On the Expressiveness of Modal Transition Systems with Variability Constraints

Maurice ter Beek, Stefania Gnesi & Franco Mazzanti
Formal Methods and Tools lab, ISTI-CNR, Pisa, Italy

Ferruccio Damiani & Luca Paolini
University of Torino, Torino, Italy

iFM’19
Bergen, Norway
5-12-2019
Outline

Behavioural SPL models (and tools)
- Featured Transition System (FTS)
- Modal Transition System (MTS)
- MTS with Variability Constraints (MTS_υ)

Expressiveness results
- FTS vs. MTS(υ)
- FTS2MTS_υ
- MTS_υ2FTS

Conclusion
Behavioural models (and tools) for product lines

Lift success stories known for single systems (products) to sets of products (families) by exploiting variability modelling and analysis

⇒ challenges models and tools by potentially high number of different products, each with a large state space in general

- **Featured Transition Systems (FTSs)**
  - SNIP/fPROMELA/fLTL, fNuSMV/fSMV/fCTL, ProVeLines
    - Classen et al. @ ICSE’11, Int. J. Softw. Tools Technol. Transf., 2012, Cordy et al. @ SPLC’13

- **Modal Transition Systems (MTSs) with variability constraints**
  - Asirelli et al. @ iFM’10, FMOODS’11, SPLC’11, ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016

- **Variability Model Checker VMC/v-ACTL**
  - ter Beek et al. @ FM’12, SPLC’12, SPLat’14

- **UML SPL profile, PL-CCS, variable I/O automata, feature nets**
  - Ziadi et al. @ PFE’03, Grüler et al. @ FMOODS’08, SPLC’08, Lauenroth et al. @ ASE’09, Muschevici et al. @ SEFM’11, Softw. Syst. Model., 2016
Behavioural models (and tools) for product lines

Lift success stories known for single systems (products) to sets of products (families) by exploiting variability modelling and analysis

⇒ challenges models and tools by potentially high number of different products, each with a large state space in general

▶ Featured Transition Systems (FTSs)
  SNIP/fPROMELA/fLTL, fNuSMV/fSMV/fCTL, ProVeLines
  Classen et al. @ ICSE’11, Int. J. Softw. Tools Technol. Transf., 2012, Cordy et al. @ SPLC’13

▶ Modal Transition Systems (MTSs) with variability constraints
  Asirelli et al. @ iFM’10, FMOODS’11, SPLC’11, ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016
  Variability Model Checker VMC/v-ACTL
  ter Beek et al. @ FM’12, SPLC’12, SPLat’14

▶ UML SPL profile, PL-CCS, variable I/O automata, feature nets
  Ziadi et al. @ PFE’03, Grüler et al. @ FMOODS’08, SPLC’08, Lauenroth et al. @ ASE’09, Muschevici et al. @ SEFM’11, Softw. Syst. Model., 2016
FTS for SPLE

Feature model:

VendingMachine

Beverages (b)
Soda s
Tea t

FreeDrinks (f)

CancelPurchase (c)

12 valid products e.g. \{v, b, s, t\}, \{v, b, s, c\}
FTS for SPLF

Feature model:

FTS of 12 valid products (LTSs) e.g. \{v, b, s, t\}, \{v, b, s, c\}

return/c → 4 cancel/c

1 pay/v∧¬f

2 change/v

3 soda/s

4 free/f

5 serveSoda/s

6 serveTea/t

7 tea/t

8 open/v∧¬f

9 take/v

close/v
FTS for SPLE

Feature model:

- **VendingMachine**
  - **Beverages**
    - **Soda**
    - **Tea**
  - **FreeDrinks**
  - **CancelPurchase**

- **serveSoda** → 1 → pay → 2 → change → 3 → **serveSoda**
- **serveTea** → 6 → tea → 7 → open → 8 → **serveTea**
- **serveTea** → 6 → tea → 7 → open → 8 → **serveTea**
- **serveTea** → 6 → tea → 7 → open → 8 → **serveTea**
- **serveSoda** → 5 → soda → 7 → open → 8 → **serveTea**

