




Algorithmic Verification

Comp4151
Lecture 11-A
Ansgar Fehnker

Comp4151 Ansgar Fehnker




Overview

Model Checking Approaches

- Explicit State Model Checking
- Symbolic Model Checking
- Bounded Model Checking
- Automatic Abstraction Refinement

- Correctness of software, hardware and protocols
- Correctness for finite state systems

Comp4151 Ansgar Fehnker



Overview

Time critical systems


Correctness of embedded and distributed systems

- correctness depends result of a computation
- **and** on the timing of events, computations, responses

Time critical systems

- railway crossing
- process control
- consumer electronics
- automotive and avionics
- wireless

Comp4151 Ansgar Fehnker



Overview

Next two weeks

Model checking real-time systems

Themes

- Decidability
- Efficient implementations and data structures
- Application examples

Today

- Real or continuous time vs discrete time models
- Syntax and semantics of timed automata
- Example: Biphase Mark Protocol

Comp4151 Ansgar Fehnker

Discrete Time vs. Real Time

Discrete time

Finite model of time

- time ranges over the natural numbers
- events can only occur at integer times
- time advances by multiples of a smallest time step
- use a special tick event to synchronize

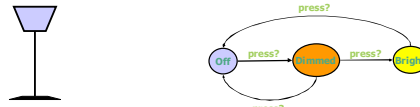
Comp4151 Arnegeer Fehnker

Discrete Time vs. Real Time

Example

Intelligent light switch

- Press button twice quickly to switch to bright
- Press it once to switch to dimmed
- If light is on or dimmed, pressing button switches light off



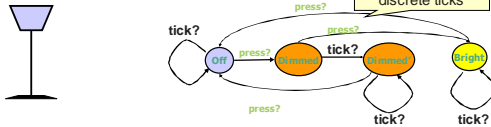
Comp4151 Arnegeer Fehnker

Discrete Time vs. Real Time

Example

Intelligent light switch

- Press button twice quickly to switch to bright
- Press it once to switch to dimmed
- If light is on or dimmed, pressing button switches light off



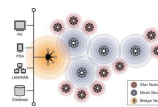
Comp4151 Arnegeer Fehnker

Discrete Time vs. Real Time

Discrete time model

- + *tick*-event just a simple addition to finite state models
- + standard model checkers and temporal logic for verification
- delays are modelled explicitly
- synchronization either to strict or to loose

⇒ suitable to model synchronous systems that evolve in lock-step
 ⇒ inadequate for many asynchronous distributed systems.



Example: wireless networks

- local clocks with drift and jitter
- discrete time may introduce absent synchrony
- however, nodes **do** synchronize on time
- complete asynchronicity inadequate as well

Comp4151 Arnegeer Fehnker

Discrete Time vs. Real Time

Real (or continuous) time

Infinite model of time

- time ranges over the positive reals
- events can take place at any point in (real) time
- multiple events can take place at the same point in time
- time may advance by any positive real amount
- no smallest time step
- timers and delays modelled using real valued variables (clocks)

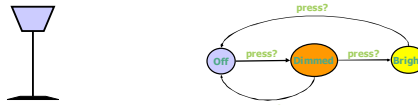
Comp4151 Ansgar Fehker

Discrete Time vs. Real Time

Example

Intelligent light switch

- Press button twice quickly to switch to bright
- Press it once to switch to dimmed
- If light is on or dimmed, pressing button switches light off



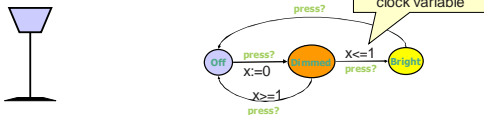
Comp4151 Ansgar Fehker

Discrete Time vs. Real Time

Example

Intelligent light switch

- Press button twice quickly to switch to bright
- Press it once to switch to dimmed
- If light is on or dimmed, pressing button switches light off



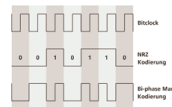
Comp4151 Ansgar Fehker

Discrete Time vs. Real Time

Continuous time model

- + clock-variables natural to model timers and delays
- + delay is modelled implicitly
- real-valued variables lead to infinite state system
- model checking algorithms become more complicated

⇒ suitable to model systems that synchronize on real time
 ⇒ too much good for completely discrete systems.



