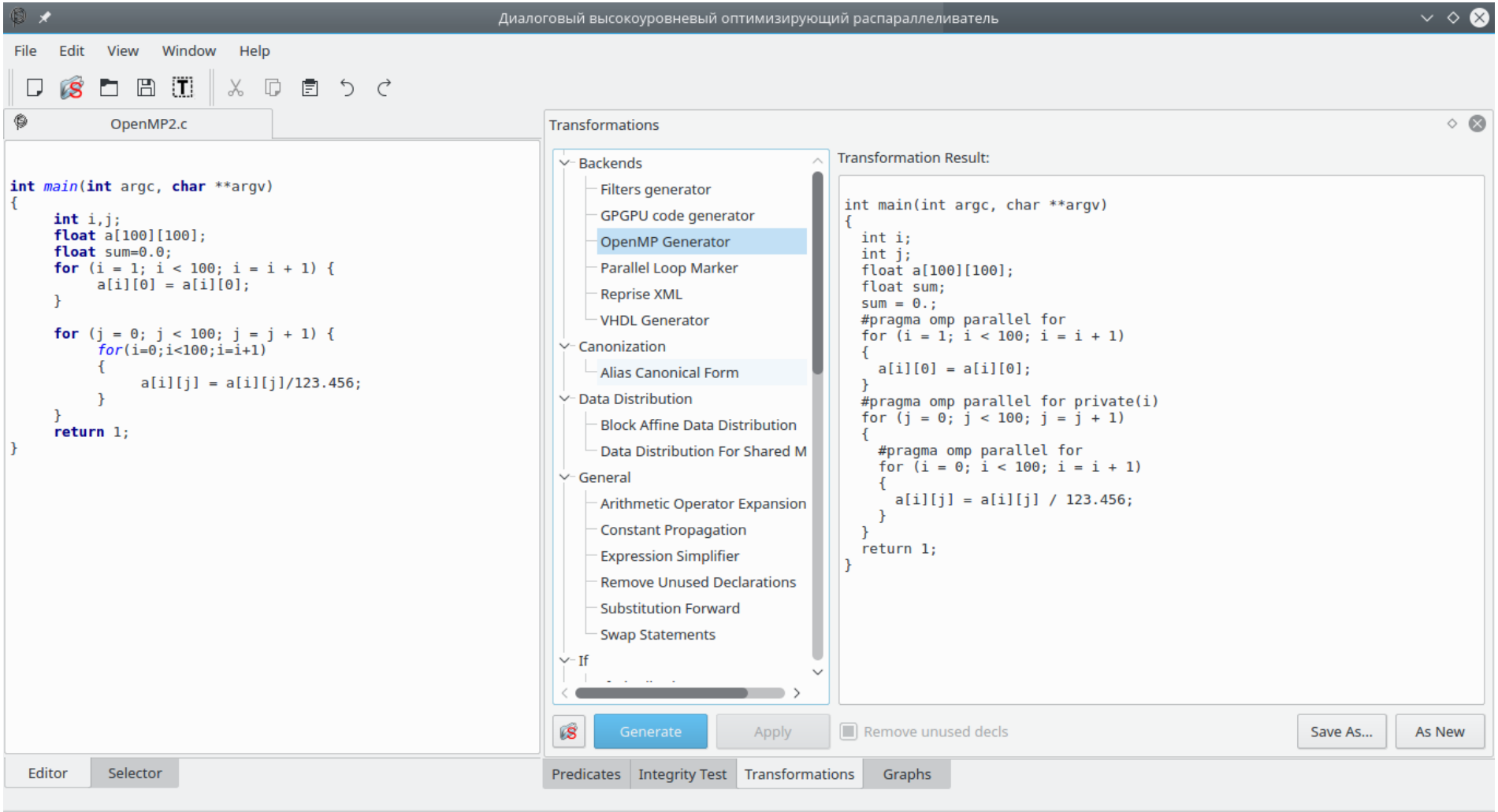
Интеграция компилятора clang со сторонней библиотекой оптимизирующих преобразований

Дубров Д. В., Патерикин А. Е.

