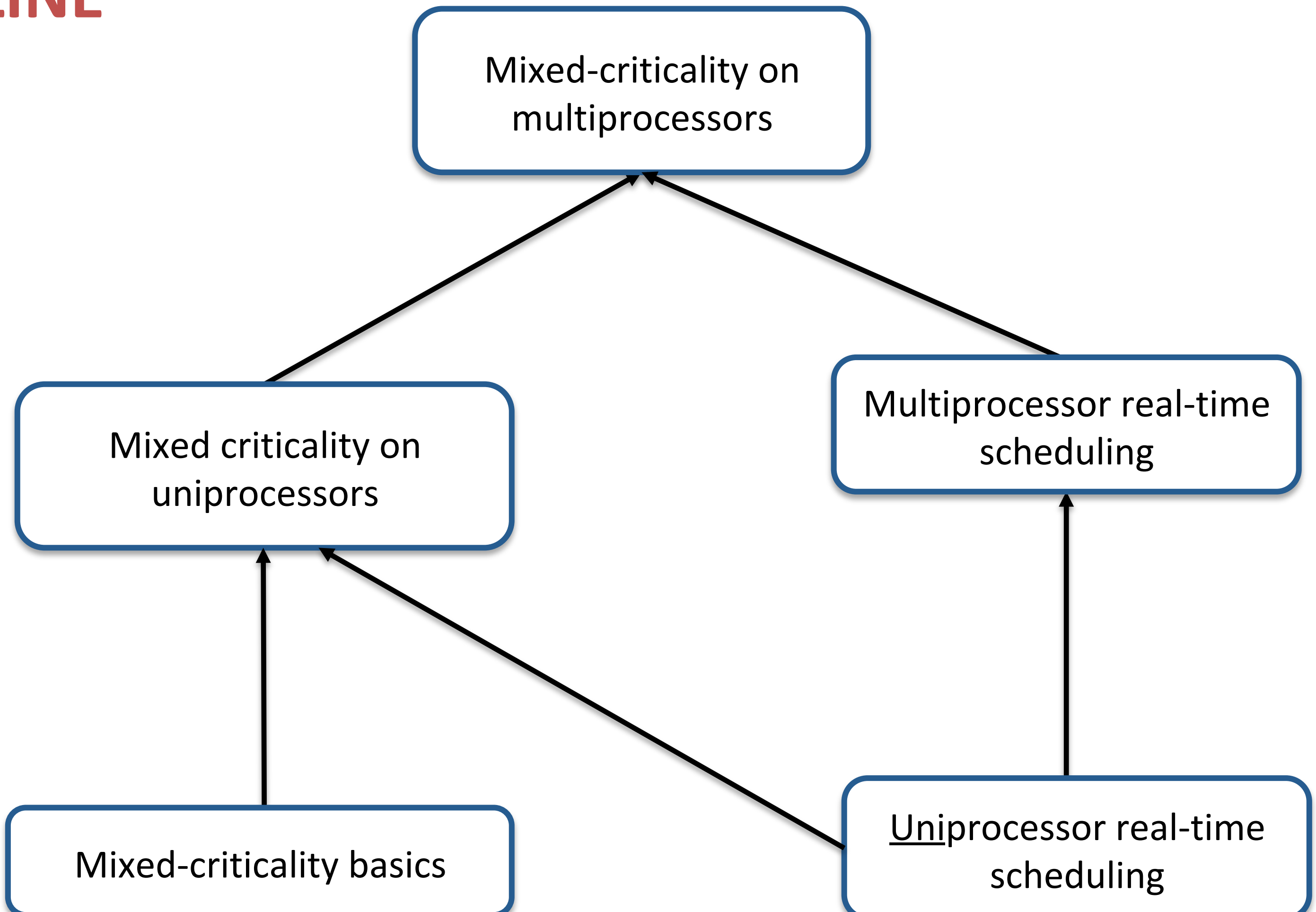
Multiprocessor Mixed-Criticality Scheduling

Sanjoy Baruah

Washington University in St. Louis

Multiprocessor Mixed-criticality Scheduling

OUTLINE



Predictability

- Functional predictability
- Timing predictability

The artefacts of computing are designed for functional predictability

Example

Input (float x, float y, time duration t)

