CS 6501 Machine Learning for Software Reliability (Fall 2024)

Wenxi Wang

University of Virginia

wenxiw@virginia.edu

Logistics

- **Instructor:** Wenxi Wang (wenxiw@virginia.edu)
- **TA:** Mingtian Tan (wtd3gz@virginia.edu)
- **Time:** Mondays & Wednesdays 11:00am - 12:15pm
- **Location:** Rice Hall 011
- **Office Hour:** appointment on demand

Course Objective

Objective: for you to gain an understanding of how research is conducted in the field of machine learning for software engineering

General steps of doing research in CS:

Step 1: learn the fundamental knowledge and classic techniques in the field;

Step 2: find a specific research topic and a specific research problem

Step 3: do a thorough literature review, learn the state-of-the-art techniques in the field

Step 4: find out what needs to be improved, propose new approach;

Step 5: design the algorithms, do the implementation;

Step 6: do the experimental evaluation: design your experimental setup, take the state-of-the-art techniques as the baselines, evaluate your technique with the baselines;

Step 7: Write up the paper

Step 8: Present the paper!

Course Structure

General steps of doing research in CS:

Step 1: learn the fundamental knowledge and classic techniques in the field;
Step 2: find a specific research topic and a specific research problem
Step 3: do a thorough literature review, learn the state-of-the-art techniques in the field
Step 4: find out what needs to be improved, propose new approach;
Step 5: design the algorithms, do the implementation;
Step 6: do the experimental evaluation
Step 7: Write up the paper
Step 8: Present the paper!

In-class timeline:

Part 1: introducing basic concepts, fundamental knowledge, classic techniques in FM, SE, and ML **(Step 1)**

Part 2: presenting papers in interdisciplinary research in various topics, introducing the state-of-the-art techniques **(Step 3, 8)**
Also, Learn the SOTA techniques from others' presentation by just attending the class!

Project Presentation **(Step 8)**

Off-class timeline:

Part 1: Read the provided materials, talk to me, and find a specific research topic **(Step 1 and 2)**

Part 2: Talk to me, do literature review Write the proposal **(Step 3)**

Part 3: More literature review, talk to me, propose new approach, do the initial implementation **(Step 3, 4, and 5)**

Part 4: Initial evaluation results and write the report **(Step 6 and 7)**

Course Structure

General steps of doing research in CS:

Step 1: learn the fundamental knowledge and classic techniques in the field;
Step 2: find a specific research topic and a specific research problem
Step 3: do a thorough literature review, learn the state-of-the-art techniques in the field
Step 4: find out what needs to be improved, propose new approach;
Step 5: design the algorithms, do the implementation;
Step 6: do the experimental evaluation
Step 7: Write up the paper
Step 8: Present the paper!

In-class timeline:

Part 1: introducing basic concepts, fundamental knowledge, classic techniques in FM, SE, and ML **(Step 1)**

Part 2: presenting papers in interdisciplinary research in various topics, introducing the state-of-the-art techniques **(Step 3, 8)**
Also, Learn the SOTA techniques from others' presentation by just attending the class!

Project Presentation **(Step 8)**

Off-class timeline:

Part 1: Read the provided materials, talk to me, and find a specific research topic **(Step 1 and 2)**

Part 2: Talk to me, do literature review Write the proposal **(Step 3)**

Part 3: More literature review, talk to me, propose new approach, do the initial implementation **(Step 3, 4, and 5)**

Part 4: Initial evaluation results and write the report **(Step 6 and 7)**

Course Structure

General steps of doing research in CS:

Step 1: learn the fundamental knowledge and classic techniques in the field;
Step 2: find a specific research topic and a specific research problem
Step 3: do a thorough literature review, learn the state-of-the-art techniques in the field
Step 4: find out what needs to be improved, propose new approach;
Step 5: design the algorithms, do the implementation;
Step 6: do the experimental evaluation
Step 7: Write up the paper
Step 8: Present the paper!

In-class timeline:

Part 1: introducing basic concepts, fundamental knowledge, classic techniques in FM, SE, and ML **(Step 1)**

Part 2: presenting papers in interdisciplinary research in various topics, introducing the state-of-the-art techniques **(Step 3, 8)**
Also, Learn the SOTA techniques from others' presentation by just attending the class!

Project Presentation **(Step 8)**

Off-class timeline:

Part 1: Read the provided materials, talk to me, and find a specific research topic **(Step 1 and 2)**

Part 2: Talk to me, do literature review Write the proposal **(Step 3)**

Part 3: More literature review, talk to me, propose new approach, do the initial implementation **(Step 3, 4, and 5)**

Part 4: Initial evaluation results and write the report **(Step 6 and 7)**

Course Structure

General steps of doing research in CS:

Step 1: learn the fundamental knowledge and classic techniques in the field;
Step 2: find a specific research topic and a specific research problem
Step 3: do a thorough literature review, learn the state-of-the-art techniques in the field
Step 4: find out what needs to be improved, propose new approach;
Step 5: design the algorithms, do the implementation;
Step 6: do the experimental evaluation
Step 7: Write up the paper
Step 8: Present the paper!

In-class timeline:

Part 1: introducing basic concepts, fundamental knowledge, classic techniques in FM, SE, and ML **(Step 1)**

Part 2: presenting papers in interdisciplinary research in various topics, introducing the state-of-the-art techniques **(Step 3, 8)**
Also, Learn the SOTA techniques from others' presentation by just attending the class!

Project Presentation **(Step 8)**

Off-class timeline:

Part 1: Read the provided materials, talk to me, and find a specific research topic **(Step 1 and 2)**

Part 2: Talk to me, do literature review Write the proposal **(Step 3)**

Part 3: More literature review, talk to me, propose new approach, do the initial implementation **(Step 3, 4, and 5)**

Part 4: Initial evaluation results and write the report **(Step 6 and 7)**

Course Evaluation

In-class timeline:

Part 1: introducing basic concepts, fundamental knowledge, classic techniques in FM, SE, and ML

Part 2: presenting papers in interdisciplinary research in the field, introducing the state-of-the-art techniques
Also, Learn the SOTA techniques from others' presentation by just attending the class!

Don't stress! You can share your thoughts, ideas, after the reading!

Quiz (5%): to see if you've done the reading

Presentation (25%): teach others the SOTA techniques in the topic you are interested in

Participation (20%): Learn the SOTA techniques from other topics by actively attending the class!

