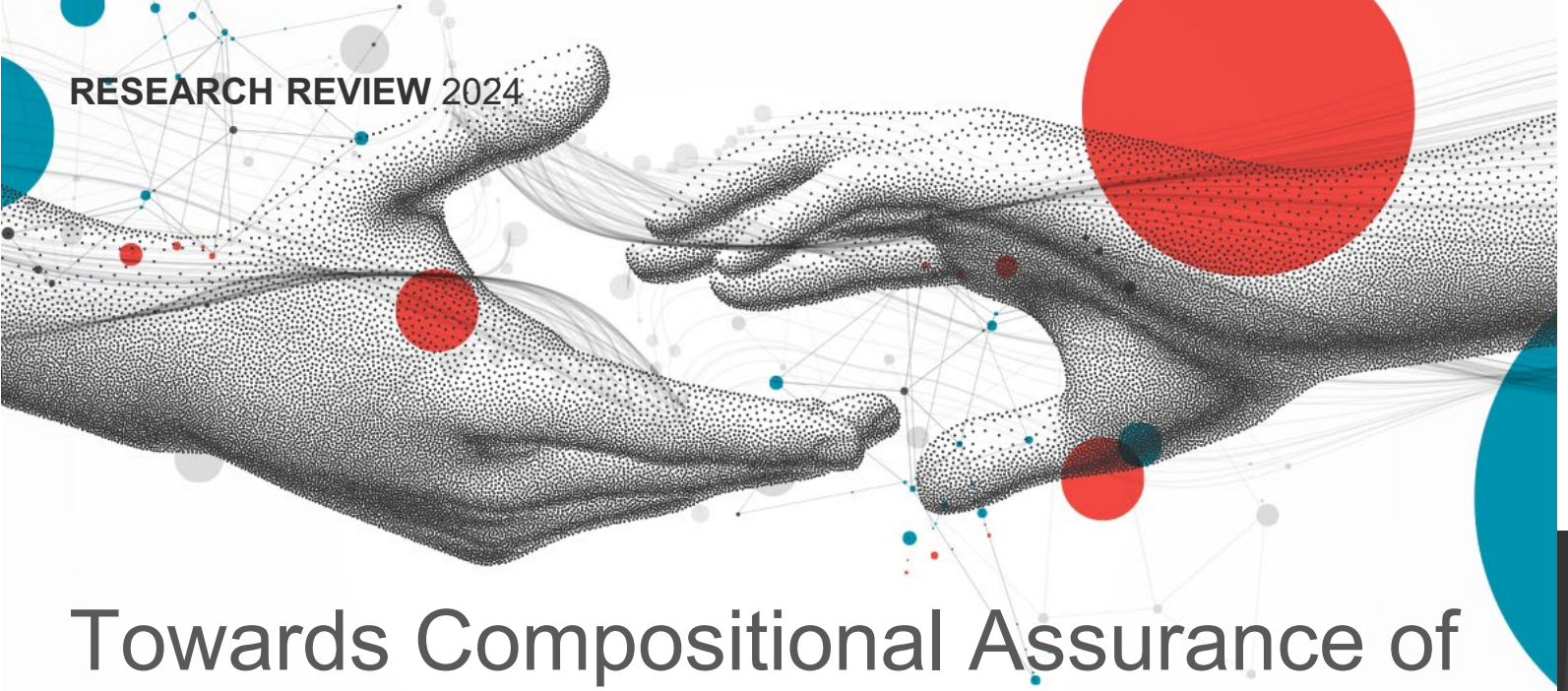
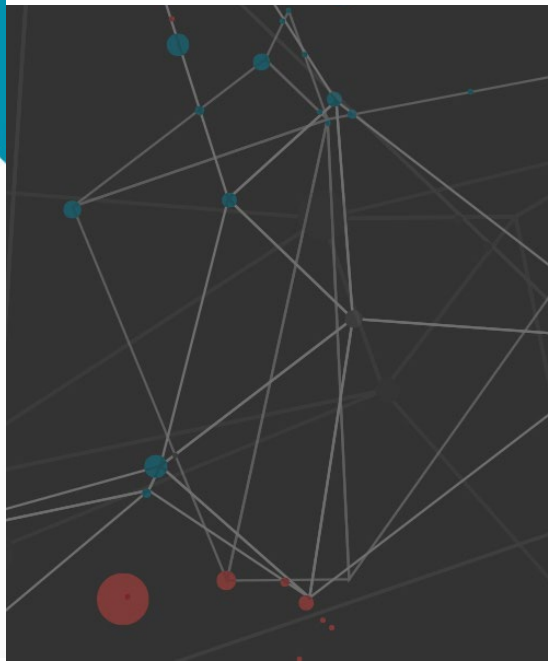


RESEARCH REVIEW 2024

Carnegie Mellon University
Software Engineering Institute

Towards Compositional Assurance of Large-Scale Systems



NOVEMBER 13, 2024

Dr. Gabriel Moreno
Principal Researcher

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM24-1447

Introduction

Delivering capability at speed and scale is a top priority for the Department of Defense (DoD).