Compute x*y within t time units

Functional correctness is the constraint and timing behavior the optimization objective

The formalisms of computing abstract away the concept of physical time

Timing predictability

A CPS = program + platform + environment

The environment

The program

The platform

- Enforce deterministic programming

Use special-purpose languages

E.g, the synchronous reactive (SR) languages

-A computation is a partial order of atomic actions

-Time advances in discrete steps of sufficient duration

Example: `x := a + b` on the Motorola PowerPC 755

- Best case: 3 cycles
- Worst case: 321 cycles

Advantage: simplicity

Disadvantages: resource under-utilization

Timing predictability

A CPS = program + platform + environment

- The environment
- The program
- **The platform**

- Enforce deterministic programming
- Enforce deterministic behavior
  - Cache partitioning
  - CAST-32 multicore recommendation

Trading off efficiency for determinism

Certification Authorities Software Team (2014). *Multi-core Processors*. Position paper CAST-32.

Timing predictability

A CPS = program + platform + environment

- The environment
- The program
- The platform

- Enforce deterministic programming
- Enforce deterministic behavior

Is the physical world deterministic?

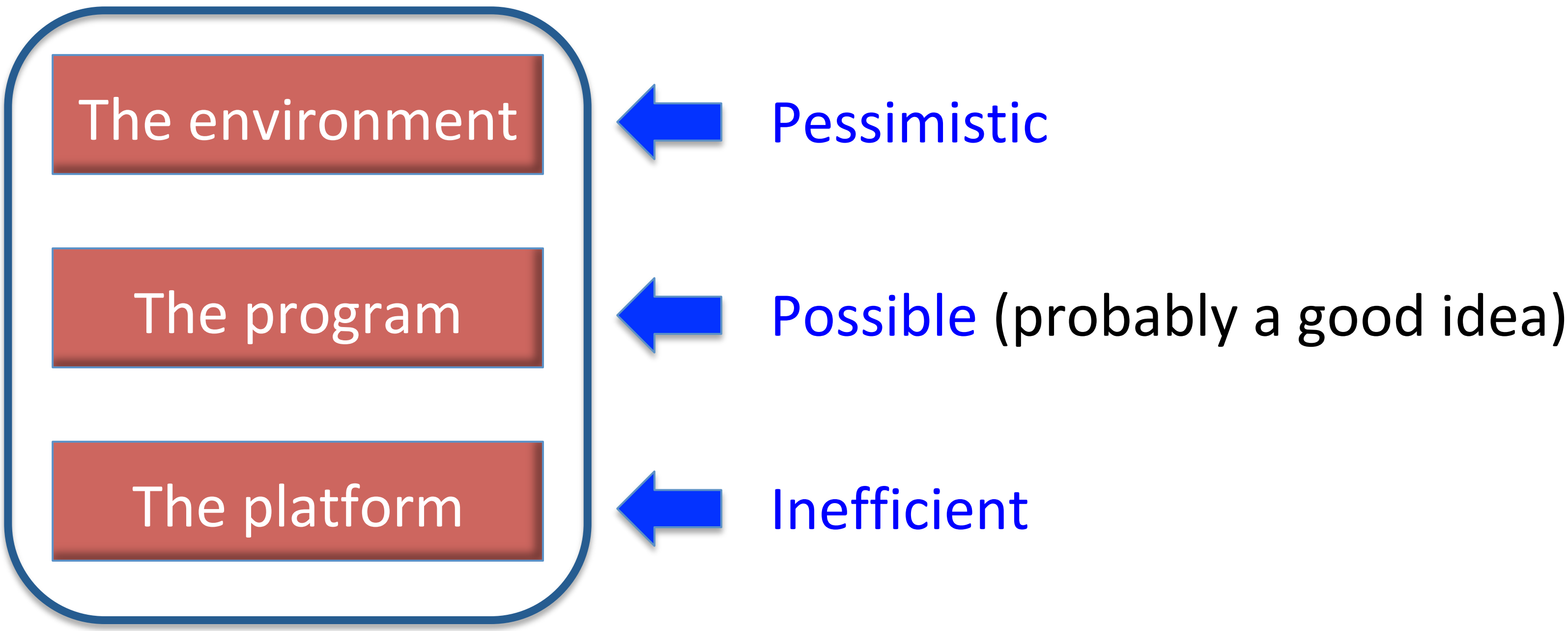
- We don't know
- It doesn't matter!
- Too complex to represent exactly

