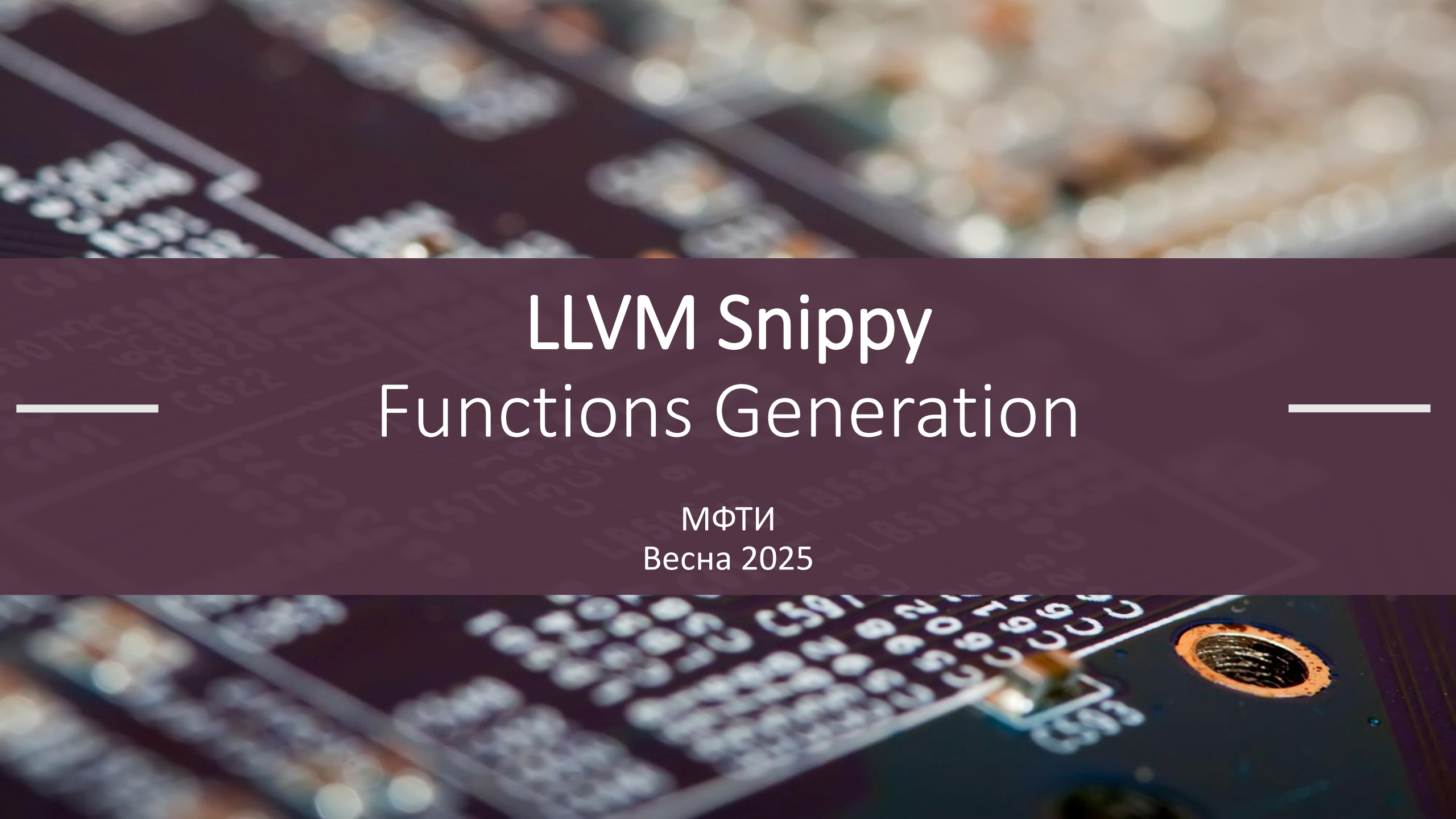


LLVM Snippy
Functions Generation

МФТИ
Весна 2025

Генерация функций

До этого мы рассматривали только генерацию одной функции

Что если мы хотим сгенерировать сниппет, содержащий несколько функций?

Генерация функций

До этого мы рассматривали только генерацию одной функции

Что если мы хотим сгенерировать сниппет, содержащий несколько функций?

Наивный подход: сгенерировать несколько функций, используя существующий подход к генерации одной функции

Генерация функций

До этого мы рассматривали только генерацию одной функции

Что если мы хотим сгенерировать сниппет, содержащий несколько функций?

Наивный подход: сгенерировать несколько функций, используя существующий подход к генерации одной функции

Проблема наивного подхода: такой сниппет с несколькими функциями равнозначен нескольким сниппетам с одной функцией

Генерация функций

До этого мы рассматривали только генерацию одной функции

Что если мы хотим сгенерировать сниппет, содержащий несколько функций?