Example: Bi-Phase mark protocol

- protocol for sending bits as square wave
- local clocks for sender and receiver
- clock drift and non-deterministic delay
- to be discussed later in this lecture

Comp4151 Ansgar Fehker

Real-Time vs Real-Time

The term *real-time* often refers to

- **supervisory systems** that update information at the same rate as they receive data
- **operating systems** that employ reliable and **predictable scheduling** to minimize the number of missed deadlines, tardiness, ...
- **control systems** that react to input within some small upper limit on the response time
- **simulators** that ensure that simulation-time proceeds at the same rate as the simulated time.

These use typically a discrete model of time
 ≠
 Real-time models with real-valued clock-variables

Comp4151, Aneel Agarwal, Fehker

Outline

Today

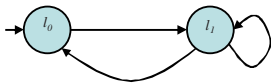
- Real or continuous time vs discrete time models
- Syntax and semantics of timed automata
 - Syntax of timed automata
 - Invariants and guards
 - Semantics of timed automata
 - Executions and runs
 - Reachability
 - Timed Languages
 - Composition
- Example: Biphase Mark Protocol

Comp4151, Aneel Agarwal, Fehker

Introduction to Timed Automata

The basics

- A finite control graph with locations and edges
- Instantaneous transitions along edges,
- Delays while in location

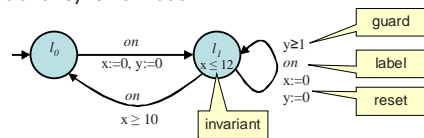


Comp4151, Aneel Agarwal, Fehker

Introduction to Timed Automata

The basics

- Real-valued clocks, that increase at the same rate
- Constraints on clocks as guard on edges
- Clock resets to measure time between transitions
- Invariants in locations to enforce progress
- Labels for synchronization



Comp4151, Aneel Agarwal, Fehker

Syntax of Timed Automata

Definitions

Clocks

A clock is a variable ranging over the positive reals $\mathbf{R}_{>0}$

Constraints

Given a set of clocks C let $\Psi(C)$ be defined by

$$\varphi := \varphi \wedge \varphi \mid \neg \varphi \mid x \leq n \mid x < n \mid x - y \leq m \mid x - y < m$$

where $x, y \in C, n, m \in \mathbf{Z}$

- Restriction to simple and diagonal constraints to ensure decidability
- Constraints of the form $x+y < n$ would make model checking TAs undecidable
- [AD94] defines TAs without diagonal constraints
- TAs without diagonal constraints are called *diagonal-free*
- Each TA with diagonal constraints is bisimilar to diagonal-free TA

Syntax of Timed Automata

Definitions

Clock valuations

A clock valuation v is a mapping $C \rightarrow \mathbf{R}_{>0}$.

Increment

For $d \in \mathbf{R}_{>0}$ valuation $v+d$ maps clock x to $v(x) + d$.

Reset

Given $r \subseteq C$ valuation $v[r:=0]$ maps x to 0 if $x \in r$ and $v(x)$ otherwise

Comp4151: Arnegeer Feherker

Syntax of Timed Automata

Definition

Timed Automata

Given a set C of clocks a *timed automaton* is a tuple

$(Loc, l_0, \Sigma, E, Inv)$, where

- Loc is a finite set of locations
- l_0 is the initial location
- Σ is a set of labels

(to be continued)

Comp4151: Arnegeer Feherker

Syntax of Timed Automata

Definition

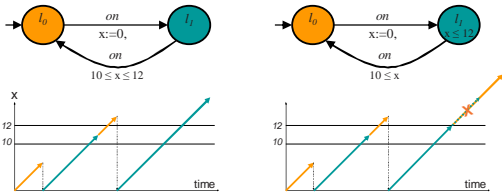
Timed Automata (cont)

- $E \subseteq Loc \times \Psi(C) \times \Sigma \times 2^C \times Loc$ a set of edges (l, g, σ, r, l') with
 - source location l
 - guard g
 - label σ (used for synchronization)
 - reset set r
 - target location l'
- $Inv: Loc \rightarrow \Psi(C)$ a mapping from locations to invariants

