Методы поиска клонов кода на основе семантического анализа программы

[1] Ереванский государственный университет
[2] Институт системного программирования Российской Академии Наук
Саргсян Севак аспирант, sevaksargsyan@ispras.ru
Курмангалеев Шамиль к. ф-м. н., kursh@ispras.ru

Типы клонов кода

Определение (Фрагмент кода) : Фрагмент кода это некая последовательность строк кода.

Тип-1 : Фрагменты кода идентичны за исключением вариаций пробелов и комментариев.

Тип-2: Фрагменты кода идентичны за исключением вариаций идентификаторов, литералов, пробелов и комментарий.

Тип-3 : Скопированный фрагмент кода был изменен путем добавления или удаления инструкции. Так же могут быть отличия литералов, пробелов и комментариев.

Примеры клонов кода

Оригинальный код

```
4: void sumProd(int n) {
5:     float sum = 0.0;
6:     float prod = 1.0;
7:     for (int i = 1; i<=n; i++) {
8:         sum = sum + i;
9:         prod = prod * i;
10:        foo(sum, prod);
11:    }
12: }
```

Тип-1

```
void sumProd(int n) {
  float sum = 0.0; //C1
  float prod = 1.0; // C2
  for (int i = 1; i <= n; i++) {
  ____    sum = sum + i;
  ____    prod = prod * i;
  ____    foo(sum, prod);
  }
}
```

Добавлены пробелы и комментарии

Тип-2

```
void sumProd(int n) {
  int s = 0; //C1
  int p = 1; // C2
  for (int i = 1; i <= n; i++) {
  ____    s = s + i;
  ____    p = p * i;
  ____    foo(s, p);
  }
}
```

Добавлены пробелы и комментарии

Имена и типы переменных изменены

Тип-3

```
void sumProd(int n) {
  int s = 0; //C1
  int p = 1; // C2
  for (int i = 1; i <= n; i++) {
  ____    s = s + i * i;
  ____    foo(s, p);
  }
}
```

Добавлены пробелы и комментарии

Имена и типы переменных изменены

Инструкции изменены и удалены

Применения поиска клонов кода

1. **Поиск функционально похожих частей программы.**
2. **Поиск лицензионных нарушений**
3. **Решении задач автоматического рефакторинга (refactoring)**
4. **Поиска семантических ошибок, возникающих при некорректном копировании участка кода.**

Подходы к поиску клонов кода

Текстовой подход (тип-1)

1. S. Ducasse, M. Rieger, S. Demeyer, A language independent approach for detecting duplicated code, in: Proceedings of the 15th International Conference on Software Maintenance.

Лексический подход (тип-1,2)

1. T.Kamiya, S.Kusumoto, K.Inoue, CCFinder : A multilinguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering.

Подход основанный на ACT (тип-1,2 тип-3 с низкой точностью)

1. I. Baxter, A. Yahin, L. Moura, M. Anna, Clone detection using abstract syntax trees, in: Proceedings of the 14th International Conference on Software.

Метрический подход (тип-1,2,3 с низкой точностью)

1. N. Davey, P. Barson, S. Field, R. Frank, The development of a software clone detector, International Journal of Applied Software Technology.

Подход основанный на графах (семантический) (тип-1,2,3 но имеет большую вычислительную сложность)

1. M. Gabel, L. Jiang, Z. Su, Scalable detection of semantic clones, in: Proceedings of the 30th International Conference on Software Engineering, ICSE 2008

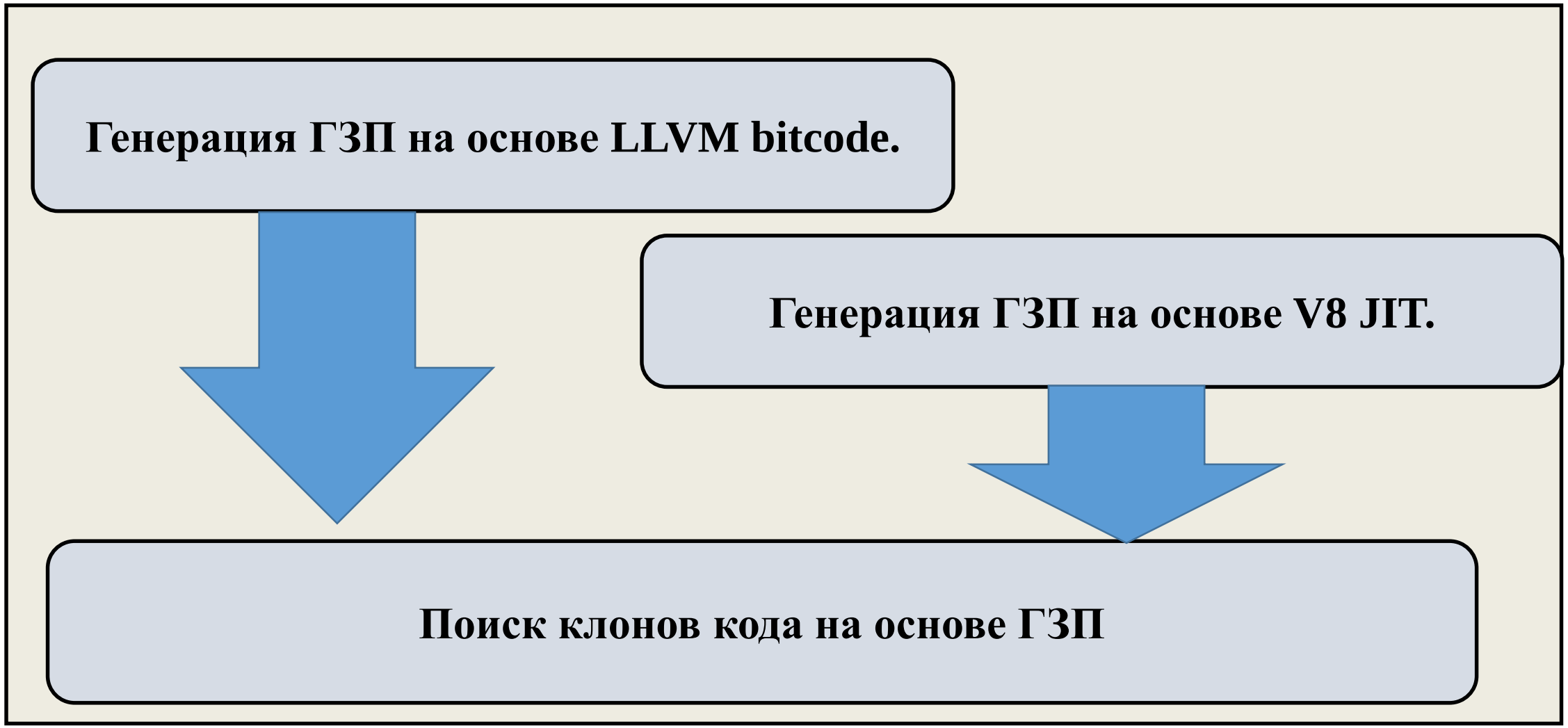
Постановка задачи

Создать инструмент поиска клонов кода.

