lts) \subseteq Act^*$, iff

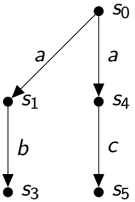
$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \cdots \xrightarrow{a_n} s_n = s_0 \xrightarrow{tr} s_n$$

Labeled Transition Systems (LTS)

- Labeled State-Transition Graph $(Proc, Act, \rightarrow)$
- LTS trace semantics $tr = (a_1, a_2, \ldots, a_n) \in Tr(s_0, lts) \subseteq Act^*$, iff

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \cdots \xrightarrow{a_n} s_n = s_0 \xrightarrow{tr} s_n$$

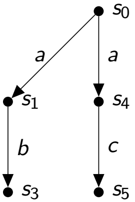
- $Tr(s_0, lts_1) = \{a, ab, ac\}$



Labeled Transition Systems (LTS)

- Labeled State-Transition Graph $(Proc, Act, \rightarrow)$
- LTS trace semantics $tr = (a_1, a_2, \ldots, a_n) \in Tr(s_0, lts) \subseteq Act^*$, iff

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \cdots \xrightarrow{a_n} s_n = s_0 \xrightarrow{tr} s_n$$



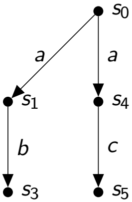
- $Tr(s_0, lts_1) = \{a, ab, ac\}$
- Trace Preorder as Testing Preorder Relation:

$$im \sqsubseteq_T tm \;:\Leftrightarrow\; Tr(s_0, im) \subseteq Tr(s_0, tm)$$

Labeled Transition Systems (LTS)

- Labeled State-Transition Graph $(Proc, Act, \rightarrow)$
- LTS trace semantics $tr = (a_1, a_2, \ldots, a_n) \in Tr(s_0, lts) \subseteq Act^*$, iff

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \cdots \xrightarrow{a_n} s_n = s_0 \xrightarrow{tr} s_n$$



- $Tr(s_0, lts_1) = \{a, ab, ac\}$
- Trace Preorder as Testing Preorder Relation:

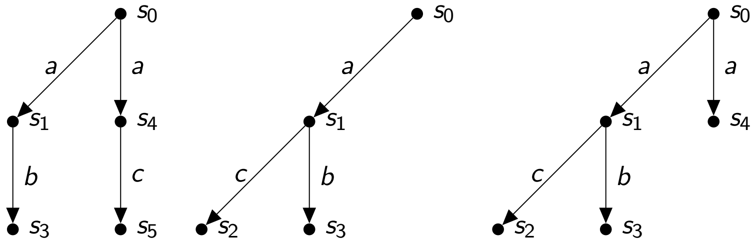
$$im \sqsubseteq_T tm \ :\Leftrightarrow \ Tr(s_0, im) \subseteq Tr(s_0, tm)$$

- Parameterized Testing Preorder Relation:

$$im \sqsubseteq_T^{TC} tm \ :\Leftrightarrow \ (Tr(s_0, im) \cap TC) \subseteq (Tr(s_0, tm) \cap TC)$$

where $TC \subseteq Tr(s_0, im)$

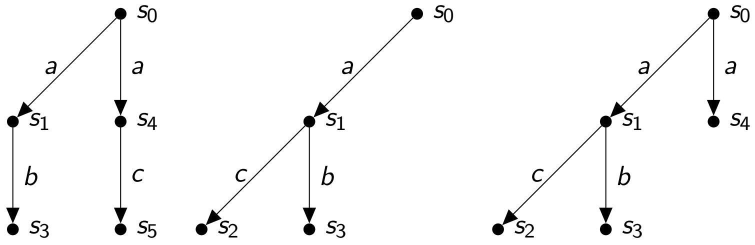
Example



(e) lts_1 (f) lts_2 (g) lts_3

- $lts_1 \equiv_T lts_2 \equiv_T lts_3$

Example



(h) lts_1 (i) lts_2 (j) lts_3

- $lts_1 \equiv_T lts_2 \equiv_T lts_3$
- **But:** different behaviors after composition with environment emitting input action a.