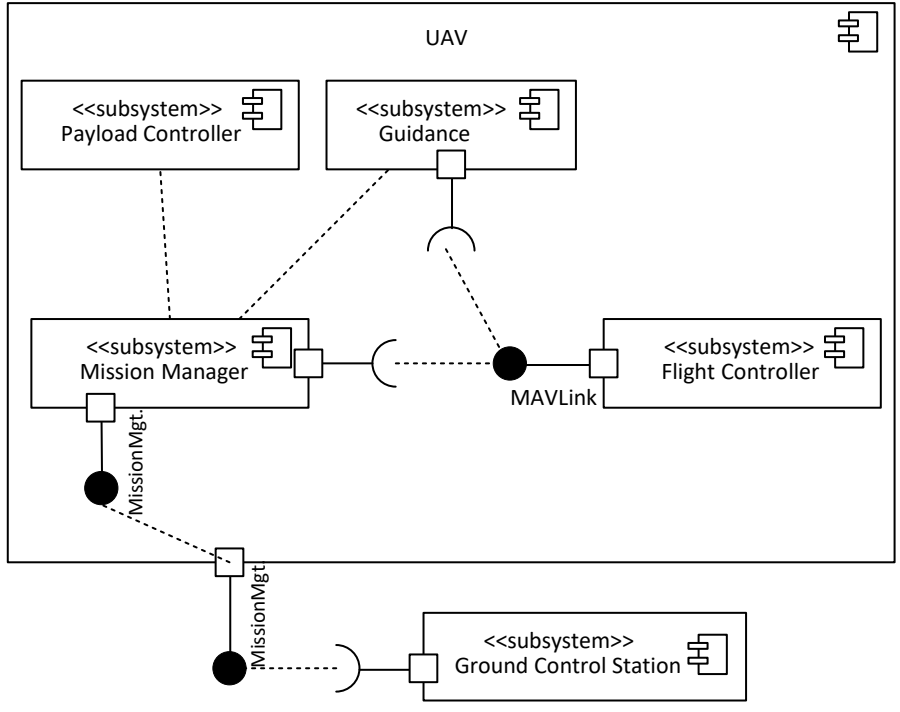
Assurance of evolving large-scale systems is a bottleneck in achieving that goal.

We are developing an approach to overcome the barriers that slow down assurance.

Carnegie Mellon University
Software Engineering Institute

Overview

Drone Example



Drone Component Diagram

Problem

Assurance of evolving large-scale software-intensive systems is a bottleneck in **deploying DoD capabilities with speed and confidence**.

Factors include the following:

- lack of effective reuse of assurance results
- inability to integrate multiple types of assurance analyses
- no notion of different levels of trust
- assurance interdependence between subsystems

Two Analysis Levels

Domain-Specific Analysis

- different analysis domains
  - timing, safety, security, etc.
- different artifacts
  - binary code, source code, timing models, state machines, etc.
- different analysis mechanisms
  - inspection, testing, simulation, model checking, theorem-based

Composition of Analyses

- Integrate results from diverse domain-specific analyses.
- Integrate analysis results for different parts of the system.
- Check the logical soundness of the integration.
- Determine the trust level of composed analysis results.

Key Considerations for Composing Analyses

- Identify the specific analyses used.
- For each analysis, do the following:
  - Identify **assumptions** about inputs needed for the analysis.
  - Be specific about the **guarantee** offered by the analysis.
  - Be as comprehensive as possible in identifying all **resources**—the factors that affect the computed result.
  - Specify the values for those resources for which the analysis shows that the guarantee holds. This determines the **trust level** of the guarantee.
- Make sure that all analysis **assumptions are satisfied** by other guarantees or axioms.
- Check that there are **no logical inconsistencies**.

Solution Approach

- Structure assurance analyses in an **argument architecture**.
- Hide analysis details using logical **judgments** as components in an argument architecture.
- Formally introduce **levels of trust**.
- **Automate the analysis of the composition** of judgments.