Off-class timeline:

Part 1: Read the provided materials, talk to me, and find a specific research topic **(step 1 and 2)**

Part 2: Do literature review. Write the proposal **(Step 3)**

Project proposal and presentation (15%)

Part 3: More literature review, propose new approach, do the initial implementation **(Step 3, 4, and 5)**

Part 4: Initial evaluation results and write the report **(Step 6 and 7)**

Final Project report and presentation (35%)

Overview of Course Content

Software is everywhere

Bugs can cause horrible consequences…

CNN BUSINESS Markets Tech Media Calculators Videos

A hacker gained access to 100 million Capit
One credit card applications and accounts

yahoo!news Search the web

Yet another FDA Class 1 recall for Minnesota-made infusi
syringe pumps

NBC NEWS United Airlines issued nationwide ground stop due to 'systemwide technology issue'

United Airlines issued nationwide ground stop due to 'systemwide technology issue'

The airline held all its aircraft in the U.S. and Canada at their departure gates Tuesday but lifted th

FOX BUSINESS Personal Finance Economy Markets Watchlist Lifestyle Real Estate Tech Video

Cruise robotaxi crashes into firetruck in San Francisco

The Cruise driverless vehicle was transporting one person at the time of the crash

Software is everywhere

Bugs can cause horrible consequences…

What can we do to help?

For software reliability, What can we do to help?

Part 1

Direction 1: Software Verification

Direction 2: Software Testing

For software reliability, What can we do to help?

Part 1

Direction 1: Software Verification

Direction 2: Software Testing

Direction 1: Software Verification

Make software reliable

using formal reasoning

formal reasoning

Formal Reasoning

Involves various research domains

- Model Checking
- Automated Reasoning
- Theorem Proving
- Verification
- Static Analysis

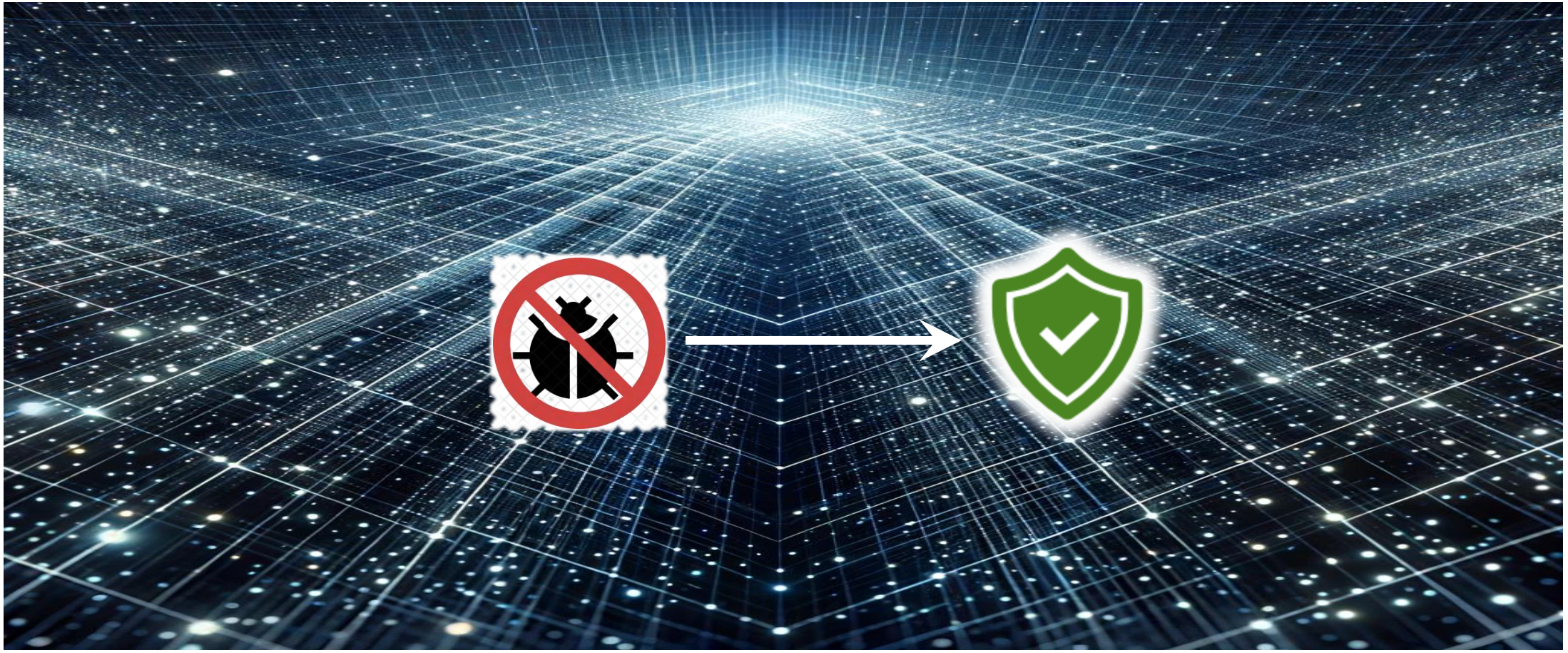
And more ……

Direction 1: Software Verification

Searches the ***entire*** state space for bugs

Direction 1: Software Verification

If no bug is found, the system is safe!

