1 Добрый день, меня зовут Ступников Александр, и я представляю свою бакалаврскую работу под названием ”Разработка библиотеки комбинаторных парсеров высшего порядка нечувствительных к левой рекурсии”.

2 Использующиеся сегодня подходы к синтаксическому анализу можно разделить на две большие категории: генераторы парсеров и комбинаторные парсеры. Как правило, эти подходы отличаются тем, что генераторы парсеров работают быстрее, но на меньшем классе грамматик, чем комбинаторные парсеры. Однако более важным отличием является тот факт, что при синтаксическом анализе при помощи генераторов парсеров пользователь должен описывать грамматику разбираемого языка с использованием некоторого предметно-ориентированного языка. Комбинаторные же парсеры являются обычными функциями, которые полностью написаны на языка интрумента, который используется для синтаксического анализа, то есть, как правило, на языке компилятора. Это значительно облегчает использование комбинаторных парсеров, позволяя с их помощью легко описывать сложные синтаксис и семантику языка.

3 В работе речь будет идти именно о комбинаторных парсеров. Их можно поделить на два вида: аппликативные и монадические. В целом монадические парсеры позволяют описывать не только контекстно-свободные грамматики. Однако это приводит к тому, что структуру монадических парсеров невозможно анализировать статически, совершая всяческие оптимизации, что нельзя сказать об аппликативных парсерах. В рамках ВКР разработан именно монадический парсер.

4 Важным понятием в рамках ВКР является левая рекурсия в грамматиках. Говорят, что грамматика леворекурсивна, если для разобра некоторого нетерминала в ней необходимо разобрать тот же нетерминал без изменения входного потока. Справа на слайде вы можете увидеть пример леворекурсивной грамматики, состоящей из единственного нетерминала E. Большинство реализаций комбинаторных парсеров не поддерживают использование леворекурсивных грамматик. Это усложняет описание грамматик с помощью таких парсеров, ведь синтаксис и семантику многих формальных языков удобнее и естественнее записывать с помощью леворекурсивных грамматик. Кроме того левая рекурсия может возникнуть при использовании комбинаторных парсеров высшего порядка.

Парсер высшего порядка seq в качестве аргументов принимает два парсера, которые пытается последовательно применить, разобрав между ними символ точки с запятой. Сам по себе парсер seq нельзя назвать леворекурсивным, однако его неаккуратное использование может привести к левой рекурсии, как это происходит в парсере stmt. Грамматика, описанная stmt является леворекурсивной и, как следствие, не может быть разобрана большинством существующих реализаций комбинаторных парсеров.

5 Сегодня появляется на свет всё больше различных языков программирования и предметно-ориентированных языков. Зачастую грамматика таких языков схожа. Вместе с тем из-за невозможности разбирать леворекурсивные грамматики, а также из-за других тонкостей, на текущий момент не существует инструмента синтаксического анализа, позволяющего писать абстрактные парсеры, удобным образом композировать грамматики, то есть подставлять их друг в друга, объединять и т.д., как это делает, например, комбинатор seq. Цель моей работы исправить это положение вещей, тем самым облегчив создание инструментов синтаксического анализа. Задачей, которая поставлена в рамках ВКР, является создание бибилиотеки монадических комбинаторных парсеров высшего порядка.

6 На текущий момент существует несколько решений, которые потенциально подходят для решения поставленной задачи. Все они имеют свои недостатки. В итоге для достижения задачи ВКР было принято решение взять за основу парсеры в стиле передачи продолжения или, как их ещё называют, CPS парсеры, дополнив и адаптировав их под свои нужды.

7 На слайде представлена архитектура полученного решения. Парсер абстрагирован от деталей реализации потока ввода, а также поддерживает использование регулярных выражений. Парсер использует таблицу мемоизации, чтобы хранить там промежуточные результаты.

8 Теперь пара слов о полученном в работе алгоритме. Вообще любой парсер это функция из состояния в изменённое состояние и результат. Разработанный парсер в CPS стиле помимо состояния принимает также продолжение. Продолжение по сути является функцией, которая на основе результатов разбора одного парсера возвращает другой парсер. Таким образом парсеры можно последовательно соединять между собой с помощью монадического комбина-

тора bind, который вызывается на результате первого парсера, переданного в bind.

Парсеры в CPS стиле позволяют вести учёт того, что вызывается непосредственно после некоторого парсера, то есть его продолжение. Чтобы хранить такие продолжения, а также результаты разбора парсера, используется хэш таблица, в которой по ключу, генерирующемся уникальным образом для каждого мемоизированного парсера, хранится список продолжений, а также список результатов этого парсера. При добавлении продолжения в список продолжений парсера, необходимо вызвать это продолжение на всех имеющихся результатах парсера. При добавлении результата в список результатов парсера, необходимо вызвать все продолжения парсера на этом результате. За счёт подобного алгоритма достигается полиномиальное время работы парсера, а также поддерживается левая рекурсия.

9 Итак, в рамках работы был написан монадический комбинаторный парсер, который способен распознавать любые однозначные контекстно-свободные грамматики за кубическое время, нечувствителен к левой рекурсии, а также поддерживает парсеры высшего порядка. Более того парсер позволяет пользователю самому задавать семантику. При этом следует отметить, что на многих типичных грамматиках, парсер работает за линейной время. Это следует из схожести использованного алгоритма с алгоритмом Эрли. На графике справа указано время разбора арифметических выражений с длиной от полутора до двухсот тысяч символов. Можно заметить, что оно линейно. Все эти свойства позволяют думать о парсерах, как о грамматике, композируя их произвольным образом.

10 В таблице собраны некоторые аналоги разработанного решения, которые существуют на рынке. Они сравниваются с моим решением. Можно заметить, что из существующих комбинаторных парсеров никто не поддерживает левую рекурсию, пользовательскую семантику и при этом разбирает любую контекстно-свободную грамматику.

11 В итоге была создана библиотека комбинаторных парсеров на Haskell, были выполнены все задачи ВКР и создан инструмент, лишённый некоторых недостатков существующих реализаций парсеров.