SS 2017
Software Verification
Finite State Machines / Semantics / Equivalence notions

Prof. Dr. Holger Schlingloff $^{1,2}$
Dr. Esteban Pavese $^1$ 

$^1$ Institut für Informatik der Humboldt Universität
$^2$ Fraunhofer Institut für offene Kommunikationssysteme FOKUS
Quick recap

- What is a finite state machine?
- FSM vs. Kripke models/structures?
  - Can we distinguish between different Kripke structures with ML?
- What are the semantics of FSMs?
  - When are two FSMs equal?
- What is the parallel composition of FSMs?
- How about FSMs with variables?
Parallelism vs. Concurrency

- Concurrency entails *logically simultaneous behaviour*. Only one processing unit is necessary, and interleaving gives the illusion of simultaneity.

- Parallelism entails *actual, physical simultaneous behaviour*. It requires more than one physical processing unit.
Parallelism vs. Concurrency

- Both concurrency and parallelism require controlled access to shared resources.
- Both suffer from derived problems such as deadlock, livelock, etc.
- FSMs can in essence model both. The difference is evident in the synchronisation semantics.
  - In this course, we will focus on concurrency. We will use the term parallel to mean concurrent.
FSM composition

Recall the definition of FSM composition
FSM composition

Interleaving on non-shared labels

ITCH  CONVERSE

ITCH II CONVERSE

ITCH

CONVERSE

(x1, x2, x3, x4, x5)
FSM composition

Synchronisation on shared labels

What happened to pairs (0,1), (0,2), (1,0), (2,1), (2,2)?
Whenever two (or more) processes share an action label, this shared label is said to be **synchronising**. These synchronising actions allow for **communicating processes**.

Non-shared actions can be interleaved freely. Shared actions are taken by all participating processes **at the same time**.

Synchronising actions not only model concurrent communication. They also **restrict global behaviour** of components.
How is concurrency modelled by FSMs?
Through non-shared interleaving and shared action synchronisation.

How is the individual process’ speed modelled?
It is **not**! Processes are assumed to run as fast as they want/can.

What is the result of this abstraction?
An **asynchronous**, scheduler-independent process model.
Parallel composition properties

Is the parallel composition...

- Commutative? i.e. $A \parallel B = B \parallel A$ ?
- Associative? i.e. $(A \parallel B) \parallel C = A \parallel (B \parallel C)$ ?
- Idempotent? i.e. $A \parallel A = A$ ?
Parallel composition properties

Is the parallel composition...

\[ A \parallel A = A? \]

\[ \times \text{Idempotent? i.e. } A \parallel A = A? \]

H. Schlingloff, E. Pavese SS2017: Software-Verifikation
A state is **non-deterministic** if it contains more than one outgoing transition with **the same action label**.

*Is there an *equivalent*, but deterministic, FSM?*
Do we *need* non-determinism?

Can we find an *equivalent*, but deterministic model?

We already did say that DFA and NFA were equivalent to FSM, right?
Do we \textit{need} non-determinism?

Can we find an \textit{equivalent}, but deterministic model in this case as well?
Do we *need* non-determinism?

So in turns out we can... sort of
What *is* equivalence anyway?

From the examples, we can see that *trace equivalence* is too weak.

Are we *really* modelling the same process?
What *is* equivalence anyway?

What about graph isomorphism?
What *is* equivalence anyway?

What about graph isomorphism?

Even considering only the *reachable* portion, it seems too strict
What *is* equivalence anyway?

Recall how we *could not* distinguish between some Kripke structures using modal logic. This will be the core idea of model equivalence.

The model equivalence notion should:
- Actually be an equivalence relation (reflexive, symmetric, transitive)
- Abstract away from graph structure or trace sets
- Be decidable
- Be invariant w.r.t. both modal logics and parallel composition
The imitation game
Simulation

Given two models and a state within each model $\mathcal{M}, s$ and $\mathcal{N}, t$, we say $\mathcal{M}, s$ simulates $\mathcal{N}, t$ (noted $\mathcal{M}, s \sqsubseteq \mathcal{N}, t$) iff there exists a relation $S \subseteq \mathcal{W}_\mathcal{N} \times \mathcal{W}_\mathcal{M}$ such that