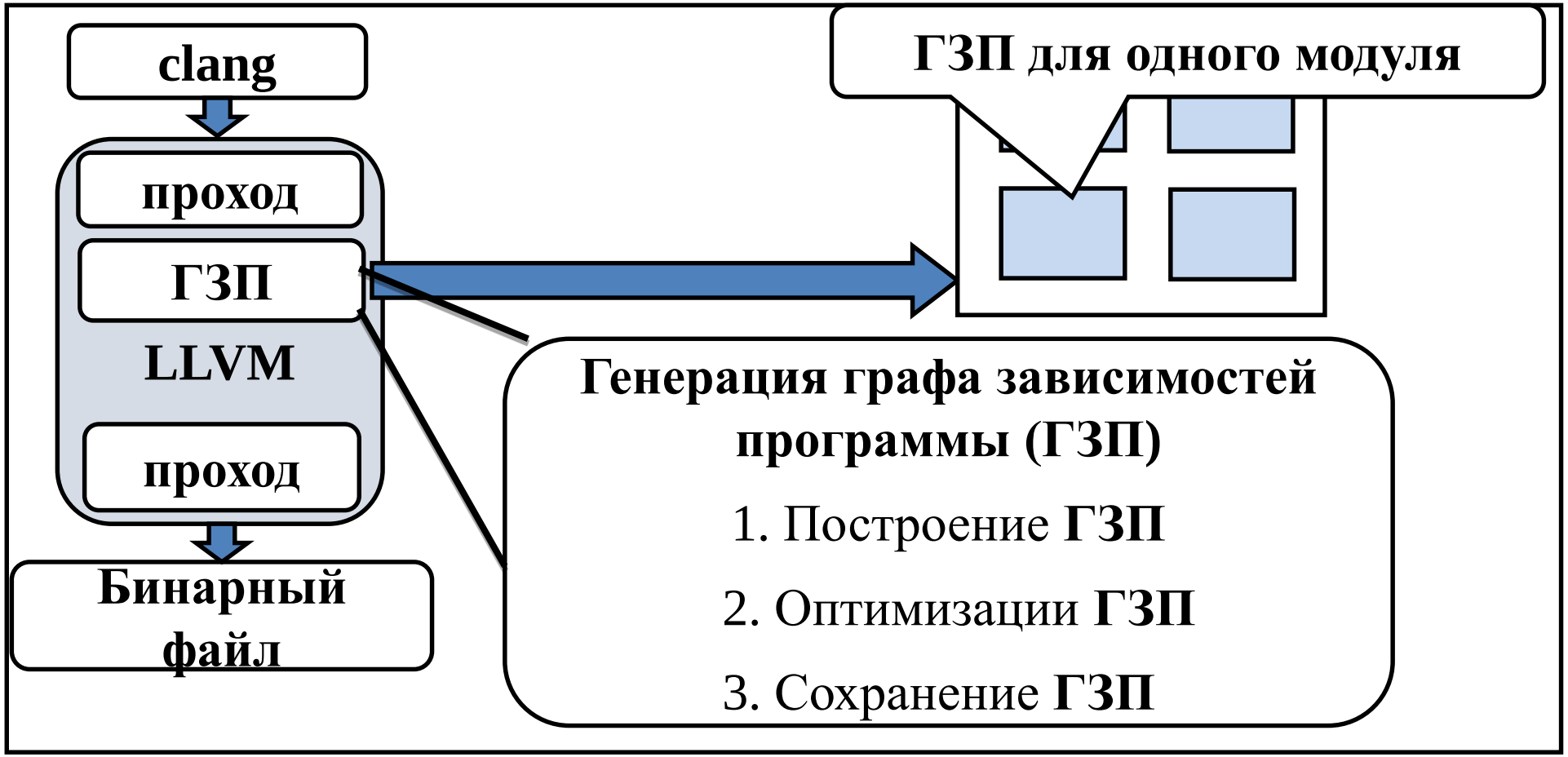
Требования:

- **Основан на семантическом анализе (ГЗП)**
- **Большая точность**
- **Применим для анализа промышленных объемов кода**
- **Находит клоны в пределах нескольких проектов**
- **Работает для всех языков программирования, если поддержать генерацию ГЗП**