Альтернативный подход: генерировать не только функции, но и вызовы сгенерированных функций

Генерация функций

До этого мы рассматривали только генерацию одной функции

Что если мы хотим сгенерировать сниппет, содержащий несколько функций?

Альтернативный подход: генерировать не только функции, но и вызовы сгенерированных функций

Не может ли получившийся сниппет выполняться *бесконечно*?

Теорминимум: граф вызовов

Граф вызовов (**Call Graph** или **CG**) – направленный граф, отображающий вызовы функций в программе

Вершины CG соответствуют функциям в программе

Дуга (*f*, *g*) в CG означает, что функция *f* вызывает функцию *g*

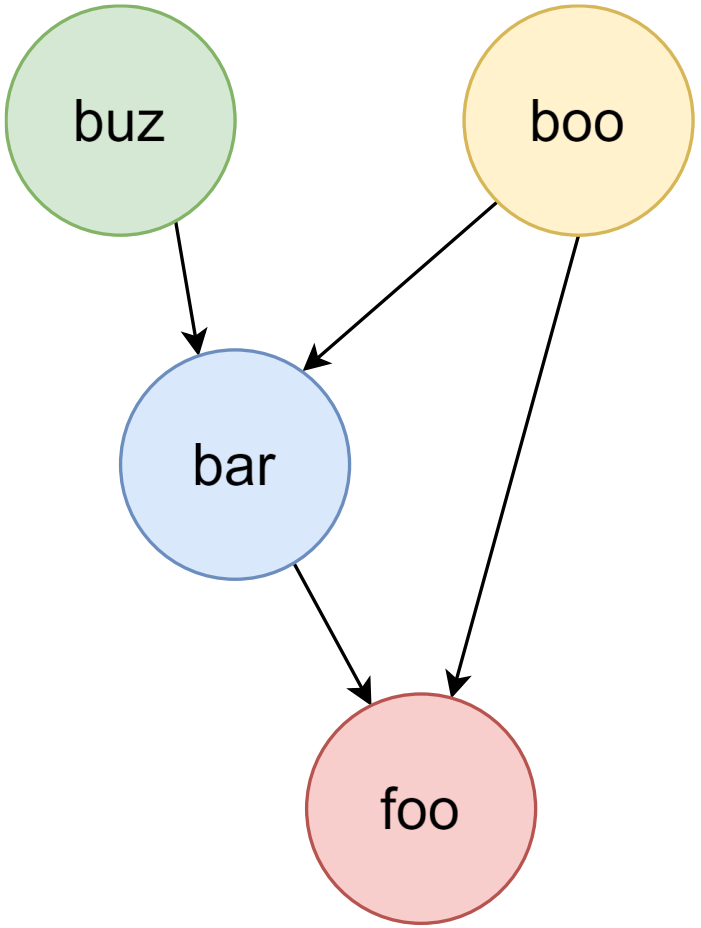
Теорминимум: граф вызовов

```
void foo() {}

void bar() { foo(); }

void buz() { bar(); }

void boo() {
  foo();
  bar();
}
```



Теорминимум: граф вызовов

Как вы думаете, обладает ли граф вызовов какими-то особыми свойствами?

Теорминимум: два вида графа вызовов

Граф вызовов бывает двух видов:

1. Статический
   - Отображает *все* возможные вызовы в программе
   - Может строить и использовать компилятор, например, для удаления недостижимых статических функций
2. Динамический
   - Отображает *все произошедшие* вызовы в программе во время ее выполнения
   - Строит, например, профилировщик

Теорминимум: статический граф вызовов

Какие вы видите проблемы с построением *статических* графов вызовов?

