Real-Time Process Control Systems

(Real-Time Controller ASM for the Railroad Crossing Case Study)

Egon Börger

Dipartimento di Informatica, Universita di Pisa
http://www.di.unipi.it/~boerger

For details see Chapter 5 (Synchronous Multi-Agent ASMs) of:

E. Börger, R. Stärk

Abstract State Machines

A Method for High-Level System Design and Analysis

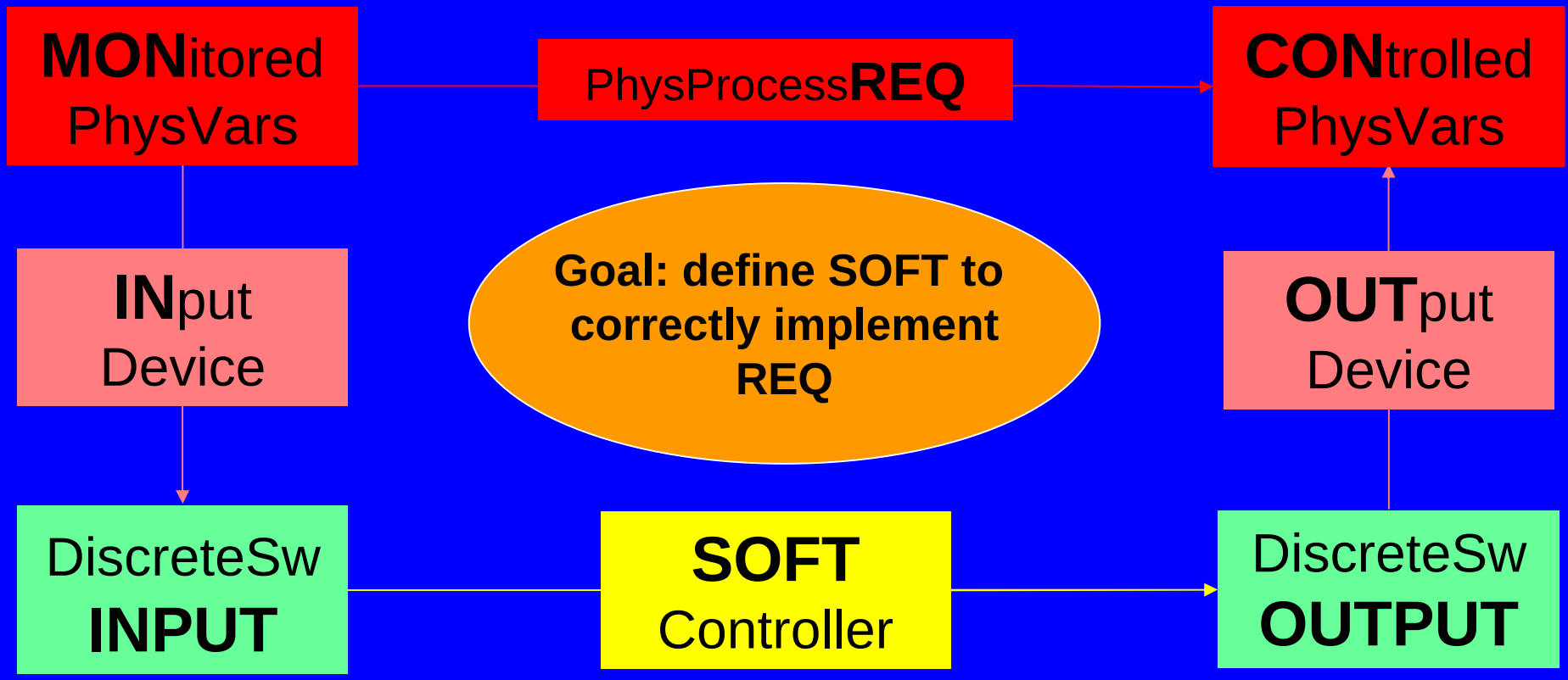
Springer-Verlag 2003

For update info see AsmBook web page:

http://www.di.unipi.it/AsmBook

- Process Control Systems
- Real-Time Controller ASMs (with atomic actions)
- Railroad Crossing Controller ASM
- Analysis of the Railroad Crossing Controller ASM (Safety & Liveness Proof)
- Exercises and References