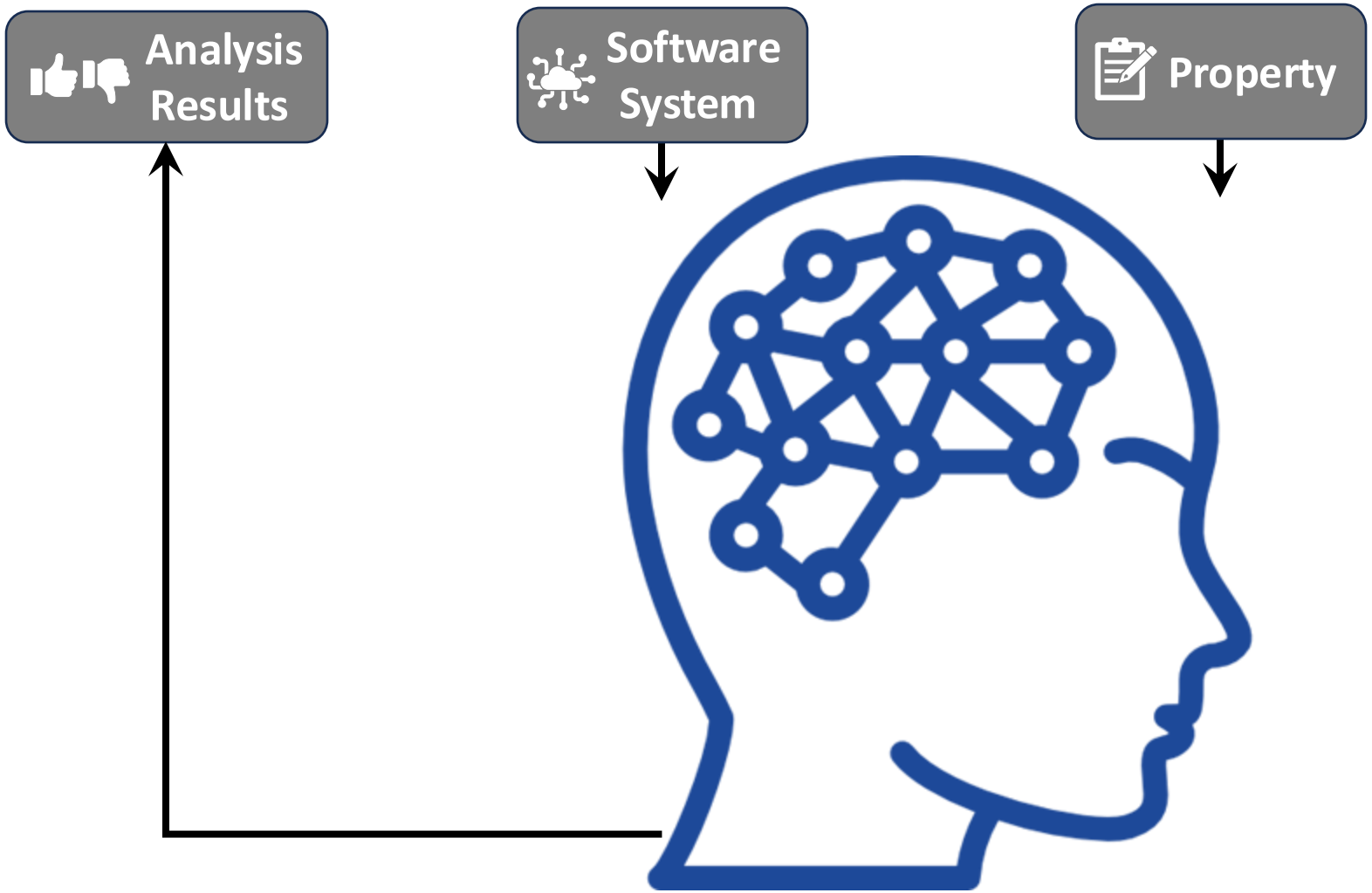


Direction 1: Software Verification

Provides correctness ***guarantees***!