Decorated Trace Semantics

- Trace equivalence is a *weak* equivalence
- Stricter notions of behavioral equivalence discriminate different decision structures within the state-transition graphs [Abr87]

Decorated Trace Semantics

- Trace equivalence is a *weak* equivalence
- Stricter notions of behavioral equivalence discriminate different decision structures within the state-transition graphs [Abr87]
- **But:** testing is limited to observable behaviors

$$initials(s) = \{a \in Act \mid s \xrightarrow{a} \} \subseteq Act$$

Decorated Trace Semantics

- Trace equivalence is a *weak* equivalence
- Stricter notions of behavioral equivalence discriminate different decision structures within the state-transition graphs [Abr87]
- **But:** testing is limited to observable behaviors

$$initials(s) = \{a \in Act \mid s \xrightarrow{a} \} \subseteq Act$$

Example: Failures and Readies

Decorated Trace Semantics

- Trace equivalence is a *weak* equivalence
- Stricter notions of behavioral equivalence discriminate different decision structures within the state-transition graphs [Abr87]
- **But:** testing is limited to observable behaviors

$$initials(s) = \{a \in Act \mid s \xrightarrow{a} \} \subseteq Act$$

Example: Failures and Readies

- A pair (tr, X) with $tr \in Act^*$ and $X \subseteq Act$ is a *failure* of state s_0 if $s_0 \xrightarrow{tr} s_n$ for some state s_n and $initials(s_n) \cap X = \emptyset$.

Decorated Trace Semantics

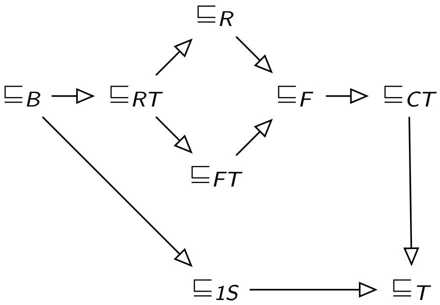
- Trace equivalence is a *weak* equivalence
- Stricter notions of behavioral equivalence discriminate different decision structures within the state-transition graphs [Abr87]
- **But:** testing is limited to observable behaviors

$$initials(s) = \{a \in Act \mid s \xrightarrow{a} \} \subseteq Act$$

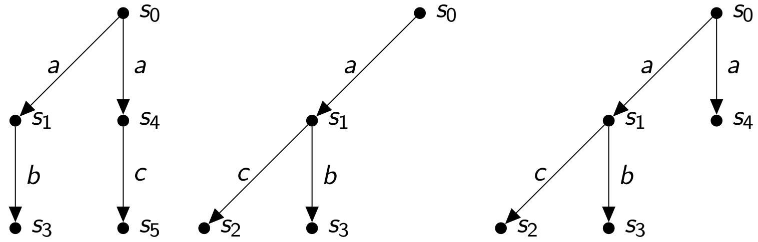
Example: Failures and Readies

- A pair (tr, X) with $tr \in Act^*$ and $X \subseteq Act$ is a *failure* of state s_0 if $s_0 \xrightarrow{tr} s_n$ for some state s_n and $initials(s_n) \cap X = \emptyset$.
- A pair (tr, X) with $tr \in Act^*$ and $X \subseteq Act$ is a *ready* of state s_0 if $s_0 \xrightarrow{tr} s_n$ for some state s_n and $initials(s_n) = X$.

Preorder Relation Inclusion Hierarchy [BFvG04]



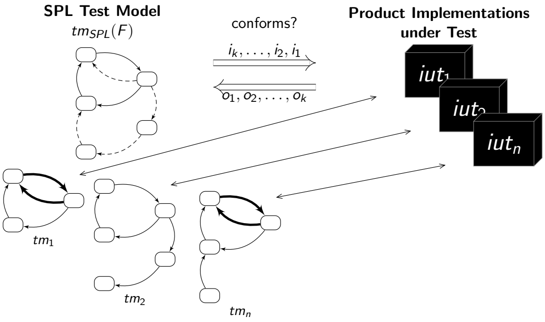
Example – Revisited



(k) lts_1 (l) lts_2 (m) lts_3

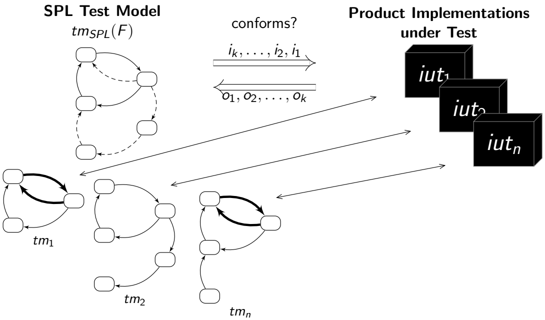
- lts_3 has completed trace a
- $lts_2 \sqsubseteq_F lts_1$
- lts_2 and lts_1 are incomparable under $\sqsubseteq_R$