Four-Variable Model for Process Control Systems



- IN transforms continuous physical quantities, measured by e.g. sensors, into discrete software INPUT
- OUT transforms discrete sw OUTPUT into values for actuators that manipulate the continuous physical process

Sampling & Synchronization Problem for Process Control

- **Sampling Problem** : **approximate continuous flow of physical process data by (a finite number of) discrete samples taken at real-time moments that form an increasing discrete sequence**
- **Synchronization Problem** : synchronize the SOFT controller with the IN/OUT devices to guarantee a correct reactive behavior of the entire software system wrt the env, with stable states for
  - reading INPUT
  - internal controller steps
  - sending OUTPUT

Env-Controller Separation Principle for Process Control Systems

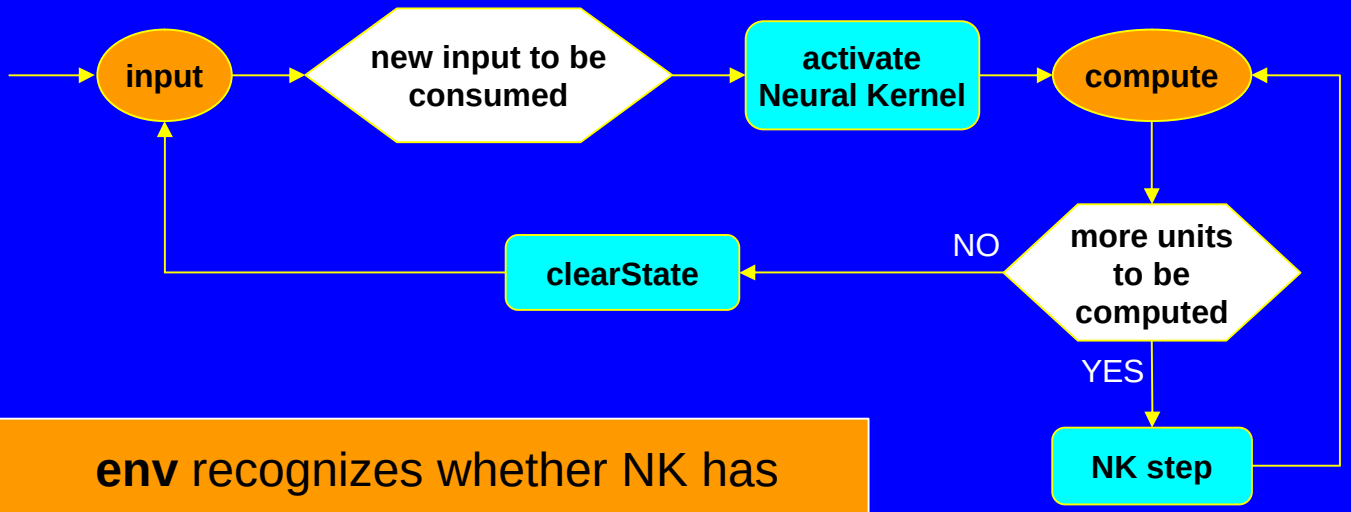
- **Environment-Controller Separation Principle**
  At each moment, the software controller
  - **either through its input device takes INPUT from the environment, without involving any further internal computation step**
  - **or makes an internal computation step (including the preparation of OUTPUT to be sent to the output device), without involving any further INPUT reading**

This principle is applied by many process control systems. Its appropriateness for a physical system to be controlled depends on

- **correctness of sampling mechanism**
- **sufficient hw speed (for the internal controller steps) wrt sampling**