Comp4151: Arnegeer Feherker

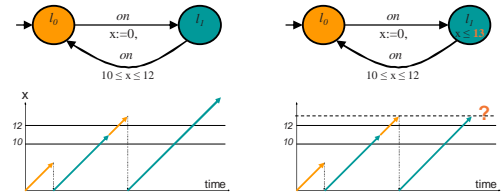
Guards and Invariants

- Guards enable progress, invariants enforce progress



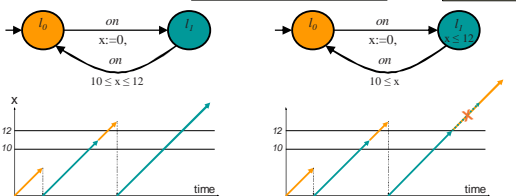
Guards and Invariants

- Guards enable progress, invariants enforce progress
- Invariants may lead to **deadlocks**



Guards and Invariants

- [AD90] used a Muller acceptance condition to ensure progress
- [Henzinger et al] introduced **Timed Safety Automata** with invariants



Semantics

Definition

The operational *semantics* of a timed automaton $(Loc, l_0, \Sigma, E, Inv)$ is given as a (timed) transition system with

- set of states $S = \{ (l, v) \mid l \in Loc, v \models Inv(l) \}$
- initial state $s_0 = (l_0, \mathbf{0})$

(to be continued)

Comp4151 Aneagr Fehriker

Semantics

Definition (cont)

- transition relation $R \subseteq S \times \Sigma \cup \mathbf{R}_{\geq 0} \times S$ that contains the following
 - discrete transitions $(l, v) \xrightarrow{\sigma} (l', v')$
if there exist $(l, g, \sigma, r, l') \in E$ s.t. $v' = g$, and $\forall r: = 0 = v'$
 - delay transitions $(l, v) \xrightarrow{d} (l, v + d)$
for $d \in \mathbf{R}_{\geq 0}$ if for all $0 \leq d' \leq d$ holds $v + d' \models \text{Inv}(l)$

Comp4151 Aregor Fehske

Runs and Executions

Run (or execution)

A finite or infinite sequence of transitions

$$(l_0, v_0) \xrightarrow{a^0} (l_1, v_1) \xrightarrow{a^1} (l_2, v_2) \xrightarrow{a^2} (l_3, v_3) \xrightarrow{a^3} \dots$$

with initial state (l_0, v_0) and $a_i \in \Sigma \cup \mathbf{R}_{\geq 0}$

Dense time

Transitions may occur at any point in real time

$$(l_0, v_0) \xrightarrow{\sqrt{2}} (l_1, v_1) \xrightarrow{\pi} (l_2, v_2) \xrightarrow{\pi} (l_3, v_3) \xrightarrow{42} \dots$$

Super dense time

Multiple transitions may occur at any point in real time

$$(l_0, v_0) \xrightarrow{\text{on}} (l_1, v_1) \xrightarrow{0} (l_2, v_2) \xrightarrow{\text{on}} (l_3, v_3) \xrightarrow{\text{on}} \dots$$

Comp4151 Aregor Fehske

Zeno

Time divergent

An infinite run is time-divergent if it has an infinite number of delays d_i such that

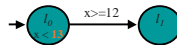
$$\lim_{n \rightarrow \infty} \sum_{i=0}^n d_i = \infty$$


Counterexample

$$(l_0, v_0) \xrightarrow{1/2} (l_0, v_1) \xrightarrow{1/2} (l_0, v_2) \xrightarrow{1/4} (l_0, v_3) \xrightarrow{1/8} \dots$$

Non-Zenoness

A timed automaton is **non-zeno**, if each finite run can be extended into a time-divergent run



Comp4151 Aregor Fehske

Reachability

Reachability

A state (l, v) is reachable if $(l_0, v_0) \xrightarrow{(\underline{d} \cup \underline{\sigma})^*} (l, v)$

A location l is reachable if there exist a v such that (l, v) is reachable.

