Runtime Verification Challenges towards RTA

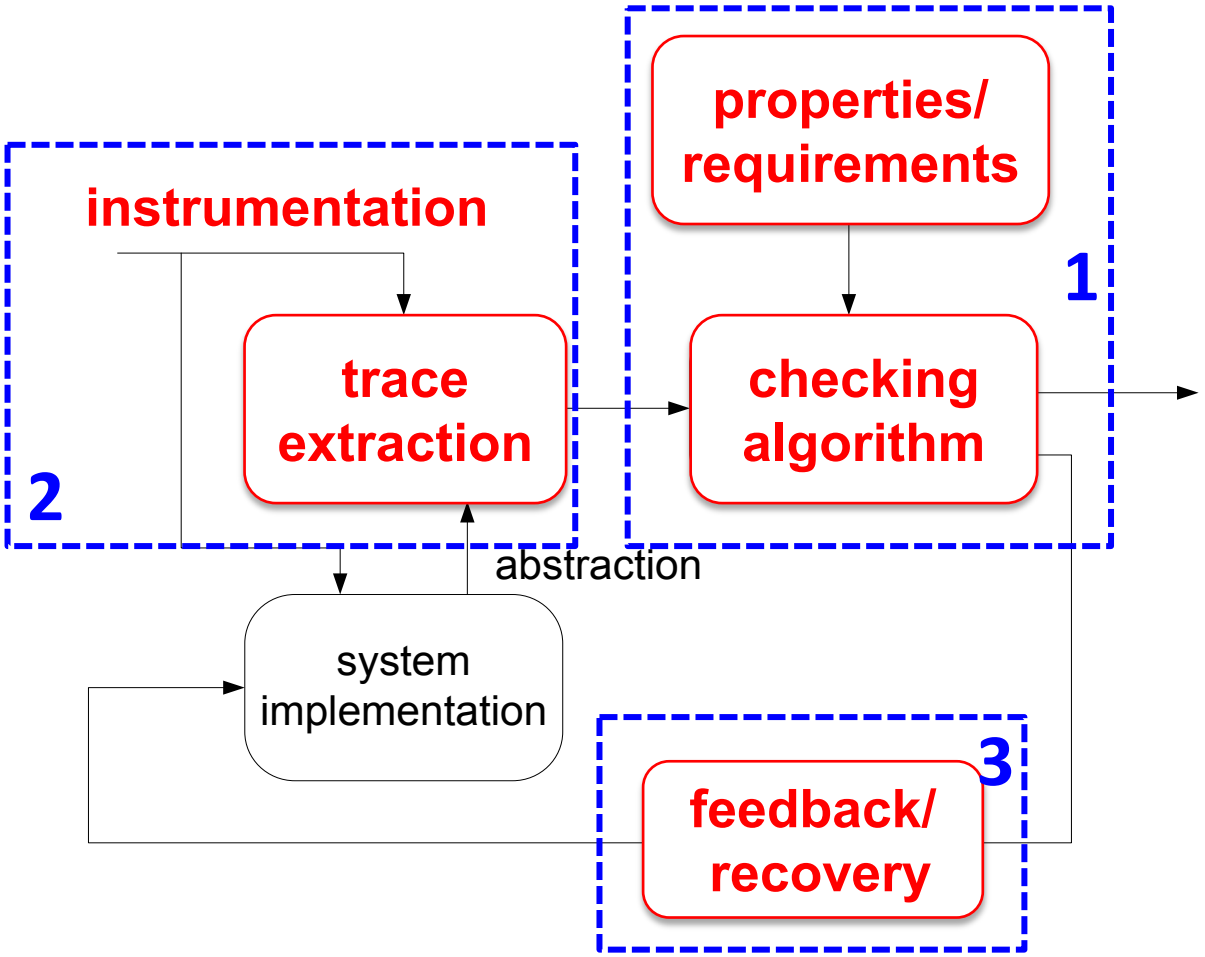
Insup Lee **Oleg Sokolsky**

PRECISE Center
University of Pennsylvania

Objectives

- Identify challenges and potential approaches related specifically to runtime verification within the RTA problem domain

RV building blocks revisited



There are cross-cutting and specific challenges associated with the three aspects, and all of them are **inter-dependent**.
(Solutions to one directly influence the issues + solutions of the other.)

Terminology

- Properties of interest
  - Properties that the **overall system needs to guarantee**
  - e.g., stability
- Run Time Verification (RV) properties
  - Properties that the **RV component needs to check**
  - e.g., switching conditions

What properties of interest (to guarantee)?