Compositional Assurance

Carnegie Mellon University
Software Engineering Institute

Argument Architecture

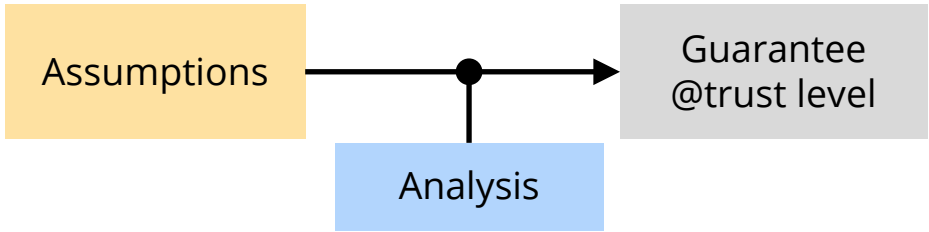
The argument architecture is the set of arguments needed to reason about the system, which comprises the following:

- the judgments that encapsulate the results of domain-specific analyses
- the logical connectives that combine judgments

The argument architecture describes the following:

- how guarantees from one analysis satisfy assumptions of another
- how trust flows from assumptions, through analyses, to guarantees

Judgment



A judgment is a first-class component in an argument architecture motivated by a new logic system that hides the details of an assurance analysis and provides a composition interface based on the following:

- assumptions
- guarantees
- trust levels (TLs)

A judgment in logic is a declaration that the expressed proposition is true. We use it to represent that an analysis has already shown that the guarantee holds given some assumptions.

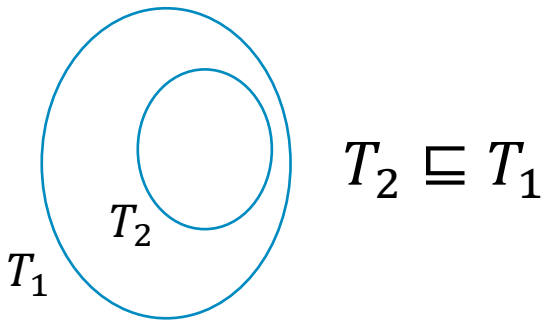
Trust Level

- Several factors affect the trust we have in the assurance guarantee:
  - rigor, fidelity, coverage
- In this work, we focus on the **coverage of the analysis** to define the trust level of a guarantee.
- We characterize coverage with the concept of **resource** (anything that influences a computation):
  - for example, inputs, configuration parameters, internal state

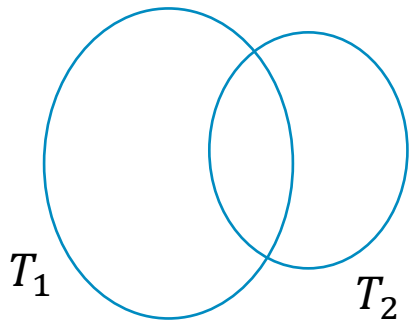
The trust level qualifies the guarantee in terms of the context under which the guarantee has been shown to be true.

Trust Level Formalization

- A single situation or test case covered by an analysis is a function that assigns values to each of the component's resources.
  - For example, if the resources of component C are $\{vel, alt\}$, an example of a test case is the function $\{(vel, 40), (alt, 200)\}$.
- The trust level of a guarantee is defined as a set of those functions.
- A partial order over trust levels can be defined.
  - An increasing trust level means a larger context in which a guarantee holds.



