Course work

- Weekly homeworks

- Final exam (3:30pm-6:30pm on Friday, June 6)

No collaboration on homeworks & exam (but welcome otherwise).

No late homeworks.

Textbooks

Manna & Pnueli Springer


Copies of lecture slides.

Papers.
Textbook Overview
(Volume II)

Chapter 0: Preliminary Concepts
[Summary of volume I]

Chapter 1: Invariance: Proof Methods

Chapter 2: Invariance: Applications

Chapter 3: Precedence

[Chapter 4: General Safety]

Chapter 5: Algorithmic Verification
(“Model Checking”)

Extra:
• ω-automata
• branching time logic CTL; BDDs

Transformational Systems

Observable only at the beginning and the end of their execution (“black box”)

\[
\text{input} \rightarrow \text{system} \rightarrow \text{output}
\]

with no interaction with the environment.

• specified by

input-output relations
\[\downarrow\]
state formulas (assertions)
First-Order Logic

• typically

terminating sequential programs
e.g., input \(x \geq 0\) → output \(z = \sqrt{x}\)

Reactive Systems

Observable throughout their execution (“black cactus”)

\[
\downarrow \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow
\]

system

\[
\downarrow \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow
\]

environment

| → time

Interaction with the environment

• specified by

their on-going behaviors
(histories of interactions with their environment)
\[\downarrow\]
sequence formulas
Temporal Logic

• Typically

– Airline reservation systems
– Operating systems
– Process control programs
– Communication networks
Overview of the Verification Process

The Components

- **System Description Language**
  SPL (Simple Programming Language)
  Pascal-like high-level language with constructs for
  - concurrency
  - nondeterminism
  - synchronous/asynchronous communication

- **Computational Model**
  FTS (Fair Transition System)
  Compact first-order representation of all sequences of states that can be generated by a system

The Components (cont.)

- **Specification Language**
  TL (temporal logic)
  models of a TL formula are infinite sequences of states

- **Verification Techniques**
  - algorithmic (model checking)
    search a state-graph for counterexample
  - deductive (theorem proving)
    prove first-order verification conditions

Reactive System

\[ \text{SPL Program } P \downarrow \]
\[ \text{Fair Transition System (FTS) } \Phi \downarrow \]
\[ \text{Verification} \]

| Proof
| Com(\(\Phi\)) \(\subseteq\) Mod(\(\psi\)) |
| i.e., all computations of \(\Phi\) are models of \(\psi\) |

| Counterexample
| computation \(\sigma\) of \(\Phi\), s.t. \(\sigma \not\in\) Mod(\(\psi\)) |
States

• vocabulary $V$ — set of typed variables
  (type defines the domain over which the values can range)
  - expression over $V$  $x + y$
  - assertion over $V$  $x > y$

• states $s$ — interpretation over $V$

Example:

$V = \{ x, y : \text{integer} \}$
$s = \{ x : 2, y : 3 \}$
(also written as $s[x] = 2, s[y] = 3$)
$x + y$ is 5 on $s$
$x > y$ false on $s$

• $\Sigma$ — set of all states

Fair Transition System (FTS)

$\Phi = \langle V, \Theta, T, J, C \rangle$
(represents a Reactive Program)

• $V = \{ u_1, \ldots , u_n \} \subseteq V$ — vocabulary
  A finite set of system variables
  System variables = data variables + control variables

• $\Theta$ — initial condition
  First-order assertion over $V$ that characterizes all initial states

Example:

$\Theta : x = 5 \land 3 \leq y \leq 5$
initial states: $\{ x : 5, y : 3 \}$
$\{ x : 5, y : 4 \}$
$\{ x : 5, y : 5 \}$

$T$ — finite set of transitions

For each $\tau \in T$,
$\tau : \Sigma \rightarrow 2^\Sigma$
($\tau$ is a function from states to sets of states)
- $s'$ is a $\tau$-successor of $s$ if $s' \in \tau(s)$
- $\tau$ is represented by the transition relation
  ("next-state" relation) $\rho_{\tau}(V, V')$ where
  $V$ — values of variables in the current state
  $V'$ — values of variables in the next state

Example:

$\rho_{\tau} : x' = x + 1$ means
$s'[x] = s[x] + 1$
- special idling (stuttering) transition $\tau_I$,
  $\rho_{\tau_I} : V = V'$

Example:
\( \langle x : 5, y : 3 \rangle \xrightarrow{\tau} \{ \langle x : 5, y : 4 \rangle, \langle x : 5, y : 5 \rangle \} \)

"When in state \( \langle x : 5, y : 3 \rangle \) \( \tau \) may increment \( y \) by either 1 or 2, and keep \( x \) unchanged."