- Safety is composed of many different properties
  - e.g., stability, causality, within bounded area of state space.
- Are we interested in properties other than those related to safety?
  - e.g., performance, Quality of Service (QoS)
- The answers to all aspects of RV depend on what these properties are
- **We assume these properties are given to us**
  - e.g., properties related to safety should be provided by the control-layer

What RV properties (to check)?

- Given a property of interest, the main challenge is:

  How to "transform" a property of interest into a set of RV properties?

- We can only observe the past; however, in general, guarantees are about the future, not about the past
  - e.g., we need to guarantee *stability*, but *stability* is not what we monitor
  - Hence, RV properties are derived from, but are not the same as the properties we need to guarantee.

- The transformation must ensure that, *given a certain steering mechanism and assumptions about the environment, the RV properties will guarantee the property of interest over a certain assumption*

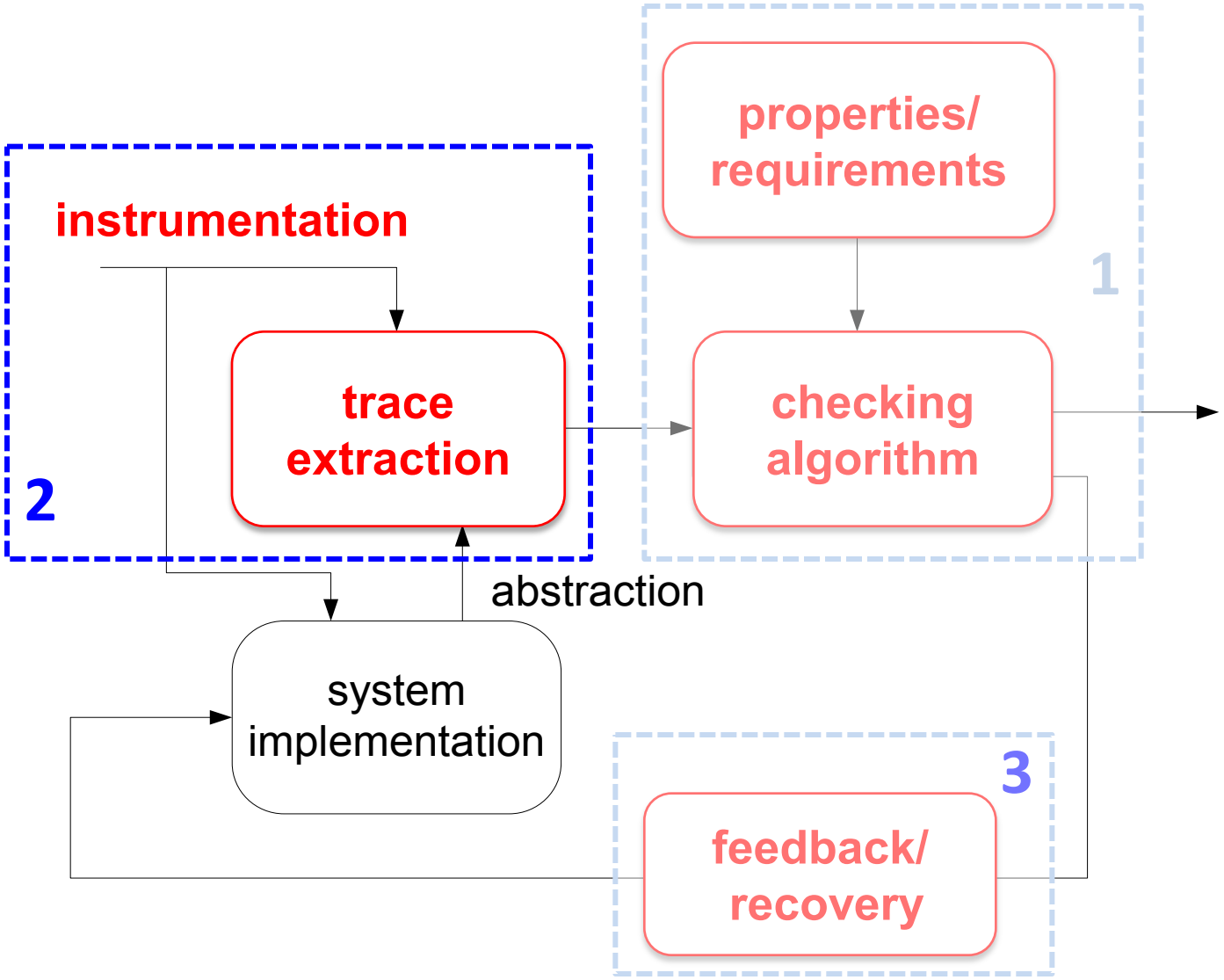
Which specification languages are suitable for the RV properties?