E.g. \{v, b, s, t\}, \{v, b, s, c\}
FTS for SPLE

Feature model:

```
{v, b, s, t}, {v, b, s, c}
```

return ↓

serveSoda →

1 →

pay →

change →

3 →

soda →

5 serveSoda →

7 open →

take →

9 →

cancel

1 return

3 pay

5 soda

7 open

9 take

close

Soda s

Tea t

Beverages b

FreeDrinks f

CancelPurchase c

VendingMachine v
Dedicated FTS model checker SNIP (now ProVeLines)

fPROMELA:

typedef features {
    bool v;
    bool f;
    ...
};
features F;

\[
gd :: F.v && !F.f;
\]
\[
\text{pay} :: F.f;
\]
\[
\text{free} :: \text{else};
\]
\[
\text{skip};
\]
\[
dg;
\]

\[
\text{return/c} \quad \text{cancel/c}
\]
\[
\text{free/f} \\
\text{pay/v} \land \neg f \quad \text{change/v}
\]
\[
\text{soda/s} \quad \text{selected} \quad \text{serveSoda/s}
\]
\[
\text{tea/t} \quad \text{serveTea/t}
\]
\[
\text{open/v} \land \neg f \quad \text{take/v}
\]
\[
\text{close/v}
\]

fLTL:
\[
[\neg f] \Box (\text{selected} \Rightarrow \Diamond \text{open})
\]

Similarly fNuSMV with fSMV/fCTL
Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible (may) / necessary (must) transitions
  Larsen & Thomsen @ LICS’88
- Recognised as useful model to compactly describe the possible behaviour of all products (LTSs) of product line (a.k.a. family)
  Fischbein et al. @ ROSATEA’06, Fantechi & Gnesi @ ESEC/FSE’07, SPLC’08

MTS cannot model variability constraints regarding alternative features, nor regarding requires/excludes cross-tree constraints, resulting in several variants and extensions
  Larsen et al. @ ESOP’07, Lauenroth et al. @ ASE’09, ...

Our solution: add a set of variability constraints to the MTS to be able to decide which derivable products (LTSs) are valid ones
  Asirelli et al. @ SPLC’11, ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016
Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible (may) / necessary (must) transitions
  Larsen & Thomsen @ LICS’88

- Recognised as useful model to compactly describe the possible behaviour of all products (LTSs) of product line (a.k.a. family)
  Fischbein et al. @ ROSATEA’06, Fantechi & Gnesi @ ESEC/FSE’07, SPLC’08

👎 MTS cannot model variability constraints regarding alternative features, nor regarding requires/excludes cross-tree constraints, resulting in several variants and extensions
  Larsen et al. @ ESOP’07, Lauenroth et al. @ ASE’09, ...

👍 Our solution: add a set of variability constraints to the MTS to be able to decide which derivable products (LTSs) are valid ones
  Asirelli et al. @ SPLC’11, ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016
Main ingredient: Modal Transition Systems (MTS)

- LTS distinguishing admissible (may) / necessary (must) transitions
  Larsen & Thomsen @ LICS'88

- Recognised as useful model to *compactly* describe the possible behaviour of all products (LTSs) of product line (a.k.a. family)
  Fischbein et al. @ ROSATEA'06, Fantechi & Gnesi @ ESEC/FSE'07, SPLC'08

👎 MTS cannot model variability constraints regarding *alternative* features, nor regarding *requires/excludes* cross-tree constraints, resulting in several variants and extensions
  Larsen et al. @ ESOP’07, Lauenroth et al. @ ASE'09, …

👍 Our solution: add a set of *variability constraints* to the MTS to be able to decide which derivable products (LTSs) are valid ones
  Asirelli et al. @ SPLC’11, ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016
Dedicated MTS\(\nu\) Variability Model Checker VMC

