L5: Undecidability, Precision, Recall and F-Measure

Software Analysis

IIT Guwahati · August 6, 2026

Undecidability of Program Properties

- Can a program analysis be **sound** *and* **complete**?

  Not if it is also required to **terminate** on every input program.

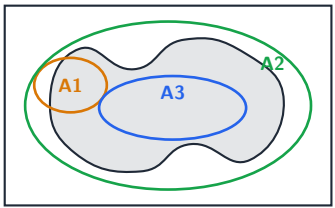
- Questions such as *"is this program point reachable on some input?"* are **undecidable**.

- Designing a program analysis therefore requires a **trade-off** between **termination**, **soundness** and **completeness**.

- Which of the three is relaxed is determined by the **consumer** of the analysis

[1] Rice, *Trans. AMS* 74(2):358–366, 1953 · Naik, CIS 5470, Penn

Quiz (1/3): Classifying Three Analyses



rectangle = all programs
shaded region = programs satisfying φ

For each analysis, state whether it is **sound**, **complete**, both, or neither.

	Sound?	Complete?
A1	No	No
A2	No	Yes
A3	Yes	No

An oval is the set of programs the analysis *accepts*. **Sound**: the oval lies inside the shaded region. **Complete**: the oval contains it.

After Naik, CIS 5470, Penn

Quiz (2/3): Certification of a Medical Device

A certification engineer must establish that the firmware of an **implantable cardiac device** contains **no divide-by-zero error** before clinical approval.

	Sound?	Complete?
A1	No	No
A2	No	Yes
A3	Yes	No

Which analysis should the engineer use?

A3

· The claim is the *absence* of the error. Only a **sound** analysis supports it: if A3 accepts the firmware, no divide-by-zero occurs on any run.

· A3 is incomplete, so some reports are spurious. The cost is **engineering time** — acceptable under a certification regime.

· A2 accepting the firmware would mean nothing: it also accepts programs that do divide by zero.

After Naik, CIS 5470, Penn

Quiz (3/3): Triage in a Large Codebase

A team maintaining a **large, actively developed codebase** will act on divide-by-zero reports only if the tool produces **no false alarms**; they will not staff a triage process.

	Sound?	Complete?
A1	No	No
A2	No	Yes
A3	Yes	No

Which analysis should the team use?

A2

- "No false alarms" is the definition of **completeness**: every program A2 rejects does contain the error.
- A2 is unsound, so some defects are missed. The team accepted that: their requirement was *report quality*, not coverage.
- The two scenarios differ only in the consumer. The property, the programs and the theory are identical; the **requirement** is reversed.

After Naik, CIS 5470, Penn, · Bessey et al., CACM 53(2), 2010

Choosing an Analysis

The requirement determines which guarantee is needed; the guarantee determines the residual cost.

Requirement	Guarantee needed	Cost accepted
Establish absence of an error	**Soundness**	spurious reports
Act on every report without triage	**Completeness**	missed errors
Scan a large codebase cheaply	*neither*	both kinds of error

· The third row is a legitimate engineering position, not a failure. Giving up both guarantees buys **speed** and **scale**: rules that run on every commit, over any language, with no whole-program analysis.

· What is **not** legitimate is leaving the position unstated. An analysis whose errors have no known direction supports no conclusion at all.

cf. L2, *Conservative Approximation* and *Soundness Depends on Your Goal*

Precision and Recall

		Analysis Outcome	
		Accept	**Reject**
Ground Truth	**Good**	True Negative	False Positive
	Bad	False Negative	True Positive

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F-Measure

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

- The F-measure is the **harmonic mean** of precision and recall.
- A low precision or recall greatly reduces F_1.
- Standard F_1 assumes precision and recall are **equally important**.