$T_2 \sqsubseteq T_1$

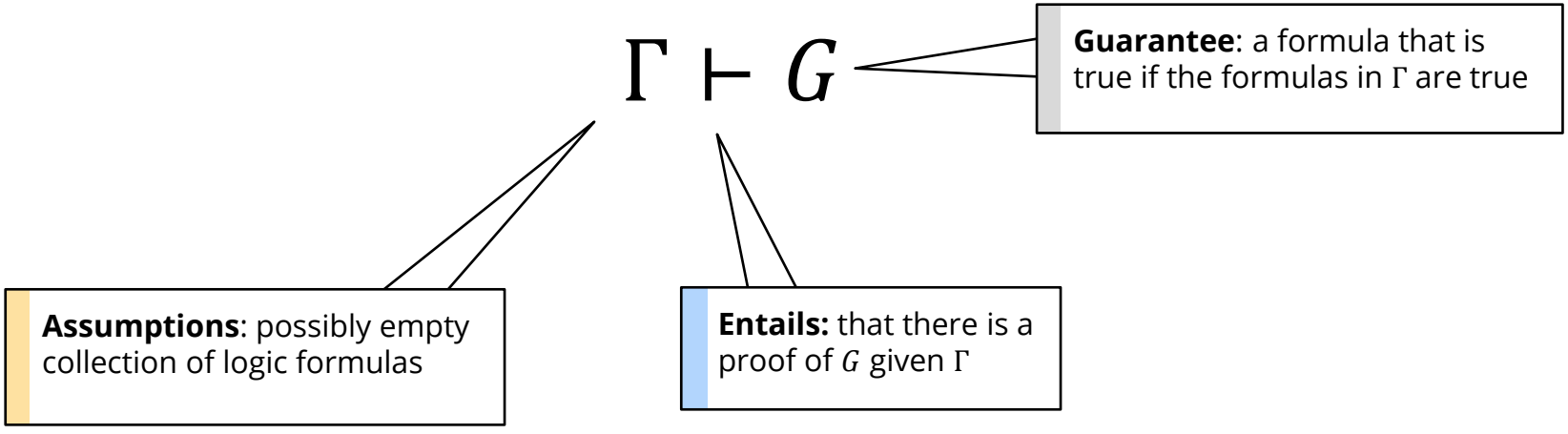


$T_1 \cap T_2 \sqsubseteq T_1$

$T_1 \cap T_2 \sqsubseteq T_2$

Logic Foundations

General Form of a Judgment

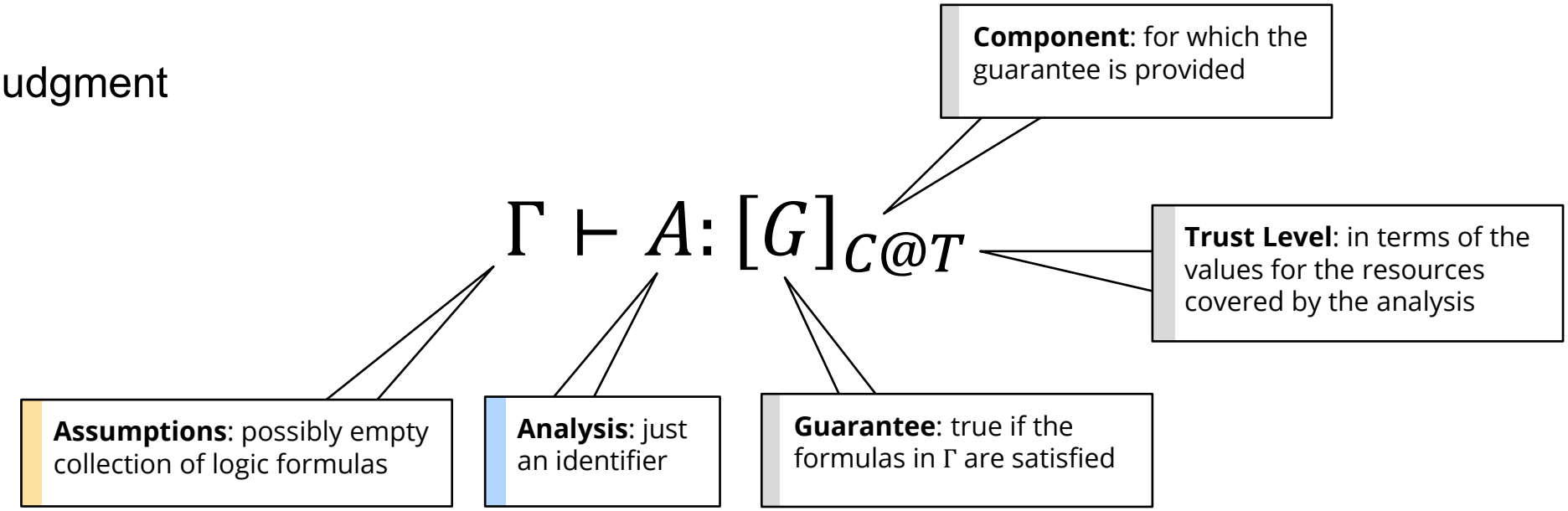


The judgment asserts that G can be proven given Γ.