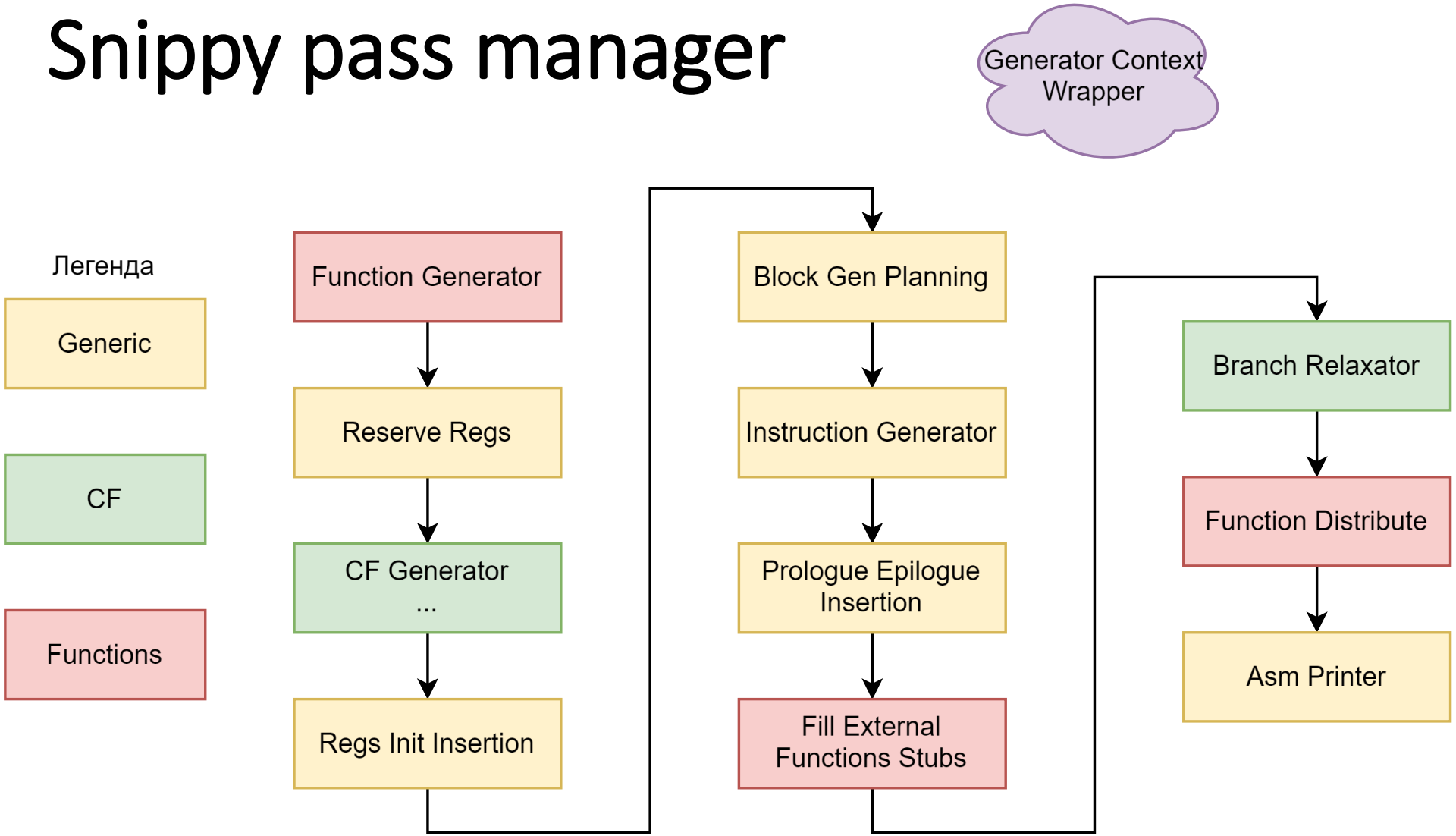
Теорминимум: статический граф вызовов

Какие вы видите проблемы с построением *статических* графов вызовов?

Статически очень сложно построить дуги для

1. Вызовов функций по указателю
2. Вызовов виртуальных функций

Snippy pass manager



[llvm-snippy/lib/FlowGenerator.cpp]

Passes: FunctionGenerator

Как видно из названия – генерирует функции

Passes: FunctionGenerator

Как видно из названия – генерирует функции

На самом деле генерирует *особый* граф вызовов и функции, являющиеся его вершинами

Passes: FunctionGenerator

Как видно из названия – генерирует функции

На самом деле генерирует *особый* граф вызовов и функции, являющиеся его вершинами

Почему граф вызовов *особый*?

Теорминимум: рекурсия

Рекурсия — вызов функции из неё же самой, непосредственно (**простая рекурсия**) или через другие функции (**сложная или косвенная рекурсия**)

Теорминимум: рекурсия

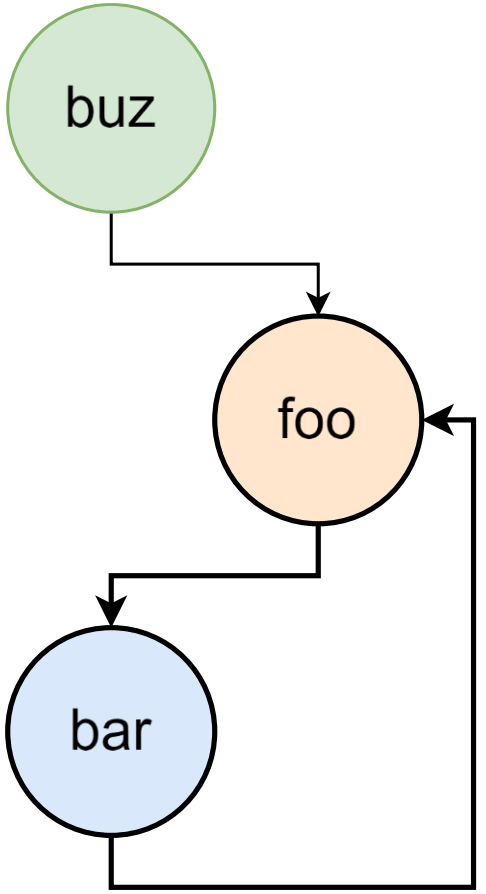
Рекурсия — вызов функции из неё же самой, непосредственно (**простая рекурсия**) или через другие функции (**сложная или косвенная рекурсия**)

Как выглядит рекурсия на графе вызовов?