Env-Controller Separation Principle in Neural Nets

- A **Neural Net** is usually seen as a black-box yielding output to the env, as result of an internal computation which is triggered by an input taken from the env. The internal computation consists of a finite sequence of atomic actions performed by the basic computing elements (nodes of a directed data-flow graph)
  - In forward propagation mode, the network input is transmitted by the input units to the internal units which propagate their results through the graph until the output units are reached



env signals presence of new input
NK consumes input

env recognizes whether NK has terminated its internal computation

Env-Controller Separation Principle in Circuits

- **Circuit Synchronization** by a clock which alternates between low and high pulse
- circuit viewed as
  - black box
  - input lines (representing the environment)
  - output lines
- When clock pulse is low
  - box and output idle, input may change
- When clock pulse is high
  - box and output may change, input is idle

- Control Systems
- Real-Time Controller ASMs (with atomic actions)
- Railroad Crossing Controller ASM
- Analysis of the Railroad Crossing Controller ASM (Safety & Liveness Proof)
- Exercises and References

Goals of Real-Time Controller ASMs (with Atomic Actions)

- **atomic actions**
  - **controller executes instantaneously**
  - **environment changes take place instantaneously**
- **at each real-time moment, there is a well defined state**
- **runs which satisfy the environment-controller separation principle**

In the definition below, we will allow that the controller is constituted by multiple agents, each of which executes a basic ASM. Atomicity of actions constitutes an idealization, to be refined by durative actions.

Recalling Basic ASMs

- A **Basic ASM** is a finite set of rules, executed by a single agent self, and having the form

  If Condition Then Updates

  with Updates a finite set of $f(t_1,\ldots,t_n):=t$ with arbitrary functions,
  Condition a Boolean valued expression (1st order formula)

- In the current state (structure) S:
  - determine all the fireable rules in S (s.t. Cond is true in S)
  - compute all exprs t_i,t occuring in updates $f(t_1,\ldots,t_n):=t$
  - execute simultaneously all these function updates
- The updating yields the next state S'

Runs of Real-Time Controller ASMs

Run of a Real-Time Controller ASM M given by

- **a discrete sequence $0 = t_0 < t_1 < t_2$,... of reals – consisting of (all & only) the computationally significant real-time "moments" of M - and associated states $S(t_n)$ where currtime = t_n and**
- **$s(t_{n+1})$ is obtained from $s(t_n)$ (= $S(t_n)$ minus currtime) by**
  - **either Case 1: a computation step (of some agents) of M, without any change of monitored fcts between t_n and t_{n+1}**
  - **or Case 2: a change of monitored fcts, without any M-comp step between t_n and t_{n+1}**
- standard extension to states S(t) for $t_n < t < t_{n+1}$

Standard Extension of Runs of Real-Time Controller ASMs

- A run $(t_n, S(t_n))_{n \in N}$ of a real-time controller ASM M is naturally extended to states S(t) of arbitrary real-time moments $t_n < t < t_{n+1}$ as follows:
  - **currtime = t in S(t)**
  - $s(t) = s(t_{n+1})$ **in Case 1 (result of the instantaneous M-step executed at time t_n)**
  - $s(t) = s(t_n)$ **in Case 2 (unchanged M-state, since between t_n and t_{n+1} no M-step takes place, & the instantaneous change of monitored fcts becomes effective at t_{n+1})**
- **Lemma.** $\forall\, t_n < t < t' < t_{n+1} : s(t) = s(t')$
- NB. $(t_n)_{n \in N}$ is discrete iff $\forall\, t\ \exists$ **only finitely many $t_n < t$. This condition excludes so-called Zeno runs.**

- Control Systems
- Real-Time Controller ASMs (with atomic actions)
- Railroad Crossing Controller ASM
- Analysis of the Railroad Crossing Controller ASM (Safety & Liveness Proof)
- Exercises and References

Problem Description of Railroad Crossing Case Study

- The system to be developed operates a gate at a railroad crossing. A set of trains travel on multiple tracks in both directions. Each track has four sensors, detecting when a train enters resp. exits the crossing: L1, L2 for trains coming from the left, R1, R2 for trains coming from the right. Based on these sensor signals, a controller should signal the gate to open/close.
- Prove the following two properties for the system:
  - Safety: When a train is in the crossing, the gate is closed.
  - Liveness:The gate is open when no train is in the crossing.