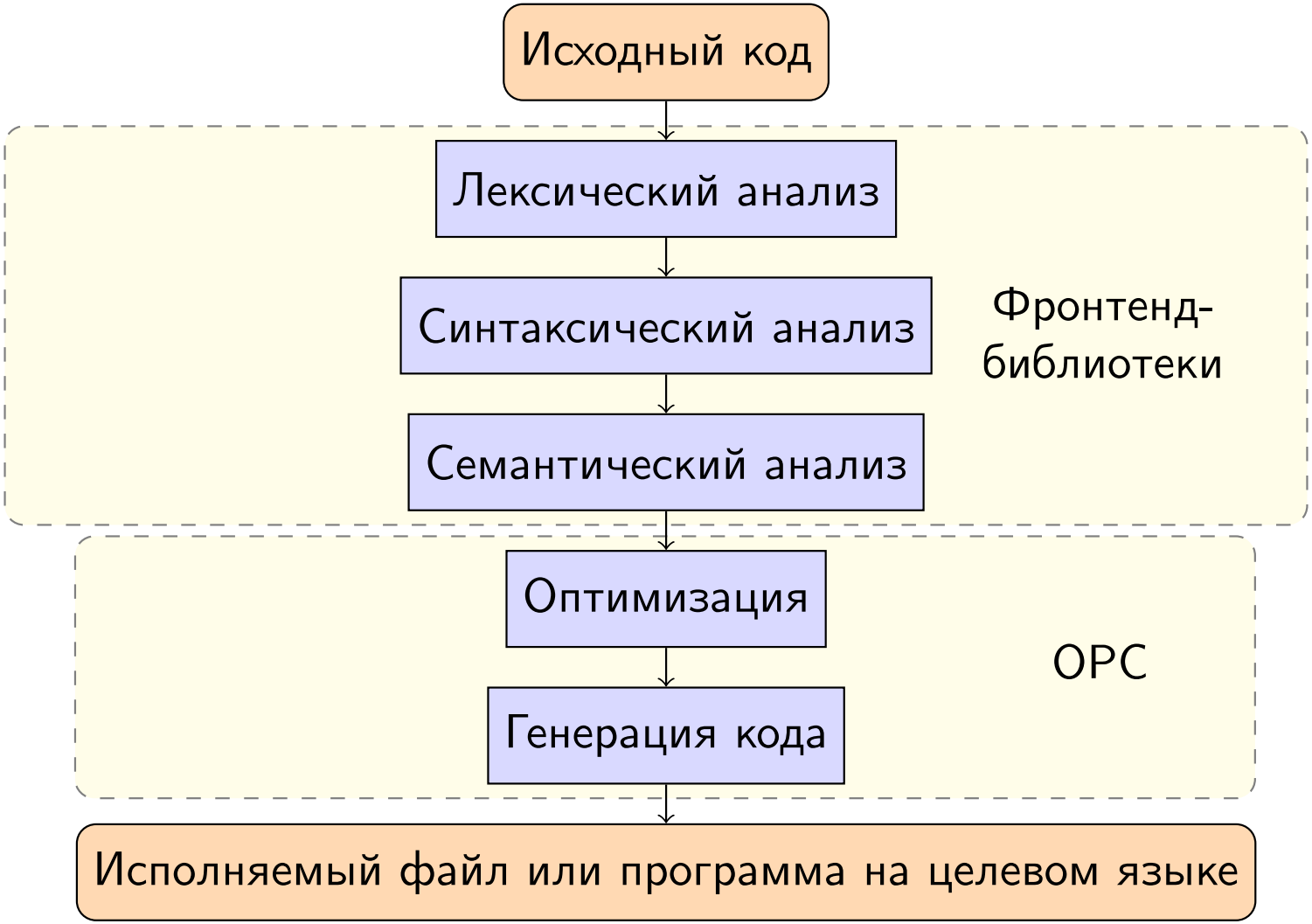
Институт математики, механики и компьютерных наук имени И.И. Воровича