- $tSs$
- Every time that $tSs$, then $V_\mathcal{N}(t) = V_\mathcal{M}(s)$ (invariance)
- Every time that $tR_\mathcal{N} t'$, then there exists $s' \in \mathcal{W}_\mathcal{M}$ such that $t'Ss'$ and $sR_\mathcal{M} s'$ (zig)

In other words, a model simulates another one if it can always imitate its behavior.
Simulation

\[ \text{toss} \]
\{heads, tails\}

\[ = \]
\[ \square \]

\[ \text{toss} \]
\[ \text{heads} \]
\[ \text{tails} \]
Ejercicio 7
Sean \( M \) (izquierda) y \( M_0 \) (derecha) las siguientes FSMs:

1. Muestre que \( M \) simula a \( M_0 \).
2. Muestre que \( M_0 \) simula a \( M \).
3. Entonces. . . vale que \( M_0 \) y \( M \), pero. . . ¿son \( M \) y \( M_0 \) bisimilares?\ Je t i f i q u e .

Ejercicio 8
Sean las siguientes dos FSMs:
Muestre que son bisimilares.
Note que la FSM de la derecha resulta de "desovillar" (unwind) un avance de la FSM de la izquierda. Muestre que cualquier ciclo unwind una cantidad finita de veces es bisimilar al ciclo original.

Ejercicio 9
Sean las siguientes dos FSMs:
Muestre que son bisimilares.
La FSM de la derecha es el resultado de duplicar exactamente sub-FSMs de la derecha. Muestre que cualquier duplicación finita es bisimilar a la máquina original.

Ejercicio 10
Supongamos que se quiere modelar el siguiente comportamiento del controlador de un horno microondas: el horno microondas posee una puerta. El horno sólo puede ponerse en marcha si la puerta está cerrada (el controlador puede detectar si la puerta está abierta o cerrada). Si se abre la puerta estando el horno prendido, el horno se apaga automáticamente. Por otra parte, en cualquier momento es posible establecer el modo de cocción. Los modos de cocción posibles son descongelar, calentar y grill.
Bisimulation

Given two models and a state within each model $\mathcal{M}, s$ and $\mathcal{N}, t$, we say $\mathcal{M}, s$ is bisimilar to $\mathcal{N}, t$ (noted $\mathcal{M}, s \sim \mathcal{N}, t$) iff there exists a relation $S \subseteq W_N \times W_M$ such that

- $tSs$
- Every time that $tSs$, then $V_N(t) = V_M(s)$ (invariance)
- Every time that $tR_N t'$, then there exists $s' \in W_M$ such that $t'Ss'$ and $sR_M s'$ (zig)
- Every time that $sR_M s'$, then there exists $t' \in W_N$ such that $t'Ss'$ and $tR_N t'$ (zag)

The idea is similar to mutual simulation, but where the simulation relation is the same in both cases.
Bisimulation properties

- Bisimulation induces an equivalence partition
  - It is reflexive $A \sim A$
  - It is symmetric $A \sim B \rightarrow B \sim A$
  - It is transitive $A \sim B \land B \sim C \rightarrow A \sim C$
- Bisimulation is invariant wrt composition
  - $A \sim B \rightarrow A \parallel C \sim B \parallel C$
  - And the converse?
- Bisimulation is invariant wrt modal logic
  - $A \sim B \land A \vDash \phi \rightarrow B \vDash \phi$
  - What about the converse?
- Theorem: FOL + bisimulation induces modal logic
Bisimulation as a game

The relational definition can be rather abstract. The *game theoretical* definition of bisimulation is also well known.

- Let $P,s$ and $Q,t$ be two models and current states we want to check for bisimilarity.

- The game consists of an *attacker* and a *defender*.

- The attacker starts at $P$, and the defender at $Q$.

- $((P,s),(Q,t))$ is then the current configuration, and in each round the players try to change the configuration.
Bisimulation as a game

The relational definition can be rather abstract. The *game theoretical* definition of bisimulation is also well known:

- Intuitively, the attacker wants to disprove bisimilarity, and the defender wants to prove it.

- In each round, the attacker chooses a process in the current configuration and makes a transition.

- The defender must mimic the transition in the other process:
  - If the defender cannot move, he loses.
  - If the defender has a strategy to make the game infinite, he wins.

- In reality, it is easier for the defender to win by just showing the bisimulation relation.