- **Expressiveness of the specification language is co-related to how easy/difficult checking is done**
  - In the past, RV property languages were temporal logics or state invariants, for which we mostly know what can be checked and how efficiently. It is not clear if they are adequate for RTA.
  - May need different formalisms for different types of properties
    - e.g., properties of hybrid systems vs of discrete event systems
  - May depend on what types of traces can be generated
  - Should we develop logics for dense-time real-value signals?
- **RV properties may change depending on the system state; how to capture this dynamism?**
- **How to translate the RV properties into the language?**

Checking an RV property

- How to check efficiently if a trace satisfies a property P?
- What is the deadline for checking P?
- How fast P can be checked (i.e., reaction time)?
- How to handle dynamism of P?
  - P may change depending on the system state
  - P depends on different sets of monitored variables in different modes of operation
  - **How to check for P during a mode switch?**
    - Mode switch can be due to recovery or due to nature of the system (e.g., different stages of operation)

Next: Challenges in observations



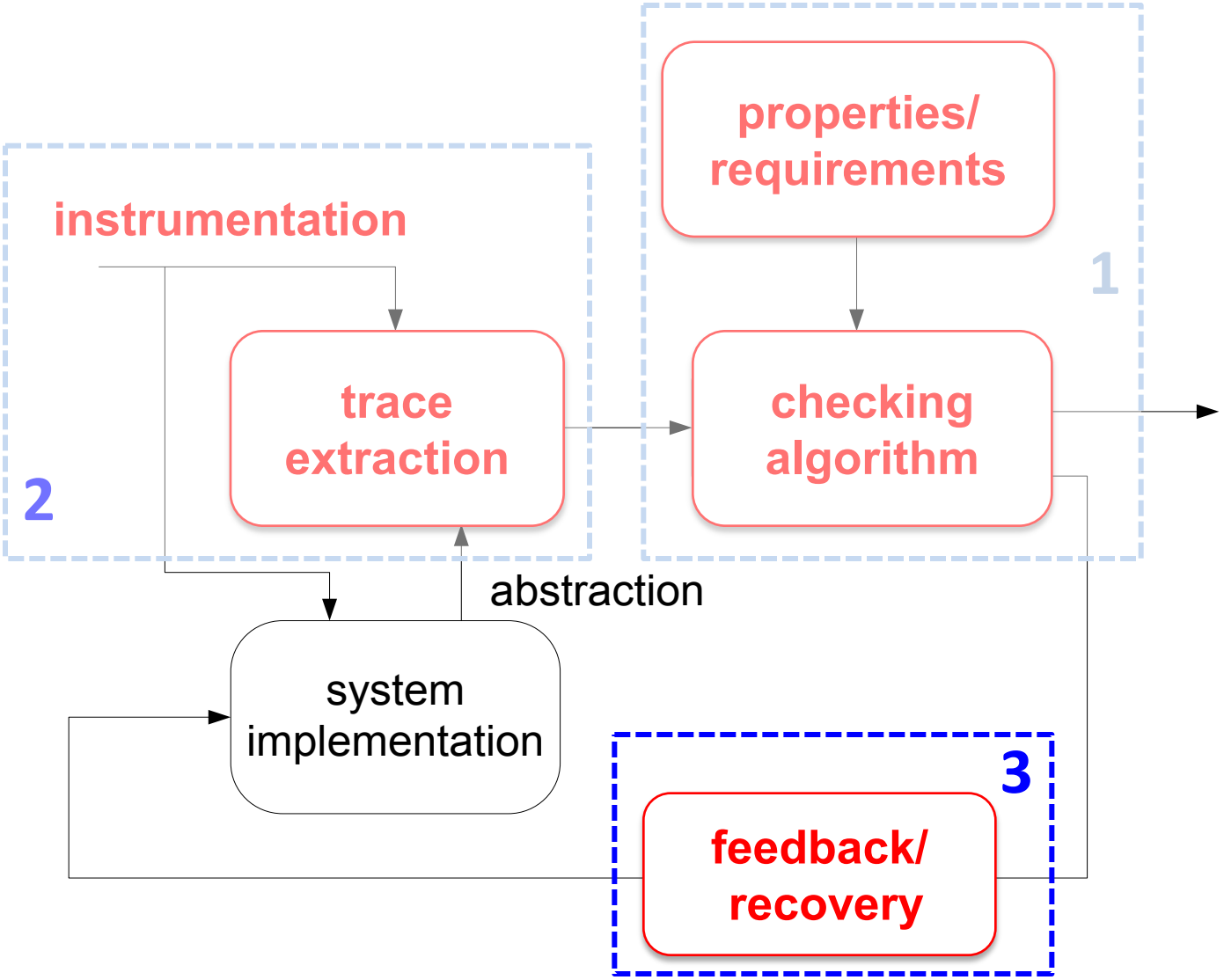
Trace generation for an RV property

- **What to monitor/observe to make the checking of an RV property P possible?**
  - What are the relevant variables to observe?
  - What system variables are observable?
  - If a variable is unobservable, can we derive it?
  - If not, can we substitute it with something else?
  - If not, can we still conservatively check P with the partial information (inferred from the observable ones)?
- The above knowledge may come from model-based design (e.g., input/output interfaces)