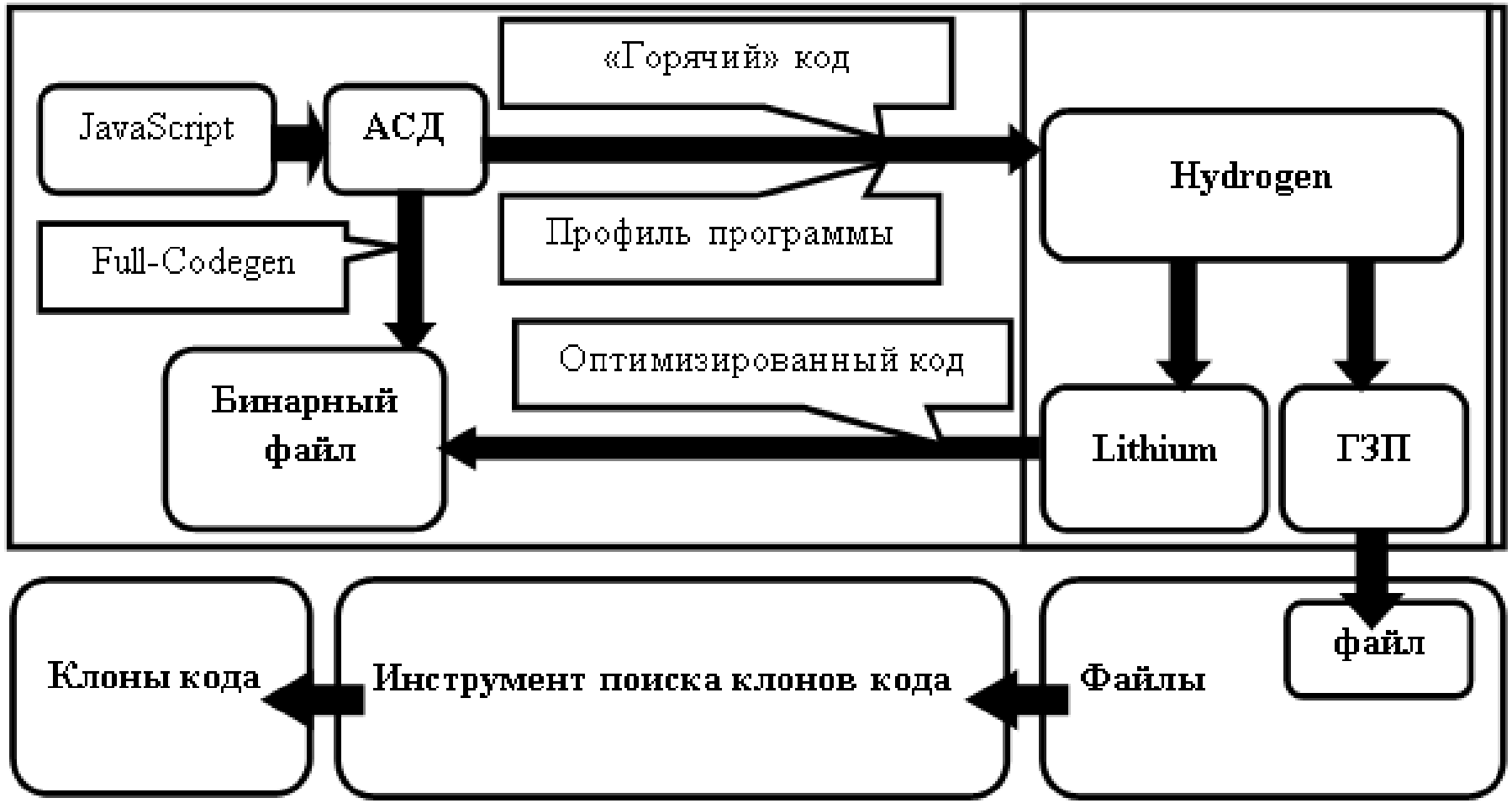
Архитектура



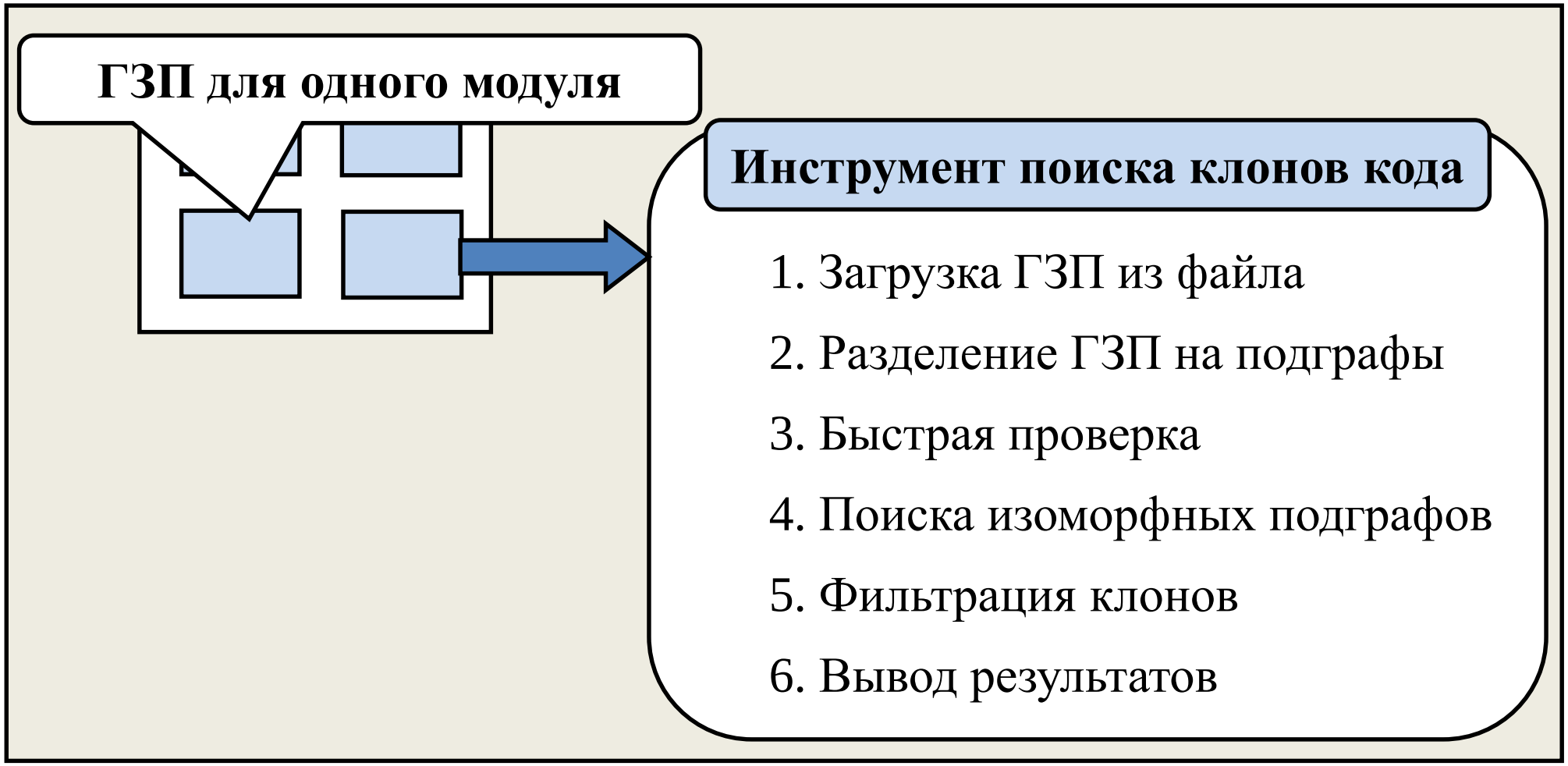
Генерация ГЗП на основе LLVM bitcode.



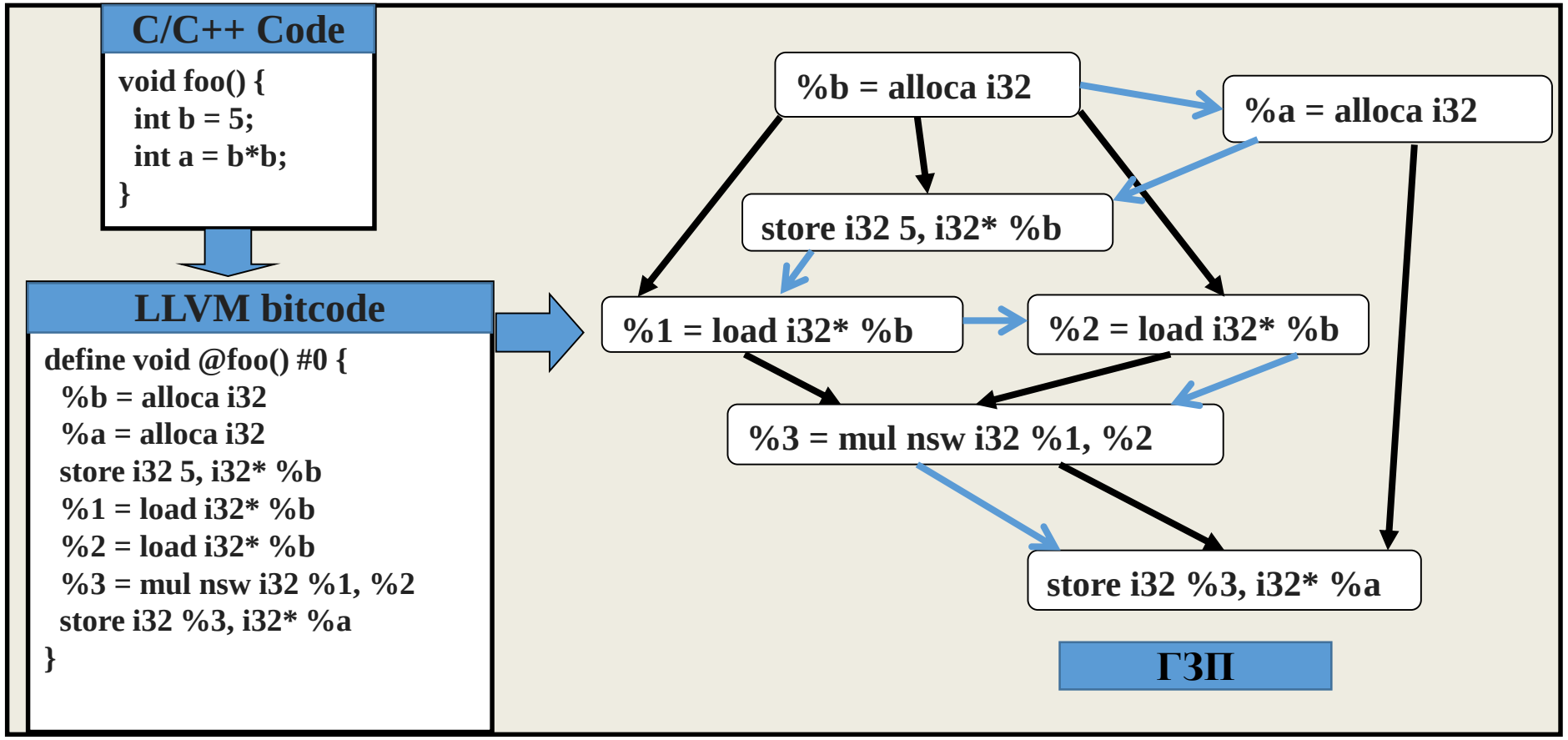
Генерация ГЗП на основе V8 JIT



Поиск клонов кода на основе ГЗП



Пример ГЗП (C++)



Зависимости по управлению

Зависимости по данным

Пример ГЗП (JavaScript)

