Correctness of FormlSlicer

FormlSlicer: A Model Slicing Tool to Support Feature-oriented Requirements in Software Product Line

Xiaoni Lai

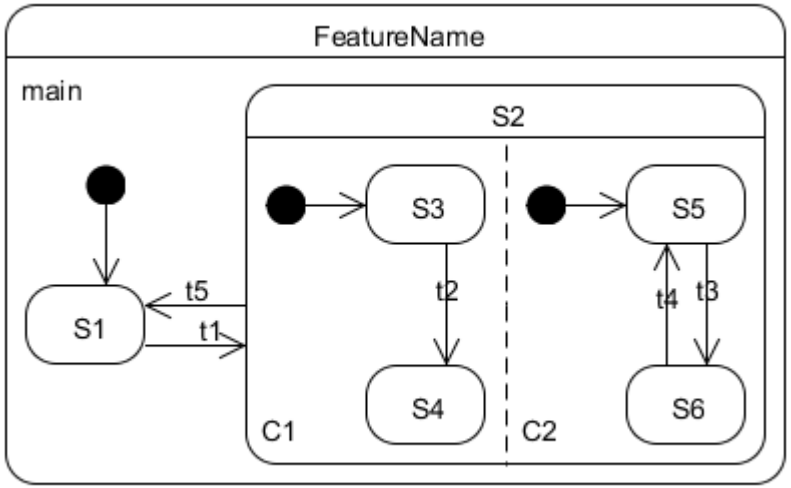
[1]The David R. Cheriton School of Computer Science
University of Waterloo

Part 2 of Master Thesis, 2015

Table of Contents

Table of Contents

The FOSMs in FORML's Behaviour Model



The model consists of many feature modules.

We call each feature module an Feature-oriented State Machine (FOSM).

FormISlicer: the Tool Developed

We have a tool that goes through multiple stages in performing slicing.
Generally, it starts from an empty slice set $\mathcal{L}$ and gradually adds model components into it.

A Comparision between the Original and Sliced Model

FOI = Feature of Interest
FOSM = Feature-oriented State Machine
ROS = Rest of System

The sliced model consists of smaller FOSMs in the ROS compared to the original.

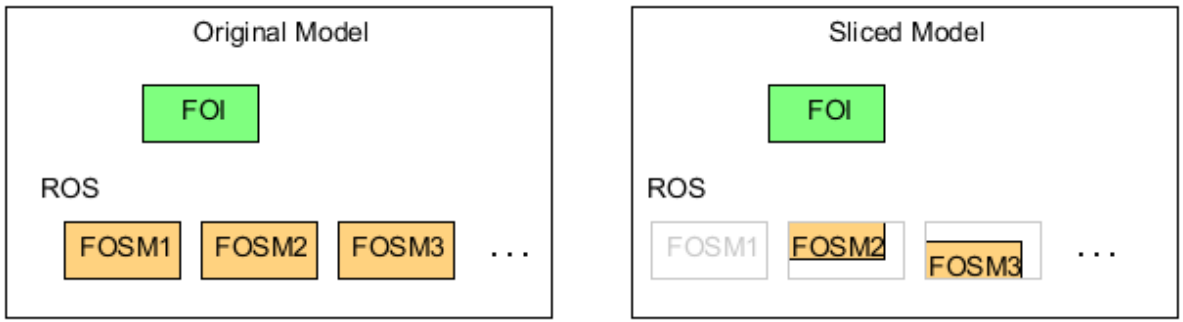
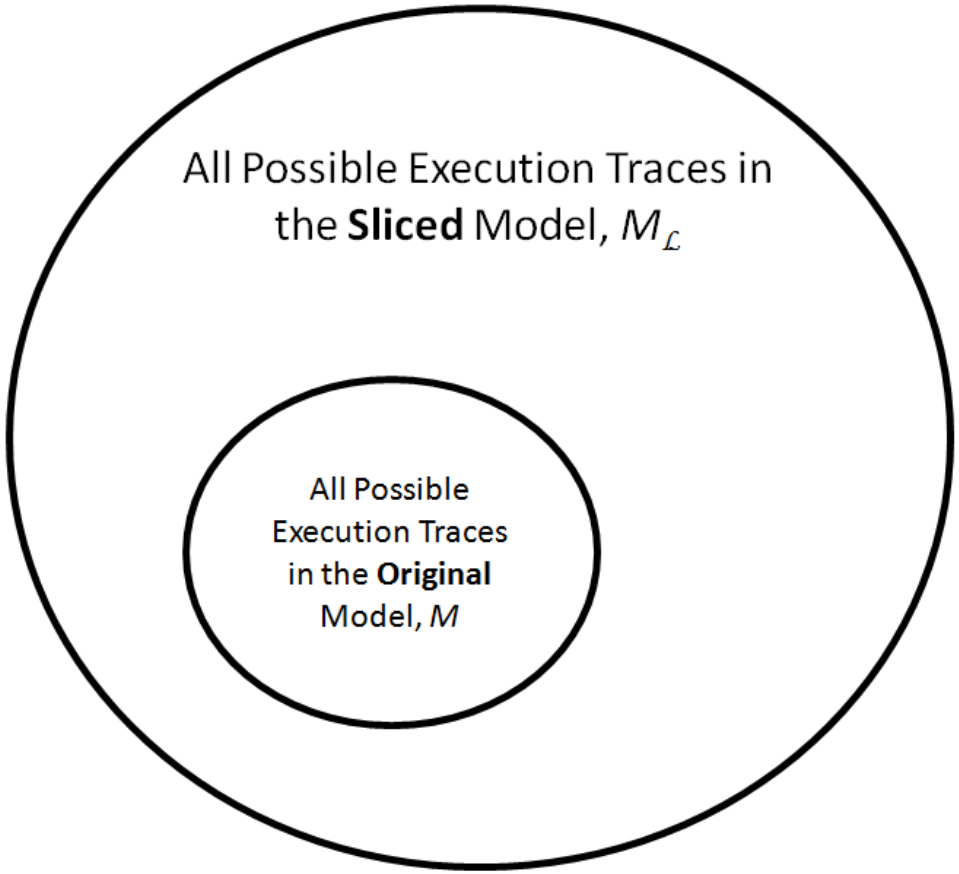


