Introduction

Course HECS3: Performance and quantitative properties

Goran Frehse

September 28, 2016

High-confidence Embedded and Cyber-Physical Systems
Master of Science in Informatics at Grenoble
Univ. Grenoble Alpes, Laboratoire Verimag
frehse@imag.fr
Introduction

Embedded and Cyber-Physical Systems
Key Features of CPS
Fundamentals of Dynamical Systems
Specifying and Analyzing Properties
Model-Based Design
What to Expect in this Lecture

**concepts** from Embedded and Cyber-Physical Systems

- standard terminology (and some buzzwords)
- informal presentation (formalization in future lectures)
- a rough map of the territory
- what it is all for...
Embedded and Cyber-Physical Systems

original computer: standalone device

**embedded system**: integrated with non-computational hardware for a specific purpose

- watches, cameras, refrigerators (integrated microcontroller), ...  

more examples?
**cyber-physical system**¹: collection of communicating computers, interacting with the physical world via feedback

- using control, computing, communication
- smart buildings, medical devices, cars, …

example: team of autonomous robots retrieving target inside house

more examples?

¹ term coined by Helen Gill at the US National Science Foundation (NSF) in 2006
CPS at Verimag’s Hybrid Systems Group

Variety of Application Domains
1. Automated vehicle of Tecnalia
2. Sandwich assembly robot of R.U.Robotics Ltd.
3. Chemical batch plants
4. MILOX™: Pipeless Production System
5. Dang, Donzé, Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. FMCAD’04
6. Bitcraze Crazyflie autonomous drones

FP7 + H2020 projects  NANO 2017 project  Collaborations
Overview

Introduction

Embedded and Cyber-Physical Systems

Key Features of CPS

Fundamentals of Dynamical Systems

Specifying and Analyzing Properties

Model-Based Design
Key Features of CPS [Alur’15]

reactiveness

• traditionally: input → computing → output → stop
• mathematically: function: inputs → outputs
• reactive: ongoing computation
• mathematically: function from sequence of inputs to sequence of outputs

examples?
Key Features of CPS

**concurrency**

- traditionally: sequential computation (Turing machine)
- concurrent: multiple threads of computation, exchanging information
- synchronous computation: all components execute in lock-step
- asynchronous computation: components act independently, communicating via messages
- both can be useful levels of abstraction

examples?
Key Features of CPS

**feedback control**

- control system interacts with physical world with sensors and actuators
- design requires modeling the dynamics of physical quantities
- traditionally: continuous dynamics
  – a small enough change in the input generates a small change in the output

examples?
Key Features of CPS

real-time

- traditionally: no explicit notion of real time
- CPS: computation needs to finish within a given time frame
- timing delays, timing-dependent coordination protocols, resources allocation → study of real-time systems

examples?
Overview

Introduction

Embedded and Cyber-Physical Systems

Key Features of CPS

Fundamentals of Dynamical Systems

Specifying and Analyzing Properties

Model-Based Design
goal: a unified view of seemingly disparate systems

• using the same concepts
• adapting techniques where necessary
• combining different techniques when systems have heterogeneous components

... which they do in cyber-physical systems!

examples?
dynamical system

• precisely identified entity
  (we know what is part of the system and what isn’t)
• defining behaviors over some notion of time
  (we know what ”before” and ”after” mean)
• with (possibly) observable outputs
• (possibly) influenced by a given set of inputs

examples for what is not a dynamical system?
behavior

• evolution of states over time
• (possibly) decorated with input or output

formalized as executions, runs, words, traces, trajectories, … examples?
disturbance

- something that modifies the inputs or outputs of the system

random changes in the environment, electromagnetic interference, sensor noise, quantization error(!)