Model-Based SPL Testing



- Reusable generic test model specification parameterized over features F

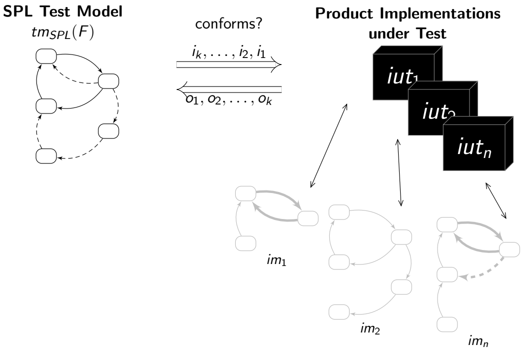
Model-Based SPL Testing



- Reusable generic test model specification parameterized over features F
- Reuse of test cases $TC' \subseteq TC$ of product *iut* for product *iut'* if

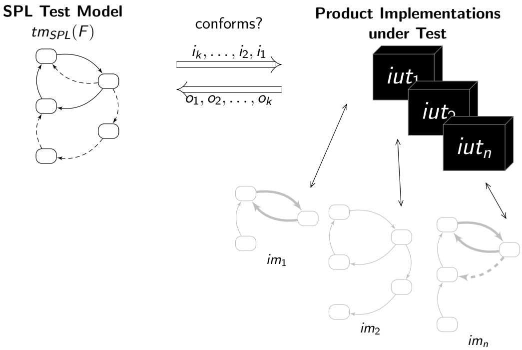
$$tm \sqsubseteq_{te}^{TC'} tm'$$

Model-Based SPL Testing



- Extending the Test Hypothesis to SPLs under test

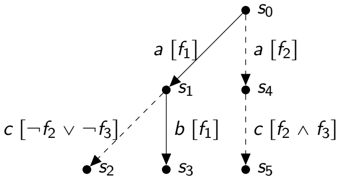
Model-Based SPL Testing



- Extending the Test Hypothesis to SPLs under test
- Reuse of test results $TC'' \subseteq TC'$ of product *iut* for product *iut'* if

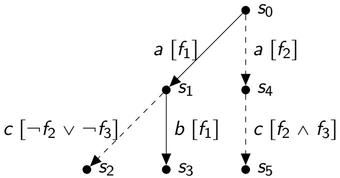
$$im \sqsubseteq_{te}^{TC''} im'$$

Feature-Annotated LTS [CHSL11]

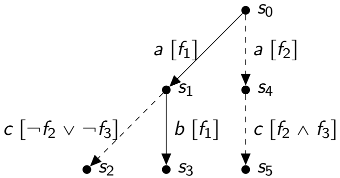


Feature-Annotated LTS [CHSL11]

- LTS with transition annotations $\sigma(s, a, s') \in \mathbb{B}(F)$

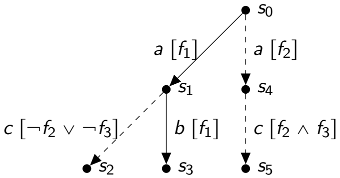


Feature-Annotated LTS [CHSL11]



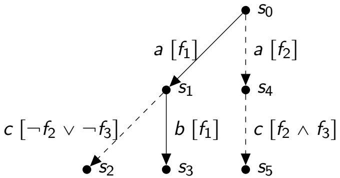
- LTS with transition annotations $\sigma(s, a, s') \in \mathbb{B}(F)$
- Constraints by feature model $fm \in \mathbb{B}(F)$

Feature-Annotated LTS [CHSL11]



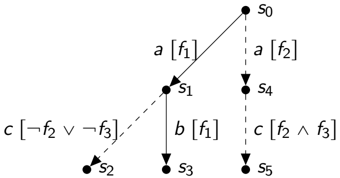
- LTS with transition annotations $\sigma(s, a, s') \in \mathbb{B}(F)$
- Constraints by feature model $fm \in \mathbb{B}(F)$
- Product configuration $\Gamma : F \to \mathbb{B}$ (full, partial)

Feature-Annotated LTS [CHSL11]



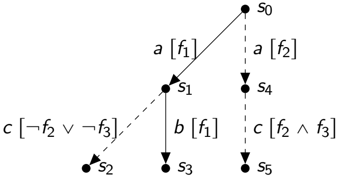
- LTS with transition annotations $\sigma(s, a, s') \in \mathbb{B}(F)$
- Constraints by feature model $fm \in \mathbb{B}(F)$
- Product configuration $\Gamma : F \to \mathbb{B}$ (full, partial)
- Product space $PC_{fm} = \{\Gamma : F \to \mathbb{B} \mid \Gamma \models fm\}$