Input: specification of an MTS\(\nu\) in process-algebraic terms together with a set of logical variability constraints

VMC offers product-based and family-based variability analyses:

1. The actual set of all valid product behaviour can explicitly be generated and the resulting LTSs can all be verified against one and the same logic property (expressed in Action-based CTL)
   De Nicola, Vaandrager @ J. ACM, 1995

2. A logic property (expressed in variability-aware ACTL) can directly be verified against the MTS, relying on the fact that under certain syntactic conditions its validity over the MTS implies validity of the same property for all derived products
   ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016

VMC v6.4 is freely usable online: http://fmt.isti.cnr.it/vmc/
Dedicated $MTS_\nu$ Variability Model Checker VMC

Input: specification of an $MTS_\nu$ in process-algebraic terms together with a set of logical variability constraints

VMC offers product-based and family-based variability analyses:

1. The actual set of all valid product behaviour can explicitly be generated and the resulting LTSs can all be verified against one and the same logic property (expressed in Action-based CTL)
   De Nicola, Vaandrager @ J. ACM, 1995

2. A logic property (expressed in variability-aware ACTL) can directly be verified against the MTS, relying on the fact that under certain syntactic conditions its validity over the MTS implies validity of the same property for all derived products
   ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016

VMC v6.4 is freely usable online: http://fmt.isti.cnr.it/vmc/
Dedicated $\text{MTS}_\nu$ Variability Model Checker VMC

Input: specification of an $\text{MTS}_\nu$ in process-algebraic terms together with a set of logical variability constraints

VMC offers product-based and family-based variability analyses:

1. The actual set of all valid product behaviour can explicitly be generated and the resulting LTSs can all be verified against one and the same logic property (expressed in Action-based CTL)

   De Nicola, Vaandrager @ J. ACM, 1995

2. A logic property (expressed in variability-aware ACTL) can directly be verified against the MTS, relying on the fact that under certain syntactic conditions its validity over the MTS implies validity of the same property for all derived products

   ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016

VMC v6.4 is freely usable online: http://fmt.isti.cnr.it/vmc/
Dedicated MTS$\nu$ Variability Model Checker VMC

Input: specification of an MTS$\nu$ in process-algebraic terms together with a set of logical variability constraints

VMC offers product-based and family-based variability analyses:

1. The actual set of all valid product behaviour can explicitly be generated and the resulting LTSs can all be verified against one and the same logic property (expressed in Action-based CTL)
   De Nicola, Vaandrager @ J. ACM, 1995

2. A logic property (expressed in variability-aware ACTL) can directly be verified against the MTS, relying on the fact that under certain syntactic conditions its validity over the MTS implies validity of the same property for all derived products
   ter Beek et al. @ J. Logic Algebr. Meth. Program., 2016

VMC v6.4 is freely usable online: http://fmt.isti.cnr.it/vmc/
Dedicated MTS\(\nu\) Variability Model Checker VMC

Input: specification of an MTS\(\nu\) in process-algebraic terms together with a set of logical variability constraints

VMC offers product-based and family-based variability analyses:

1. The actual set of all valid product behaviour can explicitly be generated and the resulting LTSs can all be verified against one and the same logic property (expressed in Action-based CTL)
   De Nicola, Vaandrager @ *J. ACM*, 1995

2. A logic property (expressed in variability-aware ACTL) can directly be verified against the MTS, relying on the fact that under certain syntactic conditions its validity over the MTS implies validity of the same property for all derived products
   ter Beek et al. @ *J. Logic Algebr. Meth. Program.*, 2016

VMC v6.4 is freely usable online: http://fmt.isti.cnr.it/vmc/
MTS: implementations

[[LICS88]]
MTS: implementations

[LICS88]

\[
\begin{array}{c}
\overset{p}{\text{a}} & \overset{b}{\text{q}} \\
\overset{a}{\text{r}} & \overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]

\[
\begin{array}{c}
\overset{p}{\text{a}} \\
\overset{b}{\text{q}} \\
\overset{a}{\text{r}} \\
\overset{b}{\text{s}} \\
\end{array}
\]
MTS: coherence and consistency [JLAMP16]
MTSν: coherence and consistency