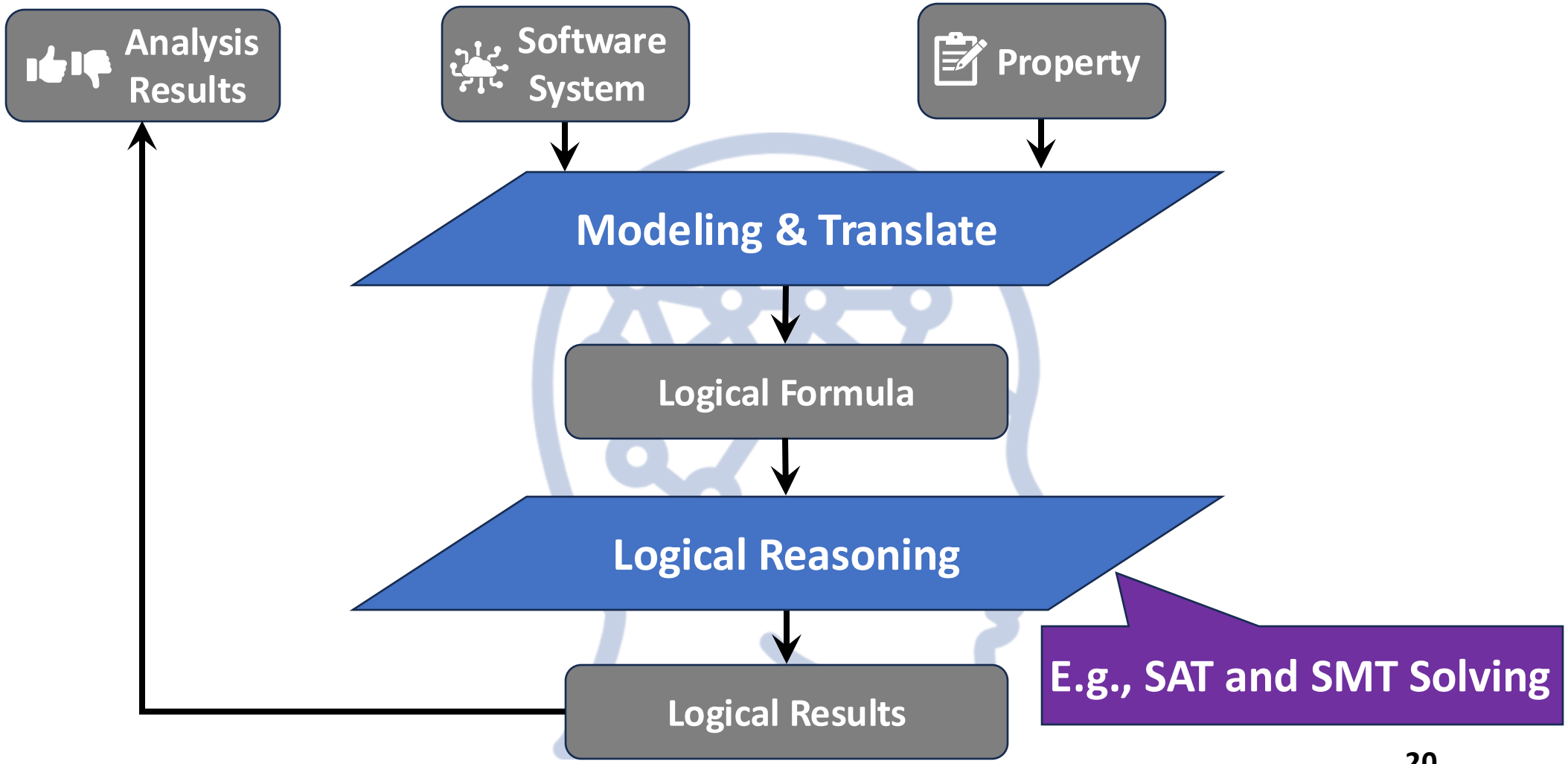
Direction 1: Software Verification

Systematically and logically analyze software systems with **properties**



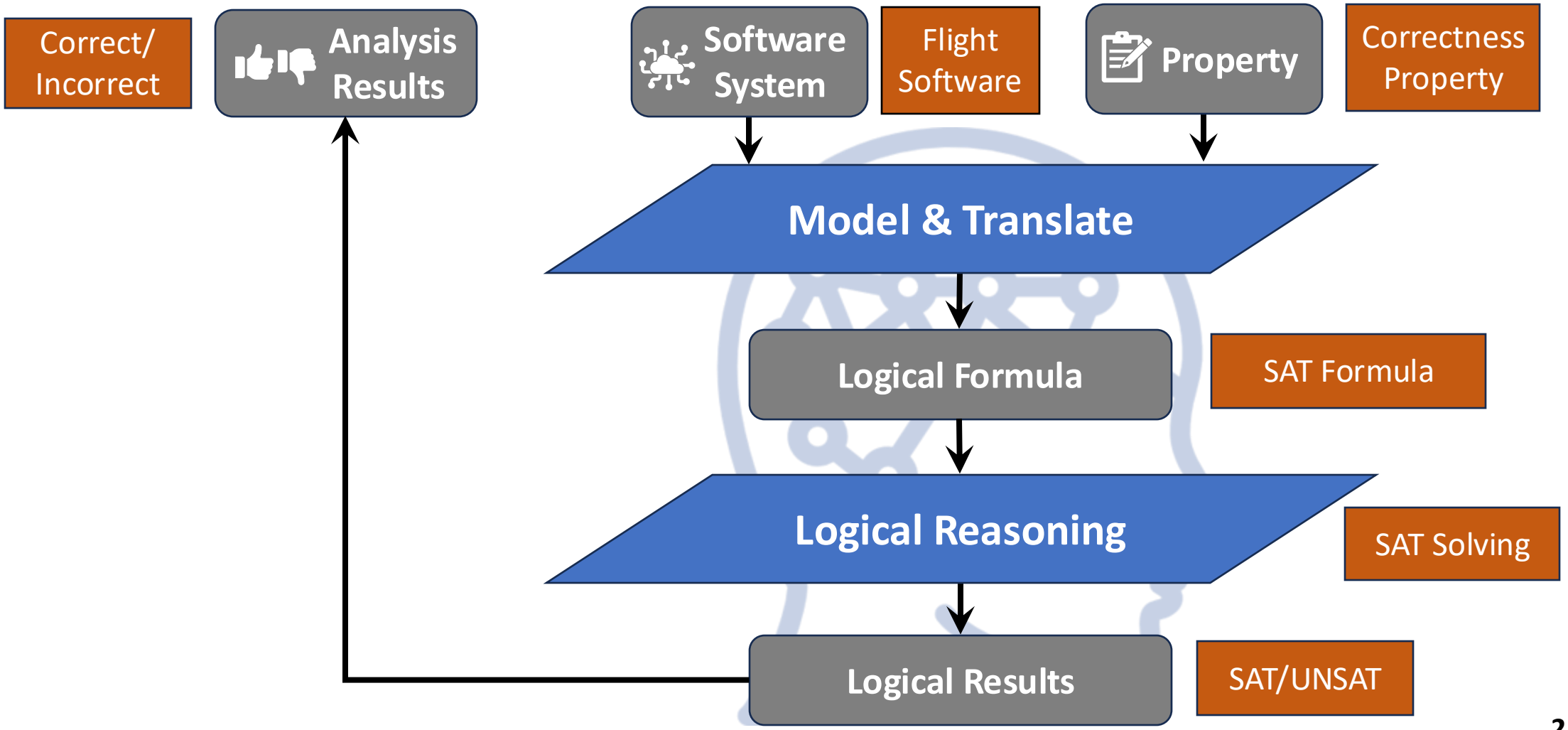
Direction 1: Software Verification

Typically models software problems into logical formulas



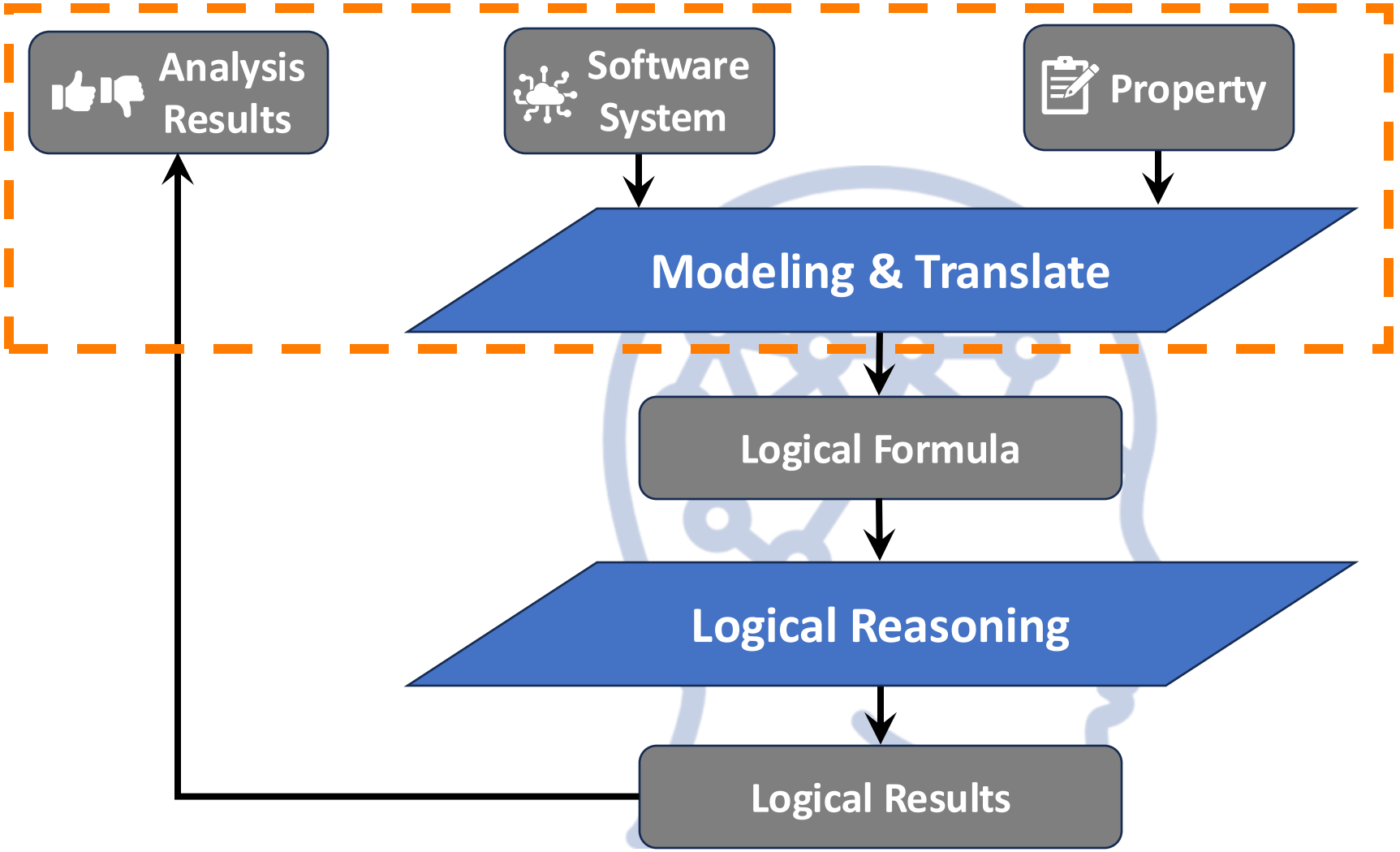
Formal Reasoning for Software Systems

For example: Flight software verification in NASA