Time additivity

Two successive delays can be combined

$$(l_0, v_0) \xrightarrow{d_0} (l_1, v_1) \xrightarrow{d_1} (l_2, v_2) \quad \text{iff} \quad (l_0, v_0) \xrightarrow{d_0 + d_1} (l_2, v_2)$$

for $d_0, d_1 \in \mathbf{R}_{\geq 0}$.

Finite runs can be rewritten as alternating sequence of transitions and delays

Comp4151 Aregor Fehske

Timed Languages

Definitions

- A **timed action** is a pair (σ, t) with $\sigma \in \Sigma$, $t \in \mathbb{R}_{\geq 0}$
- A **timed trace** of timed automaton A is a finite or infinite sequence $(\sigma_1, t_1), (\sigma_2, t_2), (\sigma_3, t_3), \dots$ with $t_0 \leq t_1 \leq t_2 \leq \dots$ s.t there exist a run

$$(l_0, v_0) \xrightarrow{d1, \sigma1} (l_1, v_1) \xrightarrow{d2, \sigma2} (l_2, v_2) \xrightarrow{d3, \sigma3} (l_3, v_3) \dots$$
 and $t_{i+1} = t_i + d_i$ for $i > 0$, and $t_0 = 0$.
- The **timed language** of A is the set of all timed traces of A .
- The **untimed language** is the restriction of the timed language to Σ .

Comp4151 Anegeer Fehske

Composition

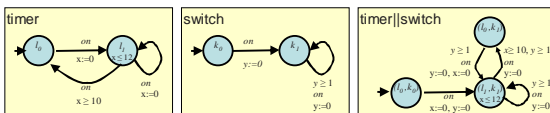
- Asynchronous and distributed systems modelled by **parallel composition** of timed automata
- Timed automata (typically) **closed** under parallel composition
- Several competing definitions of parallel composition
 - Synchronization on **common action labels** [Alur]

Comp4151 Anegeer Fehske

Composition

Composition of common labels [Alur]

- Set of locations (I^1, I^2) is the product of locations $I^1 \in \text{Loc } 1$, $I^2 \in \text{Loc } 2$
- Invariants in (I^1, I^2) are the conjunction of invariants in I^1 and I^2
- Edges **must** synchronize on shared labels,
 - guard is the conjunction of guards,
 - reset sets the union of reset sets
- Edges without shared labels may fire without synchronization



Comp4151 Anegeer Fehske

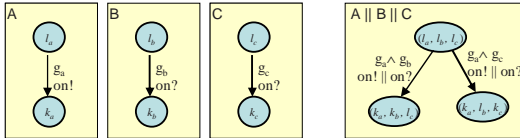
Composition

- Asynchronous and distributed systems modelled by **parallel composition** of timed automata
- Timed automata (typically) **closed** under parallel composition
- Several competing definitions of parallel composition
 - Synchronization on **common action labels** [Alur]
 - Timed I/O automata [Lynch et al]
 - Uppaal's **handshake** synchronization
 - Uppaal's **broadcast channels**

Comp4151 Anegeer Fehske

Handshake synchronization

- Uppaal defines a network of timed automata
- Synchronization via channels

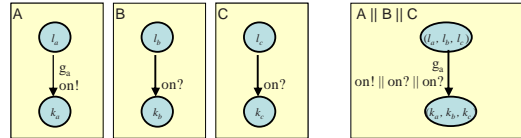


- Handshake synchronization on pairs of ! and ? label
- Both guards have to be satisfied
- If multiple pairs possible choose non-deterministically

Comp4151, Arnegeer Feherker

Broadcast synchronization

- Uppaal defines a network of timed automata
- Synchronization via channels



- Broadcast from ! channel to all ? channels
- Guards only on transition labelled !
- If multiple !-transitions enabled choose non-deterministically

Comp4151, Arnegeer Feherker

Composition

- Asynchronous and distributed systems modelled by **parallel composition** of timed automata
- Timed automata (typically) **closed** under parallel composition
- Several competing definitions of parallel composition
 - Synchronization on **common action labels** [Alur]
 - Timed I/O automata [Lynch et al]
 - Uppaal's **handshake** synchronization
 - Uppaal's **broadcast** channels
 - Many others
- Synchronization via **shared variables**