[JLAMP16]
MTS\(\nu\): variants and variability constraints

\[ \Upsilon = \emptyset \]
\[ \Upsilon = \{a \oplus b\} \]

\[ \gamma = \emptyset \]
\[ \gamma = \{a \oplus b\} \]
MTS_\nu: \text{variants and variability constraints} \quad [\text{JLAMP16}]

\[ \Upsilon = \emptyset \]

\[ \Upsilon = \{ a \oplus b \} \]
\[ F = \{ v, b, s, t, f, c \} \]
\[ \Lambda = \{ \{ v, b, s \}, \{ v, b, s, f \}, \{ v, b, s, c \}, \{ v, b, s, f, c \}, \{ v, b, t \}, \{ v, b, t, f \}, \{ v, b, t, c \}, \{ v, b, t, f, c \}, \{ v, b, s, t \}, \{ v, b, s, t, f \}, \{ v, b, s, t, c \}, \{ v, b, s, t, f, c \} \} \]

\[ F \mid _{\lambda} \text{ with } \lambda = \{ v, b, t \}, \text{ i.e. } \lambda(v) = T, \lambda(b) = T, \lambda(s) = \bot, \lambda(t) = T, \lambda(f) = \bot, \lambda(c) = \bot \]
FTS: variants

\[ \text{variants} \quad \text{ICSE10, TSE13, SCP14} \]

\[
\begin{align*}
F &= \{v, b, s, t, f, c\} \\
\Lambda &= \{\{v, b, s\}, \{v, b, s, f\}, \{v, b, s, c\}, \{v, b, s, f, c\}, \{v, b, t\}, \{v, b, t, f\}, \{v, b, t, c\}, \\
&\quad \{v, b, t, f, c\}, \{v, b, s, t\}, \{v, b, s, t, f\}, \{v, b, s, t, c\}, \{v, b, s, t, f, c\}\} \\
\end{align*}
\]

\[
\mathcal{F}_{|\lambda} \text{ with } \lambda = \{v, b, t\}, \text{ i.e. } \lambda(v) = \top, \lambda(b) = \top, \lambda(s) = \bot, \lambda(t) = \top, \lambda(f) = \bot, \lambda(c) = \bot
\]
MTS_ν vs. FTS: variants

\[ \gamma = \{ a \rightarrow c \} \]

\[ \lambda(f_a) = \top, \quad \lambda(f_b) = \bot, \quad \lambda(f_c) = \top \]

\[ \lambda'(f_a) = \top, \quad \lambda'(f_b) = \top, \quad \lambda'(f_c) = \top \]

\[ \lambda''(f_a) = \bot \]
MTS\(\nu\) vs. FTS: variants

\[ p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s \quad \gamma = \{ a \rightarrow c \} \]

\[ p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s \]

\[ p \xrightarrow{a/f_a} q \xrightarrow{b/f_b} r \xrightarrow{c/f_c} s \quad \lambda(f_a) = \top, \lambda(f_b) = \bot, \lambda(f_c) = \top \]

\[ \lambda'(f_a) = \top, \lambda'(f_b) = \top, \lambda'(f_c) = \top \]

\[ \lambda'(f_a) = \bot \]

\[ p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s \]

\[ p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s \]
MTS_\psi \text{ vs. } FTS: \text{ variants}

\begin{align*}
\sim \quad & p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s \\
& \gamma = \{a \rightarrow c\}
\end{align*}

\begin{align*}
\sim \quad & p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s
\end{align*}

\begin{align*}
\sim \quad & p \xrightarrow{a/f_a} q \xrightarrow{b/f_b} r \xrightarrow{c/f_c} s && \lambda(f_a) = \top,
& & \lambda(f_b) = \bot,
& & \lambda(f_c) = \top
\end{align*}