Теорминимум: рекурсия на графе вызовов

На графе вызовов выглядит как цикл

```
void foo(int i) {
  printf(“%d\n”, i);
  if (i != 1)
    bar(i);
}
void bar(int n) {
  foo(n % 2
          ? (n / 2)
          : (3 * n + 1));
}
void buz() { foo(345023); }
```



Passes: FunctionGenerator

Как видно из названия – генерирует функции

На самом деле генерирует *особый* граф вызовов и функции, являющиеся его вершинами

Passes: FunctionGenerator

Как видно из названия – генерирует функции

На самом деле генерирует *особый* граф вызовов и функции, являющиеся его вершинами

Почему граф вызовов *особый*?

Passes: FunctionGenerator

Как видно из названия – генерирует функции

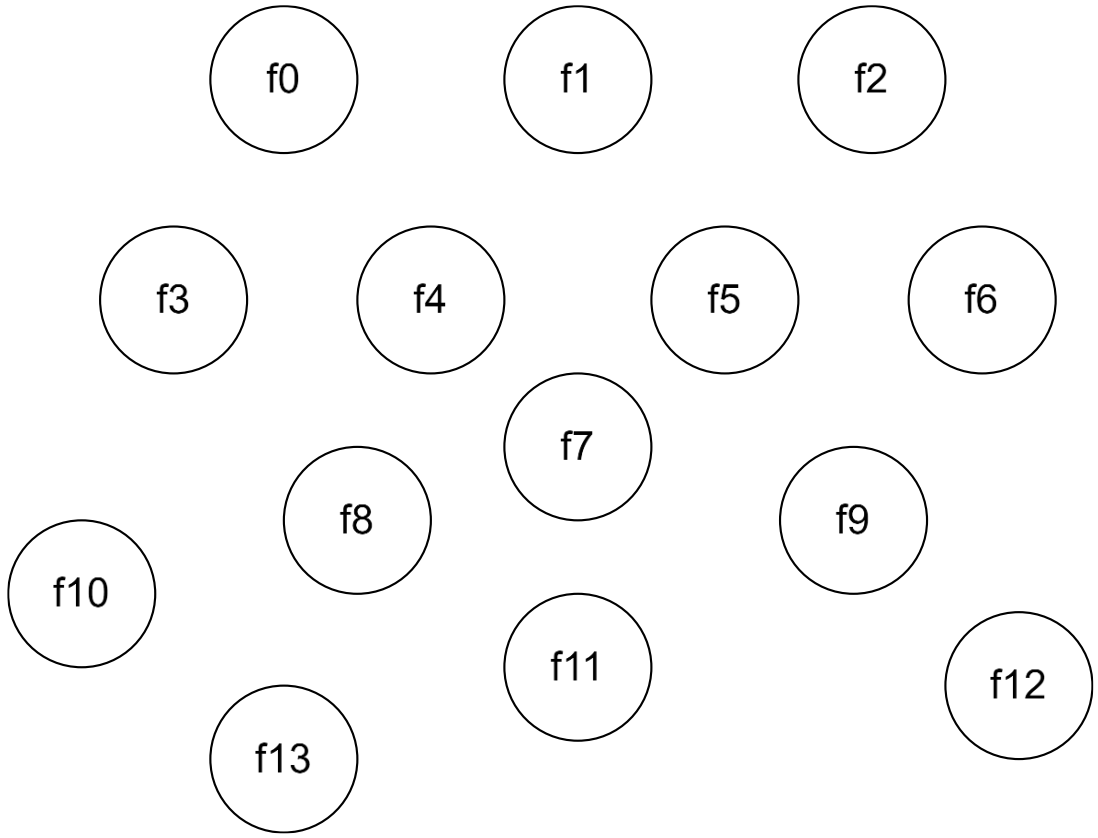
На самом деле генерирует *особый* граф вызовов и функции, являющиеся его вершинами

Почему граф вызовов *особый*?