Deterministic models of event-triggered phenomena must incorporate pessimism

Edward Lorenz (1972). Predictability: does the flap of a butterfly's wings in Brazil set off a tornado in Texas? Talk at *American Association for the Advancement of Science* 139th annual meeting. Dec. 1972

Timing predictability via **determinism**

A CPS = program + platform + environment



Timing predictability – the mixed-criticality approach

All run-time properties are not equally important

Behavior emerges from three interacting models

- The environment
- The program
- The platform

Validation of properties is done under assumptions

… that depend upon the semantics of the property

Deterministic programs executing on non-deterministic platforms, interacting with a non-deterministic environment

An Illustration

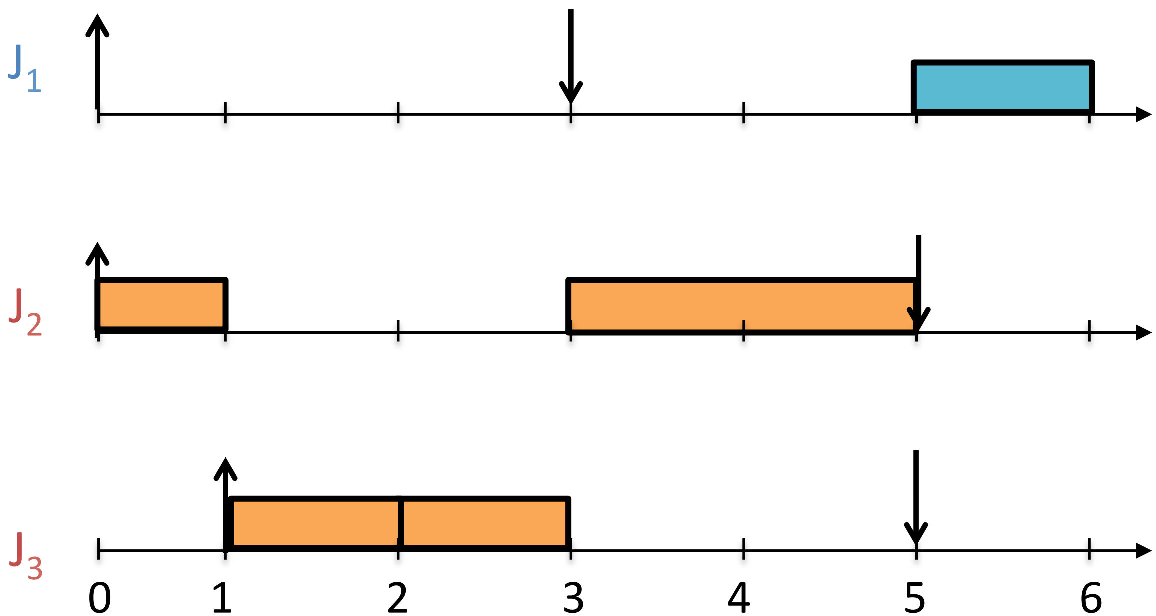
Execute J_2 over [0,1) and J_3 over [1,2)

if J_3 signals that it has completed

then execute J_1 over [2,3) and J_2 over [3,5)

else

1



Worst-case execution times (WCETs)