\begin{align*}
\sim \quad & p \xrightarrow{a/f_a} q \xrightarrow{b/f_b} r \xrightarrow{c/f_c} s && \lambda'(f_a) = \top,
& & \lambda'(f_b) = \top,
& & \lambda'(f_c) = \top
\end{align*}

\begin{align*}
\sim \quad & p \xrightarrow{a/f_a} q \xrightarrow{b/f_b} r \xrightarrow{c/f_c} s && \lambda'(f_a) = \bot.
\end{align*}
MTS$_\psi$ vs. FTS: variants

\[ \gamma = \{ a \rightarrow c \} \]

\[ \lambda(f_a) = \top, \quad \lambda(f_b) = \bot, \quad \lambda(f_c) = \top \]

\[ \lambda'(f_a) = \top, \quad \lambda'(f_b) = \top, \quad \lambda'(f_c) = \top \]

\[ \lambda''(f_a) = \bot \]
\[ \Upsilon = \{ \text{pay} \oplus \text{free}, \text{pay} \Leftrightarrow \text{open}, \text{free} \Leftrightarrow \text{takeFree}, \text{soda} \lor \text{tea} \} \]
$\tau = \{\text{pay} \oplus \text{free}, \text{pay} \Leftrightarrow \text{open}, \text{free} \Leftrightarrow \text{takeFree}, \text{soda} \lor \text{tea}\}$
References


Expressiveness

Behavioural SPL formalisms \( M \) and \( M' \):

- \( M' \) is at least as expressive as \( M \), denoted by \( M \leq M' \), iff
  \( \exists \) transformation from \( M \) into \( M' \), denoted by \( \tau: M \rightarrow M' \),
  s.t. \( \forall \) models \( \mathcal{M} \in M \), sets of variants \( \text{lts}(\mathcal{M}) = \text{lts}(\tau(\mathcal{M})) \)

- \( M' \) and \( M \) are equally expressive, denoted by \( M' = M \), if \( M \leq M' \) and \( M' \leq M \)

[SEFM15] FTS \( \leq \) MTS\(\nu \) (finite-state setting)
[SCP16] MTS \( \leq \) PL-LTS (with restricted definition) \( \leq \) FTS
  (with a generalised product-derivation relation)
[SCP18] MTS \( \leq \) PL-LTS (original liberal definition) = FTS
  (with a generalised product-derivation relation)
[JLAMP19] FTS = multiMTS (sets of multiple MTSs)
[SCP19] MTS\(\nu \) = FTS (finite-state setting)
Expressiveness

Behavioural SPL formalisms $M$ and $M'$:

- $M'$ is at least as expressive as $M$, denoted by $M \leq M'$, iff
  \(\exists\) transformation from $M$ into $M'$, denoted by \(\tau : M \rightarrow M'\),
s.t. \(\forall\) models $\mathcal{M} \in M$, sets of variants \(\text{lts}(\mathcal{M}) = \text{lts}(\tau(\mathcal{M}))\)

- $M'$ and $M$ are equally expressive, denoted by $M' = M$, if
  $M \leq M'$ and $M' \leq M$

[SEFM15] FTS $\leq$ MTS$\nu$ (finite-state setting)

[SCP16] MTS $\leq$ PL-LTS (with restricted definition) $\leq$ FTS
        (with a generalised product-derivation relation)

[SCP18] MTS $\leq$ PL-LTS (original liberal definition) = FTS
        (with a generalised product-derivation relation)

[JLAMP19] FTS = multiMTS (sets of multiple MTSs)

[SCP19] MTS$\nu$ = FTS (finite-state setting)
Expressiveness

Behavioural SPL formalisms $M$ and $M'$:

- $M'$ is at least as expressive as $M$, denoted by $M \leq M'$, iff there exists a transformation from $M$ into $M'$, denoted by $\tau : M \rightarrow M'$, s.t. for all models $\mathcal{M} \in M$, sets of variants $\text{lts}(\mathcal{M}) = \text{lts}(\tau(\mathcal{M}))$