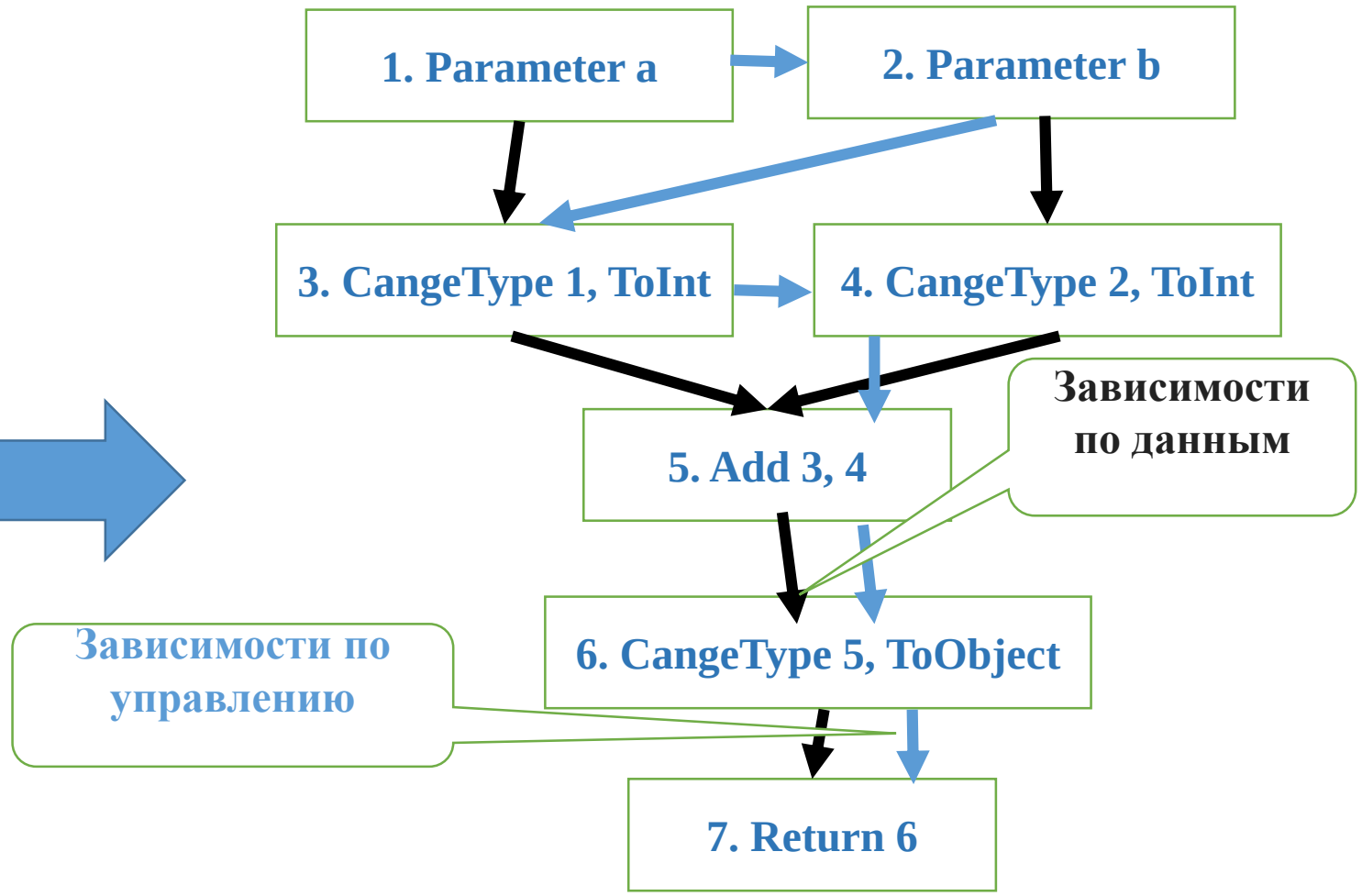
JavaScript

```
function f(a, b) {
        return a + b;
}
```