Feature-Annotated LTS [CHSL11]



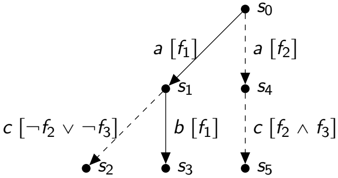
- LTS with transition annotations $\sigma(s, a, s') \in \mathbb{B}(F)$
- Constraints by feature model $fm \in \mathbb{B}(F)$
- Product configuration $\Gamma : F \to \mathbb{B}$ (full, partial)
- Product space $PC_{fm} = \{\Gamma : F \to \mathbb{B} \mid \Gamma \models fm\}$
- Feature model refinement $fm' \sqsubseteq_{fm} fm$ is product space refinement

Transition Modalities [LT88]



$$fm = f_1 \wedge (f_2 \vee f_3)$$

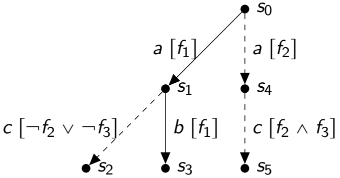
Transition Modalities [LT88]



$$fm = f_1 \wedge (f_2 \vee f_3)$$

- *may*-transitions $\rightarrow_{may} \subseteq \rightarrow$, where
  $s \xrightarrow{a}_{may} s' :\Leftrightarrow \exists \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$

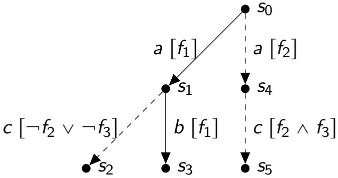
Transition Modalities [LT88]



$$fm = f_1 \wedge (f_2 \vee f_3)$$

- *may*-transitions $\rightarrow_{may} \subseteq \rightarrow$, where
  $s \xrightarrow{a}_{may} s' :\Leftrightarrow \exists \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$
- *must*-transitions $\rightarrow_{must} \subseteq \rightarrow$, where
  $s \xrightarrow{a}_{must} s' :\Leftrightarrow \forall \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$

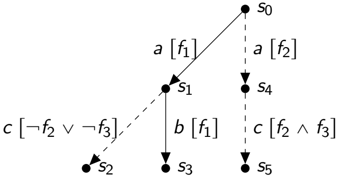
Transition Modalities [LT88]



$$fm = f_1 \wedge (f_2 \vee f_3)$$

- *may*-transitions $\rightarrow_{may} \subseteq \rightarrow$, where
  $s \xrightarrow{a}_{may} s' :\Leftrightarrow \exists \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$
- *must*-transitions $\rightarrow_{must} \subseteq \rightarrow$, where
  $s \xrightarrow{a}_{must} s' :\Leftrightarrow \forall \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$
- *prohibited*-transitions $\nrightarrow \subseteq Proc \times Act \times Proc$, where
  $s \overset{a}{\nrightarrow} s' :\Leftrightarrow \nexists \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$

Transition Modalities [LT88]



$$fm = f_1 \wedge (f_2 \vee f_3)$$

- *may*-transitions $\longrightarrow_{may} \subseteq \longrightarrow$, where
  $s \xrightarrow{a}_{may} s' :\Leftrightarrow \exists \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$
- *must*-transitions $\longrightarrow_{must} \subseteq \longrightarrow$, where
  $s \xrightarrow{a}_{must} s' :\Leftrightarrow \forall \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$
- *prohibited*-transitions $\nrightarrow \subseteq Proc \times Act \times Proc$, where
  $s \overset{a}{\nrightarrow} s' :\Leftrightarrow \nexists \Gamma \in PC_{fm} : \Gamma \models \sigma(s, a, s')$

- $\longrightarrow_{must} \subseteq \longrightarrow_{may}$
- $\nrightarrow \cap \longrightarrow_{may} = \emptyset$

F-LTS Refinement

From $fm' \sqsubseteq_{FM} fm$ it follows that $lts_{\Gamma'} \sqsubseteq_T lts_{\Gamma}$

- $\longrightarrow'_{may} \subseteq \longrightarrow_{may}$
- $\longrightarrow_{must} \subseteq \longrightarrow'_{must}$
- $\not\twoheadrightarrow \subseteq \not\twoheadrightarrow'$