5 Апреля 2017

Оптимизирующая распараллеливающая система



Оптимизирующая распараллеливающая система



Структура OPC

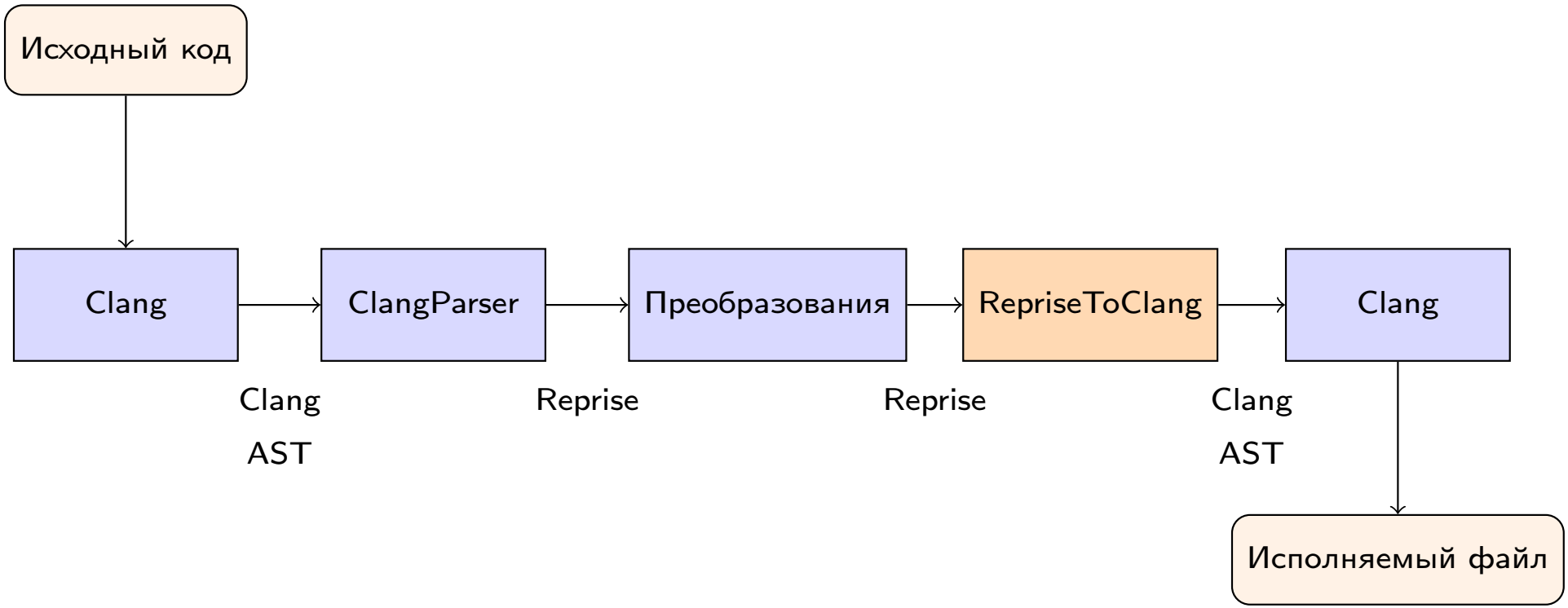
- Фронтенды $\leftarrow$ **LLVM + Clang**
- Анализы
- Трансформации
- Бекенды
- Вспомогательные модули

Абстрактное синтаксическое дерево

Абстрактное синтаксическое дерево — конечное, помеченное, ориентированное дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья с соответствующими операндами.