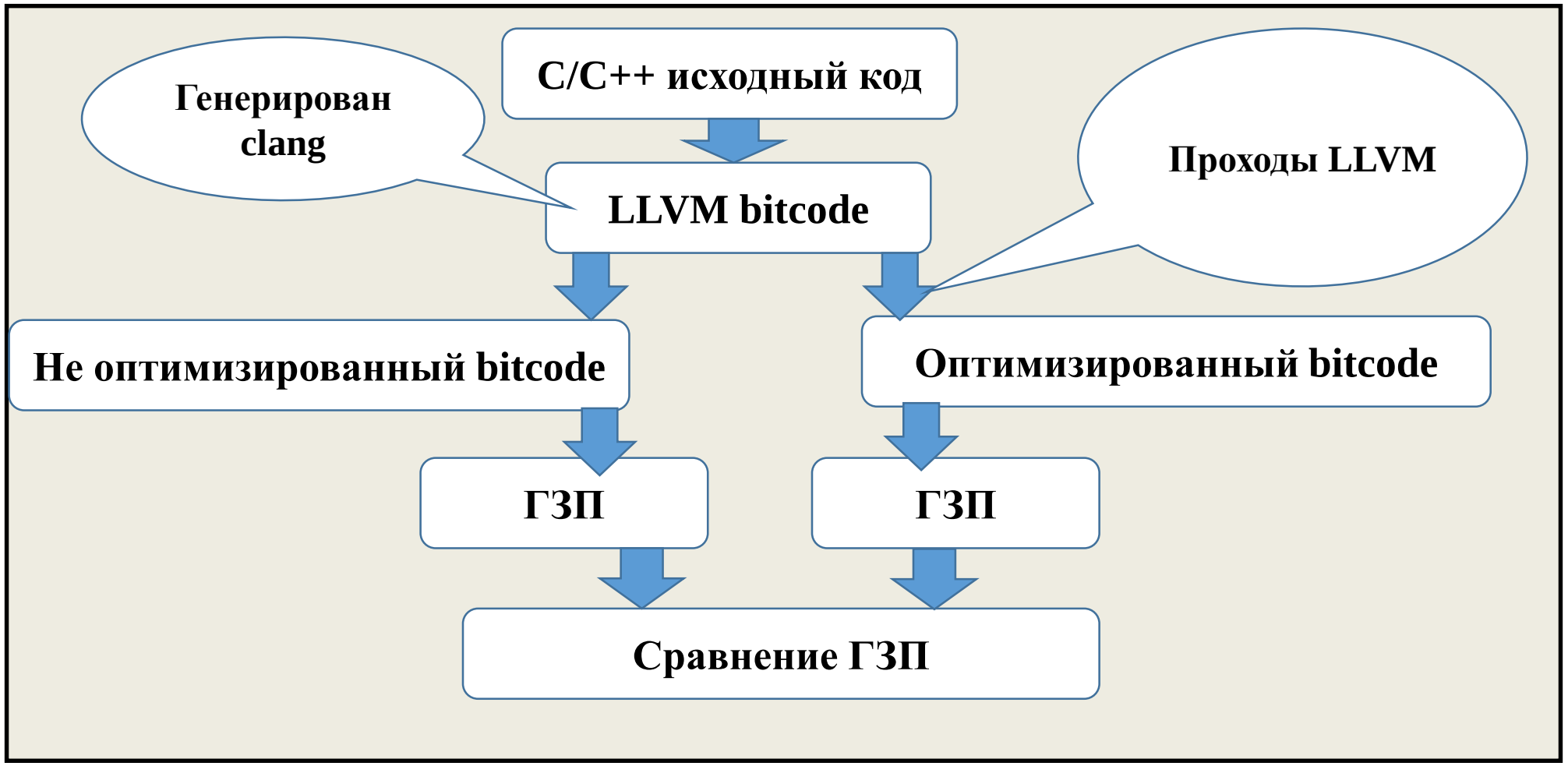
Hydrogen

Basic Block

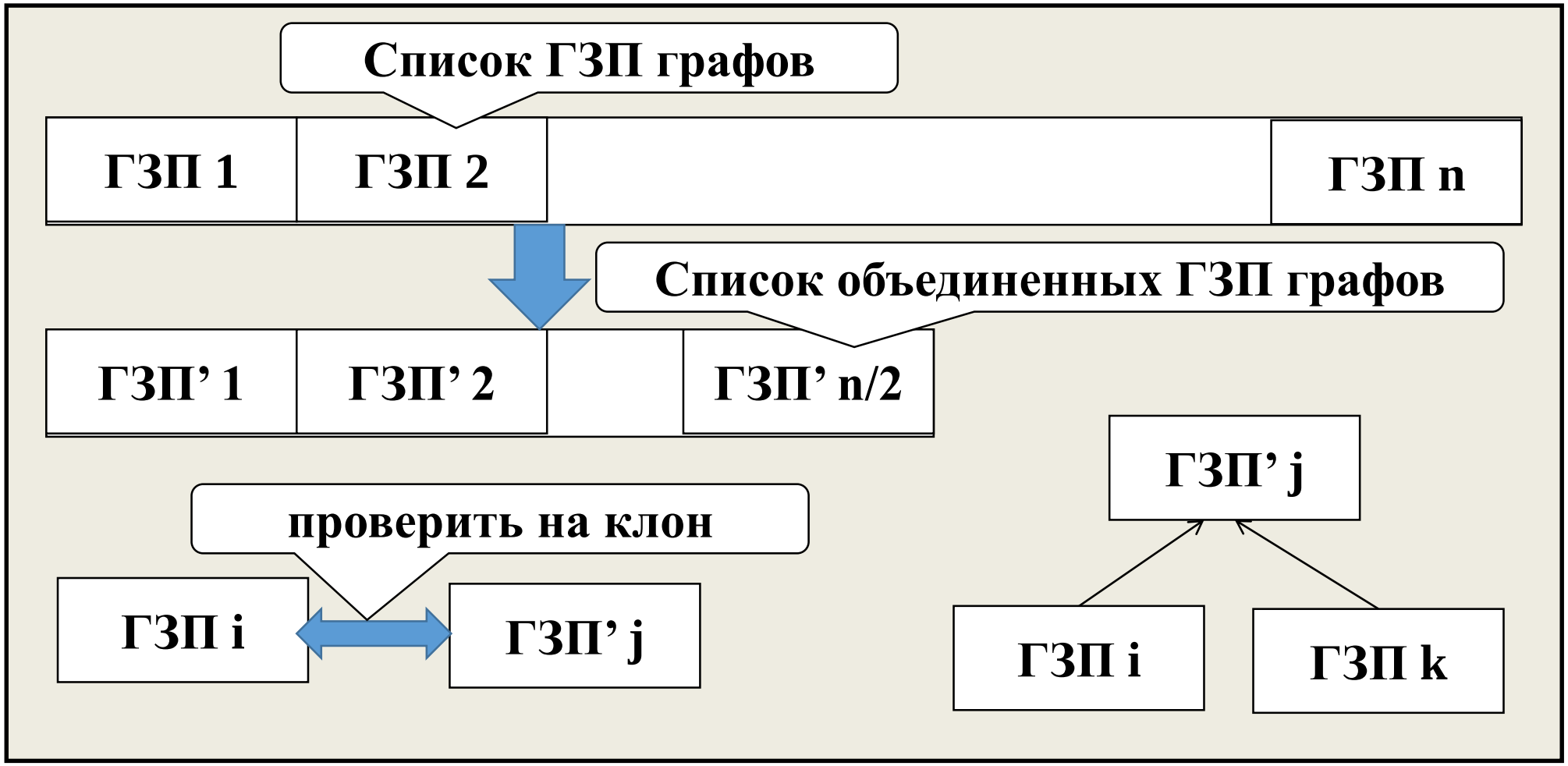
1. Parameter a
2. Parameter b
3. ChangeType 1, ToInt
4. ChangeType 2, ToInt
5. Add 3, 4
6. ChangeType 5, ToObject
7. Return 6



Автоматическое тестирование 1



Автоматическое тестирование 2



Преимущества данного подхода

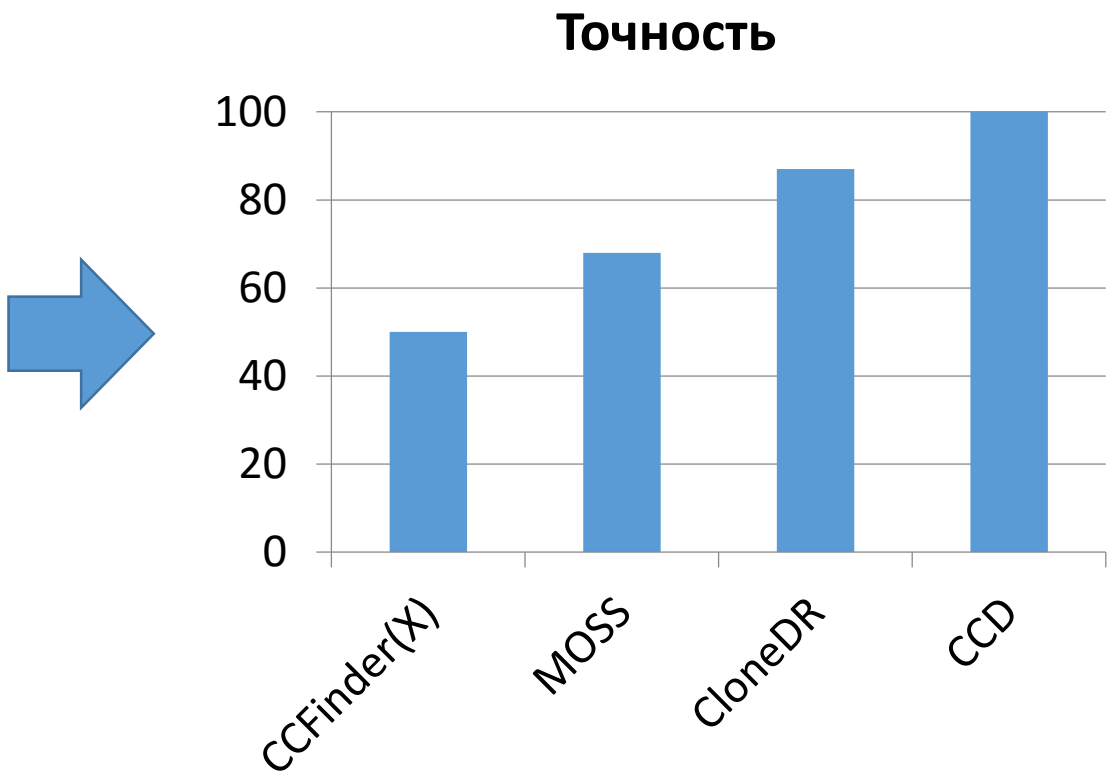
1. **Генерация ГЗП во время компиляции.**
2. **Отсутствие необходимости анализа между компилируемыми модулями.**
3. **Высокая точность (больше 90 %).**
4. **Применим для анализа промышленных объемов кода.**
5. **Находит клоны в пределах нескольких проектов.**
6. **Возможность параллельного запуска.**
7. **Автоматическая система тестирования точности реализованных алгоритмов.**

Сравнение инструментов поиска клонов

Все примеры клоны друг друга. Все тесты сравниваются с оригиналом.

