Применение генеративного ИИ в задачах анализа кода

Валерий Игнатьев

ИСП РАН

Доверие: фундаментальная или прикладная проблема?

Методы, основанные на LLM, уже **учитывают возможность галлюцинаций**, поэтому вопрос доверенного ИИ в таких подходах не такой общий, как в случае чатов

- Вопрос доверия не снимается, а смещается и конкретизируется:
  - специализированные атаки
  - отказоустойчивость
  - объяснимость

Как генеративный ИИ меняет анализ кода?

Было: поиск ошибок ограничен правилами и известными шаблонами.

Может ли ИИ что-то улучшить?

Проблемы GAI:

- Ограничения размера контекста
- Галлюцинации, недетерминизм
- Конфиденциальность, закрытый контур
- Много убедительного шума, сложно анализировать

Как генеративный ИИ меняет анализ кода?

Решение

Комплексная система

LLM + «классические методы»

LLM в статическом анализе исходного кода

Проблемы статического анализа

Человеческий фактор
- много предупреждений
- сложность их анализа
- пропущенные ошибки

Фундаментальные
- точность
- полнота
- производительность

Инфраструктурные
- много библиотек
- много языков
- проектно-специфичные ошибки

Применение LLM и ML

Developed **Валидация (разметка) предупреждений**
- с учетом релевантного контекста всего проекта;
- в закрытом контуре на базе открытых моделей;
- **генерация подробных объяснений** для предупреждений с учетом контекста;
- **генерация исправлений;**

Released Быстрая **оценка истинности предупреждений** с помощью ML
- ранжирование

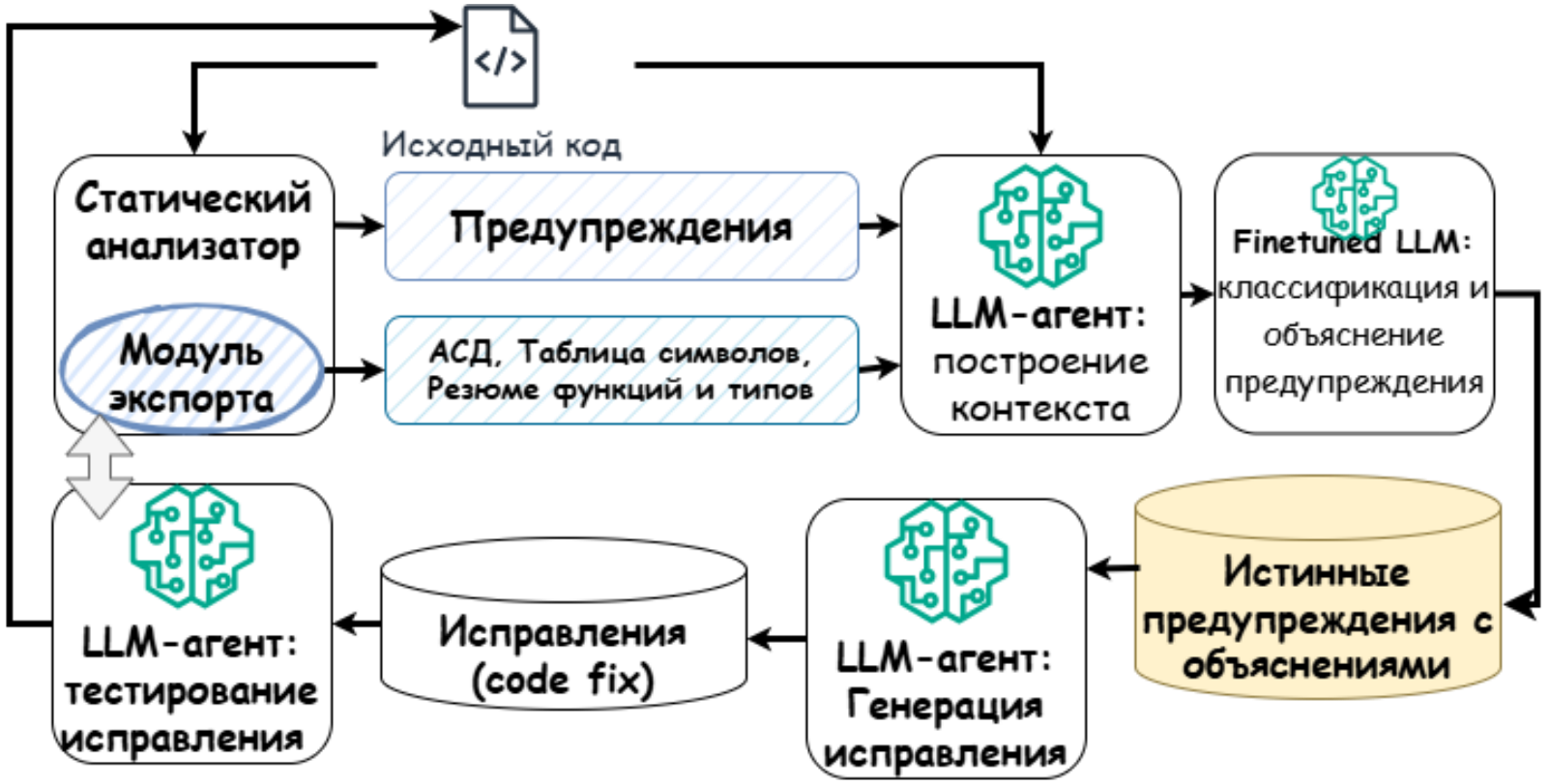
Developed **Спецификации библиотечных функций**
- по коду и документации;
- отчуждаемый инструмент генерации для закрытых библиотек;
- спецификации открытых популярных библиотек в дистрибутиве;