Trace generation for a property P

- **How to extract the observations?**
  - Too much or too little information is bad
- Potential solutions
  - Sampling-based vs event-based
  - Active vs. passive monitoring
    - e.g., snooping network for information
  - **General system architecture to support RV?**
    - e.g., a universal pub/sub framework, shared information/data bus for monitoring

Next: Challenges in feedback/recovery



Feedback/Recovery

- **How can we derive the feedback/recovery strategy from the properties of interest?**
- **What to do when an RV property P is violated?**
- Many types of recovery actions/mode switches, depending on what P is, e.g.,
  - E.g., if P is related to stability, need to switch to a safety controller
- Feedback/recovery logics needs to be determined at design time
  - Typically comes from the layer where P is derived
  - **We assume that (part of) the logics is given to us?**

Analysis of recovery

- How do we analyze effects of invoking the recovery mechanism
  - How to ensure that actions for a detected property do not adversely affect the system behavior or other properties?
- When is it safe to invoke the actions?
  - Is “immediately” always safe or possible?
  - e.g., some state information needs to be saved before switching

Feedback/Recovery Implementation Challenges

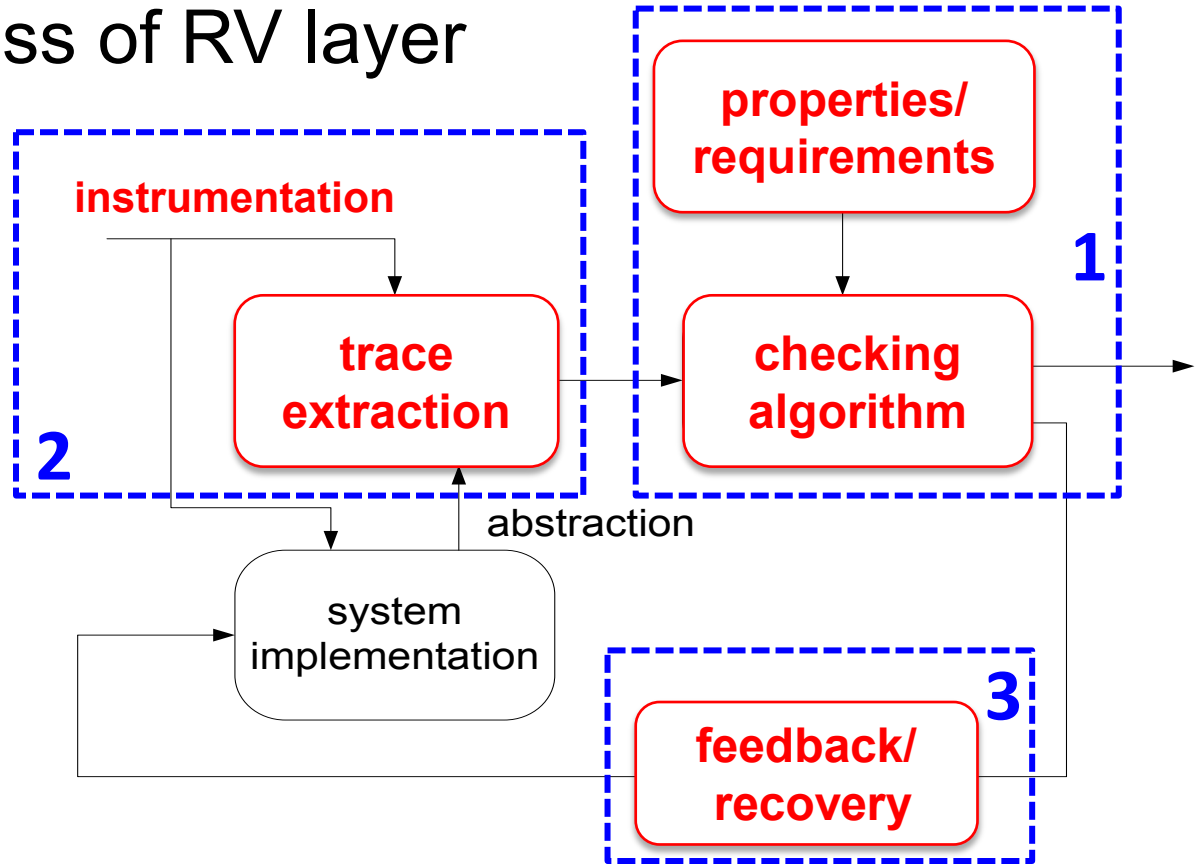
- **How to invoke recovery action?**
  - What are the components involved in this action, e.g., when switching from the experimental controller to the baseline controller?
  - How to update each component, e.g., modify the control variables?
- **How fast should the recovery actions be taken?**
  - What are the impacts on the system if this cannot be done?