Потому что он генерирует вызовы, *не создавая рекурсий*

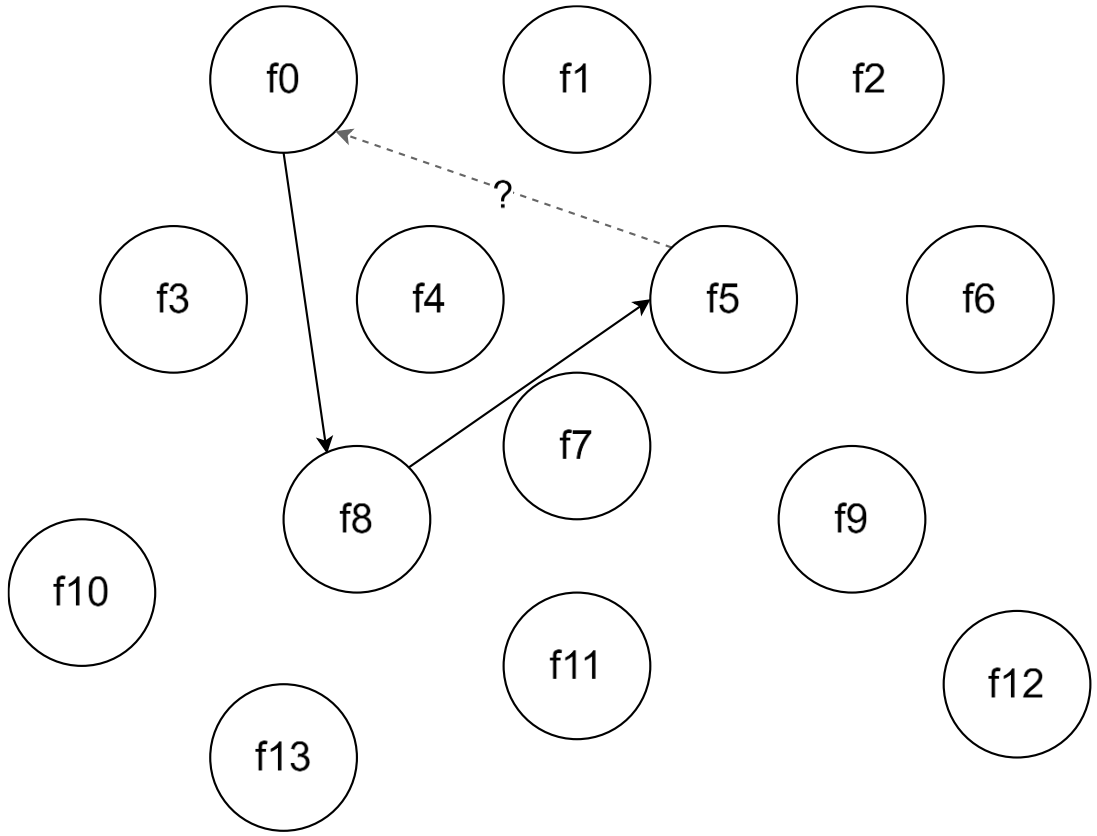
Генерация CG без рекурсий

- Генерация функций – не проблема
- Главная цель – провести ребра в графе, не создавая циклов