Research **Генерация детекторов ошибок**
- SvaceAPI, CodeQL, semgrep;
- по истории исправлений в репозитории;
- по CVE и текстовому описанию;

Released **Трансляция тестов анализатора** между языками

Работа с предупреждениями анализатора

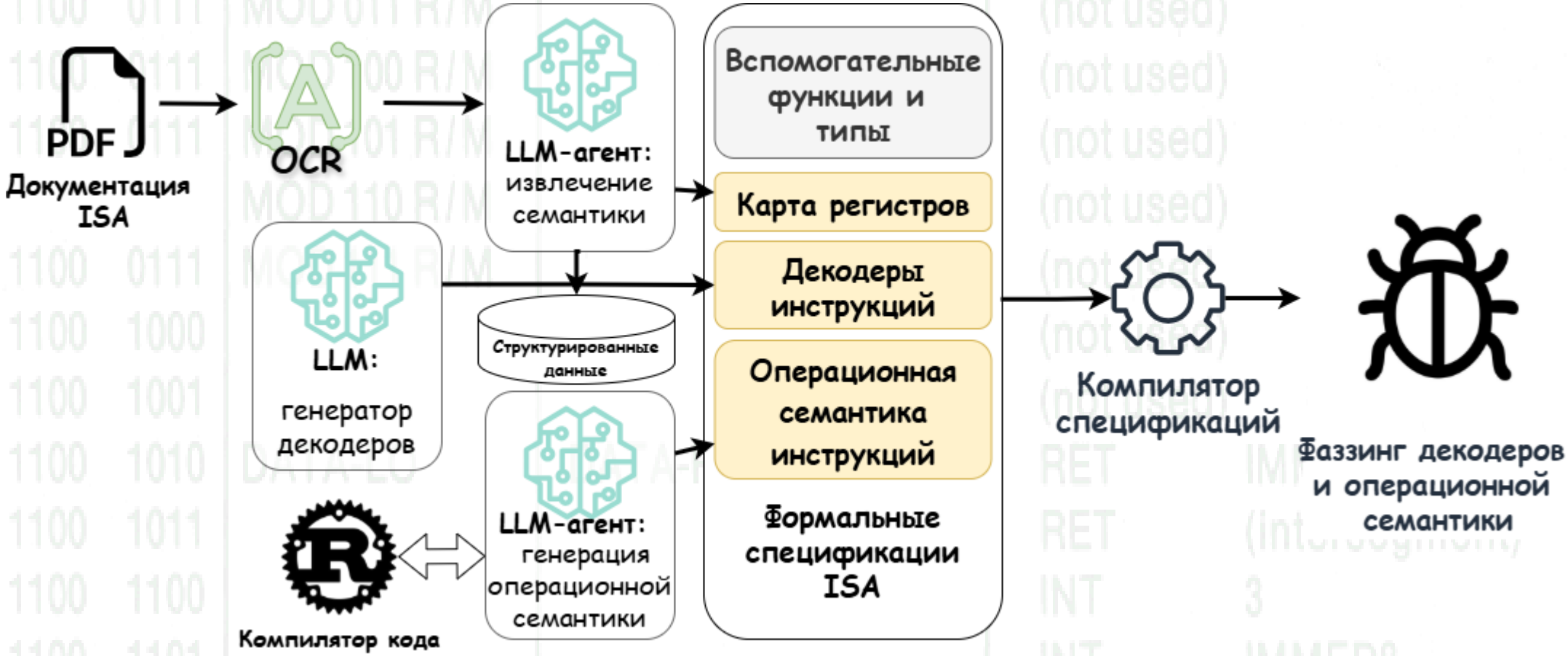
- ✓ Валидация
- ✓ Ранжирование
- ✓ Генерация подробных объяснений
- ✓ Генерация исправлений (code fix)
- ✓ Проверка исправлений



LLM в анализе бинарного кода

Проблема анализа бинарного кода:

- При разработке эмулятора, отладчика, дизассемблера, статического и динамического анализатора требуются **формальные спецификации всех поддерживаемых архитектур (ISA)**



Выводы

Вопрос **доверенности** остается открытым:

- исследования методов противодействия атакам
- оценка влияния эквивалентных преобразований кода на ответы модели
- исследование и внедрение методов внутренней оценки достоверности ответов

Стремительный **рост кодовой базы**, вызванный популярностью и доступностью **вайб-кодинг**а, требует симметричного ответа от статических анализаторов

- масштабируемость
- специфичные ошибки
- большое число предупреждений
- сложность анализа «никому незнакомого» кода