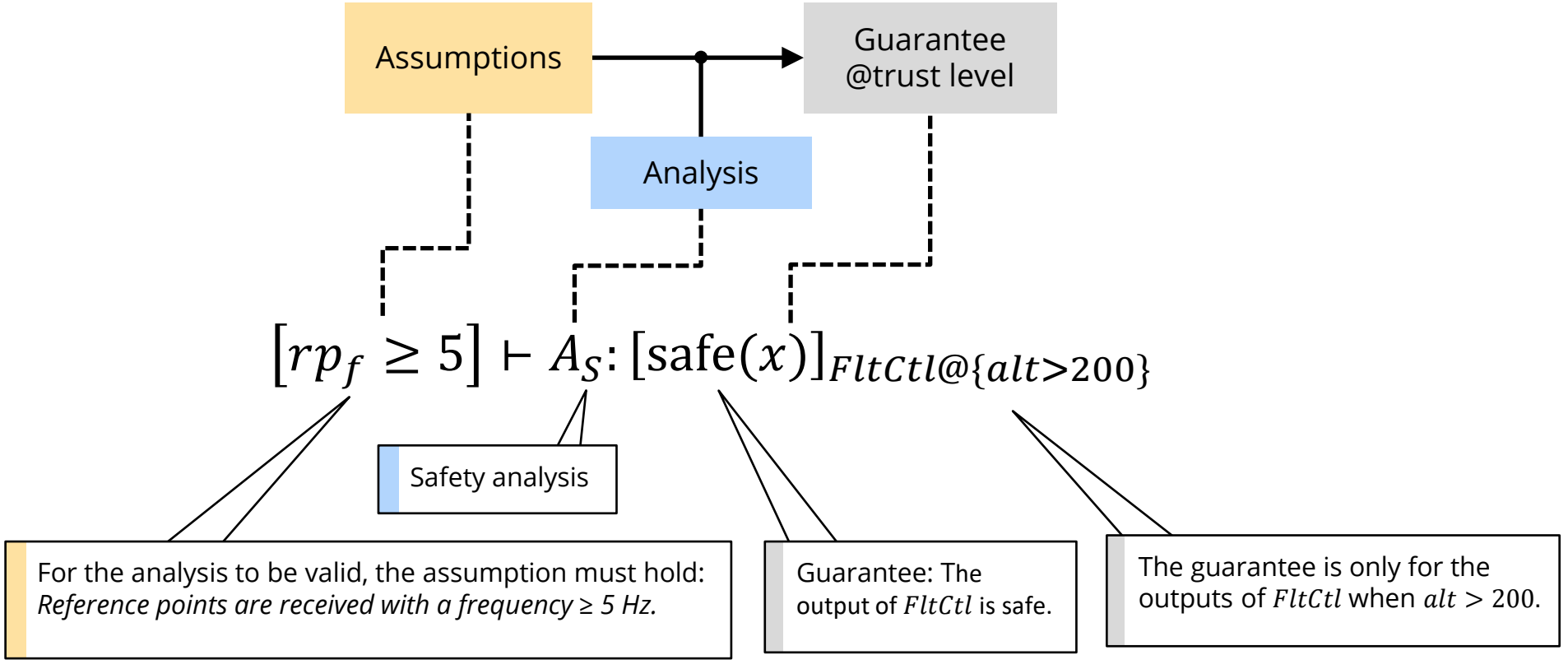
Logic Foundations: Introducing Trust Levels

We have developed a new logic system that enables rely-guarantee reasoning to compose analyses with different trust levels.