Train Motion Assumptions of Railroad Crossing

- Train Motion: For every track, there is a train motion sequence $t_0 < t_1 < t_2,...$ **of reals with the following properties:**
  - Initial State: at the initial moment t_0, the observed track segment [L1,R1] is empty.
  - Train Pattern: If t_{3i+1} appears in the sequence, then t_{3i+2} and t_{3i+3} appear in the sequence, and
    - at t_{3i+1}, an incoming train is detected at either L1 or R1
    - at t_{3i+2} this train reaches the crossing
    - at t_{3i+3} this train is detected to have left the crossing at L2 resp. R2
  - Completeness : The sequence $t_0 < t_1 < t_2,...$ covers all the (and only) trains that appear on this track.

Train & Gate Assumptions of Railroad Crossing

- Train Approaching Time: $d_{min} \leq d_{max}$ are the minimal/maximal time for a train to reach the crossing after having been detected at L1/R1:
  - $\mathbf{t_{3i+2} - t_{3i+1}} \in [d_{min}, d_{max}]$ for all i in a train motion sequence
- Gate Closure/Opening Time: during no interval $[t, t+d_{close}]$ resp. $[t, t+d_{open}]$ is a signal to close/open in force without at some moment in this interval the gate being closed/opened.
- Gate versus Train Time: $d_{close} < d_{min}$

NB. Gate may react to closure/opening commands with some (bounded) delay, whereas wlog we will assume the controller to react immediately, i.e. at the instant when it is enabled.

Signature of Railroad Crossing ASM

- currtime: REAL is the central monitored time function
- TRACK : static domain with
  - monitored fct TrackStatus : TRACK $\rightarrow$ {empty, coming, inCrossing}
  - controlled fct Deadline : TRACK $\rightarrow$ REAL $\cup$ {∞} to measure the allowable WaitingTime bw the train appearance and the latest possible moment to start the gate closing (for the gate to be closed in time)
- two dynamic fcts for gate moving directions and final gate positions:
  - Dir $\in$ {open, close} controlled by the controller
  - GateStatus $\in$ {opened, closed} controlled by the gate

Controller ASM for Railroad Crossing

- **Controller** ≡ forall x ∈ TRACK
  SetDeadline(x)
  SignalClose(x)
  ClearDeadline(x)
  SignalOpen
- SetDeadline(x) ≡ If TrackStatus (x) = coming & Deadline (x) = ∞
  then Deadline (x) := currtime + $d_{min} - d_{close}$ (WaitTime)
- SignalClose(x) ≡ If currtime = Deadline (x) then Dir := close
- ClearDeadline (x) ≡ If TrackStatus (x) = empty & Deadline (x) < ∞
  then Deadline (x) := ∞
- SignalOpen ≡ If Dir = close & SafeToOpen then Dir:=open

where SafeToOpen ≡ " x∈ TRACK: TrackStatus (x) = empty or
enough time to open before next closure currtime + d_{open} < Deadline (x)