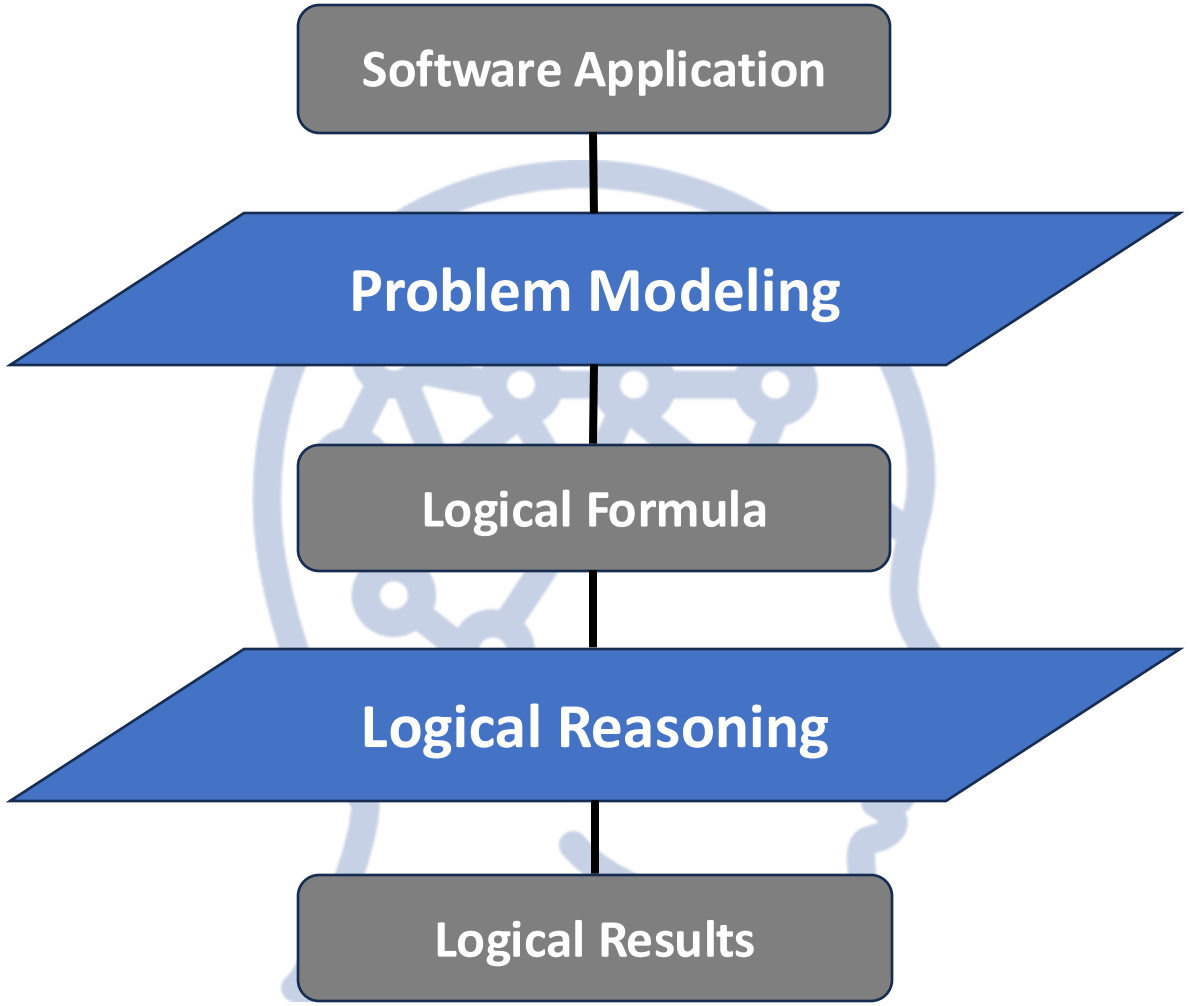
Direction 1: Software Verification

Typically models software problems into logical formulas



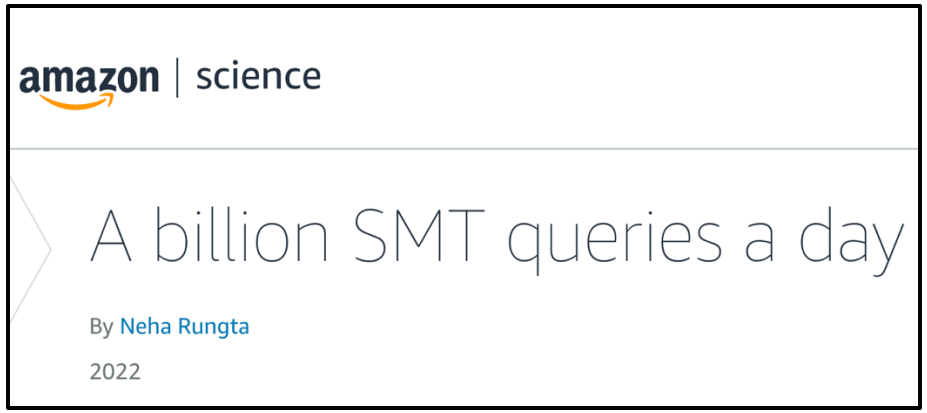
Direction 1: Software Verification

Simplified view: we focus on both analysis layers



Software Verification is Applied in Industry

Amazon Web Services makes a **billion** SMT queries daily to ensure its cloud service security