- $M'$ and $M$ are equally expressive, denoted by $M' = M$, if $M \leq M'$ and $M' \leq M$

[SEFM15] $\text{FTS} \leq \text{MTS}_\upsilon$ (finite-state setting)
[SCP16] $\text{MTS} \leq \text{PL-LTS}$ (with restricted definition) $\leq \text{FTS}$ (with a generalised product-derivation relation)
[SCP18] $\text{MTS} \leq \text{PL-LTS}$ (original liberal definition) $= \text{FTS}$ (with a generalised product-derivation relation)
[JLAMP19] $\text{FTS} = \text{multiMTS}$ (sets of multiple MTSs)
[SCP19] $\text{MTS}_\upsilon = \text{FTS}$ (finite-state setting)
FTSs

[SCP16, SCP18]
FTSs vs. MTSs

[LiCS88]
FTSs vs. MTS

\[ \gamma = \emptyset \]
\[ \tau_\ast = \{ (\text{pay, } v \land \neg f) \iff (v \land \neg f), (\text{free, } f) \iff f, (\text{change, } v) \iff v, (\text{cancel, } c) \iff c, (\text{return, } c) \iff c, \ldots, (\text{open, } v \land \neg f) \iff (v \land \neg f), (\text{take, } v) \iff v, (\text{close, } v) \iff v \} \]

\[ \cup \{ (v \land b \land \neg s \land t \land \neg f \land \neg c) \lor \cdots \lor (v \land b \land \neg s \land t \land \neg f \land c) \} \]

\[ \text{FTS2MTS}_\upsilon (\text{upto dummy transitions and action relabelling}) \]
FTS2MTS\(\nu\) (upto dummy transitions and action relabelling)

\[\{(pay, v \land \neg f), (free, f), (change, v), (cancel, c), (return, c), (soda, s), (tea, t), (serveSoda, s), (serveTea, t), (take, f), (open, v \land \neg f), (take, v), (close, v), v, b, s, t, f, c\}\]

\[\mathcal{T}_* = \{(pay, v \land \neg f) \Leftrightarrow (v \land \neg f), (free, f) \Leftrightarrow f, (change, v) \Leftrightarrow v, (cancel, c) \Leftrightarrow c, (return, c) \Leftrightarrow c, \ldots, (open, v \land \neg f) \Leftrightarrow (v \land \neg f), (take, v) \Leftrightarrow v, (close, v) \Leftrightarrow v\} \cup \{(v \land b \land \neg s \land t \land \neg f \land \neg c) \lor \cdots \lor (v \land b \land \neg s \land t \land \neg f \land c)\} \]

\[\{(pay, v \land \neg f), (change, v), (tea, t), (serveTea, t), (open, v \land \neg f), (take, v), (close, v), v, b, t\}\]
FTS2MTS$\upsilon$: soundness and completeness

Theorem [SCP19]

Let $\mathcal{F} = (Q, \Sigma, \bar{q}, \delta, F, \Lambda)$ be an FTS and let $\mathcal{M}_\star = (Q_\star, \Sigma_\star, \bar{q}, \delta_\diamond, \delta_\Box, \Upsilon_\star)$ be the MTS$\upsilon$ generated from $\mathcal{F}$ according to the FTS2MTS$\upsilon$ Algorithm.