more examples?
Fundamentals of Dynamical Systems

deterministic system

• if the inputs are known, there is only one future behavior

nondeterministic system

• if the inputs are known, there is a known set of future behaviors
  (actual behavior may be different each time we run the system, but belong to the same set)

examples?
stochastic system

- if the inputs are known, the future behavior is known with a certain probability 
  (it’s the same behavior xyz% of the times we run the system)

examples?
state

- set of values that suffices to predict the (sets of) future behavior of the system if the inputs are known

state-space

- the set of states of the system

example: motion of a car (with accelerator and brake pedals)
Fundamentals of Dynamical Systems

transition

- relates a state to a successor state
- may depend on time and inputs

transition relation

- defines for each state the possible successor states
- a subset of states $\times$ time $\times$ inputs $\times$ states

\[
\text{state} \xrightarrow{\text{time, input}} \text{state'}
\]
Fundamentals of Dynamical Systems

reachable states

- states in the closure of the transition relation
- starting from a given set of initial states
finite state system

- the state space and the inputs are finite sets

What is maximum size of the transition relation (deterministic/nondeterministic)?
Fundamentals of Dynamical Systems

state-space exploration (enumerative)

- starting from a given initial state, visit all reachable states, trying out all possible inputs
- = graph traversal, e.g., breadth-first search

always **terminates** if the state space is **finite**

example: check if the system can go to a given "bad" state
Fundamentals of Dynamical Systems

infinite state system

- the state space is an infinite set (enumerable or not)

state-space exploration no longer terminates
symbolic state-space exploration

- like state-space exploration, but using sets of states
- terminate if successors $\subseteq$ visited states or bad states overlap visited states
- often uses overapproximation to operate on sets with simple descriptions (intervals…)

may terminate even if state space is infinite
Example: A program computing $\sqrt{x_0}$ using the babylonian method

$$x_{k+1} = \frac{1}{2} \left( x_k + \frac{x_0}{x_k} \right).$$

implemented using int, float, rationals, reals, ...

- state-space? initial state? inputs? outputs? time?
- transition relation? behaviors?
- deterministic? finite?
Exercises:

Given an implementation using int, float, rationals, reals, …

1. When is enumerative state-space exploration applicable?
2. What are the ”bad states” for checking if the sequence converges to $\sqrt{x_0}$?
3. Apply symbolic state-space exploration starting from $x_0 = 8$. Use integer intervals to describe sets of states. Overapproximate if necessary.
4. Start from $x_0 = 9$. How can the precision be increased?
5. Does always rounding up or always rounding down cover all possibilities?
Discrete-Time Dynamical System:

\[ x_{k+1} = f(x_k, u_k). \]

• state-space? initial state? inputs? outputs? time?
• why "discrete-time"?
• transition relation?
• deterministic? finite?

Examples: Finite state machine (digital computer)
Discrete-Time **Continuous** Dynamical System:

\[ x_{k+1} = f(x_k, u_k). \]

- \( f \) is a continuous function of \( x \) and \( u \):
- a small enough change in the input (or in time) generates an arbitrarily small change in the output

Examples: Digital controller (considering floating point as real numbers); sun position at noon every day
Discrete-Time **Continuous** Dynamical System:

\[ x_{k+1} = f(x_k, u_k). \]

two main categories:

- **f** is **linear**: \( x_{k+1} = Ax_k + Bu_k \)
  
either converging, diverging, or periodic
- **f** is **nonlinear**:
  
possibly chaotic behavior
Linear Map

scalar case:

\[ x_{k+1} = ax_k \]

for which values of \( a \):

- converging,
- diverging,
- periodic?
Logistic Map

demographic model with reproduction and starvation
[R. May, 1976]

\[ x_{k+1} = rx_k(1 - x_k) \]

\[ x_0 = 0.6, \quad r = 4 \]
Fundamentals of Dynamical Systems

Discrete-Time **Piecewise Continuous** Dynamical System:

\[
x_{k+1} = \begin{cases} 
  f_1(x_k, u_k), & x_k \leq c_1 \\
  \vdots \\
  f_i(x_k, u_k), & c_{i-1} < x_k \leq c_i \\
  \vdots \\
  f_m(x_k, u_k), & x_k > c_m 
\end{cases}
\]

- may exhibit complex behavior even for simple \( f_i \)

Example: continuous systems with saturation of signals
Tent Map

\[ x_{k+1} = \begin{cases} 
\mu x_k, & x_k < \frac{1}{2}, \\
\mu(1 - x_k), & x_k \geq \frac{1}{2} 
\end{cases} \]

