# Доказательство корректности функциональной реализации библиотеки принтер-комбинаторов с выбором

Королихин Владимир Игоревич, 17.Б11-мм

Научный руководитель: д.т.н., доцент Д.В. Кознов
Консультант: к.ф.-м.н. А.В. Подкопаев

# Обзор тематики работы

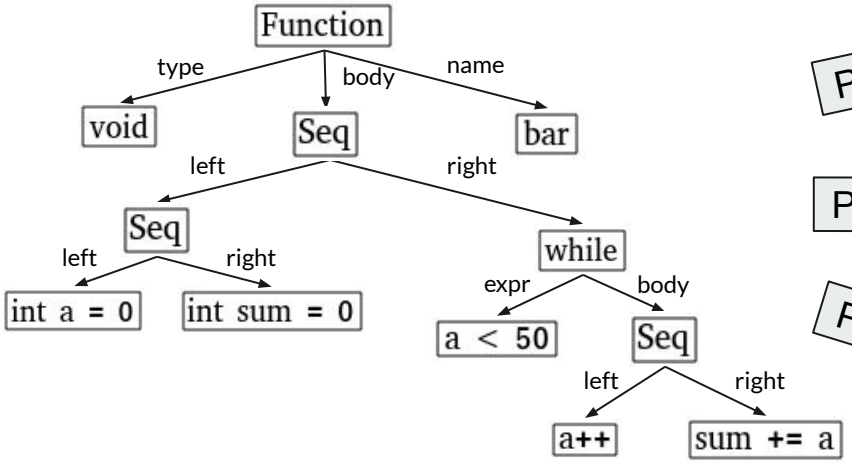
- синтаксическое дерево
- язык разметки
- таблицы

Pretty Printing

```
int main() {
    short n, m;
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        graph.push_back({ from: x - 1,  to: y - 1});
    }
    vector<int> res(LEN,  value: 0);
    vector<vector<int> > mas( n: V + 1);
    for (int i = 1; i < LEN; i++) {
        for (int j = 0; j < LEN; j++) {
            if (!isIndependent(i))
                continue;
            int c = 0;
            for (int k = 0; k < V; k++) {
                if (i == 0)
                    c++;
            }
            mas[c].push_back(i);
        }
    }
}
```

Инструмент: Pretty Printer

# Варианты форматирования




Pretty Printer

Pretty Printer

Pretty Printer

```
void bar() { int a = 0; int sum = 0;
    while (a < 50) { a++; sum += a; } }
```

Неформатированный код

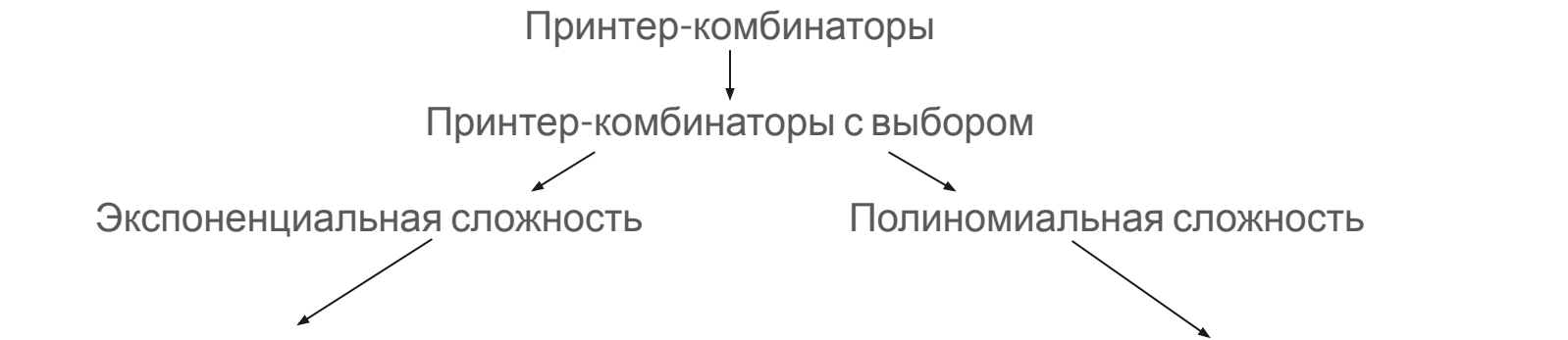
```
void bar() {
    int a = 0; int sum = 0;
    while (a < 50) { a++; sum += a; }
}
```

Форматированный без учета ширины

```
void bar() {
    int a = 0;
    int sum = 0;
    while (a < 50) {
        a++;
        sum += a;
    }
}
```