Gate ASM: an instance of a Flip Flop

- **Gate** ≡ OpenGate
  CloseGate

  where OpenGate ≡ If Dir = open then GateStatus := opened

  CloseGate ≡ If Dir = close then GateStatus := closed

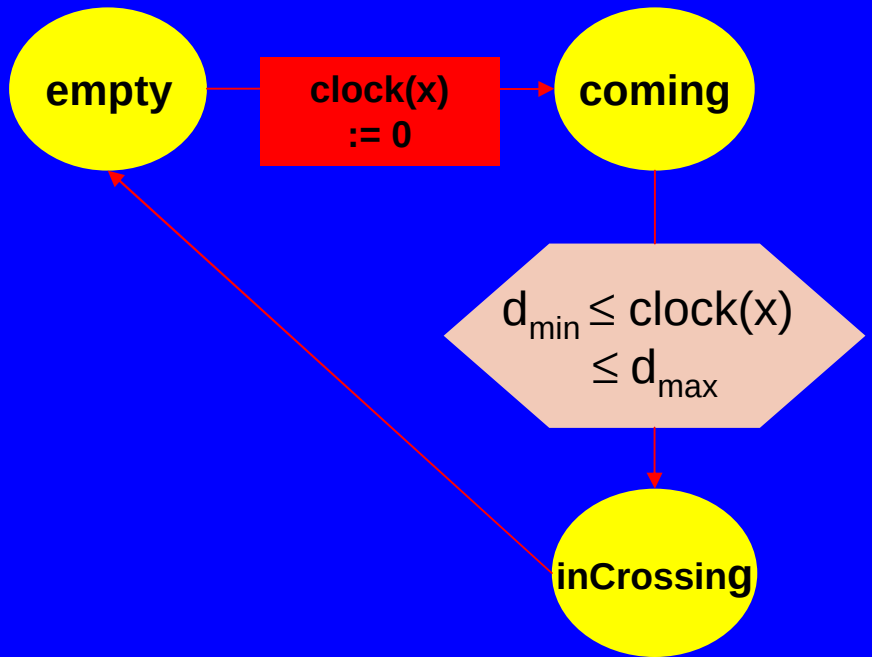
- Initialization
  - GateStatus = opened Dir = open
  - for all $x \in$ TRACK

    TrackStatus (x) = empty Deadline(x) = ∞
- NB. Sharpening the guards in
  - Gate rules by GateStatus = closed/opened
  - SignalClose (x) by Dir = open

  yields a machine where each step changes the state. We disregard this optimization, since the definition of runs of a real-time controller ASM is not restricted to machine steps which change the state.

The Agents of the Railroad Crossing ASM

- The Railroad Crossing Controller ASM has three agents
  - Controller, Gate
  - Env, represented by the monitored fct TrackStatus
- A representation of Env as agent has to internalize the timing conditions of train motion sequences, e.g. by a timed automaton ASM, consisting of one machine for each track x, each able to fire independently & simultaneously.



NB. The notion of real-time controller ASM run has to be adapted to treating env as a special team of agents

A Run of the Railroad Crossing ASM is …

… a real-time controller ASM run. These runs reflect the train motion condition through the moments of change of the monitored function TrackStatus:

- **among the computationally significant moments of the run,** every track x has a subsequence $0 = t_0 < t_1 < t_2, \ldots$ of significant moments of x such that **TrackStatus(x) =**
  - **empty over every interval $[t_{3i}, t_{3i+1})$**
  - **coming over ... $[t_{3i+1}, t_{3i+2})$ & $d_{min} \leq t_{3i+2} - t_{3i+1} \leq d_{max}$**
  - **inCrossing over ... $[t_{3i+2}, t_{3i+3})$**
  - **empty over $[t_k, \infty)$ if t_k is the final significant moment of track x, in which case 3|k**
- NB. In those runs, the Controller is assumed to react immediately, the reaction time of Gate to be bounded in the sense of the Gate Closure/Opening Time property.