Comp4151, Arnegeer Feherker

Outline

Today

- Real or continuous time vs discrete time models
- Syntax and semantics of timed automata
 - Syntax of timed automata
 - Invariants and guards
 - Semantics of timed automata
 - Executions and runs
 - Reachability
 - Timed Languages
 - Composition
- Example: Bi-phase mark protocol

Comp4151, Arnegeer Feherker

Example

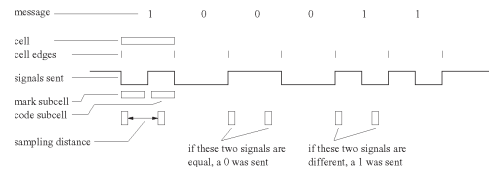
Biphase Mark Protocol

- Convention for representing both a string of bits and clock edges in a square wave.
- Used, for instance, in:
 - Intel 82530 Serial Communications Controller
 - Ethernet
 - Manchester encoding
 - Optical communications
 - Satellite telemetry applications
- Model based on work in [Vaandrager and de Groot]

Comp4151, Arnegeer Feherker

Example

Terminology



- Message encoded as square wave over as many cells as bits
- Cells are divided into mark subcell and code subcell
- Receiver should sample at the beginning of mark subcell, and somewhere within the code subcell

Comp4151, Arnegeer Feherker

Example

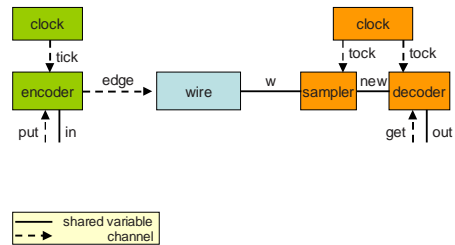
Assumptions

- Sender and receiver have each its own clocks
- Clocks with drift and jitter
- The signal takes some time after a change in voltage at stabilize.
- Sampling within this period many produce any value.
- The receiver may samples non-deterministically at some point during dock cycle.

Comp4151, Arnegeer Feherker

Example

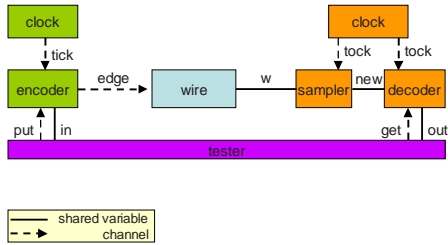
A compositional model



Comp4151, Arnegeer Feherker

Example

A compositional model

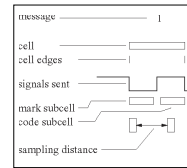


Comp4151 Ansgar Fehnker

Constants

Constants for a typical configuration

- length cell: 32 clock cycles
- length mark subcell: 16 clock cycles
- sampling point: 23 clock cycles
- min length clock cycle: 81 time units
- max length clock cycle: 100 time units
- max length unstable edge: 81 time units
- max sample delay: <81 time units

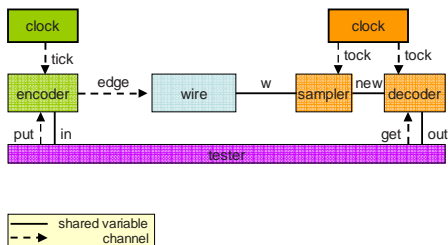


- 4 clocks (sender, receiver, wire, sampler)
- 5 channels (put, get, edge, tick, tock)

Comp4151 Ansgar Fehnker

Example

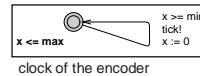
The digital clocks



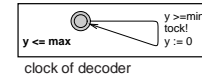
Comp4151 Ansgar Fehnker

Example

The digital clocks



clock of the encoder



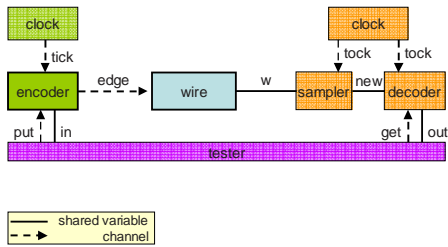
clock of decoder

- Output event: tick! and tock! (broadcast)
- Local clocks x and y
- Jitter and drift modelled as non-deterministic timing
- Clocks tick (or tock) once between min and max time units
- Broadcast channels to synchronize with other components

