ИСП РАН

Гибридный подход к анализу ПО: от поиска дефектов к пониманию поведения программ с помощью LLM

Курмангалеев Ш.Ф.

ИСП РАН, Москва 2026

ИСП РАН

Теорема Райса

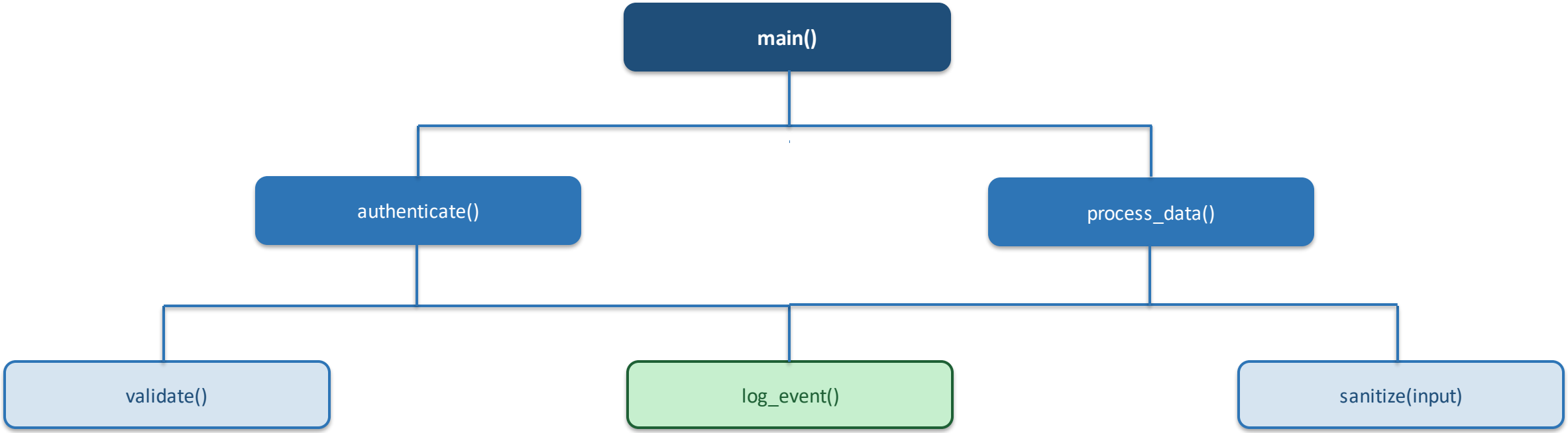
- Теорема Райса (1953): любое нетривиальное семантическое свойство частично вычислимых функций является неразрешимым.
- Доказательство: редукция к проблеме остановки Тьюринга.
- Семантическое свойство — вопрос о поведении программы:
  - «Достигают ли пользовательские данные внешнего сетевого вызова?»
  - «Корректна ли логика аутентификации при всех входных условиях?»
  - «Утекает ли память при возникновении исключения в обработчике?»
- Следствие: анализ не может быть одновременно корректным, полным и завершающимся для произвольных нетривиальных свойств.
- Любой инструмент выбирает, чем пожертвовать:
  - Корректность (soundness) — все ошибки найдены, есть ложные срабатывания.
  - Полнота (completeness) — нет ложных срабатываний, но есть пропуски.
  - Масштабируемость — ограниченная область или бюджет анализа.

От дефектов к поведению: что мы хотим знать о программе

- Традиционный анализ: поиск заранее известных классов дефектов.
- Понимание программы требует ответов на более широкий круг вопросов.
- Вопросы о безопасности — предустановленные чеккеры:
  - «Возможно ли разыменование нулевого указателя в функции f?»
  - «Достигают ли внешние данные без санирования функции exec?»
- Произвольные вопросы о поведении — требуют понимания логики:
  - «Какие функции определяют функциональное назначение этого модуля?»
  - «Есть ли труднодостижимые участки кода при динамическом тестировании?»
  - «Какие сценарии активируют интересующий фрагмент кода?»
  - «Есть ли в коде недекларированные возможности — скрытая логика?»
- Тезис: чеккеры — частный, заранее формализованный случай запроса.
- LLM поверх классического анализа открывает возможности изучения поведения ПО.