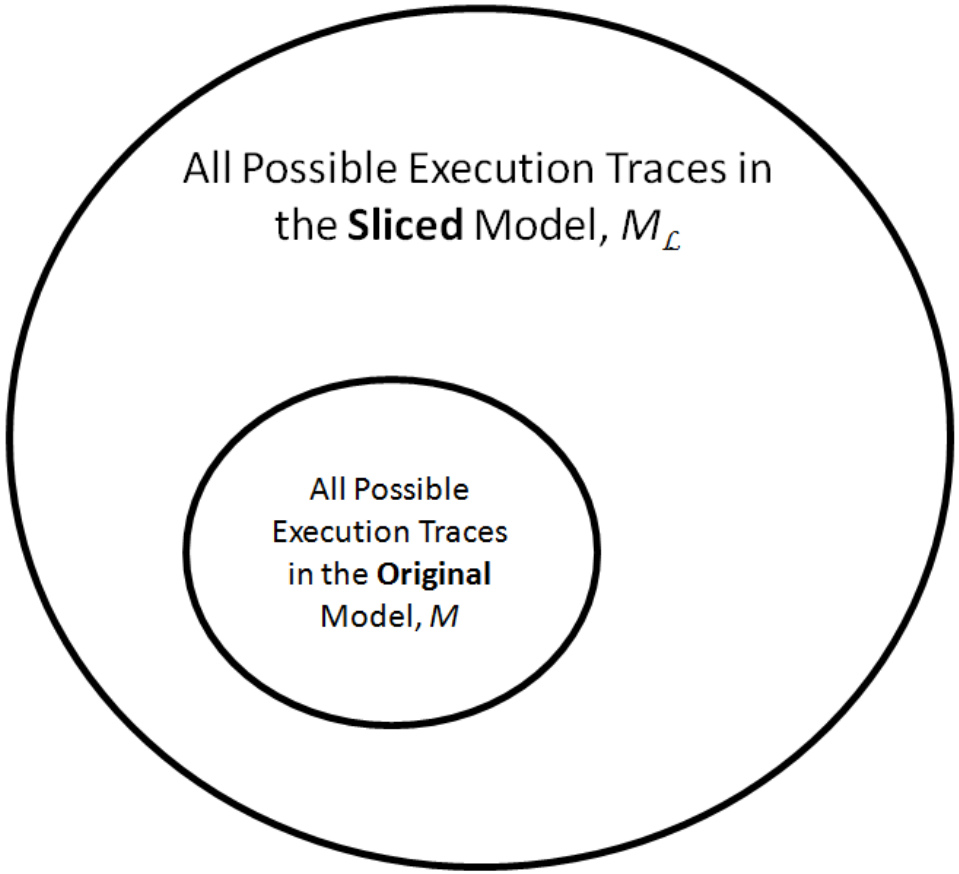
Table of Contents

Purpose of the Proof



$$M_{\mathcal{L}} \models \varphi \Rightarrow M \models \varphi$$

Purpose of the Proof



Any given execution trace in original model can be *simulated* by an execution trace in the sliced model.