\( x_0 = 0.6, \mu = 2 \)
Tent Map

\[ x_{k+1} = \begin{cases} 
\mu x_k, & x_k < \frac{1}{2}, \\
\mu(1 - x_k), & x_k \geq \frac{1}{2} 
\end{cases} \]

\[ x_0 = 0.6001, \ \mu = 2 \]
Tent Map

\[ x_{k+1} = \begin{cases} 
\mu x_k, & x_k < \frac{1}{2}, \\
\mu(1 - x_k), & x_k \geq \frac{1}{2} 
\end{cases} \]

\[ x_0 = 0.6, \, \mu = 1.5 \]
Tent Map vs Logistic Map

tent map:

\[ x_{k+1} = \begin{cases} 
\mu x_k, & x_k < \frac{1}{2}, \\
\mu (1 - x_k), & x_k \geq \frac{1}{2}
\end{cases} \]

logistic map:

\[ y_{k+1} = ry_k (1 - y_k) \]

for \( \mu = 2 \) and \( r = 4 \):

\[ x_k = \frac{2}{\pi} \sin^{-1} \sqrt{y_k} \]

relation between piecewise linear and nonlinear system
Continuous-Time (Continuous) Dynamical System:

Typically given by a differential equation system:

\[ \dot{x}(t) = f(x(t), u(t)) \].

- can be converted to discrete-time system by sampling at time points, e.g., \( t = k\delta \)

Example: Motion of a car
Discrete-Continuous Dynamical System (Hybrid System):

- mix of discrete and continuous dynamics
- discrete state changes are considered instantaneous
- discrete state determines continuous dynamics

Example: Motion of a car with gear shift
Levels of Abstraction

discrete (state) system (discrete dynamics)
continuous system (continuous dynamics)
  • discrete or continuous time
discrete-continuous (hybrid) system

What is the ”right” model?
Example: Robot in a Maze

- robot turning at exactly 90 degrees, timing irrelevant: **discrete system**
  Can the robot leave the maze?

- maze door opens and closes at specific times: **timed system**
  Can the robot leave the maze while the door is open?

- robot not turning exactly 90 degrees: **hybrid (discrete-continuous) system**
  accumulation of deviations!
  Can the robot leave the maze while the door is open?
Overview

Introduction

- Embedded and Cyber-Physical Systems
- Key Features of CPS
- Fundamentals of Dynamical Systems
- Specifying and Analyzing Properties
- Model-Based Design
Specifying and Analyzing Properties

**boolean properties**

- **state property**: state $\rightarrow \{\text{true, false}\}$
  
e.g., predicate over the state variables

- **behavior property**: behavior $\rightarrow \{\text{true, false}\}$
  
e.g., all states along the behavior satisfy the property

- **system property**: system $\rightarrow \{\text{true, false}\}$
  
e.g., all behaviors from initial states satisfy property

These generalizations to behaviors over time are called **temporal logics**

examples?
probabilistic properties

• behavior probability: behavior $\rightarrow [0, 1]$
e.g., probability that the behavior is taken (from given initial state)

• probabilistic property: system $\rightarrow [0, 1]$
e.g., probability that any behavior from the initial states satisfies the property

these generalizations to behaviors over time are called probabilistic temporal logics

examples?
Specifying and Analyzing Properties

**safety**: nothing bad ever happens

analysis techniques:

- inductive invariants,
- state-space exploration (enumerative, symbolic)

examples?
Specifying and Analyzing Properties

**liveness**: something good eventually happens

analysis techniques:

- temporal logics,
- model checking (generalization of state-space exploration),
- ranking functions

examples?
probabilistic safety and liveness: nothing bad/something good happens with a certain probability

analysis techniques:

• probabilistic temporal logics,
• model checking,
• fault-tree analysis

descriptions?
quantitative semantics of safety and liveness: distance to nothing bad/something good happening

• e.g., min distance of any behavior to violating the property

Example: \( x(t) \leq c \) for all \( t \geq 0 \) (boolean safety)

• quantitative semantics \( q = \min_{t \geq 0} c - x(t) \)

property satisfied iff \( q \geq 0 \).

measure of robustness\(^2\)

real-time scheduling: system achieves given tasks in given time frame

analysis techniques:

- model checking,
- worst-case execution time (WCET) analysis

quantitative property:

- computing worst-case execution time

examples?
Specifying and Analyzing Properties

**stability**: system will remain close to its steady state if disturbances (inputs) small enough

Analysis techniques:

- linear algebra,
- Lyapunov functions (continuous ranking functions)

**Quantitative property**:

- **stability (gain) margin**: amount that feedback can be increased while remaining stable
Overview

Introduction

Embedded and Cyber-Physical Systems
Key Features of CPS
Fundamentals of Dynamical Systems
Specifying and Analyzing Properties
Model-Based Design
Model-Based Design

• traditionally: design, implementation, testing, validation
• model-based: formal (mathematically precise) requirements, models of the system and its environment, analysis tools for checking requirements on the model
• detect design errors earlier, ensure higher reliability
Model-Based Design

- different from programming: may incorporate nondeterminism and environment behavior
- structured design: complex tasks accomplished by composing simple components (and conversely for properties)
- requirements-based design: requirements are specified up front and guide the design (choice between design alternatives) and debugging
Model-Based Design

Development Vision for Systems Mixing
Software, Circuits and Mechanics (Fujitsu 2006)

Model-Based Design

Development Vision for Systems Mixing Software, Circuits and Mechanics (Fujitsu 2006)

Formal Models

- unambiguous – not open to interpretation
- mathematically precise, can be analyzed rigorously
- block diagrams, code, state machines, differential equations
Formal Verification

check if a formal model satisfies a property using mathematical reasoning

- rigorous (sound)
- exhaustive (all behavior is covered)
- (possibly) algorithmic or with computer support (theorem prover)
Formal Verification

drawbacks:

• generally: not scalable (or not even decidable)
• a suitable model needs to be constructed first
• typically requires expert knowledge

hard questions:

• Does the model match reality?
• Who verifies the verifier?
Boeing & Tupolew Collision

- Überlingen, July 1, 2002
- 21:33:03
- Alarm from Traffic Collision Avoidance System (TCAS)

B757-200  !  TU154M
Boeing & Tupolew Collision

- Überlingen, July 1, 2002
- 21:33:03
  - Alarm from Traffic Collision Avoidance System (TCAS)
- 21:34:49
  - Human air traffic controller command
Boeing & Tupolew Collision

- Überlingen, July 1, 2002
- 21:33:03
  - Alarm from Traffic Collision Avoidance System (TCAS)
- 21:34:49
  - Human air traffic controller command
- 21:34:56
  - TCAS recommendation
Boeing & Tupolew Collision

Überlingen, July 1, 2002

21:33:03
- Alarm from Traffic Collision Avoidance System (TCAS)

21:34:49
- Human air traffic controller command

21:34:56
- TCAS recommendation

21:35:32
- Collision

B757-200

TU154M

!
Boeing & Tupolew Collision

Überlingen, July 1, 2002

21:33:03 – Alarm from Traffic Collision Avoidance System (TCAS)

21:34:49 – Human air traffic controller command

21:34:56 – TCAS recommendation

21:35:32 – Collision

B757-200

TU154M

Official Inquiry Recommendation:

“pilots are to obey and **follow TCAS advisories, regardless of whether contrary instruction is given**”

⇒ Requires high confidence design
Formal Verification

Model of System

Formal Specification

Incorrect / Unknown

Correct

Verification (algorithmic)

Revise Design

TCAS verified in part [Livadas, Lygeros, Lynch, ’00]
Join Manoeuvre [Tomlin et al.]

- Traffic Coordination Problem
  - join paths at different speed
- Goals
  - avoid collision
  - join with sufficient separation
Join Maneuver

- Traffic Coordination Problem
  - join paths at different speed

- Goals
  - avoid collision
  - join with sufficient separation

- Models
  - Environment: Planes
  - Software: Controller
    - switches fast/slow

- Specification
  - keep min. distance

Disturbances
Join Manoeuvre

reachable states
blue plane

time

reachable states
yellow plane
Join Manoeuvre

Possible collision!

reachable states
blue plane

time

reachable states
yellow plane

[Join Maneuver [Tomlin et al.]]
"systems that users can bet their life on"

-D. Corman, NSF

cars…
airplanes…
pacemakers…