- Control Systems
- Real-Time Controller ASMs (with atomic actions)
- Railcroad Crossing Controller ASM
- Analysis of the Railroad Crossing Controller ASM (Safety & Liveness Proof)
- Exercises and References

Safety & Liveness for Railroad Crossing ASM

For arbitrary runs of the Railroad Crossing ASM, the following two properties hold:

- Safety Theorem. At any moment when a train is in the crossing (Gate Status (x) = inCrossing for some track x), the gate is closed (GateStatus = closed).
- Liveness Theorem. Whenever the crossing is empty (i.e.TrackStatus(x) ^{1} inCrossing for every x) in an open real-time interval (α,β) with $\alpha + d_{open} < \beta - D_{close}$, the gate is open in the closed interval $[\alpha+d_{open}, \beta - D_{close}]$ where
  $D_{close} = d_{max} - \text{WaitTime} = d_{close} + (d_{max} - d_{min})$
- NB. It can be shown that the Liveness Thm fails if d_{open} or D_{close} are replaced with smaller values.
- NB.One can also prove that the closing of the gate is never interrupted; under appropriate assumptions the same holds for opening.

Proving Safety for the Railroad Crossing ASM

For some track x, let TrackStatus(x) = inCrossing over an interval $[t_{3i+2}, t_{3i+3})$. We show: GateStatus = closed over $[t_{3i+1} + d_{min}, t_{3i+3}] \supseteq [t_{3i+2}, t_{3i+3}]$.

Proof.

- SetDeadline (x) fires at $\mathbf{t_{3i+1}}$ (Controller is immediate) setting Deadline(x) to $\alpha = \mathbf{t_{3i+1}} + \text{WaitTime}$
- TrackStatus(x) ^{1} empty **over I = (a, t_{3i+3})**

- Hence **over I, x makes SafeToOpen false, hence SignalOpen disabled**.
- Then also **OpenGate** is **disabled** over I
  - because SignalClose(x) fires at α so that immediately after α, Dir=close holds (& remains unchanged over I)
- Dir=close immediately after α implies that
  - GateStatus = closed for some $a < t < a + d_{close} = t_{3i+1} + d_{min}$
  - & remains so as long as SignalOpen is disabled, i.e. over (α, t_{3i+3}) and therefore over $[t_{3i+1} + d_{min}, t_{3i+3}]$
    - since only OpenGate, fireable only after SignalOpen, can change it to opened

Proving Liveness for the Railroad Crossing ASM

By the Deadline Lemma (see below), for every track x holds:

- Deadline(x)$\geq b - D_{close} > a + d_{open}$ **over** (a, b)
  - Therefore SafeToOpen holds at currtime = a
  - Thus Dir=open holds immediately after a (possibly through firing of SignalOpen)
- Deadline(x)$\geq b - D_{close} >$ **currtime over** $(a, b - D_{close})$
  - Hence over $(a, b - D_{close})$: every SignalClose(x) is disabled, so that Dir remains open and CloseGate is disabled
- Dir = open over $(a, b - D_{close})$ and d_{open} time assumption imply that
  - GateStatus = opened for some $a < t < a + d_{open}$
  - & remains so as long as CloseGate is disabled, namely over $(a, b - D_{close})$
  - Therefore GateStatus = opened over $[a + d_{open}, b - D_{close}]$
    - since only CloseGate, fireable only after some SignalClose(x), can change it to closed