Межпроцедурный анализ: граф вызовов и стратегия обхода

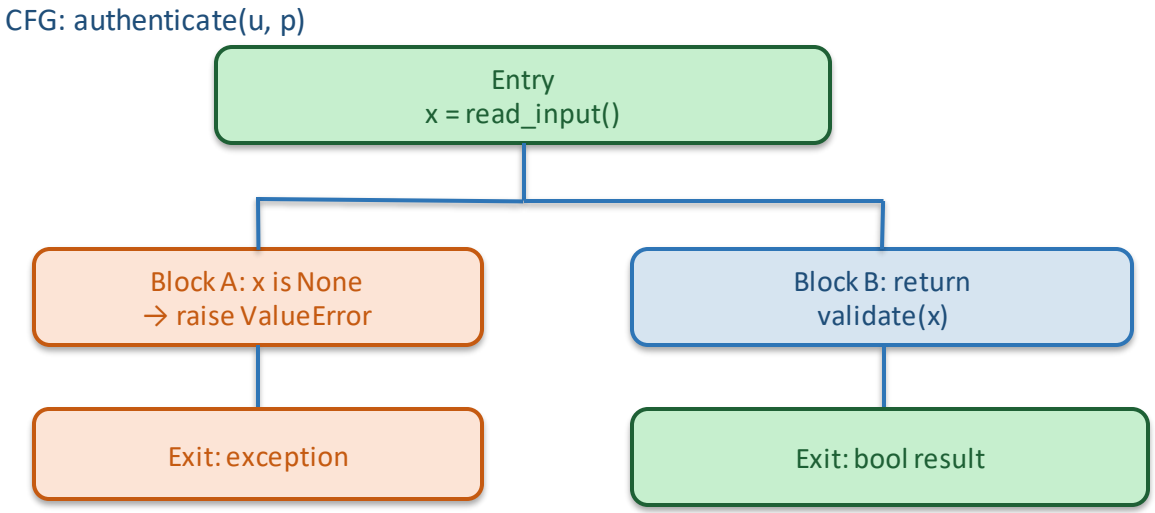
- Граф вызовов, CFG, Dataflow, Points-to — фундамент анализа.
- Без них LLM видит только текст файлов, не отслеживая зависимости через границы модулей и функций.
- Стратегия обхода снизу вверх: вызываемая функция анализируется раньше вызывающей — результаты нижних уровней становятся контекстом для верхних.

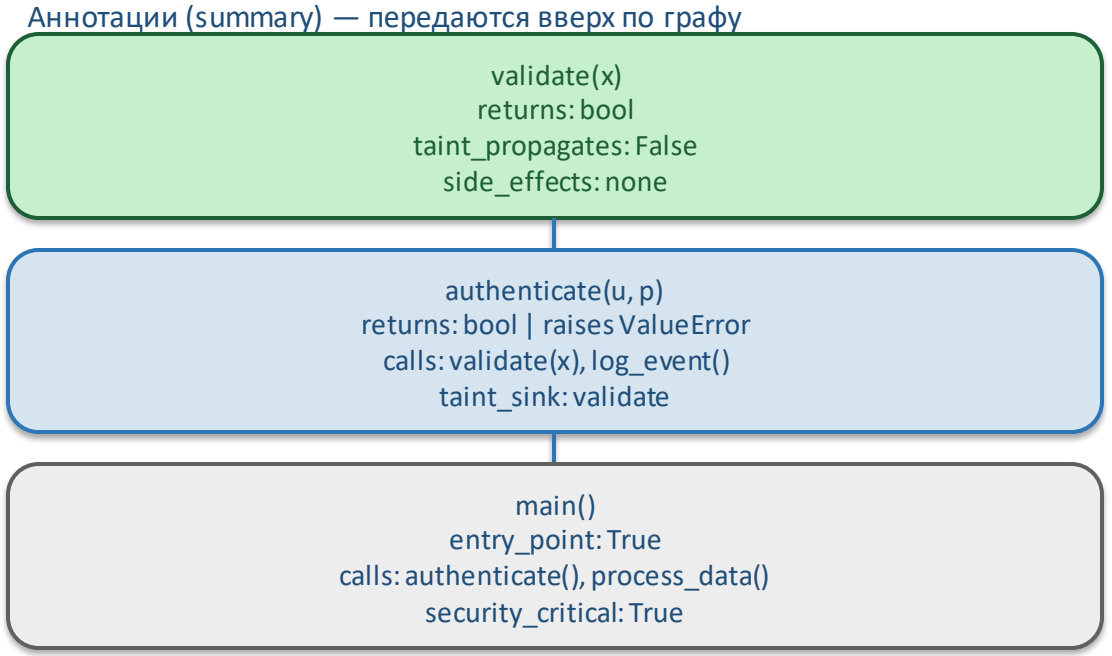


↑ обход снизу вверх: validate / sanitize / log_event → authenticate / process_data → main

Граф потока управления и межпроцедурные аннотации

- CFG (граф потока управления): базовые блоки, рёбра-переходы, Entry/Exit.
- Аннотация (summary) — компактное описание функции для вызывающих: что возвращает, какие аргументы изменяет (side effects), какие данные распространяет наружу (taint). Передаётся вверх по графу вызовов.