F-LTS Refinement

From $fm' \sqsubseteq_{FM} fm$ it follows that $lts_{\Gamma'} \sqsubseteq_T lts_\Gamma$

- $\longrightarrow'_{may} \subseteq \longrightarrow_{may}$
- $\longrightarrow_{must} \subseteq \longrightarrow'_{must}$
- $\nrightarrow \subseteq \nrightarrow'$

But: this does not hold for decorated trace semantics

- Set of failures increases under refinement
- Set of readies is not subset closed

$\Rightarrow$ May-transitions may become failures as well as readies after refinement

Decorated May-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-failure* of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \notin X$

Decorated May-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-failure* of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \notin X$
- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-ready* of state s_0 if (1) for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \in X$, and (2) for each $a \in Act$ with $s \overset{a}{\nrightarrow} s'$ it holds that $a \notin X$

Decorated May-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-failure* of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \notin X$
- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-ready* of state s_0 if (1) for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \in X$, and (2) for each $a \in Act$ with $s \stackrel{a}{\nrightarrow} s'$ it holds that $a \notin X$

From $fm' \sqsubseteq_{FM} fm$ it follows that $lts_{\Gamma'} \sqsubseteq_{te\text{-}may} lts_{\Gamma}$ holds.

Decorated May-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-failure* of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \notin X$
- A pair (tr, X) with $s_0 \xrightarrow{tr} s$ and $X \subseteq Act$ is a *may-ready* of state s_0 if (1) for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \in X$, and (2) for each $a \in Act$ with $s \overset{a}{\nrightarrow} s'$ it holds that $a \notin X$

From $fm' \sqsubseteq_{FM} fm$ it follows that $lts_{\Gamma'} \sqsubseteq_{te\text{-}may} lts_{\Gamma}$ holds.

- **But:** full product configurations are incomparable under $\sqsubseteq_{te\text{-}may}$

Must-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$, where $s_i \xrightarrow{a}_{must} s_{i+1}$, for $0 \leqslant i < n$, and $X \subseteq Act$ is a *must*-failure of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{may} s'$ it holds that $a \notin X$
- A pair (tr, X) with $s_0 \xrightarrow{tr} s$, where $s_i \xrightarrow{a}_{must} s_{i+1}$, for $0 \leqslant i < n$, and $X \subseteq Act$ is a *must*-ready of state s_0 if (1) for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \in X$, and (2) there is no $a' \in Act$ with $s \xrightarrow{a'}_{may}$ and not $s \xrightarrow{a'}_{must}$

Must-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$, where $s_i \xrightarrow{a}_{must} s_{i+1}$, for $0 \leqslant i < n$, and $X \subseteq Act$ is a *must*-failure of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{may} s'$ it holds that $a \notin X$
- A pair (tr, X) with $s_0 \xrightarrow{tr} s$, where $s_i \xrightarrow{a}_{must} s_{i+1}$, for $0 \leqslant i < n$, and $X \subseteq Act$ is a *must*-ready of state s_0 if (1) for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \in X$, and (2) there is no $a' \in Act$ with $s \xrightarrow{a'}_{may}$ and not $s \xrightarrow{a'}_{must}$

$\Rightarrow$ From $fm' \sqsubseteq_{FM} fm$ it follows that $lts_{\Gamma} \sqsubseteq_{te\text{-}must} lts_{\Gamma'}$ holds.

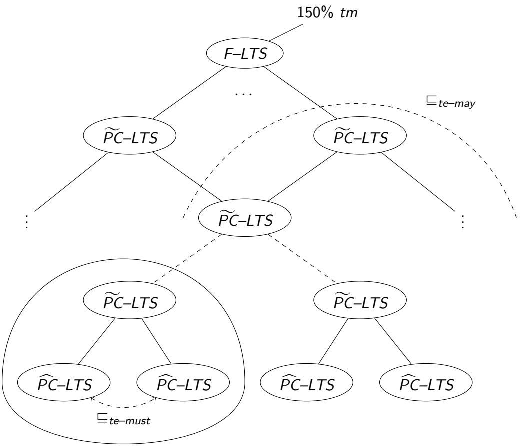
Must-Trace Semantics

- A pair (tr, X) with $s_0 \xrightarrow{tr} s$, where $s_i \xrightarrow{a}_{must} s_{i+1}$, for $0 \leqslant i < n$, and $X \subseteq Act$ is a *must*-failure of state s_0 if for each $a \in Act$ with $s \xrightarrow{a}_{may} s'$ it holds that $a \notin X$
- A pair (tr, X) with $s_0 \xrightarrow{tr} s$, where $s_i \xrightarrow{a}_{must} s_{i+1}$, for $0 \leqslant i < n$, and $X \subseteq Act$ is a *must*-ready of state s_0 if (1) for each $a \in Act$ with $s \xrightarrow{a}_{must} s'$ it holds that $a \in X$, and (2) there is no $a' \in Act$ with $s \xrightarrow{a'}_{may}$ and not $s \xrightarrow{a'}_{must}$

