

Статический анализ кода в IntelliJ Rust

Студент: Артем Мухин, 444 группа
Руководитель: к. т. н. Литвинов Ю.В.
Консультант: Бескровный В.Н.
Рецензент: Кладов А.А.

Санкт-Петербург, 2019

Введение

—

Rust

- Современный язык системного программирования
- Небольшая среда исполнения
- Нет сборщика мусора
- Безопасность работы с памятью

Введение

—

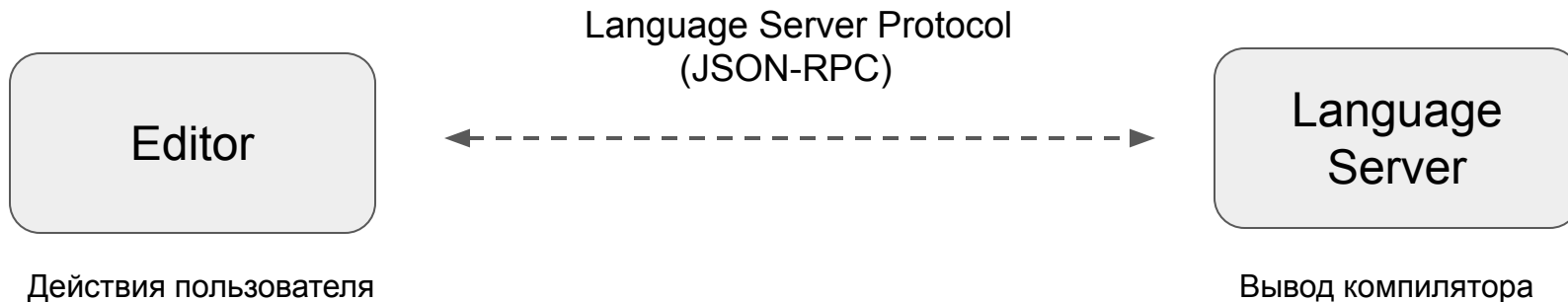
Анализатор заимствований

- Статический анализатор кода
- Реализован в компиляторе Rust
- Использует граф потока управления
- Проверяет правильность работы с памятью:
 - перемещения
 - заимствования

Анализатор перемещений проверяет только перемещения

Среды разработки для Rust

- **Rust Language Server** – реализация LSP для Rust
- Множество RLS плагинов для VS Code, Vim, Emacs



Rust Language Server

- Компилятор Rust не подходит для анализа кода “на лету”:
 - Не умеет работать в режиме сервера
 - Слишком бедный API
 - Не умеет работать с неполным кодом
- Минусы RLS:
 - Перезапускает компиляцию проекта после каждого изменения
 - Плохое автодополнение кода
 - Не анализирует неполный код
 - Не умеет работать с синтаксическим деревом

IntelliJ Rust

Плагин для поддержки Rust в IntelliJ IDEA, CLion и др.

<https://github.com/intellij-rust>

- Собственная реализация анализа кода
- Не использует компилятор
- Анализирует код “на лету”
- Умное автодополнение кода

Постановка задачи

- Разработать анализатор перемещений для IntelliJ Rust
 - Категоризация памяти
 - Построение графа потока управления
 - Анализ потока данных
- Внедрить анализатор в существующую архитектуру плагина
- Реализовать соответствующие инспекции в IDE

Модель владения в языке Rust

- Каждый ресурс имеет владельца
- В каждый момент времени владелец у ресурса только один
- Когда владелец выходит из области видимости, ресурс освобождается

```
struct S;  
fn foo(a: S) {}
```

```
let x = S;  
let y = x;  
let z = x; // Error: use of moved value  
foo(x);    // Error: use of moved value
```


Категоризация памяти

Категории:

- Временное значение
- Адрес локальной переменной
- Разыменованный указатель
- Обращение к полю

Категории изменяемости:

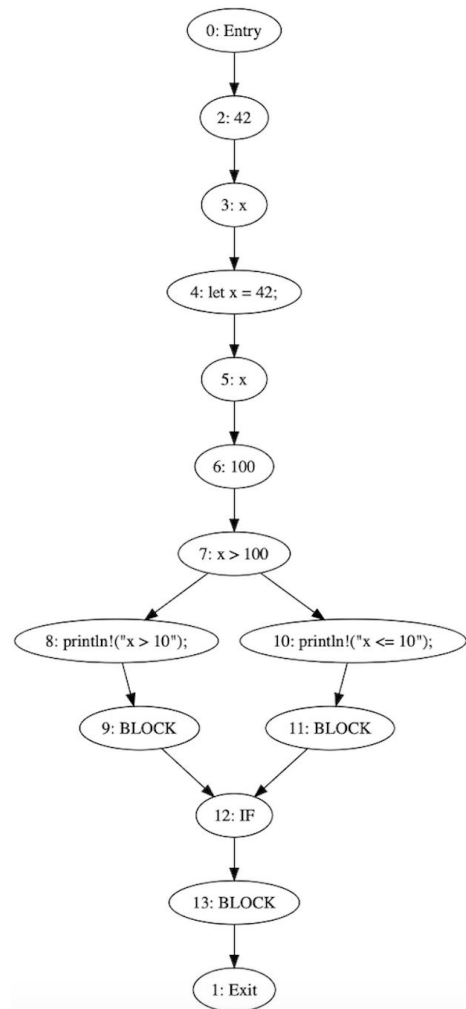
- Неизменяемое
- Изменяемое по определению
- Изменяемое как содержимое изменяемого

Граф потока управления

Вершины графа:

- Выражения и операторы
- Входная и выходная вершины

```
fn main() {  
    let x = 42;  
    if x > 100 {  
        println!("x > 10");  
    } else {  
        println!("x <= 10");  
    }  
}
```





Анализ перемещений

—

- **Сбор перемещений**
 - Проверка наличия перемещения
 - Проверка возможности перемещения
 - Запись информации о перемещении
- Построение графа потока управления
- Построение потока данных
- **Проверка перемещений**
 - Проверка инициализации памяти
 - Проверка корректности присваивания



Анализ перемещений: инспекция

```
1  struct S;  
2  
3 fn f(s: S) {}  
4  
5  fn main() {  
6     let s = S;  
7     f(s);  
8     print(s);
```

Use of moved value [more...](#) (⌘F1)

Апробация и тестирование

- Компонентные тесты для всех модулей
- Запуск анализа на нескольких реальных проектах
- Было обнаружено и исправлено несколько ложных срабатываний
- Новая функциональность доступна в стабильной версии плагина

Результат



В IntelliJ Rust была добавлена новая функциональность:

- Анализ перемещений
- Несколько инспекций и исправлений
- Категоризация памяти
- Построение графа потока управления
- Модуль анализа потока данных