Test Name	CCFinder(X)	MOSS	CloneDR	CCD
copy00.cpp	да	да	да	да
copy01.cpp	да	да	да	да
copy02.cpp	да	да	да	да
copy03.cpp	да	да	да	да
copy04.cpp	да	да	да	да
copy05.cpp	да	да	да	да
copy06.cpp	нет	нет	да	да
copy07.cpp	нет	да	да	да
copy08.cpp	нет	нет	нет	да
copy09.cpp	нет	нет	да	да
copy10.cpp	нет	нет	да	да
copy11.cpp	нет	нет	нет	yes
copy12.cpp	нет	да	да	да
copy13.cpp	нет	да	да	да
copy14.cpp	да	да	да	да
copy15.cpp	да	да	да	да

1. Chanchal K. Roy : Comparison and evaluation of code clone detection techniques and tools : A qualitative approach

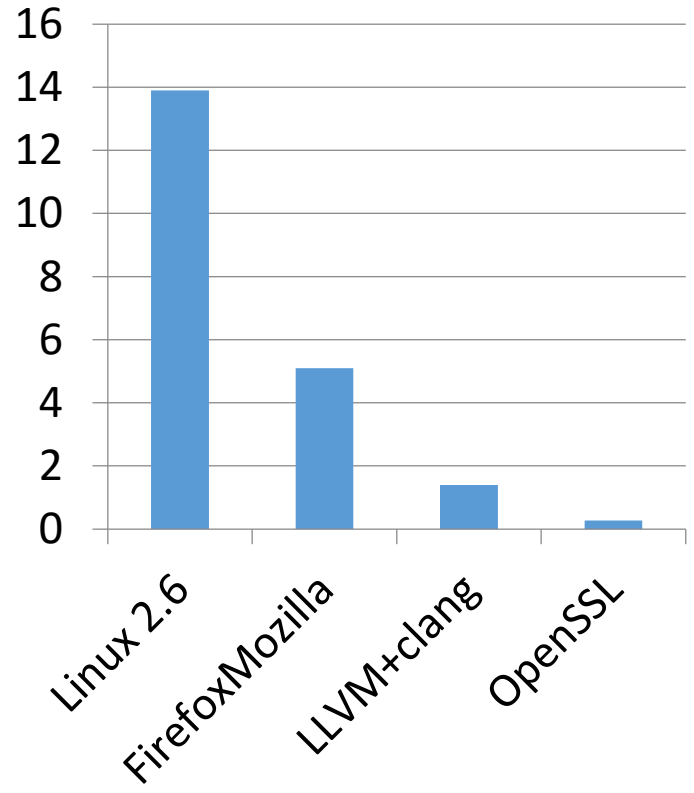


да– тест обнаружен как клон оригинала.
no – тест на обнаружен

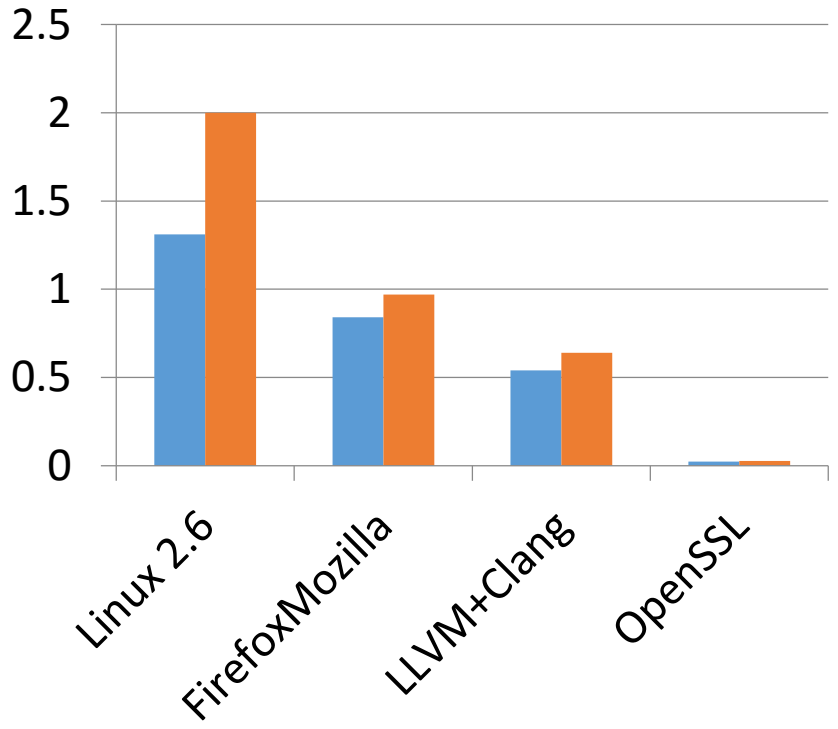
Генерация ГЗП (C/C++)

Intel core i3, 8GB Ram.

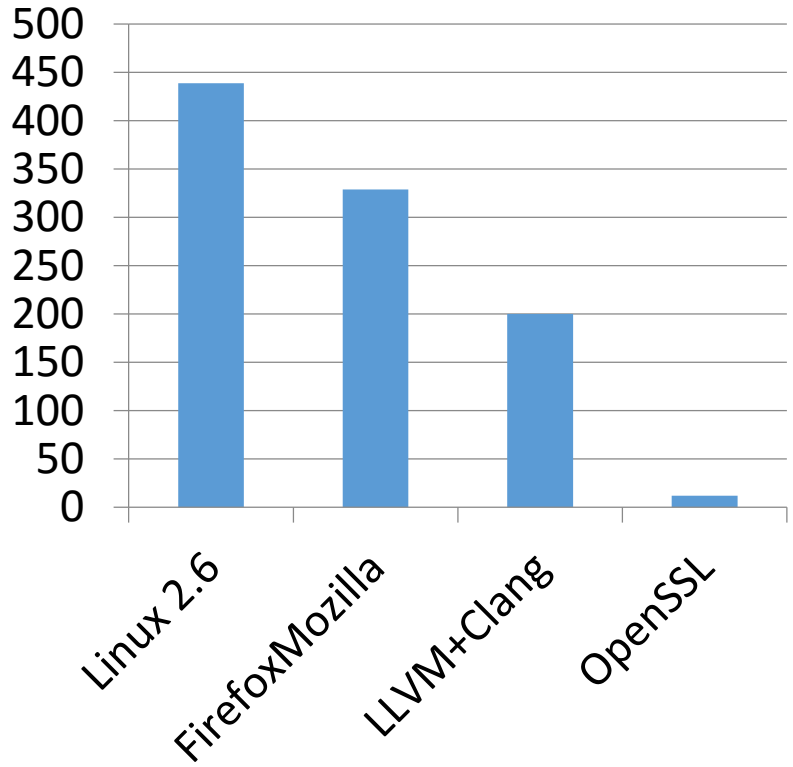
Строки исходного кода (миллионы)



Время генерации ГЗП (часы)