The sets of variants $\text{lts}(\mathcal{F})$ and $\text{lts}(\mathcal{M}_\star)$ coincide, up to dummy transitions and action relabelling.
Example FTS $\not\preceq$ MTS

\[ \lambda(f) = \top, \quad \lambda(f') = \bot, \quad \lambda'(f') = \top \]

\[ \gamma = \{ f \oplus f', \quad (a, f) \leftrightarrow f, \quad (b, f') \leftrightarrow f' \} \]
Example FTS $\leq$ MTS\textsubscript{$\nu$} [SCP19]

\[
\begin{align*}
\lambda(f) &= \top, \\
\lambda(f') &= \bot,
\end{align*}
\]

\[
\begin{align*}
\lambda'(f) &= \bot, \\
\lambda'(f') &= \top
\end{align*}
\]
MTS\textsubscript{v2FTS} (worst-case: exponential in number of features)

\[ F = \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{takeNotFree}}, f_{\text{close}} \} \]

\[ \Lambda = \{ \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{tea}}, f_{\text{serveTea}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{tea}}, f_{\text{serveTea}}, f_{\text{takeNotFree}}, f_{\text{close}} \}, \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{tea}}, f_{\text{serveTea}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{tea}}, f_{\text{serveTea}}, f_{\text{takeNotFree}}, f_{\text{close}} \}, \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{soda}}, f_{\text{serveSoda}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{soda}}, f_{\text{serveSoda}}, f_{\text{takeNotFree}}, f_{\text{close}} \}, \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{soda}}, f_{\text{serveSoda}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{soda}}, f_{\text{serveSoda}}, f_{\text{takeNotFree}}, f_{\text{close}} \}, \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{takeNotFree}}, f_{\text{close}} \}, \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{return}}, f_{\text{cancel}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{takeNotFree}}, f_{\text{close}} \}, \{ f_{\text{pay}}, f_{\text{change}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{open}}, f_{\text{takeFree}}, f_{\text{close}} \}, \{ f_{\text{free}}, f_{\text{tea}}, f_{\text{soda}}, f_{\text{serveTea}}, f_{\text{serveSoda}}, f_{\text{takeNotFree}}, f_{\text{close}} \} \} \]
Theorem [SCP19]

Let $M = (Q, \Sigma, \bar{q}, \delta^\diamond, \delta^\Box, \Upsilon)$ be an MTS$\nu$ and let $F^\circ = (Q, \Sigma, \bar{q}, \delta^\circ, F^\circ, \Lambda^\circ)$ be the FTS generated from $M$ according to the MTS$\nu$2FTS Algorithm.

The sets of variants $\text{lts}(M)$ and $\text{lts}(F^\circ)$ coincide, i.e. $\text{lts}(M) = \text{lts}(F^\circ)$. 
Conclusion

[SCP19]:
- $\text{MTS}_v = \text{FTS}$ contributes to the expressiveness hierarchy
- $\text{FTS2MTS}_v$ preserves original (compact) branching structure
$\Rightarrow$ family-based model checking of FTSs with VMC

[SPLC19] (best paper):
- Static analysis of FTSs detecting hidden deadlocks
- $\text{FTS2MTS}$ preserving liveness (i.e. no deadlocks)
$\Rightarrow$ family-based model checking FTSs/MTSs

Future journal paper:
$\Rightarrow$ Any valid v-ACTL$\square$ formula preserved by live FTSs
$\Rightarrow$ Any valid LTL formula preserved by live FTSs
Conclusion

[SCP19]:
- $\text{MTS}_\nu = \text{FTS}$ contributes to the expressiveness hierarchy
- $\text{FTS}_2\text{MTS}_\nu$ preserves original (compact) branching structure
  ⇒ family-based model checking of FTSs with VMC

[SPLC19] (best paper):
- Static analysis of FTSs detecting hidden deadlocks
- $\text{FTS}_2\text{MTS}$ preserving liveness (i.e. no deadlocks)
  ⇒ family-based model checking FTSs/MTSs

Future journal paper:
- Any valid $v$-ACTLive formula preserved by live FTSs
- Any valid LTL formula preserved by live FTSs
Conclusion

[SCP19]:
- $\text{MTS}_\nu = \text{FTS}$ contributes to the expressiveness hierarchy
- $\text{FTS2MTS}_\nu$ preserves original (compact) branching structure
  $\implies$ family-based model checking of FTSs with VMC