Система	Внутреннее представление
clang	Clang AST
OPC	Reprise

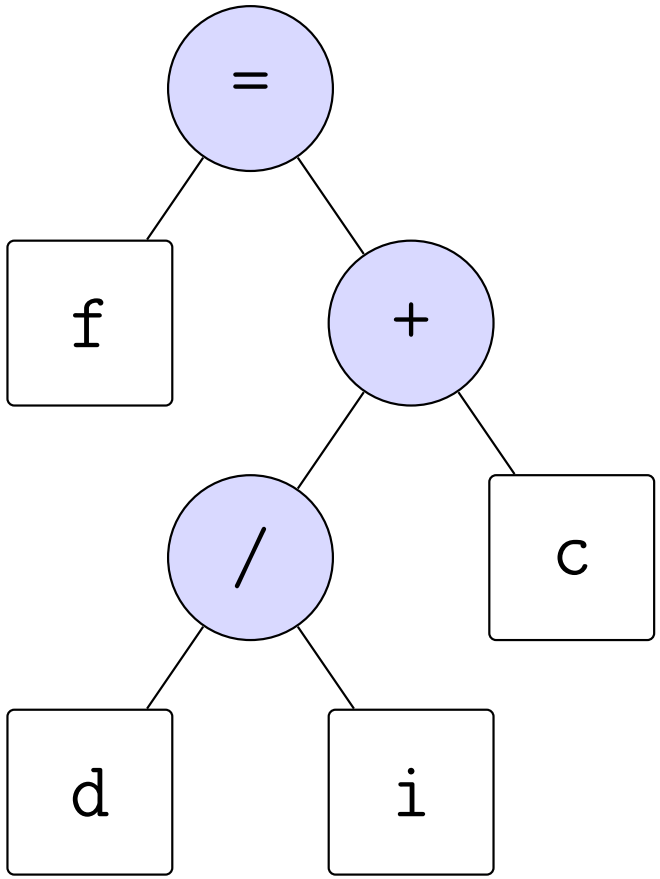
RepriseToClang



Сравнение деревьев Reprise и Clang AST

```
f = d/i + c;
```

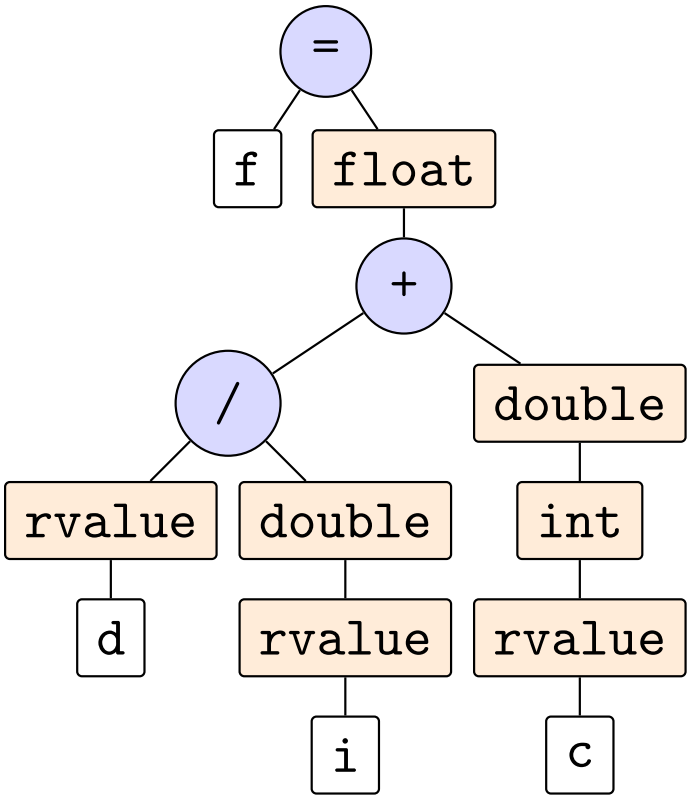
Reprise



Сравнение деревьев Reprise и Clang AST

```
f = d/i + c;
```

Clang AST



Неявные преобразования

Преобразования lvalue к rvalue C99 6.3.2.1	`double d;`
	`char c;`
	`d = c;`

Продвижение целочисленных типов C99 6.3.1.1 п2	`double d;`
	`char c;`
	`d = d + (double) (int) c;`