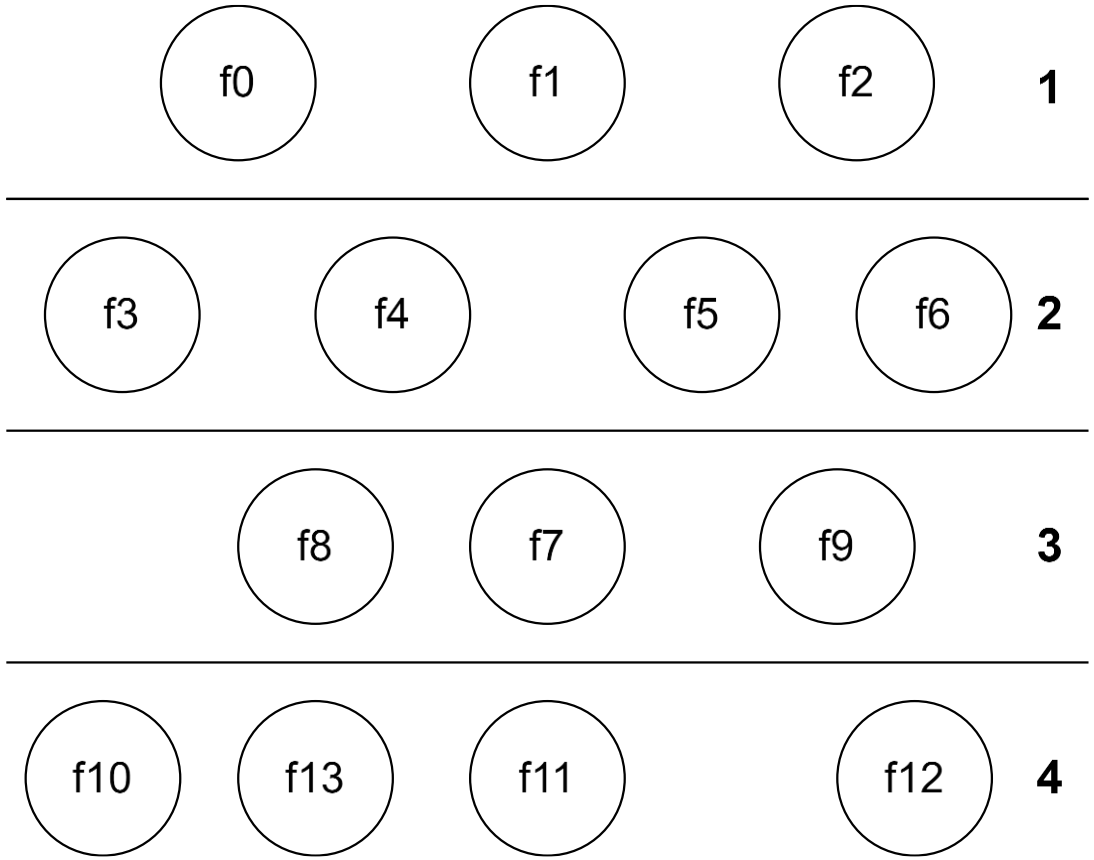
Генерация CG без рекурсий

- Генерация функций – не проблема
- Главная цель – провести ребра в графе, не создавая циклов
- **Наивный подход**:
  - Проводим ребро и проверяем, добавляет ли оно цикл



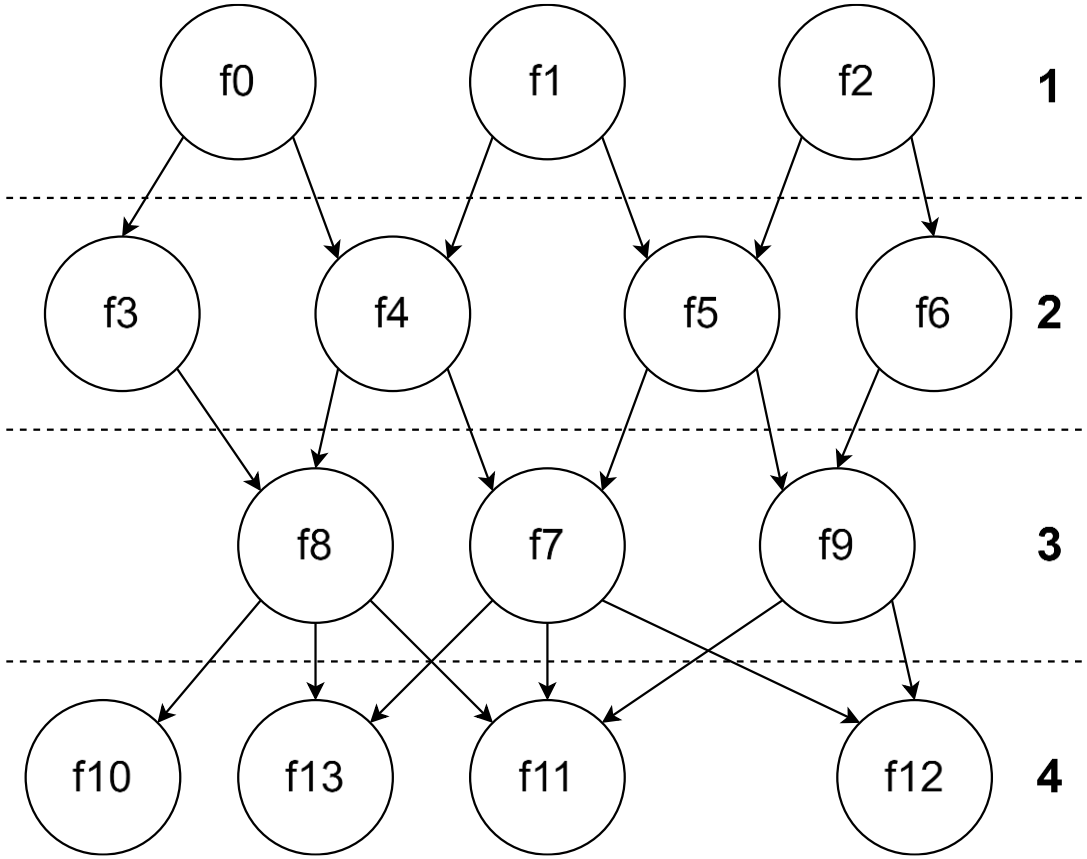
Генерация CG без рекурсий

- Генерация функций – не проблема
- Главная цель – провести ребра в графе, не создавая циклов
- **Альтернативный подход**:
  - Разбиваем ноды на «уровни»