1	1
3	3
1	2

CA's WCET estimates

J_2 and J_3 are subject to certification

An Illustration

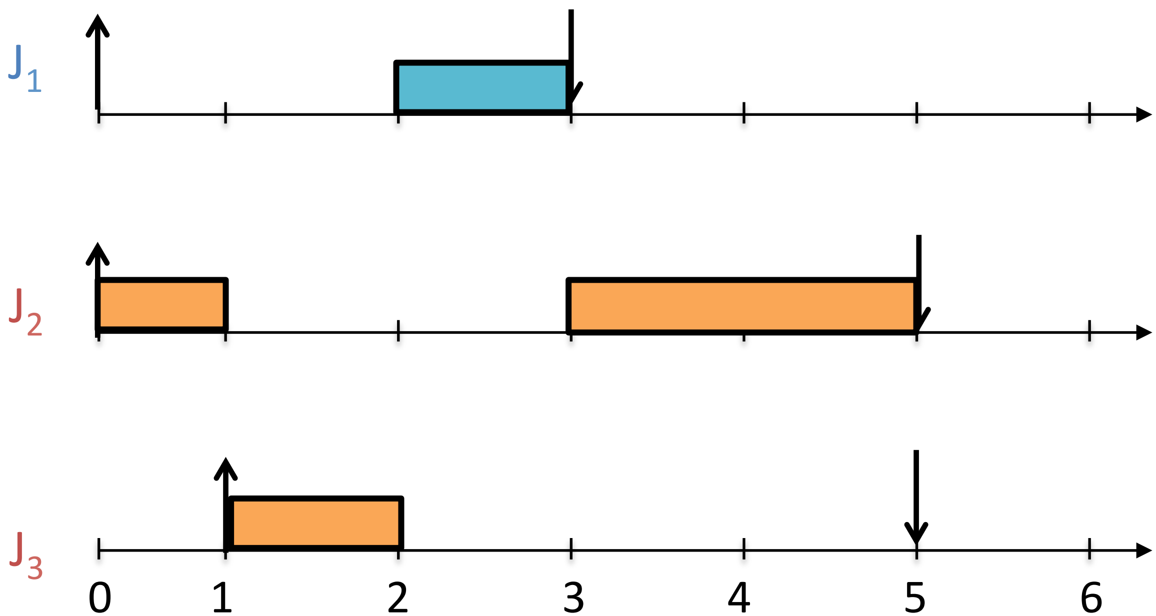
Execute J_2 over [0,1) and J_3 over [1,2)

if J_3 signals that it has completed

then execute J_1 over [2,3) and J_2 over [3,5)

else execute J_2 over [2,3) and J_2 over [3,5)

Validation by System Developer:



Worst-case execution times (WCETs)

1	1
3	3
1	2

J_2 and J_3 are subject to certification

An Illustration

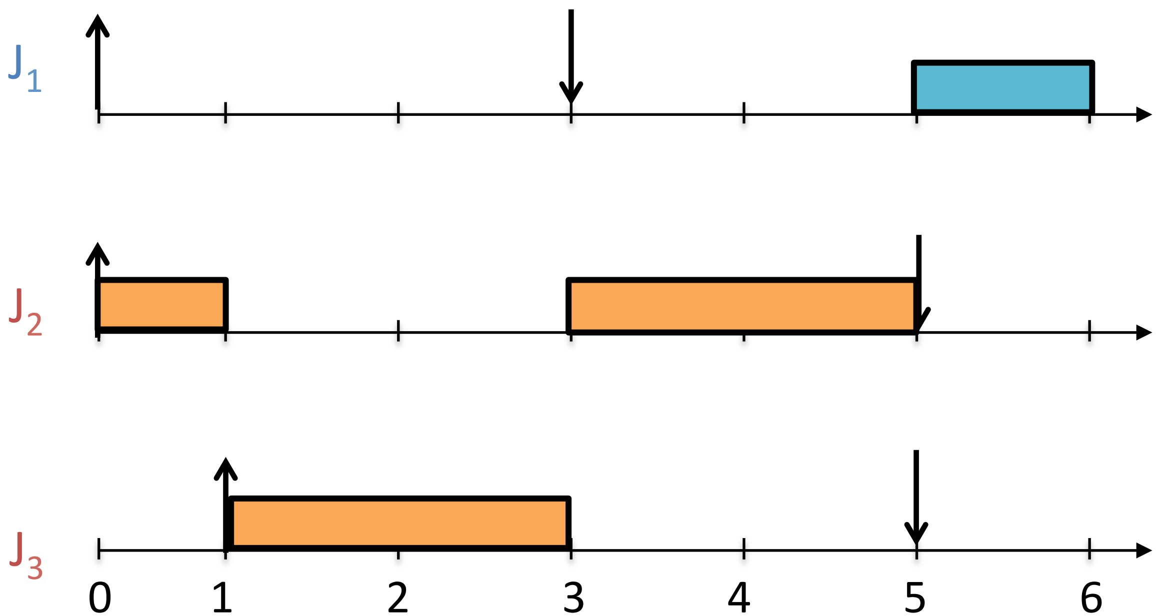
Execute J_2 over [0,1) and J_3 over [1,2)

if J_3 signals that it has completed

then execute J_1 over [2,3) and J_2 over [3,5)

else execute J_2 over [2,3) and J_2 over [3,5)