Comp4151 Ansgar Fehnker

Example

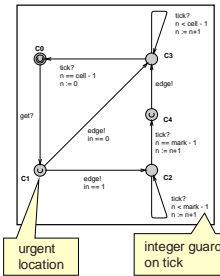
The digital clocks



Comp4151 Ansgar Fehrer

Example

The encoder

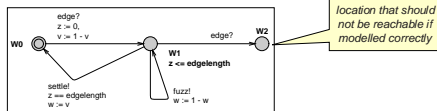


- Input event: tick?, get?
- Output event: edge!
- Input variable: in
- Local variable: n
- No clock (!) guards on tick?
- Time may not advance while in an urgent location
- Equivalent to a reset a clock x on entry and invariant $x \leq 0$ in location
- Reduction on the number of used clocks.

Comp4151 Ansgar Fehrer

Example

The wire

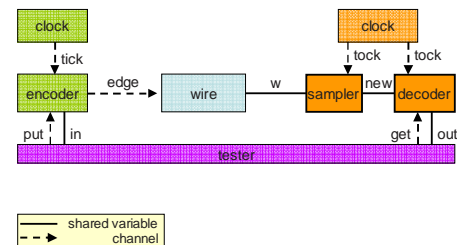


- Input event: edge?
- Local variable: v (voltage)
- Output variable: w (output voltage)
- Local events: fuzz! and stable!
- Voltage changes upon input edge!
- Output voltage may change (fuzz) during edgelenlength time after edge!

Comp4151 Ansgar Fehrer

Example

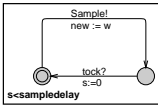
The digital clocks



Comp4151 Ansgar Fehrer

Example

The sampler

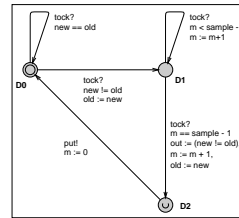


- Input variable: w (voltage)
- Output variable: new
- Input event: tock?
- Local clock: s
- Local event: Sample!
- Samples variable w less than sampledelay time after tock?
- Point of sampling non-deterministically

Comp4151 Ansgar Fehrer

Example

The decoder

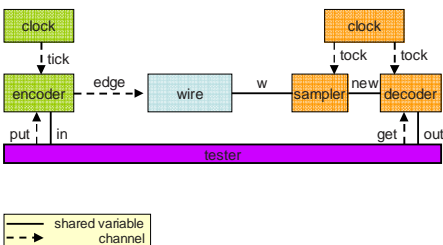


- Input variable: new
- Local variable: old, m
- Output variable: out
- Input event: tock?
- When change in new is detected, wait for sample.
- The output is 0 if sampled value equals old, 1 otherwise.
- Copy new to old, wait for next edge

Comp4151 Ansgar Fehrer

Example

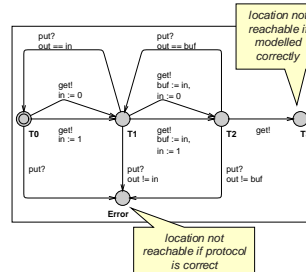
The digital clocks



Comp4151 Ansgar Fehrer

Example

▪ The tester



- Input variable: out
- Local variable: buf
- Output variable: in
- Input event: put?
- Output event: get!
- Send non-deterministically 0 or 1
- One place-buffer to send while waiting for feedback

Comp4151 Ansgar Fehrer

Model Checking

How to check automatically if an *error* location is reachable?

Main problems

The state space is infinite: $S = \{ (l, v) \mid l \in \text{Loc}, v \models \text{Inv}(l), v: C \rightarrow \mathbf{R}_{\geq 0} \}$

The transition relation is infinite: $R \subseteq S \times \Sigma \cup \mathbf{R}_{\geq 0} \times S$

Alur and Dill have a solution to this problem