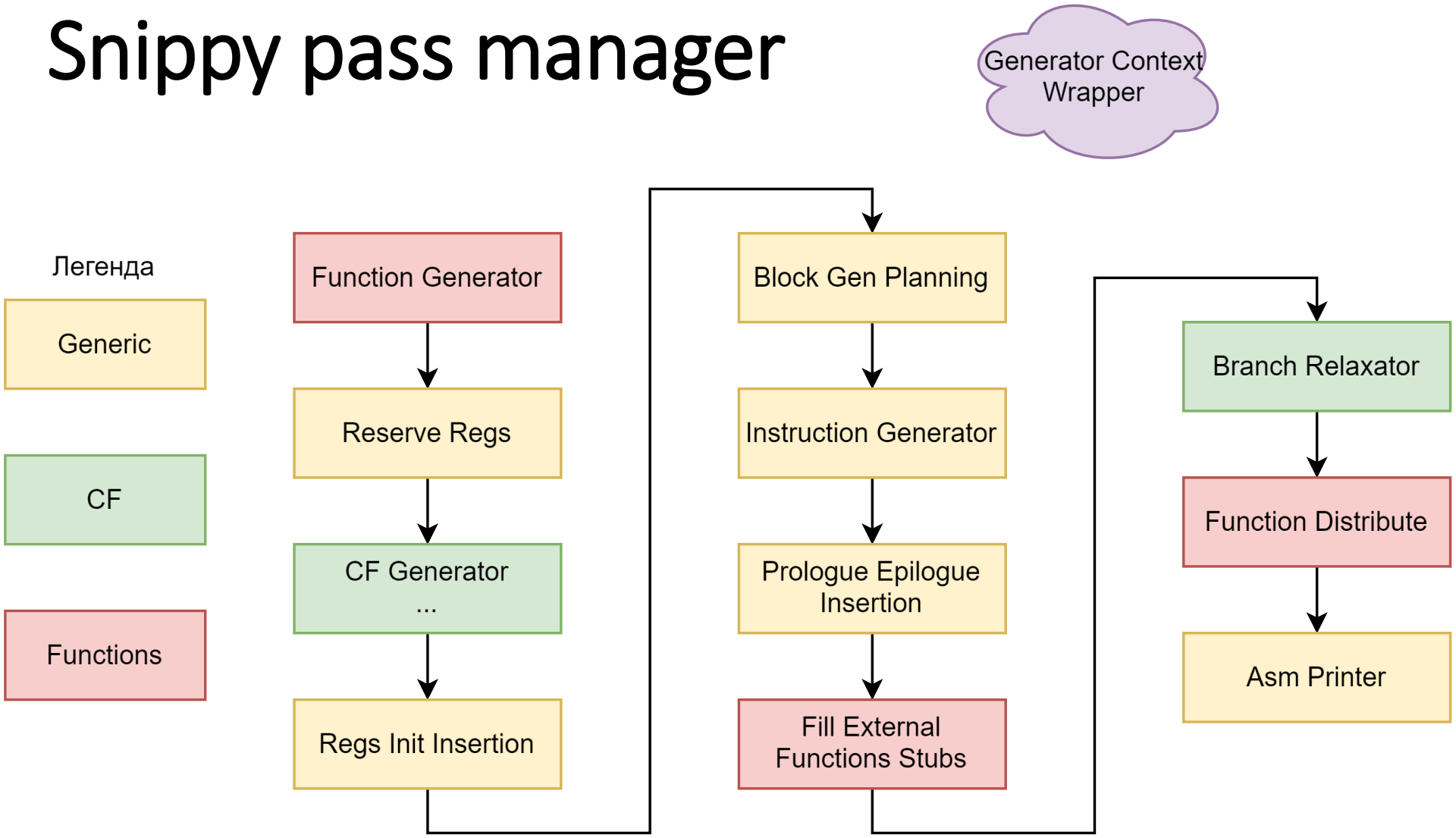


Генерация CG без рекурсий

- Генерация функций – не проблема
- Главная цель – провести ребра в графе, не создавая циклов
- **Альтернативный подход**:
  - Разбиваем ноды на «уровни»
  - Проводим ребра **только** от меньшего уровня к большему
  - Ребра между нодами на одном уровне запрещены



Snippy pass manager



llvm-snippy/lib/FlowGenerator.cpp

Passes: FillExternalFunctionsStubs

- Сниппи может встраивать внешние функции в call graph
- Как в таком случае генерировать трассу?

Passes: FillExternalFunctionsStubs

- Сниппи может встраивать внешние функции в call graph
- Как в таком случае генерировать трассу?
- А какие операции разрешены в подключаемых внешних функциях?