Validation by Certification Authority:



Worst-case execution times (WCETs)

1	1
3	3
1	2

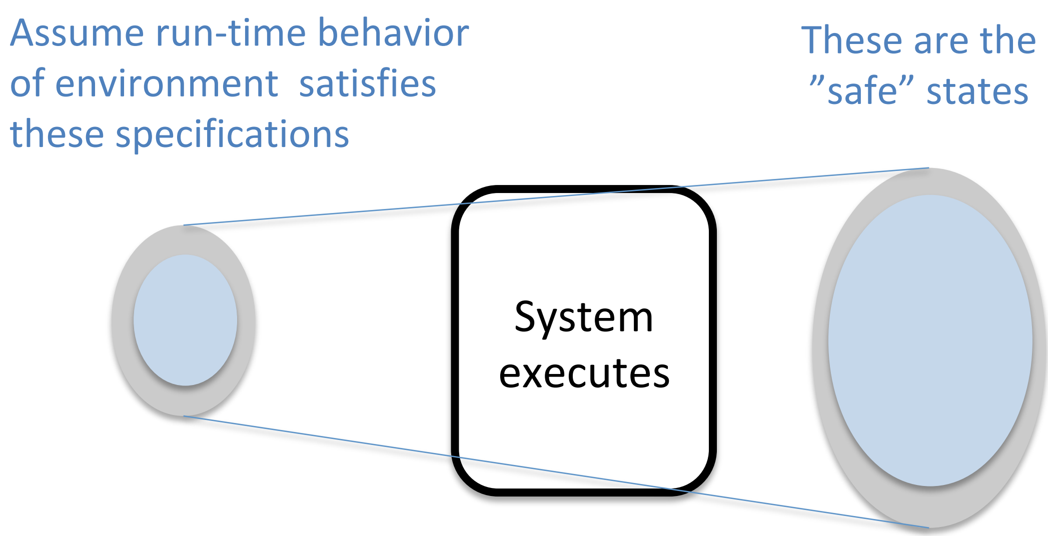
J_2 and J_3 are subject to certification

Mixed criticality: the verification perspective

Assume-guarantee reasoning

ROBUSTNESS: Guarantee holds even when assumptions do not

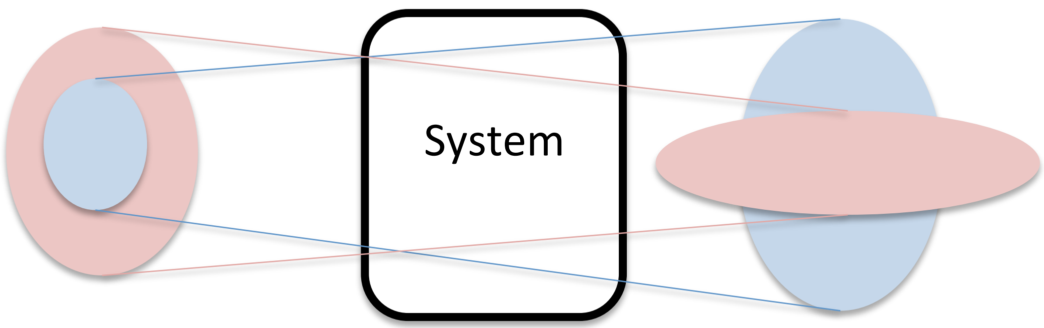
RESILIENCE: A graceful degradation of the guarantee when the assumptions do not hold



"Standard" mixed-criticality scheduling theory does **not** address robustness or resilience