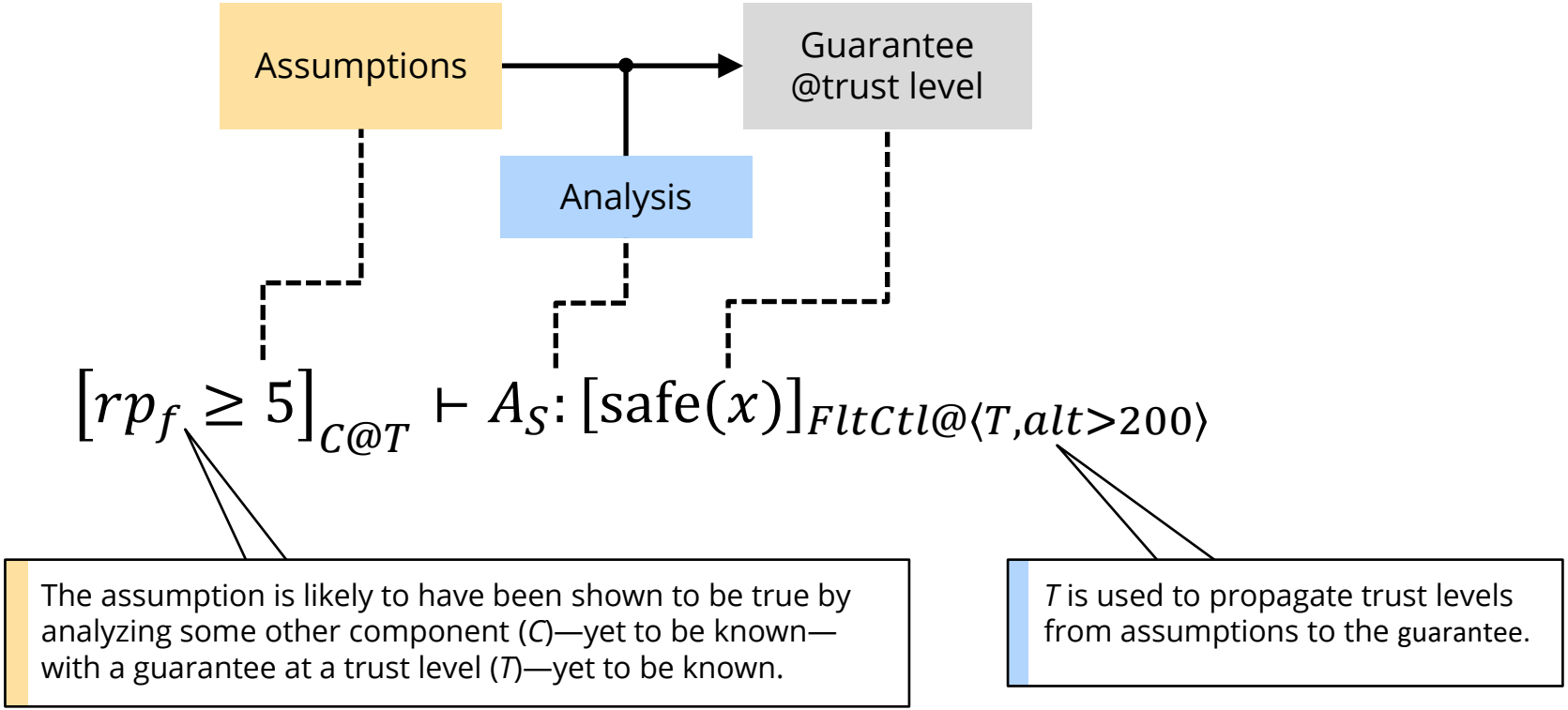
Judgment



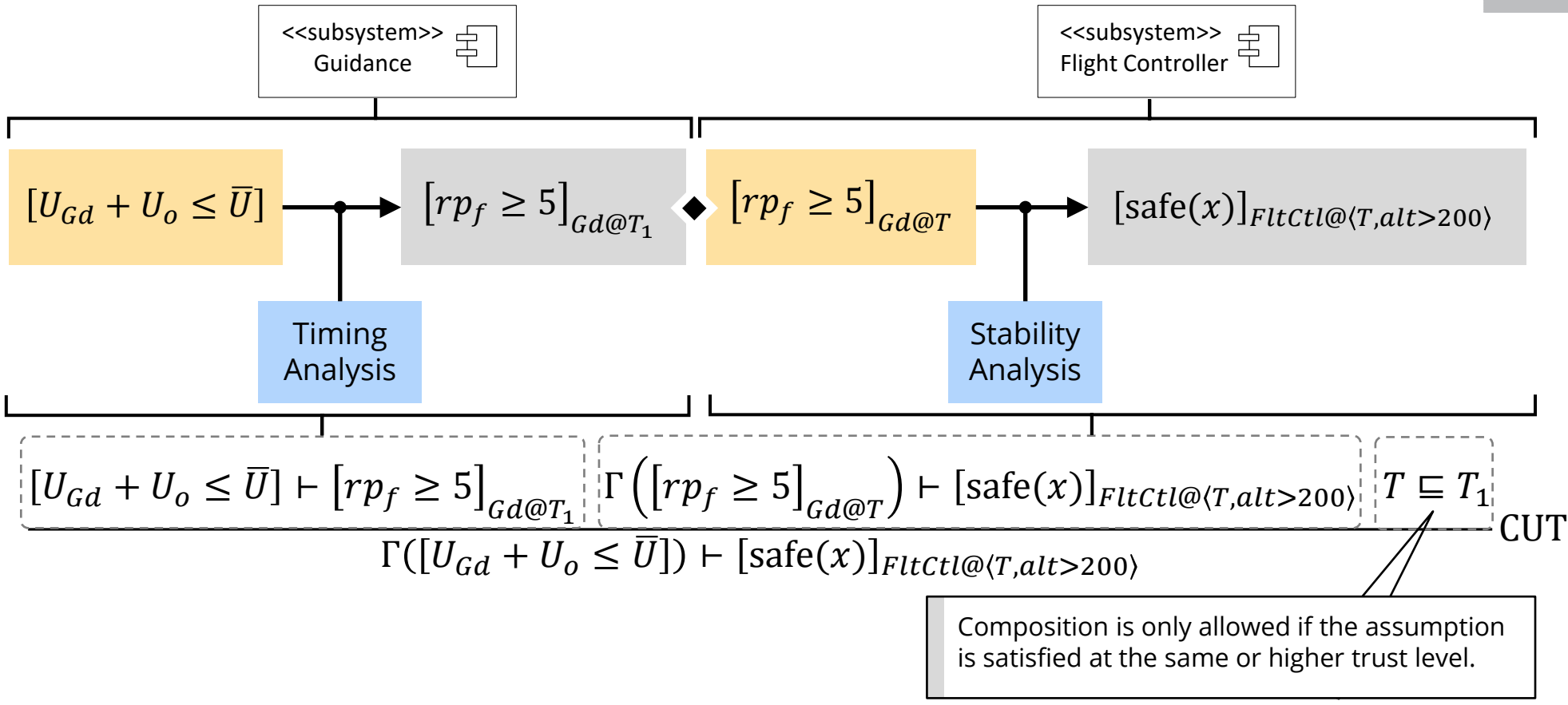
Judgment Example –1



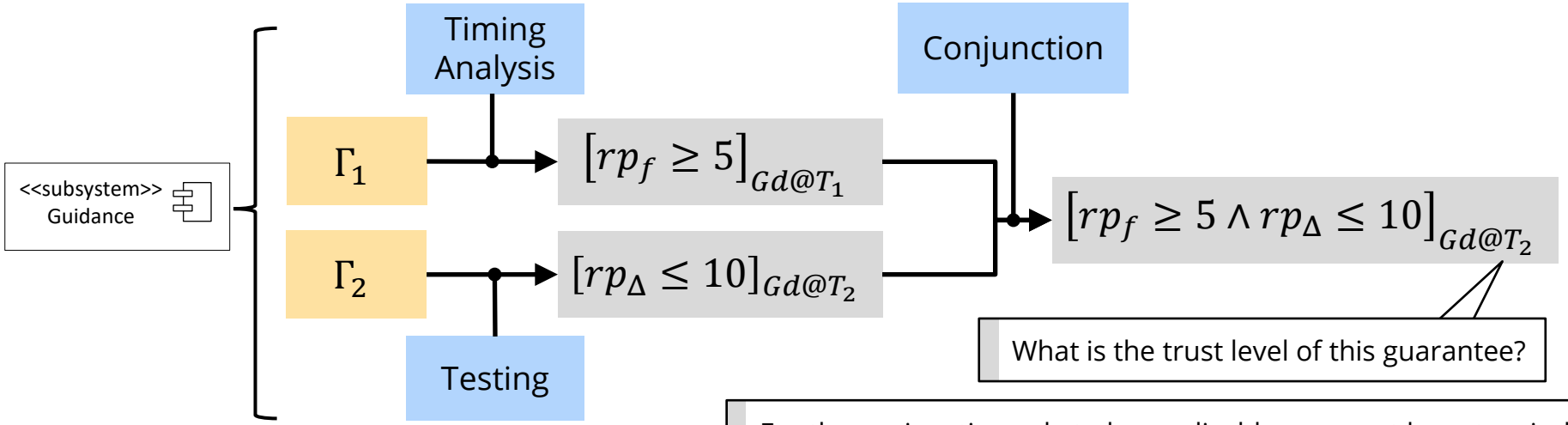
Judgment Example –2



Composing Judgments –1