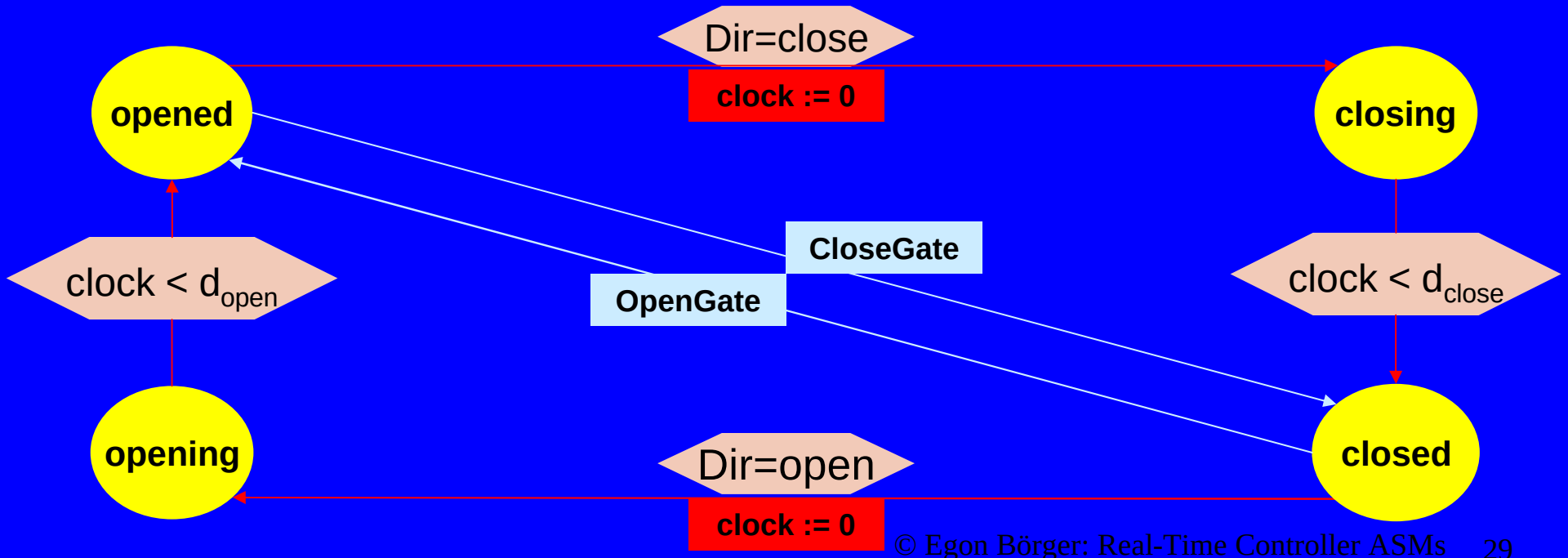
Deadline Lemma for Railroad Crossing ASM

- **Deadline Lemma**. In a run of the Railroad Crossing ASM, let x be a track with significant moments $0 = t_0 < t_1 < t_2, \ldots$
  - **Deadline(x) =**
    - ∞ **over** $(t_{3i}, t_{3i+1}]$
    - t_{3i+1} + WaitTime **over** $(t_{3i+1}, t_{3i+3}]$
  - **If TrackStatus(x)** ^{1} **inCrossing over an interval** (a,b), **then over this interval Deadline(x)** ^{3} $b - D_{close}$ **where** $D_{close} = d_{close} + (d_{max} - d_{min}) = d_{max}$ - WaitTime
- **Prove (a) by** induction on **i**
- **For (b) assume Deadline(x)<**b **+** WaitTime - d_{max} **at some t** $\hat{I}$ (a,b)**.**
- **Then Deadline(x) =** t_{3i+1} + WaitTime **at t for some** $t_{3i+1} < t £ t_{3i+3}$ **.**
- **Then** $t_{3i+1} < t <$ b $£ t_{3i+2}$ **since** by hypothesis, (a,b) and the inCrossing interval $[t_{3i+2}, t_{3i+3})$ **are disjoint.**
- Therefore $b - t_{3i+1} £ t_{3i+2} - t_{3i+1} £ d_{max}$ **and** $b - d_{max} £ t_{3i+1}$ **so that**
- $b - D_{close} = b - d_{max}$ **+WaitTime** $£ t_{3i+1}$ **+WaitTime = Deadline(x) (at t)** $< b - D_{close}$ **(by assumption): a contradiction**

- Control Systems
- Real-Time Controller ASMs (with atomic actions)
- Railcroad Crossing Controller ASM
- Analysis of the Railroad Crossing Controller ASM (Safety & Liveness Proof)
- Exercises and References

Railroad Crossing Case Study : Refinement Exercise

- Refine the Railroad Crossing ASM by introducing gate positions between Closed = 0^o and Opened = 90^o. Clarify the real-world timing assumptions which allow that Dir=close/open only when GateStatus=opened/closed.
- Hint. Consider the following timed automaton ASM, with additional GateStatus values closing, opening, and refined rules OpenGate, CloseGate.