Форматированный с учетом ширины

# Принтер-комбинаторы с выбором




- Пабло Азеро, Дойце Свирстра

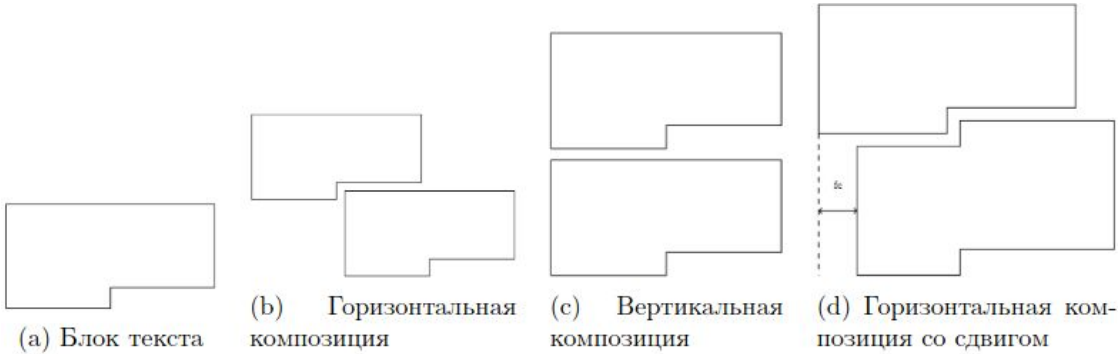
- Фильтрация по Парето (*Жан-Филипп Бернарди*)
- Метод BURS (*Антон Подкопаев*)

# Постановка задачи

- Исследовать работу библиотек принтер-комбинаторов с выбором
- Реализовать на Coq библиотеку Жана-Филиппа Бернарди и тривиальный алгоритм принтер-комбинаторов с выбором
- Исследовать свойства библиотек и определить требования, накладываемые на принтер
- Механизировать доказательство корректности библиотеки в Coq
- Предоставить верифицированную версию библиотеки на языке Haskell

# Основные составляющие принтер-комбинаторной библиотеки

```
Inductive Doc : Type :=
  | Text   (s: string)
  | Indent (t: nat) (d: Doc)
  | Beside (d: Doc) (d: Doc)
  | Above  (d: Doc) (d: Doc)
  | Fill   (d: Doc) (d: Doc) (s: nat)
  | Choice (d: Doc) (d: Doc).
```



(a) Блок текста (b) Горизонтальная композиция (c) Вертикальная композиция (d) Горизонтальная композиция со сдвигом

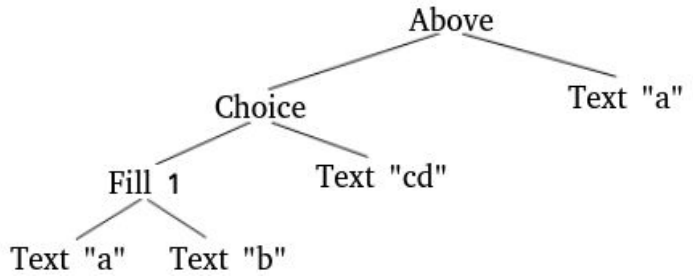
Работа с блоками текста *Format*

Processor: $\mathrm{Doc} \to [Format]$

# Фильтрация по Парето

Оптимальное форматирование как задача поиска множества Парето.

- Работа на списках
- Удаление больших раскладок
- Частичный порядок $a \preccurlyeq b$
- Множество Парето:
  $\{x \in X \mid \nexists y \in X\ x \neq y \wedge y \preccurlyeq x\}$




Узлы дерева представляют список из раскладок

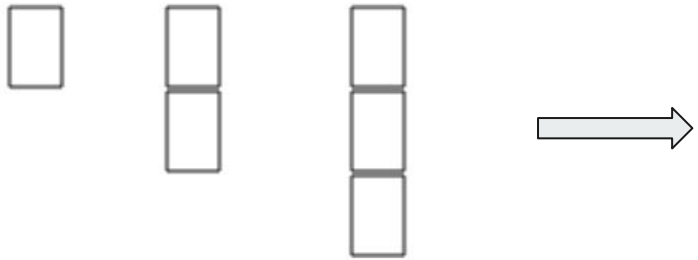
# Корректность библиотеки (1)

**Свойства библиотеки**:

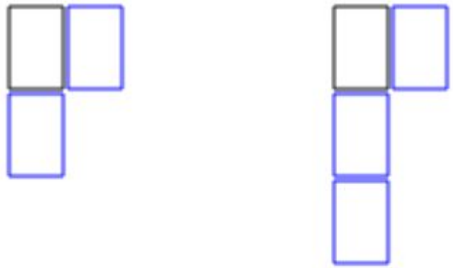
1. Корректность раскладок
2. Корректность комбинаторов

Из корректности комбинаторов следует теорема о контексте, гарантирующая, что удаление бóльших раскладок не повлияет на результат:

***Лемма*** (О контексте)**.** *Для любых* $a \preccurlyeq b$ *верно, что* $C[a] \preccurlyeq C[b]$.



Раскладки $c$, $a$ и $b$ соответственно



Результат $fill\ \ 0\ \ c\ \ a$ и $fill\ \ 0\ \ c\ \ b$ соответственно

# Корректность библиотеки (2)

1. Корректность

$$evaluatorTrivial : \mathbb{N}_0 \to Doc \to list\ \ Format$$

$$evaluatorPareto : \mathbb{N}_0 \to Doc \to list\ \ Format$$

***Теорема .*** *Для любых* $width \in \mathbb{N}_0$ *и* $Doc$ *верно, что* $evaluatorPareto\ \ width\ \ doc\ \subseteq\ evaluatorTrivial\ \ width\ \ doc.$

2. Результат не хуже

***Теорема .*** *Для любых* $width \in \mathbb{N}_0$ *и* $Doc$ *выполняется*

$$pretty\_list\ \ evaluatorPareto\ \ width\ \ doc \subseteq pretty\_list\ \ evaluatorTrivial\ \ width\ \ doc$$

# Экстракция кода

Для получения верифицированной библиотеки на языке Haskell использовался встроенный механизм экстракции кода в Coq.

- Преобразование в стандартные типы языка Haskell, посредством переопределения конструкторов
- Сборка проекта в stack

# Результаты

- Реализованы алгоритмы принтеров тривиальной реализации и с фильтрацией по Парето (Coq)
- Формализованы понятия корректности принтеров и раскладок (Coq)
- Поддержан комбинатор fill
- Выполнена механизация доказательств корректности в Coq
- Выполнена экстракция кода библиотеки с языка Coq на язык Haskell

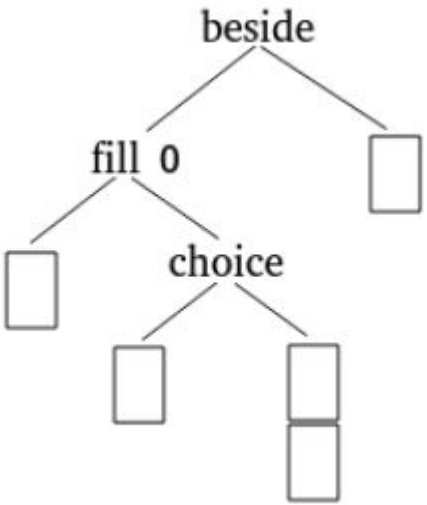
**Дальнейшие планы:**

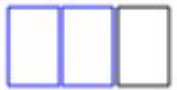
1. Доказать корректность библиотеки на основе BURS
2. Доказать корректность императивной реализации на C с помощью логики Хоара

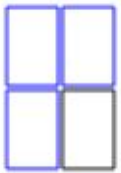
# APPENDIX

# Частичный порядок

$$a \preccurlyeq b \overset{\text{def}}{\Leftrightarrow} \begin{array}{l} height(a) \leqslant height(b) \ \wedge \\ first\_line\_width(a) \leqslant first\_line\_width(b) \ \wedge \\ middle\_width(a) \leqslant middle\_width(b) \ \wedge \\ last\_line\_width(a) \leqslant last\_line\_width(b) \end{array}$$





$$a \preccurlyeq_m b = \begin{cases} a \preccurlyeq b, \ height(a) = 1 \wedge height(b) = 1 \\ a \preccurlyeq b, \ height(a) > 1 \wedge height(b) > 1 \end{cases}$$

# Корректность раскладок

- В случае, если раскладка имеет высоту 1, то все характеристики высоты считаем равными
- Если же высота 2, то считаем middle_width = first_line_width
- Высота раскладки 0 недопустима

# Корректность комбинаторов

1) Сохраняется корректность раскладки: если раскладки $a$ и $b$ корректны, то раскладка $\underline{f}\ \ a\ \ b$ также корректна.

2) Сохраняется ширина раскладки: для любых корректных раскладок $a$ и $b$ из того, что $width(\underline{f}\ \ a\ \ b) \leq w$ следует $width(a) \leq w$ и $width(b) \leq w$.

3) Для любой четверки корректных раскладок $a, b, c, d$, если $a \preccurlyeq'_m b$ и $c \preccurlyeq'_m d$, тогда верно $(\underline{f}\ \ a\ \ c) \preccurlyeq'_m (\underline{f}\ \ b\ \ d)$.

$$a \preccurlyeq'_m b = \begin{cases} a \preccurlyeq' b, \ height(a) = 2 \\ a \preccurlyeq_m b, \ height(a) \neq 2 \end{cases}$$

# Контрпример

Пример того, когда равенство списков из двух реализаций нарушается.