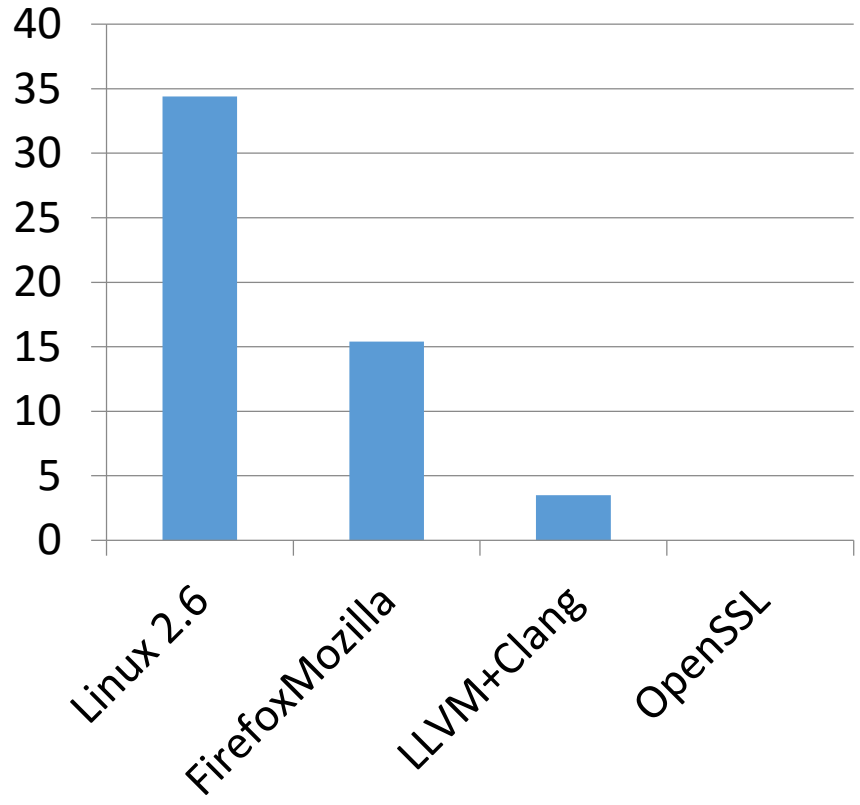
Размер ГЗП (мегабайт)



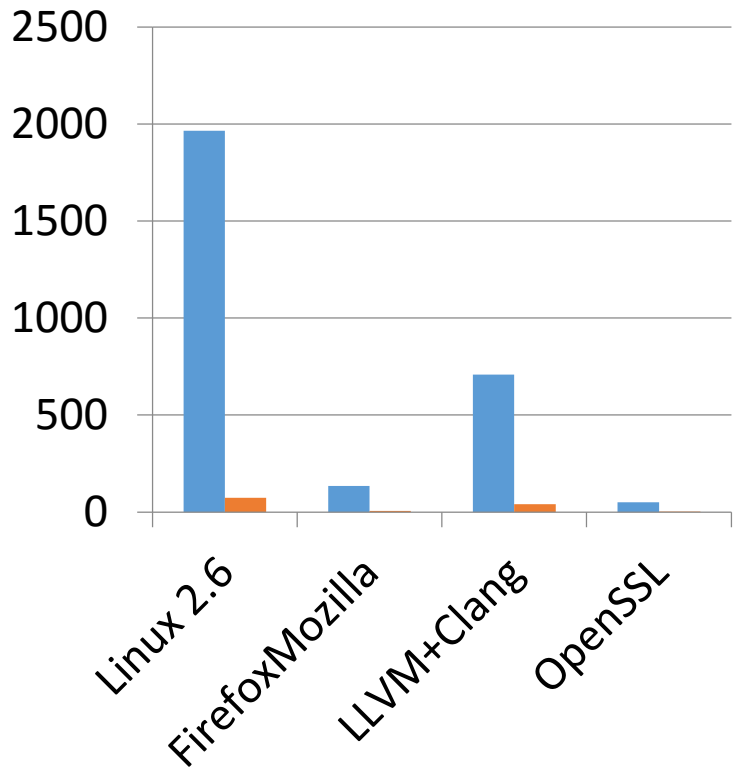
Поиск клонов кода (C/C++)

Схожесть больше 95%, минимальная длинна клона 25.
Intel core i3, 8GB Ram.

Время поиска клонов кода



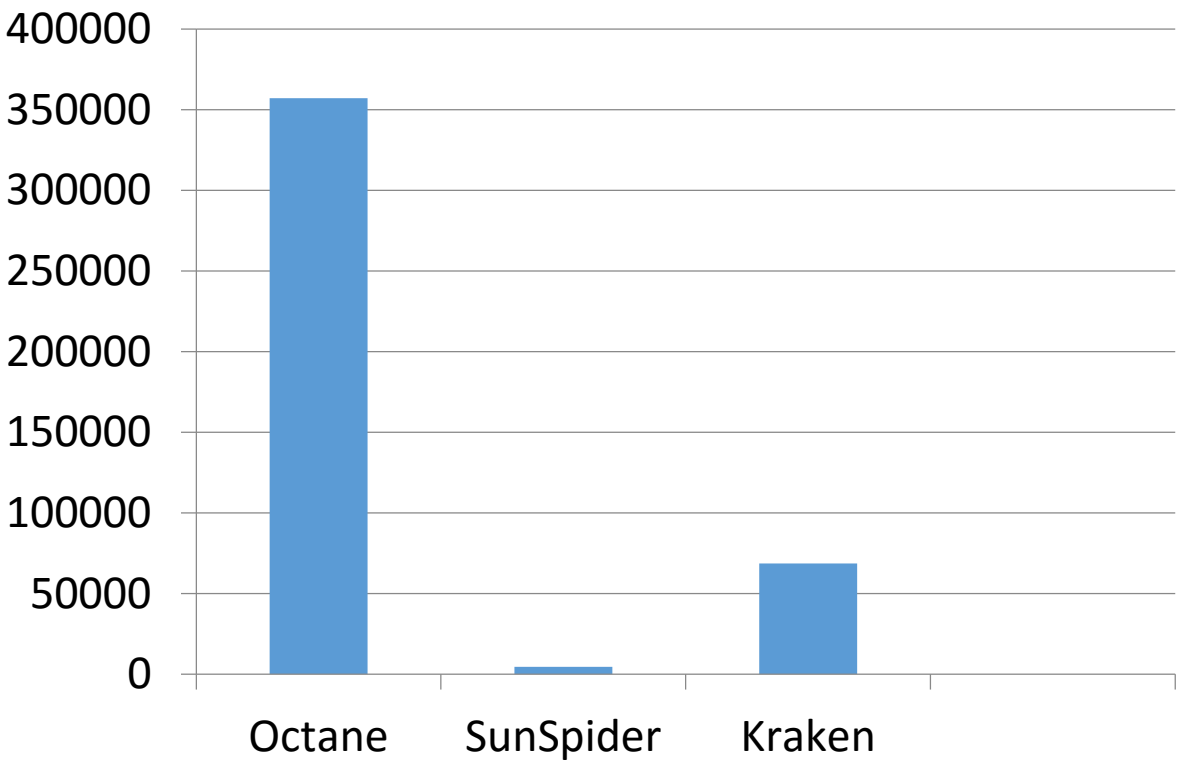
Количество найденных клонов и ложных срабатываний