Software Verification has many applications

But software verification is generally hard

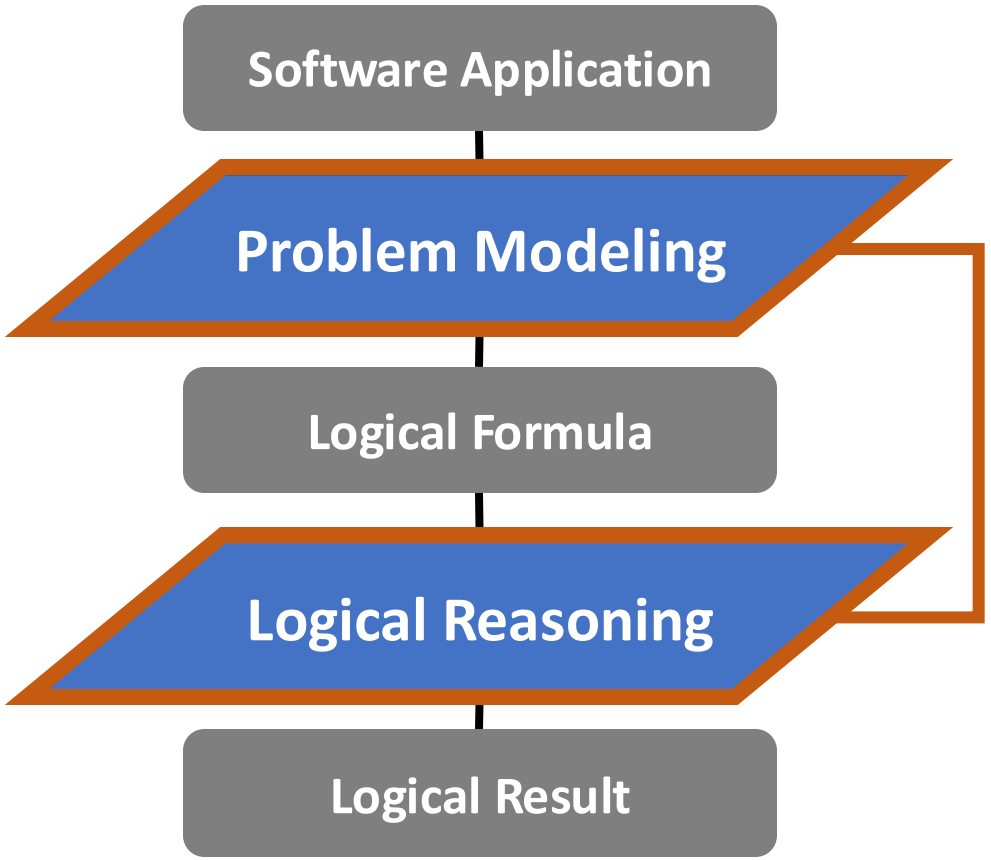
High Computational Complexity
(NP-Complete or Worse)

Direction 1: Software Verification

Improve the scalability and applicability of software verification

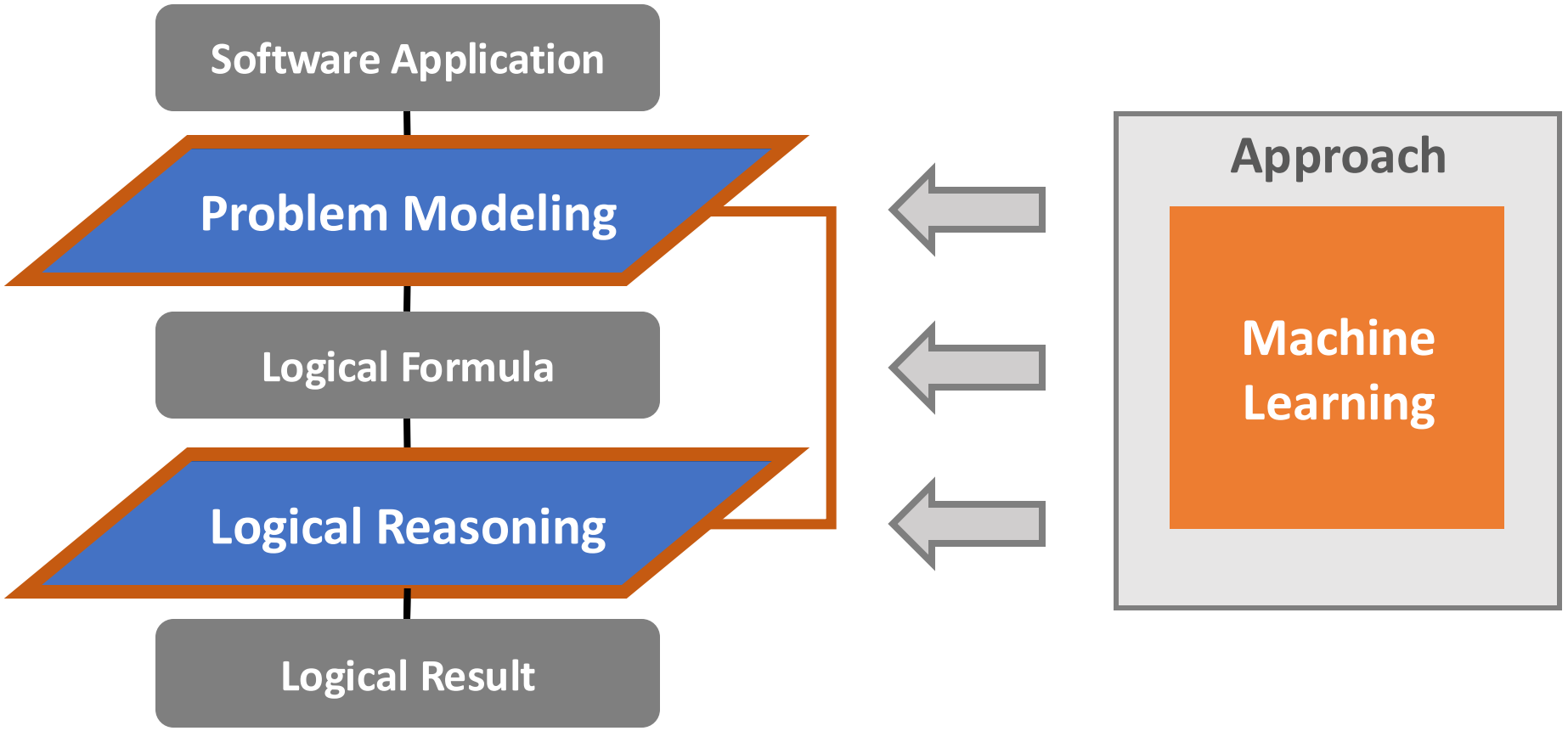
Direction 1: Software Verification

Improve the scalability of software verification
by **enhancing and bridging both analysis layers**

