1 Featured Mutants Model

A featured mutant model is a couple: featured transition system (FTS) and feature diagram (FD). A FD represents all the possible valid products of a product line. It is usually (but not always) represented as a boolean expression over a set of variables called feature. A set of variable (set to true) satisfying this boolean expression is a valid product (also called valid configuration) of the product line. The semantic of a FD ($d$) is the set of valid products of the product line and is noted $[d]$. A FTS is (basically) a transition system where transitions have been tagged with the set of products able to execute it. To compactly represent this set, we use feature expressions (boolean expressions over features).

Definition 1 (Featured Transition System (FTS)). A Featured Transition System (FTS) is a tuple $(S, \text{Act}, \text{trans}, \text{init}, d, \gamma)$, where:

- $S$ is a set of states;
- $\text{Act}$ is a set of actions;
- $\text{trans} \subseteq S \times \text{Act} \times S$ is a transition relation such as the FTS is deterministic (with $(s_1, \alpha, s_2) \in \text{trans}$ sometimes noted $s_1 \xrightarrow{\alpha} s_2$);
- $d$ is a FD;
- $\gamma : \text{trans} \mapsto [d] \mapsto \top, \bot$ is a function labelling specifying for each transition which valid product may execute it, this function is represented as a boolean expression over the features of $d$;
- $\text{init} : S \mapsto ([d] \mapsto \top, \bot)$ a total function that indicates if a state $i \in S$ is an initial state for a product $p \in [d]$, which allows one to model mutants that change the initial state of the system.

Definition 2 (Featured Mutant Model (FMM)). A Featured Mutant Model is a couple $(\text{fd}_{fmm}, \text{fts}_{fmm})$ where $\text{fd}_{fmm}$ is a feature diagram representing a set of mutation operators applications and $\text{fts}_{fmm}$ is a FTS where the feature diagram ($d$) is $\text{fd}_{fmm}$.

2 Mutation operators

Definition 3 (Mutation operator). A mutation operator is a function: $\text{op} : \text{ts} \to \text{ts}'$, where $\text{ts}$ and $\text{ts}'$ are transition systems. The operator $\text{op}$ performs a certain model transformation on $\text{ts}$ and produces $\text{ts}'$.

Definition 4 (FMM Mutation operator). A FMM mutation operator is a function: $\text{op}_{fmm} : \text{fmm} \to \text{fmm}'$, where $\text{fmm}$ and $\text{fmm}'$ are FMM'. The operator $\text{op}$ performs a certain model transformation on $\text{fmm}$ and produces a $\text{fmm}'$ representing the modifications made by previously used operators and the modification made by $\text{op}_{fmm}$.

Practically, a FMM operator will modify the FMM's FTS and add a feature in the FMM's FD representing the added transformation. In the following sections, we define mutation operators and FMM mutation operators, inspired from Fabbri et al. [Fabbri et al., 1999].

2.1 State Missing (SMI)

Removes a random state $s_i \neq i$ (except initial state). Let $\text{ts}$ a TS and $(\text{fts}_{fmm}, \text{fd}_{fmm})$ a FMM representing the previous mutations on $\text{ts}$. The result of the application $\text{smi}_{fmm} (\text{fts}_{fmm}, \text{fd}_{fmm})$ is a FMM where $\text{fts}_{fmm}$ has all the feature expression of the input transitions of the state $s_i$ (the state removed in the mutant) conjunct with $\neg \text{smi}$. 


2.2 Wrong Initial State (WIS)
Modifies the start state of the transition system to another random state.

2.3 Action Exchange (AEX)
Modifies an action on a transition and replace it by another action.
2.4 Action Missing (AMI)
Removes the actions from a transition.

\begin{align*}
\text{Input } ts & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{Input } fts_m & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{Output } ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{Output } (fts_m, f_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} : ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{Output } ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{Output } (fts_m, f_{d_{fmm}} \not\models \text{ami}_{s_1}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{Output } ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{Output } (fts_m, f_{d_{fmm}} \not\models \text{ami}_{s_1}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \not\models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} : ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \not\models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \not\models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \not\models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \not\models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\quad & 0 \xrightarrow{a/\gamma_1} s_1 \\
& \\
\text{ami} \not\models ts \rightarrow ts' & \\
\quad & 0 \xrightarrow{a} s_1 \\
& \\
\text{ami}_{fmm} : (fts_m, f_{d_{fmm}}) \rightarrow (fts'_m, f'_{d_{fmm}}) & \\
\end{align*}
2.6 Transition Add (TAD)

Adds a transition to the system by randomly picking up two states and an action. Note: this corresponds to
the event extra operator in [Fabbri et al., 1999]. Adding an action without adding a transition with this action
has no sense since it can not be detected without being fired.

\[
\text{Input } ts \quad \text{Input } (fts_{fmm}, fd_{fmm})
\]

\[
\begin{align*}
tad : ts & \rightarrow ts' \\
tad_{fmm} : (fts_{fmm}, fd_{fmm}) & \rightarrow (fts'_{fmm}, fd'_{fmm})
\end{align*}
\]

\[
\text{Output } ts'\quad \text{Output } (fts_{fmm}, fd_{fmm} \oplus tad_{s0})
\]

2.7 Transition Destination Exchange (TDE)

Changes the destination of a transition to the system by randomly picking up another state in the system.

\[
\begin{align*}
tdc : ts & \rightarrow ts' \\
tdc_{fmm} : (fts_{fmm}, fd_{fmm}) & \rightarrow (fts'_{fmm}, fd'_{fmm})
\end{align*}
\]

\[
\text{Output } ts'\quad \text{Output } (fts_{fmm}, fd_{fmm} \oplus tdc_{s1})
\]

3 Executing test case

Let \((fts_{fmm}, fd_{fmm})\) be a FMM of the original TS \(ts\) and \(tc = (\alpha_1, \alpha_2, ..., \alpha_n)\) be a test case over \(ts\) starting
from \(i\) (the initial state of \(ts\)) and ending in \(i\). The question is which mutants are not killed by \(tc\) when executing
it on \(fts_{fmm}\)? Let us consider:

- \(trs = \{(t_k, ..., t_l), ...\}\) the set possible sequences of transitions fired when executing \(tc\) on \(fts_{fmm}\), where
each sequence ends in \(i_{ts}\), the initial state \(i\) in \(ts\); Sequences in \(trs\) return to the initial state, meaning that the original system and mutants with their feature
expression activated by transitions in \(trs\) may execute \(tc\) (and are not killed by \(trs\)). The set of mutants not
killed by \(tc\) is defined using the feature expression :

\[
\text{alive} = \bigvee_{(t_k, ..., t_l) \in trs} \left( \bigwedge_{t \in \{t_k, ..., t_l\}} \gamma_t \right)
\]

The alive feature expression is conjunct with the feature diagram \(fd_{fmm}\) (representing all the possible
mutants in \(ts\)) to compute the set of mutants not killed by a test case \((tc)\).
References