[SPLC19] (best paper):
- Static analysis of FTSs detecting hidden deadlocks
- $\text{FTS2MTS}$ preserving liveness (i.e. no deadlocks)
  $\implies$ family-based model checking FTSs/MTSs

Future journal paper:
- Any valid $\nu$-ACTLive$^\square$ formula preserved by live FTSs
- Any valid LTL formula preserved by live FTSs
The aim of the FMICS conference series is to provide a forum for researchers who are interested in the development and application of formal methods in industry. FMICS brings together scientists and engineers who are active in the area of formal methods and interested in exchanging their experiences in the industrial usage of these methods. The FMICS conference series also strives to promote research and development for the improvement of formal methods and tools for industrial applications.

FMICS 25:
This 25th edition of FMICS will be celebrated in a special way.

Topics of interest include (but are not limited to):

- Case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions.
- Methods, techniques and tools to support automated analysis, certification, debugging, descriptions, learning, optimisation and transformation of complex, distributed, real-time, embedded, mobile and autonomous systems.
- Verification and validation methods (model checking, theorem proving, SAT/SMT constraint solving, abstract interpretation, etc.) that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues).
- Impact of the adoption of formal methods on the development process and associated costs. Application of formal methods in standardisation and industrial forums.

NEW THIS YEAR:
SPECIAL TRACK on “Formal Methods for Security in IoT”
We invite submissions in topics related to the secure development and security assessment of IoT-based applications using formal methods.

FMICS'25: 25th International Conference on Formal Methods for Industrial Critical Systems
2 - 3 September 2020 | Vienna | Austria

PC Chairs
Maurice ter Beek | ISTI-CNR
Dejan Ničković | AIT

PC Members
Bernhard Aichernig | TU Graz
Jiri Barnat | Masaryk University
Davide Basile | University of Florence & ISTI-CNR
Carlos Budde | University of Twente
Rance Cleaveland | University of Maryland & NSF
Thao Dang | VERIMAG
Michael Dierkes | Rockwell Collins
Georgios Fainekos | ASU
Alessandro Fantechi | University of Florence
Wan Fokkink | VU Amsterdam
Maria Del Mar Gallardo | University of Malaga
Ichiro Hasuo | University of Tokyo
Klaus Havelund | NASA
Thierry Lecomte | CLEARSY
Axel Legay | UCLouvain
Gabriele Lenzini | University of Luxembourg
Alberto Lluch Lafuente | DTU
Florian Lorber | Aalborg University
Tiziana Margaria | University of Limerick & Lero
Radu Mateescu | INRIA
Franco Mazzanti | ISTI-CNR
Stefan Mitsch | CMU
José N. Oliveira | University of Minho & INESC TEC
Jaco van de Pol | Aarhus University
Adam Rogaliewicz | Brno University of Technology
Markus Roggenbach | Swansea University
Matteo Rossi | Polytechnic University of Milan
Stefano Tonetta | FBK
Jan Tretmans | TNO
Andrea Vandin | Sant'Anna School of Adv. Studies
Tim Willemse | TU Eindhoven
Kirsten Winter | University of Queensland
Lijun Zhang | Chinese Academy of Science

Steering Committee
Alessandro Fantechi | University of Florence
Hubert Garavel | INRIA
Stefania Gnesi | ISTI-CNR
Diego Latella | ISTI-CNR
Tiziana Margaria | University of Limerick & Lero
Radu Mateescu | INRIA
Jaco van de Pol | Aarhus University

Important Dates
Abstract submission: May 8, 2020
Paper submission: May 15, 2020
Author notification: July 1, 2020
Camera-ready version: July 15, 2020

FMICS is part of QONFEST 2020
CONCUR | FMICS | FORMATS | QEST
August 31 – September 5, 2020

Keynote Speakers
Thomas Henzinger | IST Austria
Stefan Resch | THALES
Roderick Bloem | TU Graz

CONCUR | QEST | FMICS Speaker
FMICS Speaker
FMICS Speaker
CONCUR | FORMATS | FMICS Speaker