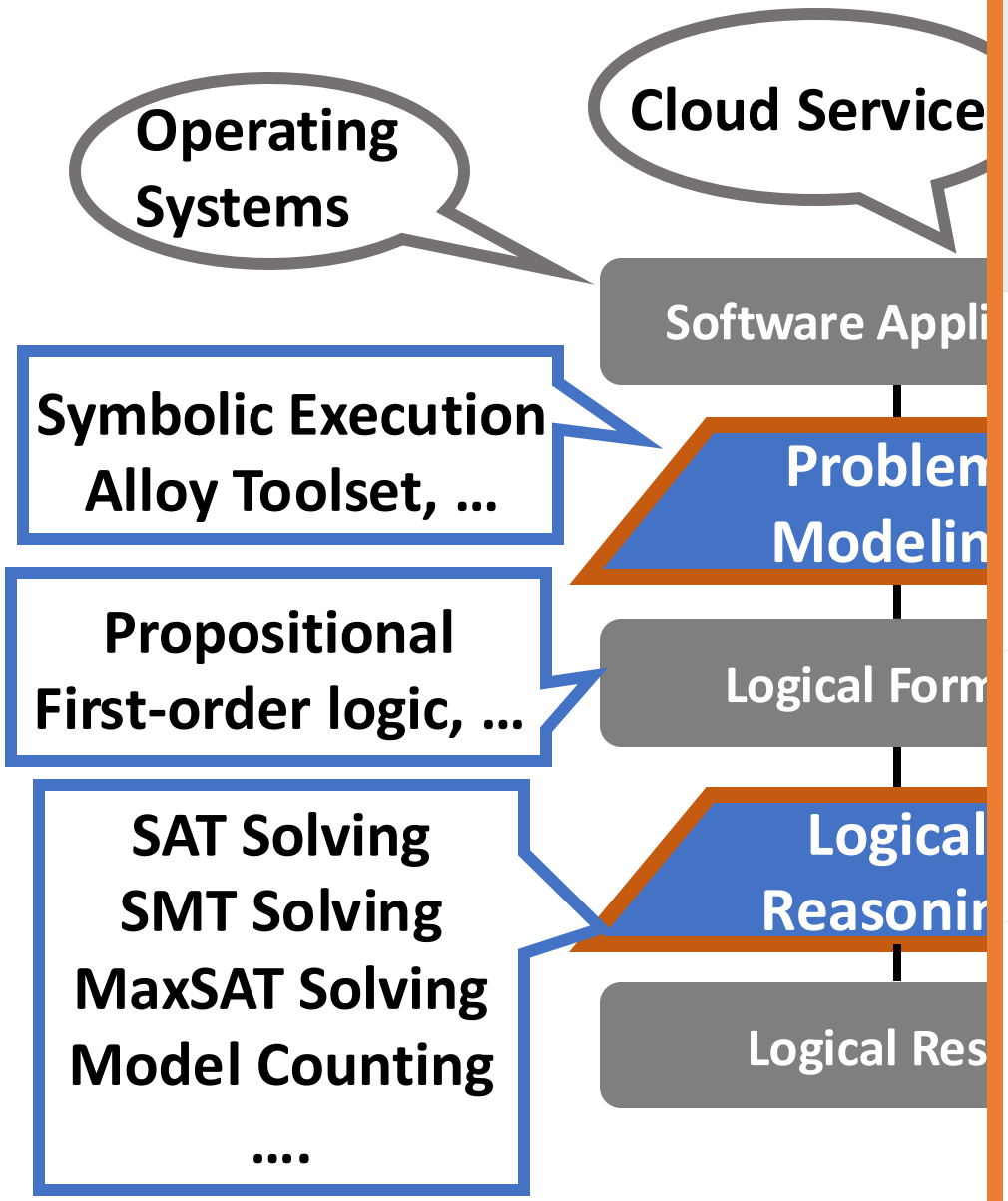


Direction 1: Software Verification

Improve the scalability of software verification by enhancing and bridging both analysis layers using **machine learning approaches**