Purpose of the Proof

An execution trace in the original model is a long sequence of execution steps:

$$e_0, e_1, e_2, \ldots, e_k$$

Each e in the original model can be projected to an $e_{\mathcal{L}}$ in the sliced model:

$$P(e) = e_{\mathcal{L}}$$

Purpose of the Proof

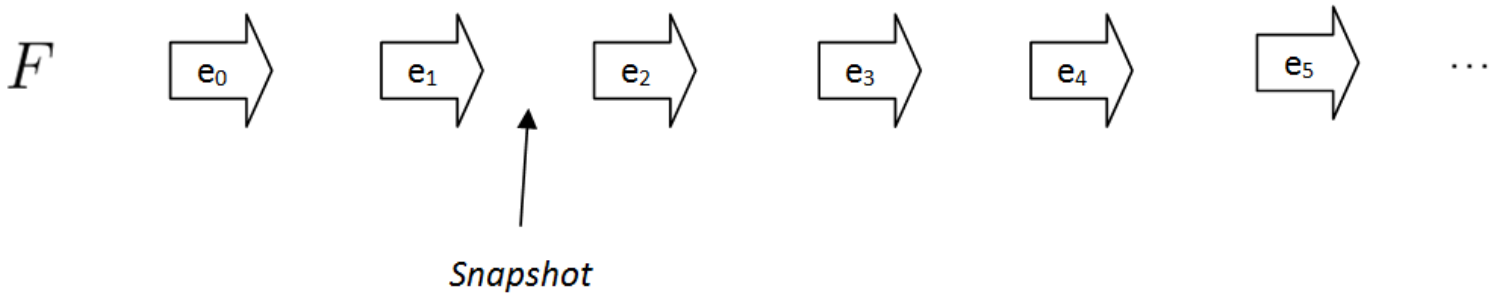
So how can we show this?

$$P(e) = e_{\mathcal{L}}$$

We need to define some semantics first.

Table of Contents

Snapshot



Informally, a **snapshot** is an observable point in a model's execution; it refers to the status of a model between execution steps.

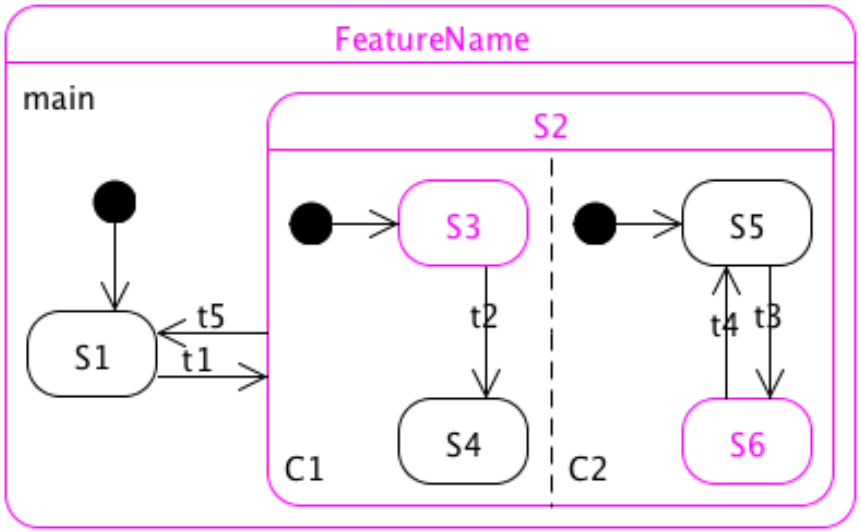
Snapshot

We define the snapshot of an FOSM as a combination of

- the state configuration of the FOSM, N,
- and the interpretation of variables, σ.

What are N and σ?

State Configuration, N



$N =$
$[FeatureName, S2, S3, S6]$.

Interpretation, σ

σ maps variables to their values.

$\sigma(v)$ represent the interpretation of the variable v in the environment.

Snapshot

A snapshot $= (N, \sigma)$.

the current state configuration $+$ the values of all variables

An Execution Step, e

An execution step e changes the current snapshot from (N, σ) evolves to (N', σ').

$$F \vdash e : (N, \sigma) \longrightarrow (N', \sigma')$$

An Execution Step, e

Each execution step involves a set of transitions which occur concurrently, denoted as:

$$e = \begin{matrix} t_1 \\ \vdots \\ t_k \end{matrix}$$

If $k = 1$? we write $e = t$. **Non-concurrent Execution Step**.
If $k > 1$?

Table of Contents

Projection of an Execution Step

In order to show

$$P(e) = e_{\mathcal{L}}$$

We will show

$$P(e) = P\begin{pmatrix} t_1 \\ \vdots \\ t_k \end{pmatrix} = \begin{pmatrix} P(t_1) \\ \vdots \\ P(t_k) \end{pmatrix} = e_{\mathcal{L}}$$

Projection of a Transition

Then we need to show $P(t) = (\epsilon \vee t_{\mathcal{L}})$

A transition in the original model is projected to epsilon or another transition in the sliced model, if:

Before the transition, the snapshot of original model is projected to the snapshot in sliced model;

After the transition, the snapshot of original model is still projected to the snapshot in sliced model.