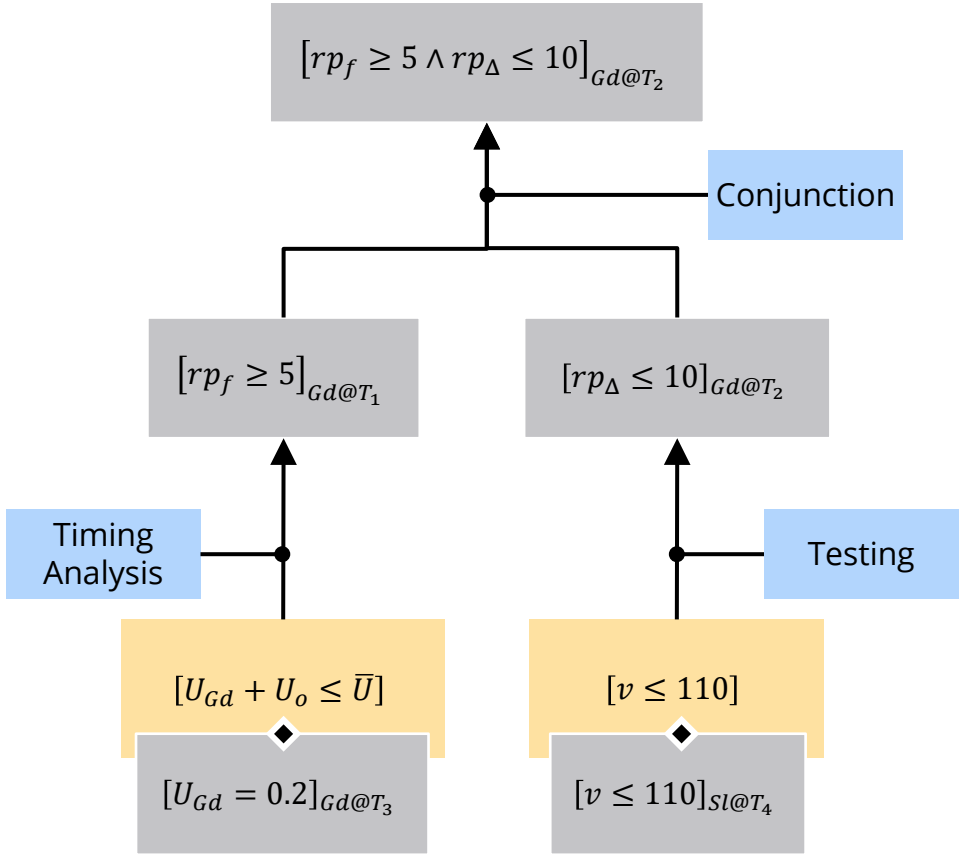
Composing Judgments –2



For the conjunction rule to be applicable, we must have two judgments with the same trust level. Here we downgrade to the lowest of the two.

