Systems and Software Verification

Chapter 5. Timed Automata
5. Timed Automata

• “Temporal”
  – “Trigger the alarm action upon detecting of a problem”

• “Real-Time”
  – “Trigger the alarm less than 5 seconds after detecting a problem”

• Timed Automata
  – An answer to this “real-time” needs

• Organization of chapter 5
  – Description of a Timed Automata
  – Networks of Timed Automata and Synchronization
  – Variants and Extensions of the Basic Model
  – Timed Temporal Logic
  – Timed Model Checking
5.1 Description of Timed Automata

- Two fundamental elements of timed automata
  1. A finite automaton (assumed instantaneous between states)
  2. Clocks

- An example

  c ≥ 5, ?msg, c := 0

  - , ?msg, c := 0
  c < 5 , ?msg, -
• Clocks and transitions
  – Clocks
    • Variables having non-negative real values in $R$
    • All clocks are null in the initial system states
    • All clocks evolve at the same speed, synchronously with time
  – Transitions
    • Three items
      • A guard
      • An action (label)
      • Reset of some clocks
  – The system operates as if equipped with
    • A global clock
    • Many individual clocks (each is synchronized with the global clock)
• Configurations and executions
  – Configuration of the system
    • \((q, v)\)
    • \(q\) : a current control state of the automaton
    • \(v\) : the value of each clock

  • We also refer to \(v\) as a valuation of the automaton clocks.
  • Time automata does not fix the time unit under consideration

  – Execution of the system
    • (usually infinite) sequence of configurations
    • A mapping \(\rho\) from \(\mathbb{R}\) to the set of configuration

  • Configurations change in two ways
    – Delay transition
    – Discrete transition (or action transition)

\[(\text{init, } 0) \rightarrow (\text{init, } 10.2) \rightarrow (\text{verify, } 0) \rightarrow (\text{verify, } 5.8) \rightarrow (\text{verify, } 0) \rightarrow (\text{verify, } 3.1) \rightarrow (\text{alarm, } 3.1) \rightarrow ...\]

  • Trajectory
    – \(\rho(0)\) : the initial state
    – \(\rho(12.3) = (\text{verify, } 2.1)\)
5.2 Networks of Timed Automata and Synchronization

- It is useful to build a timed model in a composite fashion,
  - by combining several parallel automata synchronized with one another
  - → a timed automata network

- Executions of a timed automata network
  - All automata components run in parallel at the same speed
  - Their clocks are all synchronized to the same global clock

  - \((q, v)\): a network configuration
    - \(q\): a control state vector
    - \(v\): a function associating with each network clock its value at the current time

- Synchronization
  - Timed automata synchronize on transitions (as usually) by resetting the clocks
  - The clocks which were not reset are unchanged
  - No concurrent write conflicts on clocks, since reset writes a zero value and nothing else
• Example: modeling a railroad crossing
5.3 Variants and Extensions of the Basic Models

- Many variants, and three extensions

1. Invariants
   - Liveness hypothesis in the untimed model
   - Invariant: a state’s condition on the clock values, which **must always hold** in the state
   - Example: near (invariant: $H_t < 5$), on (invariant: $H_t < 2$), lower/raise (invariant: $H_b < 2$)

2. Urgency
   - Used when cannot tolerate a time delay
   - Represented in the system configurations, not in the transitions
   - Allowing urgent/synchronized behaviors in a more natural way

3. Hybrid linear system
   - Models dynamic variables (in a form of differential equations)
   - HYTECH
5.4 Timed Temporal Logic

• Given a system described as a network of timed automata,
• We wish to be able to state/verify properties of this system
  – Temporal properties
    • “When the train is inside the crossing, the gate is always closed.”
  – Real-time properties
    • “The train always triggers an Exit signal within 7 minutes of having emitted an App signal.”

• Three ways to formally state real-time properties
  1. Express it in terms of the reachability of some sets of configurations
  2. Use observer automata in PLTL model checking
     • Given a property $\phi$, a network $R$
     • Testing reachability of some states in the product $R \parallel A\phi$
     • UPPAAL, HYTECH
  3. Use a timed logic
     • TCTL (Timed CTL)
     • Etc.
• **TCTL (Timed CTL)**

  • $\Phi, \Psi :: = P_1 \mid P_2 \mid ...$ (atomic proposition)
  
  • $\neg \Phi \mid \Phi \land \Psi \mid \Phi \Rightarrow \Psi \mid ...$ (boolean combinators)

  • $\text{EF}_{(-k)} \Phi \mid \text{EG}_{(-k)} \Phi \mid \text{E} \Phi \text{U}_{(-k)} \Psi$ (temporal combinators)

  • $\text{AF}_{(-k)} \Phi \mid \text{AG}_{(-k)} \Phi \mid \text{A} \Phi \text{U}_{(-k)} \Psi$ (path quantifiers)

  • $\sim$ : any comparison symbol from $\{<, \leq, =, \geq, >\}$

  • $k$ : any rational number from $Q$ (real number)

  • Operator X does not exist in TCTL

  • **Example:**

    • $\text{AG} (pb \Rightarrow \text{AG}_{(\leq 5)} \text{alarm})$
      • “If a problem occurs, then the alarm will sound immediately and it will sound for at least 5 time units.”

    • $\text{AG} (\neg \text{far} \Rightarrow \text{AF}_{(<7)} \text{far})$
      • “When the train is located in the railway section between the two sensors App and Exit, it will leave this section before 7 time units.”
5.5 Timed Model Checking

- With timed automata and TCTL logic
- We wish to obtain a model checking algorithm for them.

- Difficulties: Automaton has an infinite number of configurations, since
  1. Clock values are unbounded
  2. The set of real numbers used in clocks is dense

→ Overcome it with the equivalence classes, called "regions"

- Example: $x_1, x_2 \sim k$ with $k = 0, 1, 2$
• Complexity

- Model checking algorithms are complicated.
- The number of regions grows exponentially.

- $O(n!M^n)$
  - $n$: number of clocks
  - $M$: upper bounds of every constant

- No general and efficient method is likely to exist. (vs. linear complexity in CTL)
- PSPACE-complete problem

- Existing tools focus on defining adequate data structures for handing sets of regions → "zones"

- Existing tools have been successfully used
  - HYTECH
  - KRONOS
  - UPPAAL
Conclusion of Part I

• Model checking is a verification technique

• It consists of three steps:
  1. Representation of a program or a system by an automaton
  2. Representation of a property by a logical formula
  3. Model checking algorithm

• Model checking is a powerful but restricted tool:
  – Powerfulness: exhaustive and automatic verification
  – Limitation: due to complexity barriers
  – In practice, the size of system is indeed the main obstacle yet to overcome.

• Model checker users are forced to simplify the model under analysis, until it is manageable. (Abstraction)