Passes: FillExternalFunctionsStubs

- Сниппи может встраивать внешние функции в call graph
- Как в таком случае генерировать трассу?
- А какие операции разрешены в подключаемых внешних функциях?
- Внешние функции должны восстанавливать архитектурное состояние перед возвратом
- С точки зрения выполнения сниппета такие внешние функции эквивалентны пустым функциям

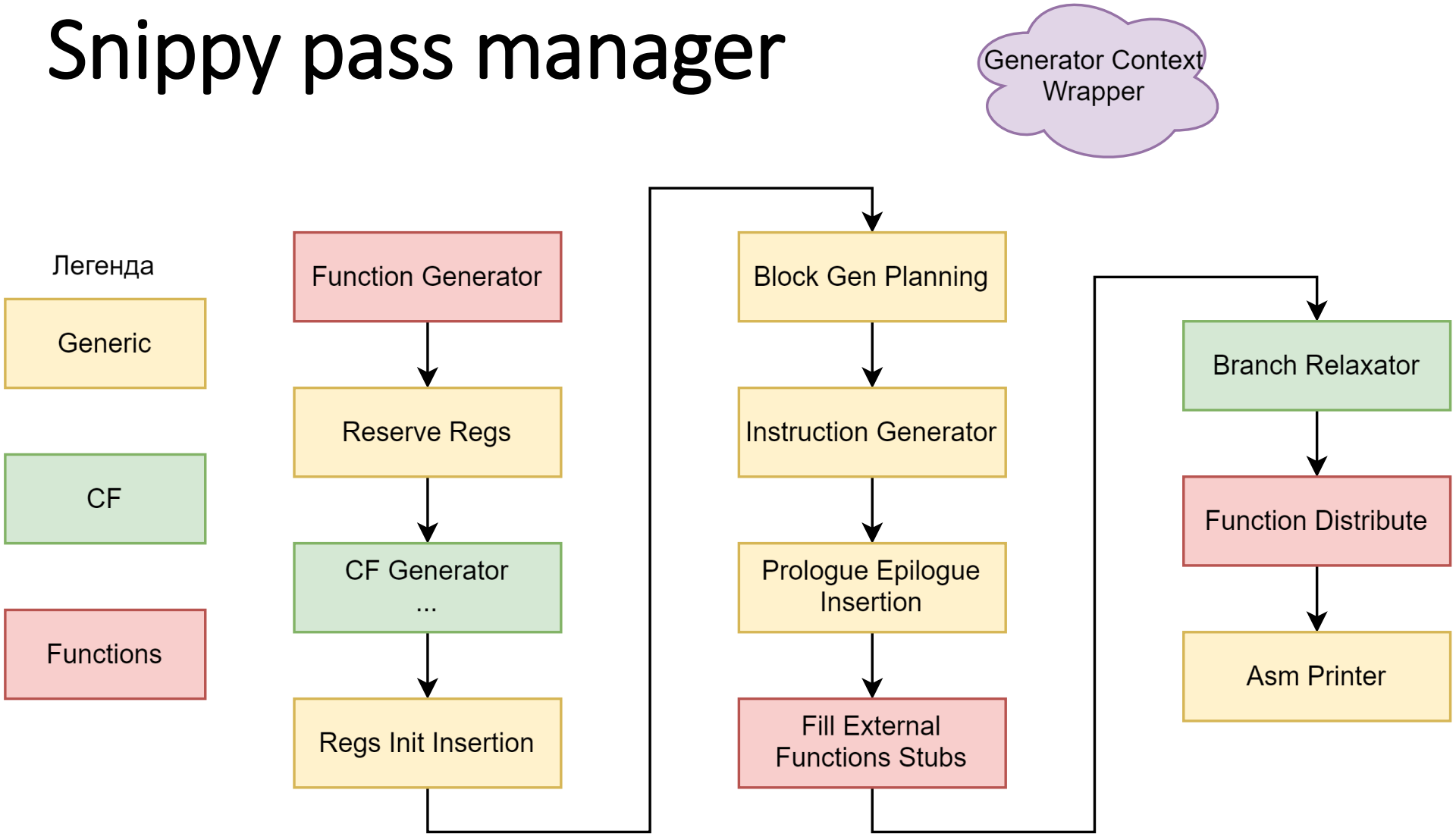
Passes: FillExternalFunctionsStubs

- Сниппи может встраивать внешние функции в call graph
- Внешние функции должны восстанавливать архитектурное состояние перед возвратом
- С точки зрения выполнения сниппета такие внешние функции эквивалентны пустым функциям
- FillExternalFunctionsStubs временно создает функции-заглушки для внешних функций, чтобы было возможно запустить модель для сбора трассы

Passes: FillExternalFunctionsStubs

- Пример применения: проверка нелегальных инструкций
- Подготавливается набор функций с нелегальными инструкциями
- Генерируется сниппет с вызовом этих функций
- Сниппет линкуется с нелегальными функциями и обработчиками прерываний/исключений
- Обработчики регистрируют все нелегальные события и восстанавливают архитектурное состояние

Snippy pass manager



llvm-snippy/lib/FlowGenerator.cpp

Passes: FunctionDistribute

- Распределение кода в обычных сниппетах довольно скучное
- Как сделать расположение кода интереснее?
- Распределить функцию по нескольким RX секциям

To be continued …

На следующем занятии

- Будем находить баги в арифметических инструкциях
- Научимся подготавливать сниппеты к запуску в тестовом окружении