$$\dfrac{\dfrac{[\Gamma_1] \vdash [rp_f \geq 5]_{Gd@T_1} \quad T_2 \sqsubseteq T_1}{[\Gamma_1] \vdash [rp_f \geq 5]_{Gd@T_2}}\,\text{DOWN}}{[\Gamma_1 \wedge \Gamma_2] \vdash [rp_f \geq 5]_{Gd@T_2}}\,\text{WEAKEN} \qquad \dfrac{[\Gamma_2] \vdash [rp_\Delta \leq 10]_{Gd@T_2}}{[\Gamma_1 \wedge \Gamma_2] \vdash [rp_\Delta \leq 10]_{Gd@T_2}}\,\text{WEAKEN}}{[\Gamma_1 \wedge \Gamma_2] \vdash [rp_f \geq 5 \wedge rp_\Delta \leq 10]_{Gd@T_2}}\,\wedge \text{R}$$

Automated Analysis of Compositions



Argument architectures for multiple subsystems can be composed and checked to detect the following:

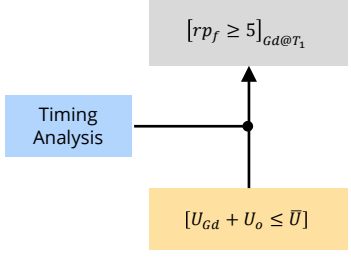
- unsatisfied or mismatched assumptions
- logical inconsistencies

The trust level of different guarantees is computed using the flows of trust throughout the architecture. This information allows focusing assurance efforts on where it matters.

Addressing Assurance Speed and Confidence

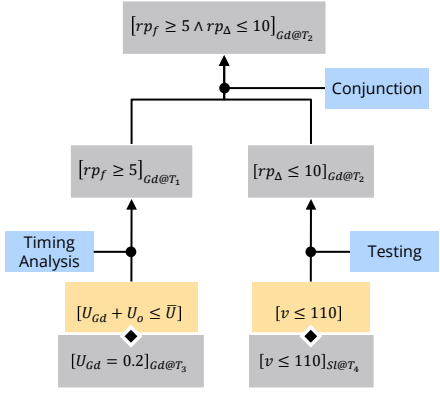
Lack of Effective Reuse of Assurance Results

- Analysis results are captured in judgments with a composition interface that allows them to be reused.



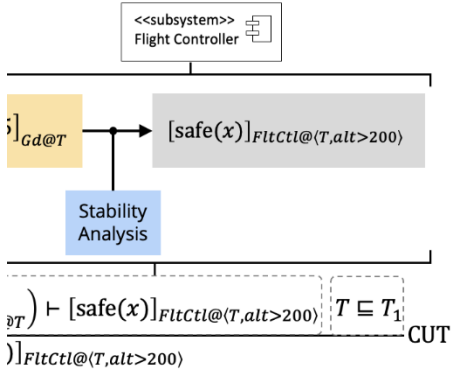
Inability to Integrate Multiple Types of Assurance Analyses

- Judgments hide the details of the analyses.
- The argument architecture captures how different analyses are composed to assure a system.



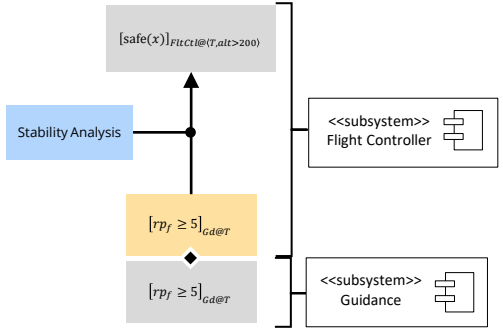
No Notion of Different Levels of Trust

- Guarantees are qualified with a trust level.
- The logic system supports rely/guarantee reasoning with trust levels.



Interdependence Between Subsystems

- Analysis assumptions are clearly stated.
- The satisfaction of assumptions and logical consistency of integrated assurance results is checked.



Summary and Next Steps

We presented an assurance approach that provides the foundations for the following:

- incremental and compositional assurance with reuse of analysis results
- explicit consideration of trust so that assurance effort is commensurate with its importance to the mission

In the next year, we will make the approach practical through tooling that analysts can use.

We believe that co-development of a system's architecture and its argument architecture holds promise for how complex evolving systems can be developed.

We're interested in working with you on systems that might benefit from this approach, including those that have already been assured and are evolving.

Team

Dr. Gabriel Moreno
Principal Researcher

Mark Klein
Principal Technical Advisor

Dr. Shambwaditya Saha
Assurance Researcher

Dr. Farzaneh Derakhshan
Assistant Professor
Illinois Institute of Technology

Dr. Limin Jia
Electrical and Computer Engineering
Carnegie Mellon University

Dr. Dionisio de Niz
Technical Director

Dr. Anton Hristozov
Senior Engineer

John Robert
Division Deputy Director

Dr. Ruben Martins
Assistant Research Professor
School of Computer Science
Carnegie Mellon University