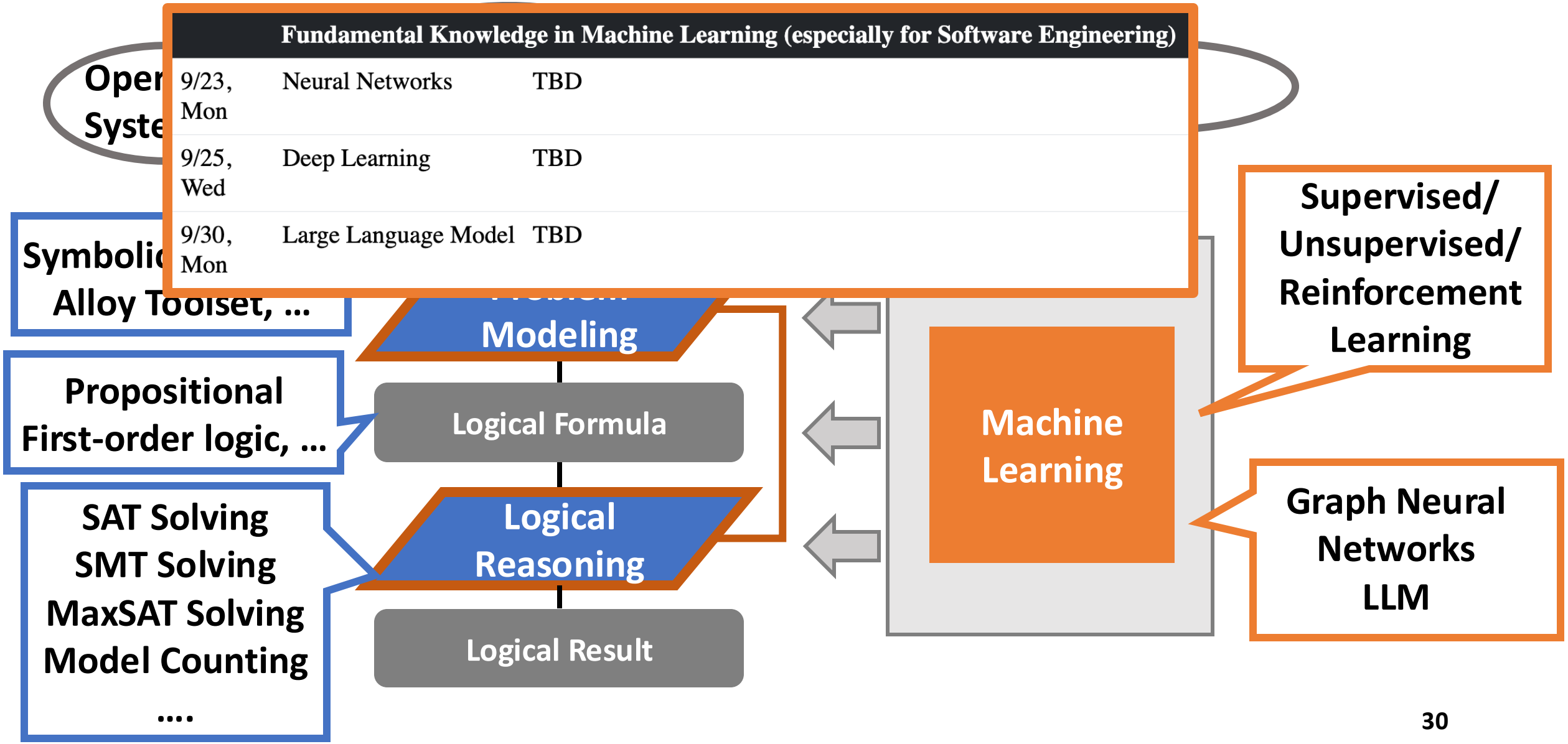


Direction 1

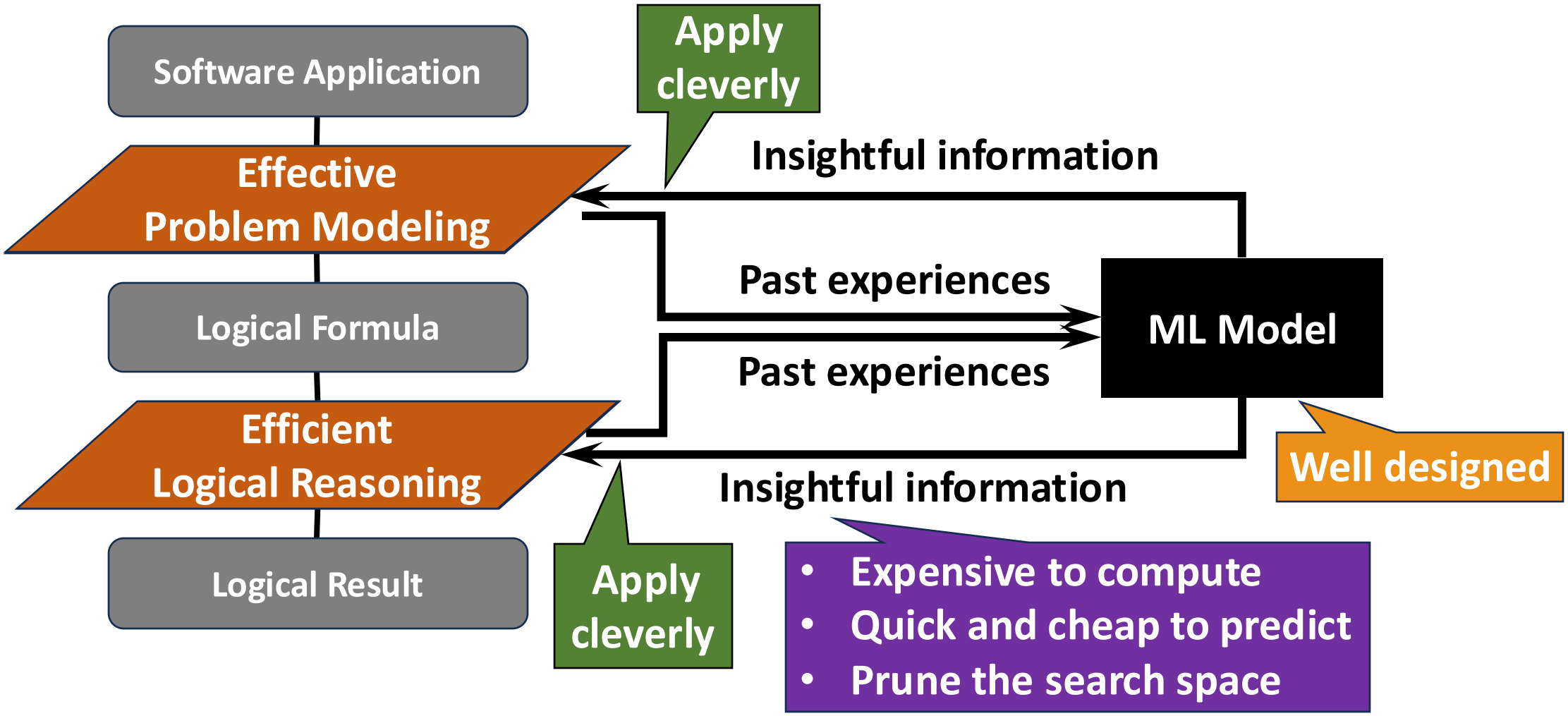


Fundamental Knowledge in Formal Methods and Software Engineering		
9/4, Wed	Symbolic Execution	Guest Lecture: Presenter: Yang Hu (Applied Scientist in AWS, PhD in UT Austin) Readings: 1. Symbolic execution slides by Michael Hicks 2. KLEE, a famous symbolic execution tool 3. KLEE webpage (optional) Quiz before the lecture
9/9, Mon	SAT Solving I	Readings: 1. Classic Book: Decision Procedure by Daniel Kroening and Ofer Strichman (Read Chapter 1 and Chapter 2) 2. Classic SAT solver MiniSat 3. MiniSat page (optional) Quiz before the lecture
9/11, Wed	SAT Solving II and SMT Solving	Readings: 1. State-of-the-art SAT solver Kissat 2. For SMT Solving, read Decision Procedure, Chapter 3 Quiz before the lecture
9/16, Mon	Software Modeling and Verification	Readings: 1. A popular software modeling tool, Alloy 2. Classic Book Reading: Software Abstractions by Daniel Jackson (optional) Quiz before the lecture

Direction 1: Software Verification



Improve formal reasoning for software systems using machine learning



For software reliability, What can we do to help?

Direction 1: Software Verification

Direction 2: Software Testing

Direction 2: Software Testing

Make software reliable

using testing

testing

Direction 2: Software Testing

Searches the ***partial*** state space for bugs

Direction 1: Software Verification

If a bug is found, the system is unsafe!

Direction 2: Software Testing

Cannot Provides ***guarantees***!

Direction 2: Software Testing

9/18, Wed	Software Testing	Guest Lecture: Presenter: Pengyu Nie (Assistant Professor at University of Waterloo) Readings: 1. Differential Testing 2. Metamorphic Testing 3. Regression Testing Quiz before the lecture

Part 2: Research Topics in the intersection

1. ML for SAT Solving
2. ML for SMT Solving
3. ML for Software Testing
4. LLM for Software Testing
5. LLM for Fuzzing
6. ML for Software Verification
7. LLM for Software Verification
8. Software Verification for ML models
9. Software Testing for ML models
10. LLM for Code Generation
11. ML for Program Repair
12. Combining two AI systems:
    ML and Formal Reasoning

Grading Details

https://wenxiwang.github.io/CS6501-016.html