Обычные арифметические преобразования C99 6.3.1.8	`f = (float)(d/(double)i + (double)(int)c)`

Пример

```
int main()
{
    int i;
    float f;
    double d;
    char c;

    f = d/i + c;

    return 0;
}
```

```
f = (float)( d/(double)i + (double)(int)c )
```

Результат RepriseToClang

```
|-BinaryOperator 'float' '='
| |-DeclRefExpr 'float' lvalue Var 'f' 'float'
| `-ImplicitCastExpr 'float' <FloatingCast>
|   `-BinaryOperator 'double' '+'
|     |-BinaryOperator 'double' '/'
|     | |-ImplicitCastExpr 'double' <LValueToRValue>
|     | | `-DeclRefExpr 'double' lvalue Var 'd' 'double'
|     | `-ImplicitCastExpr 'double' <IntegralToFloating>
|     |   `-ImplicitCastExpr 'int' <LValueToRValue>
|     |     `-DeclRefExpr 'int' lvalue Var 'i' 'int'
|     `-ImplicitCastExpr 'double' <IntegralToFloating>
|       `-ImplicitCastExpr 'int' <IntegralCast>
|         `-ImplicitCastExpr 'char' <LValueToRValue>
|           `-DeclRefExpr 'char' lvalue Var 'c' 'char'
```

Представление операторов switch

Reprise

- Условное выражение.
- Метки `case` и `default`.

Clang

- Условное выражение.
- Метки `case` и `default`.
- Тело оператора.

Поддерживается GNU-диалект языка Си.

Представление операторов switch

Reprise

- Условное выражение.
- Метки `case` и `default`.

Clang

- Условное выражение.
- Метки `case` и `default`.
- Тело оператора.

Поддерживается GNU-диалект языка Си.

```
switch(i)
{
    case 1:  i = 2;
    default:  i = 3;
}
```

```
|-SwitchStmt
| |-<<<NULL>>>
| |-ImplicitCastExpr 'int' <LValueToRValue>
| | `-DeclRefExpr 'int' lvalue Var 'i' 'int'
| `-CompoundStmt
|   |-CaseStmt
|   | |-IntegerLiteral 'int' 1
|   | |-<<<NULL>>>
|   |   ...
|   `-DefaultStmt
|       ...
```

Ассемблерные вставки

Reprise

- Формат GCC (с ограничениями).
- Формат Microsoft (с ограничениями).
- Код хранится в виде строки.

```
__asm (" mov ax,ex ");
```

Clang

- Много свойств (волатильность, список токенов, constraints, clobber list...)
- Код хранится в виде строки.

```
|-GCCAsmStmt
```

Явные преобразования типов

```
float f;
char c;
char* p_c;

(int) f;
(double) c;
(int*) p_c;
```

```
|-CStyleCastExpr 'int' <FloatingToIntegral>
| `-ImplicitCastExpr 'float' <LValueToRValue>
|   `-DeclRefExpr 'float' lvalue Var 'f' 'float'
|-CStyleCastExpr 'double' <IntegralToFloating>
| `-ImplicitCastExpr 'char' <LValueToRValue>
|   `-DeclRefExpr 'char' lvalue Var 'c' 'char'
|-CStyleCastExpr 'int *' <BitCast>
| `-ImplicitCastExpr 'char *' <LValueToRValue>
|   `-DeclRefExpr 'char *' lvalue Var 'p_c' 'char *'
```

ClangInjection