\( \langle x : 5, y : 4 \rangle \) and \( \langle x : 5, y : 5 \rangle \) are \( \tau \)-successors of \( \langle x : 5, y : 3 \rangle \).

\[\begin{align*}
\mathcal{J} \subseteq T : & \text{ set of just (weakly fair) transitions} \\
\mathcal{C} \subseteq T : & \text{ set of compassionate (strongly fair) transitions}
\end{align*}\]

\( J \subseteq T \) : set of just transitions
\( C \subseteq T \) : set of compassionate transitions

\[\begin{align*}
\text{Enabled/Disabled/Taken Transition}
\end{align*}\]

- For each \( \tau \in T \),
  \( \tau \) is enabled on \( s \) if \( \tau(s) \neq \emptyset \)
  \( \tau \) is disabled on \( s \) if \( \tau(s) = \emptyset \)

- For an infinite sequence of states
  \( \sigma : s_0, s_1, s_2, \ldots, s_k, s_{k+1}, \ldots \)
  - \( \tau \in T \) is enabled at position \( k \) of \( \sigma \)
    if \( \tau \) is enabled on \( s_k \)
  - \( \tau \in T \) is taken at position \( k \) of \( \sigma \)
    if \( s_{k+1} \) is a \( \tau \)-successor of \( s_k \)

Example:
\( \rho \tau : x = 5 \land x' = x + 1 \land y' = y \)

\( \tau \) is enabled on all states s.t. \( s[x] = 5 \)
and disabled on all other states

\[\begin{align*}
\sigma : \ldots, \langle x : 5, y : 3 \rangle, \langle x : 6, y : 3 \rangle \ldots \\
\tau \text{ is enabled at position } k \\
\tau \text{ is taken at position } k
\end{align*}\]

\[\begin{align*}
\text{Computation}
\end{align*}\]

Infinite sequence of states
\( \sigma : s_0, s_1, s_2, \ldots \)
is a computation of an FTS \( \Phi \) (\( \Phi \)-computation), if it satisfies the following:

- Initiality: \( s_0 \) is an initial state (satisfies \( \Theta \))
- Consecution: For each \( i = 0, 1, \ldots \),
  \( s_{i+1} \in \tau(s_i) \) for some \( \tau \in T \).
• **Justice**: For each \( \tau \in J \), it is **not** the case that \( \tau \) is continually enabled beyond some position \( j \) in \( \sigma \) but not taken beyond \( j \).

**Example:**

\[
V : \{ x : \text{integer} \} \\
\Theta : x = 0 \\
T : \{ \tau_I, \tau_{\text{inc}} \} \text{ with } \rho_{\tau_{\text{inc}}} : x' = x + 1 \\
J : \{ \tau_{\text{inc}} \} \\
C : \emptyset
\]

\[\sigma : \langle x : 0 \rangle \xrightarrow{\tau_I} \langle x : 0 \rangle \xrightarrow{\tau_I} \langle x : 0 \rangle \xrightarrow{\tau_I} \ldots\]

satisfies Initiality and Consecution, but not Justice. 
Therefore \( \sigma \) is not a computation.

(In any computation of this system, \( x \) grows beyond any bound.)

• **Compassion**: For each \( \tau \in C \), it is **not** the case that \( \tau \) is enabled at infinitely many positions in \( \sigma \), but taken at only finitely many positions in \( \sigma \).

**Example:**

\[
V : \{ x, y : \text{integer} \} \\
\Theta : x = 0 \land y = 0 \\
T : \{ \tau_I, \tau_x, \tau_y \} \text{ with } \rho_{\tau_x} : x' = x + 1 \mod 2 \\
\rho_{\tau_y} : x = 1 \land y' = y + 1 \\
J : \{ \tau_{\text{inc}} \} \\
C : \{ \tau_y \}
\]

\[\sigma : \langle x : 0, y : 0 \rangle \xrightarrow{\tau_x} \langle 1, 0 \rangle \xrightarrow{\tau_x} \langle 0, 0 \rangle \xrightarrow{\tau_x} \ldots\]

is not a computation: \( \tau_y \) is infinitely often enabled, but never taken. 
(Note: If \( \tau_y \) had only been just, \( \sigma \) would have been a computation, since \( \tau_y \) is not continually enabled.)

FTS \( \Phi = \langle V, \Theta, T, J, C \rangle \)

**Run** = Initiality + Consecution

**Fairness** = Justice + Compassion

**Computation** = Run + Fairness

Notation: \( s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s_2 \xrightarrow{\tau_3} s_3 \rightarrow \ldots \)

**Note:** For every two consecutive states \( s_i, s_{i+1} \), there may be more than one transition that leads from \( s_i \) to \( s_{i+1} \). 
Therefore, several different transitions can be considered as taken at the same time.
Finite-State

- For a computation $\sigma$ of $\Phi$

  $$\sigma : s_0, s_1, s_2, \ldots, s_i, \ldots,$$

  state $s_i$ is a $\Phi$-accessible state.

- $\Phi$ is finite-state if the set of $\Phi$-accessible states is finite. Otherwise, it is infinite-state.
  - If the domain of all variables of $\Phi$ is finite, (e.g., booleans, subranges, etc.), then $\Phi$ is finite-state.
  - Even if the domain of some variables of $\Phi$ is infinite (e.g., integer), $\Phi$ may still be finite-state.

**Example:**

$V : \{x : \text{integer}\}$

$\Theta : x = 1$

$T : \{\tau_1, \tau_1, \tau_2\}$ with

$\rho_{\tau_1} : x = 1 \land x' = 2$

$\rho_{\tau_2} : x = 2 \land x' = 1$

$J, C : \emptyset$

has 2 accessible states:

$\langle x : 1 \rangle$ and $\langle x : 2 \rangle$