Mixed criticality: the verification perspective

MIXED CRITICALITY: Synthesize a deterministic system to satisfy multiple assume-guarantee specifications



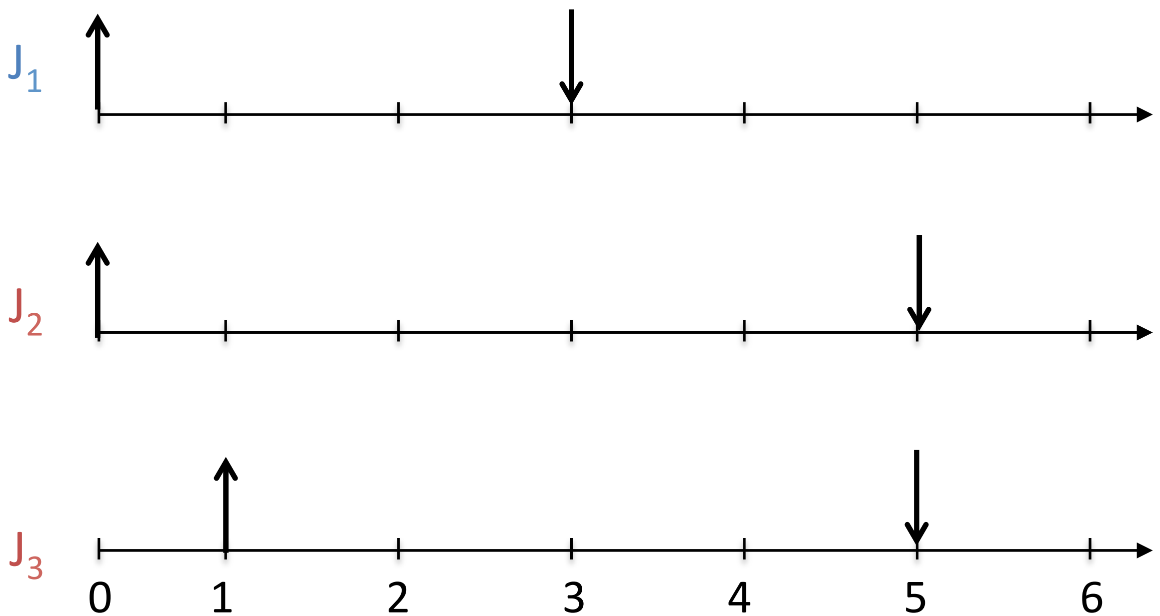
Notation

$J_i = (\chi_i, r_i, [c_i(LO), c_i(HI), d_i)$

$J_1 = (LO, 0, [1, 1], 3)$

$J_2 = (HI, 0, [3, 3], 5)$

$J_3 = (HI, 1, [1, 2], 5)$



Worst-case execution times (WCETs)

J_1	1	1
J_2	3	3
J_3	1	2

J_2 and J_3 are subject to certification

Behaviors

$J_i = (\chi_i, r_i, [c_i(LO), c_i(HI), d_i)$

During an execution of the system, J_i signals completion after executing for p_i time units