Feedback/Recovery Implementation Challenges

- **What are the impacts of recovery actions on the RV component itself?**
  - When switching to a new mode, the set of variables/properties may be modified
    - How to update the RV efficiently?
    - Observe all required variables even if they are not active at a certain mode will be too expensive
- **HW/SW layer needs to account for the effects of such actions**
  - e.g., resource must be sufficient to ensure schedulability during mode switches

Cross-cutting challenges

- Managing overhead
- Dealing with uncertainty
- Trustworthiness of RV layer



How to manage RV overheads?

- RV overheads affect the system's behavior and performance in many ways, e.g.,
  - recovery actions take too long to complete may lead to system failure
  - monitoring and checking takes too much shared resources, resulting in control tasks missing deadlines and packets dropped
  - quality of control is inherently affected due to degrading timing performance
- RV overhead optimization must not affect the behaviors of other components in the system
  - e.g., when information is shared

How to manage RV overheads?

- Potential approaches
  - Combine with offline techniques such as
    - static analysis
    - automata symbolic technique
    - complier optimization
  - Sampling-based instrumentation, time-triggered monitoring and checking
    - Other issue needs to be addressed, e.g., time synchrony

Dealing with uncertainty

- How to RV when observations are partial?
  - Observability issue (due to cost, legacy systems)
  - Due to hardware/software defects
- Can we infer hidden information from other sources
  - Sensor redundancy?
- Can we check properties with hidden information?
  - Probabilistic models?

Trustworthiness of RV layer

- Monitoring and checking layer becomes a critical part of the system
- Need to ensure it is verifiable
- There is hope:
  - Relatively simple logic
  - Amenable to formal verification
  - Model-based implementation should help
- Fault tolerance needs to be addressed
  - Currently not our focus

Dealing with legacy systems

- What can we assume about legacy systems?
  - What information is accessible?
    - Affect what observations are available and how instrumentation is done
  - Which elements can be modified?
    - Affect how feedback/recovery is possible

Discussions

Backup slides

More on Decomposing a property P

- Need to account for overheads and tradeoffs between different types of overheads
  - Different ways to decompose P may exist, some of which may be less expensive than the others
  - A central solution requires a faster processor resource, whereas a distributed architecture puts more load on the network
- Which decomposition is best for P depends on what are available
  - e.g., a fast processor or a fast network?

More on Decomposing a property P

- RV in a distributed setting: two approaches
  - Centralized checker, distributed event recognizers
    - i.e., one checker checks for the properties (that guarantee P) based on collective observations delivered by different event recognizers
  - Distributed checkers, distributed event recognizers
    - i.e., composition of the properties checked by all the checkers will guarantee P
- Two corresponding challenges
  - How to produce the observations required by P from observations that are available?
  - How to decompose P into distributed properties to be checked by each distributed checker?