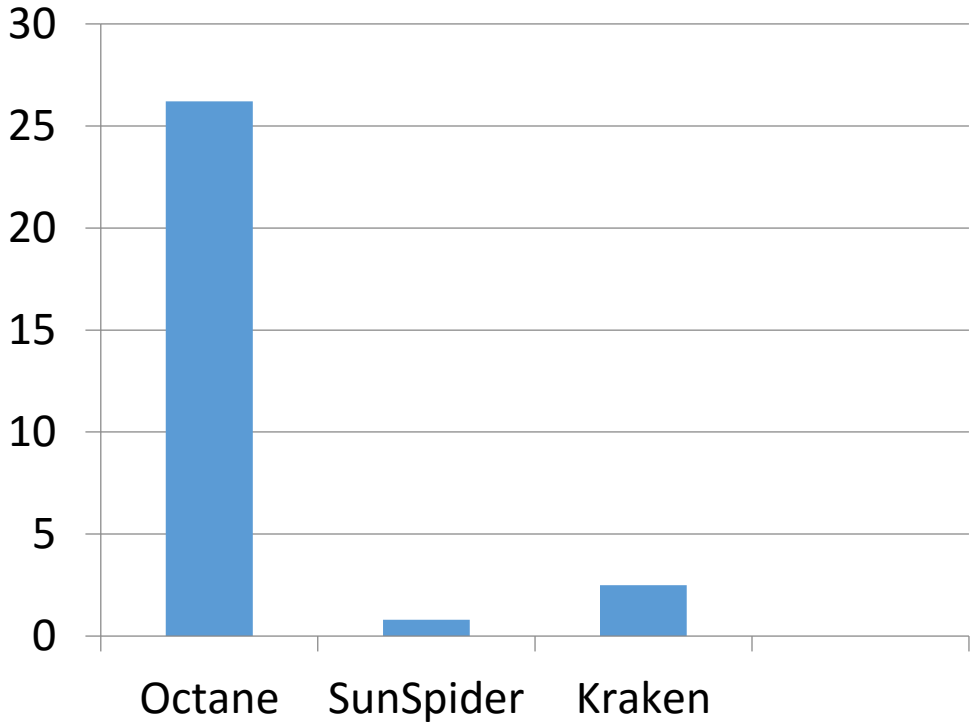
Генерация ГЗП (JavaScript)

Intel core i3, 8GB Ram.

Строки исходного кода

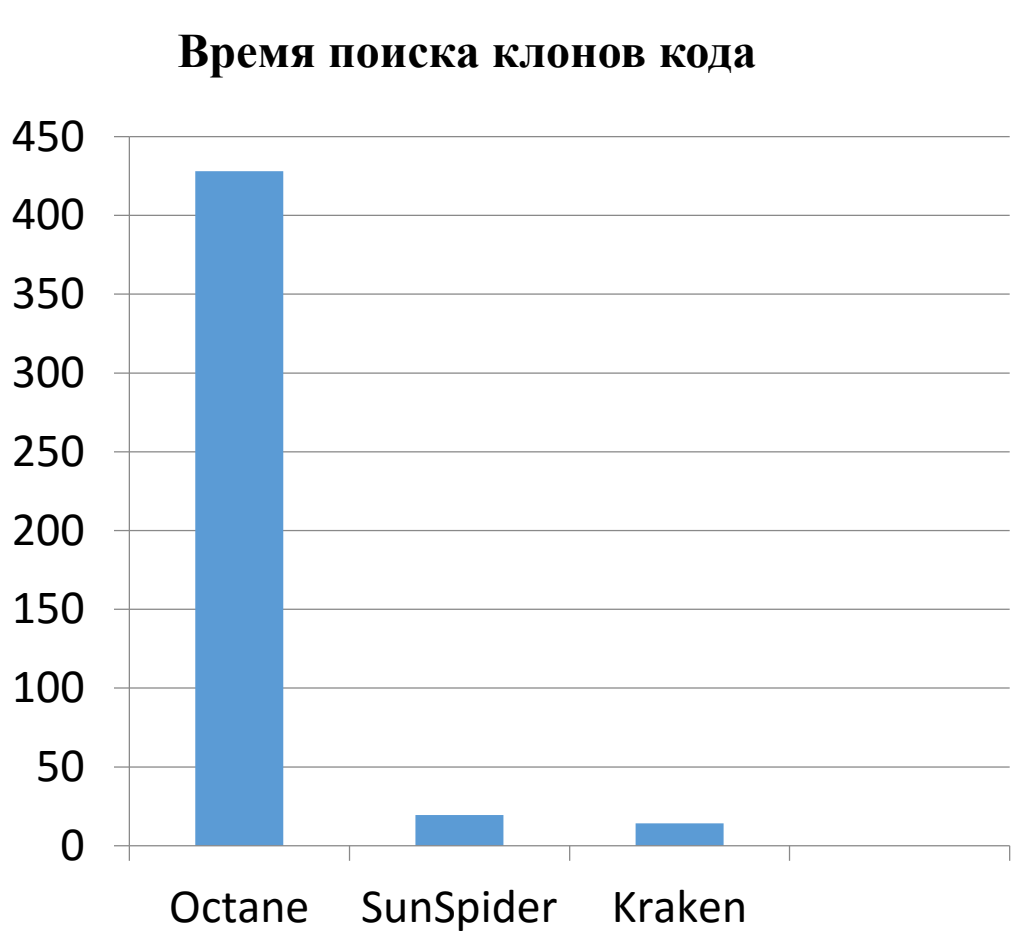


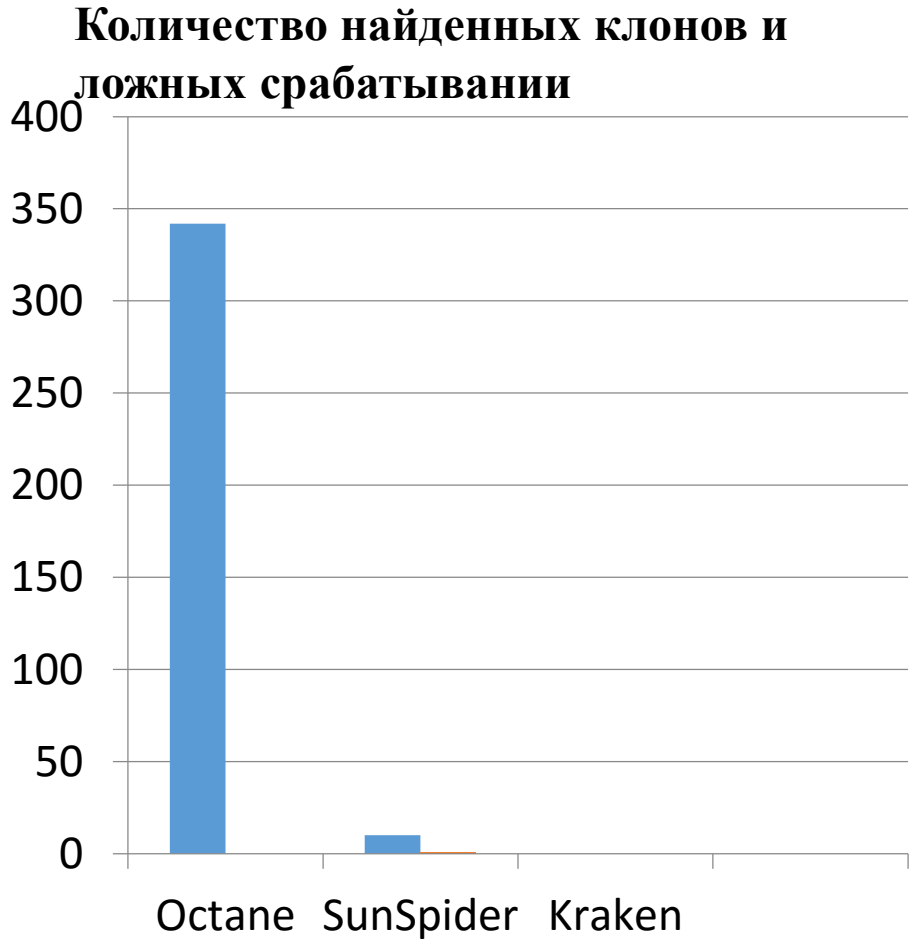
Размер ГЗП (мегабайт)



Поиск клонов кода (JavaScript)

Схожесть больше 90%, минимальная длина клона 10.
Intel core i3, 8GB Ram.





Дальнейшие работы

1. **Поиск семантических ошибок допущенных при адаптации скопированного фрагмента кода**

2. **Поиск уязвимостей по шаблону**

ISP RAS

Спасибо за внимание.