if $p_i \le c_i(LO)$ for all jobs J_i, LO-criticality behavior

else if $p_i \le c_i(HI)$ for all jobs J_i, HI-criticality behavior

else erroneous behavior

Correctness

A mixed-criticality scheduling algorithm is correct

if all jobs meet their deadlines in LO-criticality behaviors

and all HI-criticality jobs meet their deadlines in HI-criticality behaviors

Notions of schedulability

A clairvoyant scheduling algorithm knows the p_i values beforehand

- A hypothetical abstraction

An on-line(OL) algorithm only knows p_i when J_i signals completion

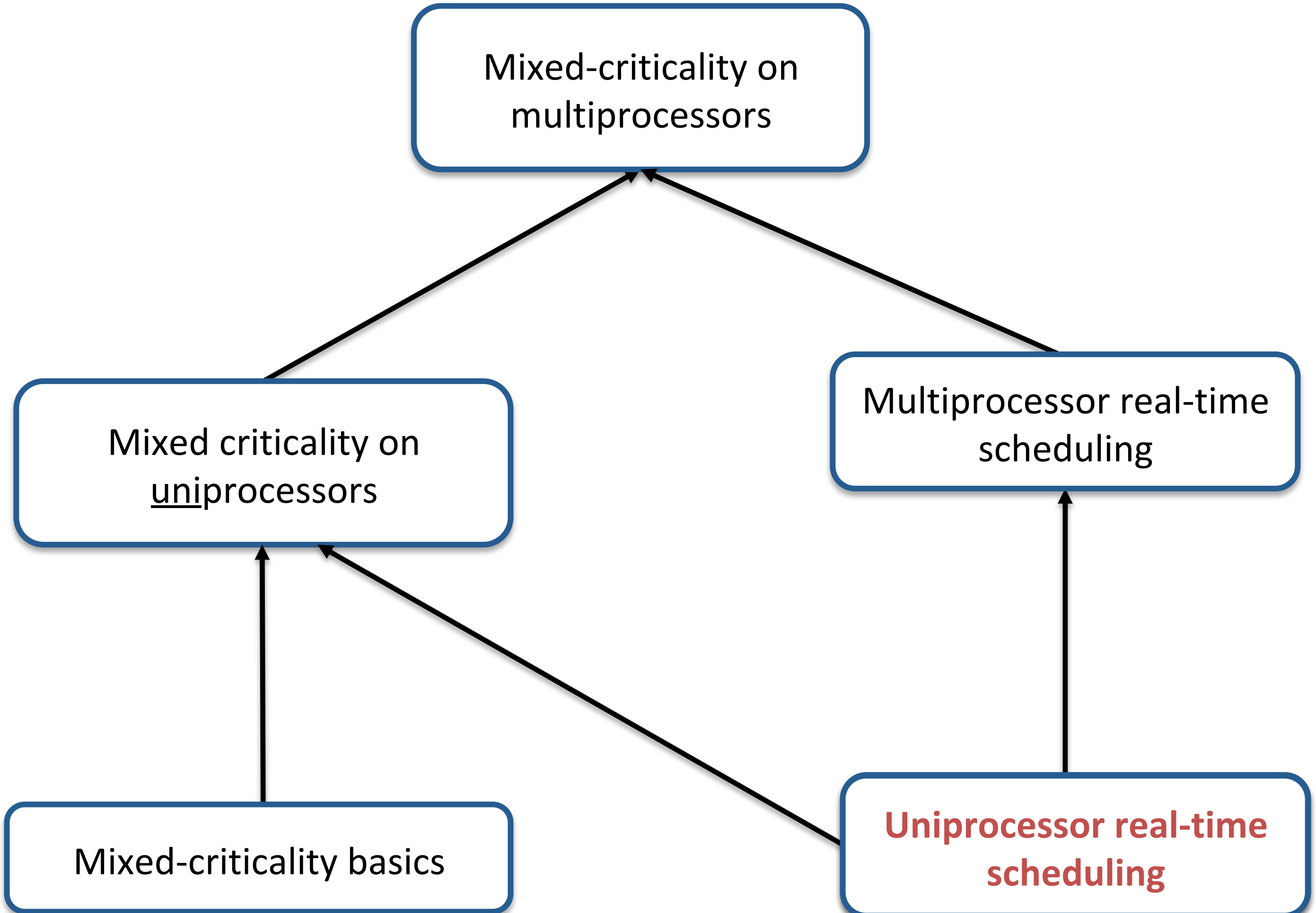
Clairvoyant-schedulable and **Mixed-criticality** (**MC**) schedulable

Result: Not all clairvoyant-schedulable instances are MC-schedulable

Speedup factor of an OL algorithm **Alg**: "*any instance that is clairvoyant-schedulable is* **Alg***-schedulable upon a processor that is s times as fast.*"

($s \geq 1$)

Outline



Mixed-criticality on multiprocessors
Mixed criticality on uniprocessors
Multiprocessor real-time scheduling
Mixed-criticality basics
Uniprocessor real-time scheduling

Uniprocessor Real-Time Scheduling

- The Earliest Deadline First (EDF) scheduling algorithm
- Optimality of EDF on preemptive uniprocessors
- The sporadic tasks model
- EDF scheduling of sporadic task systems