$\Rightarrow$ From $fm' \sqsubseteq_{FM} fm$ it follows that $lts_\Gamma \sqsubseteq_{te\text{-}must} lts_{\Gamma'}$ holds.

$\Rightarrow$ From $\Gamma'' = lub(\Gamma, \Gamma')$ and $TC = Tr_{te\text{-}must}(s_0, f\text{-}lts'')$ it follows that $lts_\Gamma \sqsubseteq_{te}^{TC} lts_{\Gamma'}$ holds.

F-LTS Refinement Hierarchy



FM-constraint May-Trace Semantics

- A trace $s_0 \xrightarrow{tr} s_n$ is an *fm-constraint may-trace* if $\bigwedge_{1 \leqslant i \leqslant n} \sigma(s_{i-1}, a_i, s_i) \models fm$ holds
- A may-failure (tr, X) is an *fm-constraint may-failure* if (1) $s_0 \xrightarrow{tr} s_n$ is an *FM-constraint may-trace*, and (2) $\bigwedge_{a \in X} \neg\sigma(s_n, a, s') \models fm$ holds
- A may-ready (tr, X) is an *fm-constraint may-ready* if (1) $s_0 \xrightarrow{tr} s_n$ is an *FM-constraint may-trace*, and (2) $\bigwedge_{a \in X} \sigma(s_n, a, s') \models fm$ holds

Conclusions & Future Work

- Sample implementation for trace preorder semantics [LSKL12, LLSG12]
- Test result reuse via test model slicing [KLB12]

Future Work

- Variability-aware test result reuse criteria
- Feature-Unit testing
- Testing Equivalences with τ-sensitivity $\rightarrow$ **pl-ioco**
- Automated SPL test suite generation

Thanks for Your Attention

Any Questions?

References I

Samson Abramsky.
Observation Equivalence as a Testing Equivalence.
Theoretical Computer Science, 53:225–241, August 1987.

Gilles Bernot.
Testing against Formal Specifications: A Theoretical View.
In S. Abramsky and T. Maibaum, editors, *TAPSOFT '91*, volume 494 of *Lecture Notes in Computer Science*, pages 99–119. Springer Berlin , Heidelberg, 1991.
10.1007/354053981663.

Bard Bloom, Wan Fokkink, and Rob J. van Glabbeek.
Precongruence Formats for Decorated Trace Semantics.
ACM Transactions on Computational Logic, 5(1):26–78, Jan. 2004.

Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay.
Symbolic Model Checking of Software Product Lines.
In *ICSE*, pages 321–330, 2011.

References II

Jochen Kamischke, Malte Lochau, and Hauke Baller.
Conditioned Model Slicing of Feature-Annotated State Machines.
In *4th International Workshop on Feature-Oriented Software Development (FOSD)*, 2012.

Sascha Lity, Malte Lochau, Ina Schaefer, and Ursula Goltz.
Delta-oriented Model-based SPL Regression Testing.
In *3rd International IEEE Workshop on Product Line Approaches in Software Engineering (PLEASE)*, 2012.

Malte Lochau, Ina Schaefer, Jochen Kamischke, and Sascha Lity.
Incremental Model-based Testing of Delta-oriented Software Product Lines.
In *6th International Conference on Tests & Proofs*, 2012.

Kim Guldstrand Larsen and Bent Thomsen.
A Modal Process Logic.
In *Logic in Computer Science, 1988. LICS '88., Proceedings of the Third Annual Symposium on*, pages 203 –210, Juli 1988.

References III

Rocco De Nicola and Matthew. C. B. Hennessy.
Testing Equivalences for Processes.
Theoretical Computer Science, pages 83–133, 1984.

Jan Tretmans.
Testing Concurrent Systems: A Formal Approach.
In *Proceedings of the 10th International Conference on Concurrency Theory*, CONCUR '99, pages 46–65, London, UK, 1999. Springer-Verlag.

Mark Utting and Bruno Legeard.
Practical Model-Based Testing. A Tools Approach.
M. Kaufmann, 2007.