Projection of Snapshot

How can we know a snapshot of the original model, (N, σ), is projected to a snapshot of the sliced model, $(N_{\mathcal{L}}, \sigma_{\mathcal{L}})$?

Projection of Snapshot

We define $P((N, \sigma)) = ((N_{\mathcal{L}}, \sigma_{\mathcal{L}}))$, when

- there is a relation between their state configurations, N and $N_{\mathcal{L}}$;
- $\sigma(v) = \sigma_{\mathcal{L}}(v)$ for a relevant variable v.

Projection of Snapshot

We define $P((N, \sigma)) = ((N_{\mathcal{L}}, \sigma_{\mathcal{L}}))$, when

- there is a relation between their state configurations, N and $N_{\mathcal{L}}$; <== Use State Transition Rule to Prove!
- $\sigma(v) = \sigma_{\mathcal{L}}(v)$ for a relevant variable v.<== Only relevant variables are concerned!

State Transition Rule

The state transition rule defines how the current state configuration N evolves to the next state configuration N' through an execution step.

$$N' = (N - exited(t)) \cup entered(t)$$

N and N are the current and next state configuration of the model.

State Transition Rule

$$N' = (N - exited(t)) \cup entered(t)$$

$exited(t)$:

- the source state itself, $ss(t)$;
- the ancestor states of $ss(t)$ up along the tree of state hierarchy before reaching the least common ancestor with $ds(t)$;
- the descendant states of $ss(t)$.

State Transition Rule

$$N' = (N - exited(t)) \cup entered(t)$$

$entered(t)$:

- the destination state itself, $ds(t)$;
- the ancestor states of $ds(t)$ up along the tree of state hierarchy before reaching the least common ancestor with $ss(t)$;
- the recursively identified default start states of $ds(t)$ and its entered descendant states.

Relevant Variables

Relevant variables are a set of variables selected by the first two steps of slicing in FormISlicer. They are monitored, or transitively monitored by the Feature of Interest (FOI).

Roadmap So Far...

Projection of an execution step

- <== Projection of a transition inside the execution step
- <== Projection of a snapshot before and after the transition
- <== State configuration and interpretation of the snapshot
  - <== state configuration: using state transition rule and steps in FormISlicer
  - <== interpretation: using relevant variables and data dependency of FormISlicer

Projection of a Transition

We will divide into 7 cases to show $P(t) = (\epsilon \vee t_{\mathcal{L}})$.

$$P(t) = \begin{cases} t_{true} & \text{if } (t \notin \mathcal{L}) \wedge (\neg SameParent(t)) \quad (1) \\ \epsilon & \text{if } (ss(t) \in \mathcal{L}) \wedge (ds(t), t \notin \mathcal{L}) \wedge (SameParent(t)) \quad (2) \\ \epsilon & \text{if } (ss(t), ds(t), t) \notin \mathcal{L} \wedge (SameParent(t)) \quad (3) \\ t_{true} & \text{if } (ss(t), t \notin \mathcal{L}) \wedge (ds(t) \in \mathcal{L}) \wedge (SameParent(t)) \quad (4) \\ t_{true} & \text{if } (ss(t), ds(t) \in \mathcal{L}) \wedge (t \notin \mathcal{L})) \wedge (SameParent(t)) \quad (5) \\ \epsilon & \text{if } (n_{merged} = (ss(t) \vee ds(t)) \in \mathcal{L} \quad (6) \\ t & \text{if otherwise} \quad (7) \end{cases}$$

Projection of a Transition

The proof for each case (i) from (1) to (7) will be:

- Because of Step X in FormISlicer, $entered(t)$ will appear in the state configuration of the sliced model...
- Because of Data Dependency in FormISlicer, $\sigma(v)$ will change value in the same fashion in the sliced model...

$=>$ the snapshot in original model is still projected to the snapshot in sliced model
$=>$ $P(t) = (\epsilon \vee t_{\mathcal{L}})$ for Case (i)

Projection of an Execution Step

$$P(e) = P\begin{pmatrix} t_1 \\ \vdots \\ t_k \end{pmatrix} = \begin{pmatrix} P(t_1) \\ \vdots \\ P(t_k) \end{pmatrix} = e_{\mathcal{L}}$$

Note that $P(t) = (\epsilon \vee t_{\mathcal{L}})$.

Induction

Base Case:
The initial snapshot in original model can be projected to the initial snapshot in the sliced model.

Inductive Case:
$P(e) = e_{\mathcal{L}}$, such that snapshot in original remains projected to the snapshot in the sliced model.

Conclusion:
The execution trace in original model can be *simulated* by an execution trace in sliced model.


THANK YOU