Exercises for the Railroad Crossing ASM

Show the following properties for runs of the Railroad Crossing ASM with significant moments $0 = t_0 < t_1 < t_2, \ldots$ for a track x:

- SetDeadline(x) fires exactly at $\mathbf{t_{3i+1}}$ (when TrackStatus(x) has become coming)
- SignalClose(x) fires exactly at $\mathbf{t_{3i+1}}$+ WaitTime
- ClearDeadline (x) fires exactly at $\mathbf{t_{3i}}$ if i>0 (when TrackStatus(x) has become empty)
- Let s(x) be the local SafeToOpen(x) condition, namely TrackStatus(x)=empty or currtime + d_{open} < Deadline (x). Show:
  - If WaitTime > d_{open}, then s(x) holds over intervals [$\mathbf{t_{3i}}$, $\mathbf{t_{3i+1}}$ +WaitTime – d_{open}) and fails over intervals [$\mathbf{t_{3i+1}}$ +WaitTime – d_{open}, $\mathbf{t_{3i+3}}$)
  - If WaitTime ≤ d_{open}, then s(x) holds over intervals [$\mathbf{t_{3i}}$, $\mathbf{t_{3i+1}}$] and fails over intervals ($\mathbf{t_{3i+1}}$, $\mathbf{t_{3i+3}}$)
  - s(x) changes from false to true at moments $\mathbf{t_{3i}}$ with i>0 (when TrackStatus(x) has become empty)
- SignalOpen fires exactly when SafeToOpen becomes true. If SafeToOpen becomes true at t, then some TrackStatus(x) has become empty at t.

Railroad Crossing ASM : Exercises (Cont'd)

- Show that the Liveness Theorem fails if d_{open} or D_{close} are replaced with smaller values.
- Prove that the closing of the gate is never interrupted, i.e. if Dir is set to close at some moment t, then Dir = close holds over the interval $(t , t+d_{close})$.
- Prove that if WaitTime $\geq d_{open}$, then the opening of the gate is never interrupted, i.e. if Dir is set to open at some moment t, then Dir = open holds over the interval $(t, t+d_{open})$.
- Show that the Railroad Crossing ASM, viewed as consisting of three agents for Controller, Gate, and the Environment (representing a team that updates the monitored functions), in a natural way satisfies the axioms for an async ASM.
- Adapt the definition of real-time controller ASMs to the case where the changes of monitored functions are treated as resulting from updates of a team env of agents.

References

- C. Heitmeyer and D. Mandrioli (Eds.) : Formal Methods for Real-Time Computin. John Wiley & Sons, Chichester 1996
- Y. Gurevich and J. K. Huggins: The Railroad Crossing Problem: An Experiment with Instantaneous Actions and Immediate Reactions
  - In: Computer Science Logic. Selected Papers from CSL'95 (Ed. H. Kleine Büning). Springer LNCS 1092, 1996, 266-290
- D. Beauquier, T. Colard, A. Slissenko: A Predicate Logic Framework for Mechanical Verification of Real-Time Gurevich Abstract State Machines: A Case Study with PVS
  - TR 00-25, University Paris 12, CS Dept, 2000. See http://www.univ-paris12.fr/lacl/

References

- E. Börger, R. Stärk: Abstract State Machines. A Method for High-Level System Design and Analysis Springer-Verlag 2003, see http://www.di.unipi.it/AsmBook
- D. L. Parnas and J. Madey: Functional Documentation for Computer Systems Engineering (Vol.2).
  - Mc Master University TR CRL 237, Hamilton, Ontario, Sept.1991
- D.L. Parnas and J. Madey: Functional documents for computer systems. In: Science of Computer Programming 25 (1995), 41-62.