Детектор дефектов — заранее сформулированные поведенческие запросы

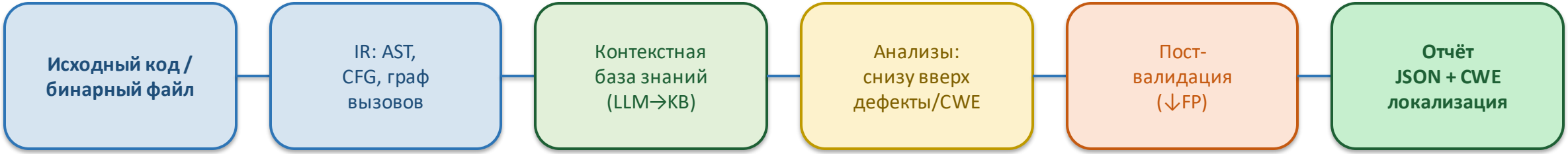
Класс / CWE	Вопрос о поведении	Метод анализа	Ограничение
CWE-476 Null Deref	Может ли указатель быть null в точке разыменования?	Dataflow + CFG; Svace, Coverity	Ложные срабатывания на сложных псевдонимах
CWE-416 Use-After-Free	Доступна ли память после освобождения?	Separation Logic; Infer	Сложные ownership-паттерны пропускаются
CWE-89 SQL Injection	Достигают ли внешние данные SQL-запроса без санирования?	Taint-анализ; Checkmarx	Неполные источники и стоки taint
CWE-190 Int Overflow	Возможно ли переполнение арифметического выражения?	Dataflow; UBSan (динамически)	Без запуска UBSan не все случаи видны
Произвольный запрос	Какие сценарии активируют интересующий участок кода?	Нет готового чеккера; LLM + граф вызовов	Нет формальных гарантий у LLM

LLM: переход от жесткой логики к запросам на естественном языке

- Детектор дефектов требует формализации: нужно знать заранее, что искать, и написать детектор на языке правил или специализированной логике.
- LLM, получая IR и аннотации функций, позволяет задавать вопросы о поведении на естественном языке:
  - «Найди функции, определяющие функциональное назначение этого модуля»
  - «Какие участки кода труднодостижимы при динамическом тестировании?»
  - «Есть ли в этом коде недекларированные возможности — скрытая логика?»
  - «Опиши поведение программы при отрицательном значении параметра»
- Принципиальное условие надёжности: входом LLM должны быть не только исходный текст, но и межпроцедурный граф вызовов, потоки данных, аннотации вызываемых функций.
- Детекторы дефектов по-прежнему нужны: для них сохраняются формальные гарантии.

ISP Aurora: архитектура гибридного анализатора

- ISP Aurora: межпроцедурный анализ + LLM — глубина экспертной проверки.
- Анализ обходит граф вызовов снизу вверх; каждый запрос-детектор может использовать результаты предыдущих через общее контекстное хранилище.
- Поддержка локальных LLM, параллельных моделей, MCP-сервера, векторных БД, пользовательских анализов. Языки: C/C++, Java, Kotlin, C#, Python, JS.



← статический анализ без запуска

LLM-анализ с контекстом →

Проект (язык)	LLM анализа	Найдено (фильтр)	Подтверждено	Доля TP, %
libarchive (C)	Qwen3-Coder	22	20	90,9 %
coreutils (Rust)	deepseek-chat	5	5	100 %
xz (C)	deepseek-chat	2	2	100 %
libfuse (C)	deepseek-chat	10	7	70 %
sudo-rs (Rust)	deepseek-chat	5	4	80 %
rustls (Rust)	deepseek-chat	10	6	60 %
ArduPilot (C++)	—	—	7	—
Среднее TP	—	—	—	**83,5 %**

Синергия: межпроцедурный анализ дает факты — LLM интерпретирует

- Роль межпроцедурного анализа:
  - Предоставляет факты: граф вызовов, потоки данных, псевдонимы.
  - Ограничивает галлюцинации: LLM не может «придумать» вызов, которого нет в графе.
  - Снижает стоимость: предфильтрация уменьшает число запросов к модели.
  - Обеспечивает порядок: обход снизу вверх — контекст всегда готов.
- Роль LLM:
  - Отвечает на произвольные вопросы о поведении, используя IR и аннотации.
  - Обнаруживает логические ошибки и недекларированные возможности без чеккеров.
  - Верифицирует результаты предыдущих анализов (пост-валидация, сокращение FP).
  - Генерирует описание функционального назначения исследуемого ПО.

Ограничения и горизонт: что остаётся открытым

- Ограничения текущего подхода:
  - Отсутствие формальных гарантий у LLM: ответ статистически правдоподобен, а не верифицирован; обязательна кросс-проверка классическим методом.
  - Зависимость от полноты фактов: ошибки в построении графа вызовов непосредственно ограничивают качество ответов LLM.
  - Масштаб: при кодовых базах порядка миллионов строк стратегия построения контекстной базы знаний критична.
  - Воспроизводимость: LLM стохастична — один вопрос может давать разные ответы.
- Открытые исследовательские направления:
  - Верификация гипотез LLM: LLM формулирует инварианты → Frama-C / Z3 проверяет.
  - Замкнутый цикл: фаззер находит контрпример → LLM уточняет запрос о поведении.
  - Пользовательские анализы: API для добавления произвольных запросов к конвейеру.
  - Оценка достоверности ответа LLM без ручной разметки (calibration).