

Формальные языки

домашнее задание до 23:59 15.10

1. Релизовать синтаксический анализатор с использованием библиотеки парсер-комбинаторов семейства `parsec` для упрощенного подмножества пролога.

- Описание синтаксиса

- Программа на прологе начинается с *объявления модуля*, за которым идет последовательность *определений типов*, за которым идет последовательность определений отношений *определений отношений*.
- Определение отношений состоит из *головы* и *тела*, разделенных штопором `(:-)`, в конце стоит точка `(.)`.
- Голова — атом.
- Тело — выражение с правоассоциативными бинарными операторами конъюнкции `(,)` и дизъюнкции `(;)` над атомами. Конъюнкция имеет более высокий приоритет, чем дизъюнкция. Возможно использование скобок `((,))` для управления порядком вычислений.
- Атом — идентификатор, за которым идет последовательность атомов, разделенных пробельными символами. Любой атом в этой последовательности может быть взят в произвольное количество скобок `((,))`.
- Тело и штопор могут отсутствовать.
- Где угодно могут встречаться пробельные символы: советую использовать лексер.

- Новый синтаксис

- Объявление модуля это ключевое слово `module`, за которым идет идентификатор (не переменная), завершается все это точкой `.: module example.`
- Определение типа это строка, начинающаяся с ключевого слова `type`, за которым идет идентификатор-имя типа, за которым идет тип, в конце — точка `..`. Тип это последовательность атомов или типов, разделенных стрелкой `->`.
 - * Корректные определения типов
 - `type filter (A -> o) -> list A -> list A -> o.`
 - `type fruit string -> o.`
 - * Некорректные определения типов
 - `type type type -> type.` — ключевое слово используется как идентификатор
 - `type x -> y -> z.` — нет имени типа
 - `type x o.` — идентификатор вместо ключевого слова `type`
- Идентификатор, который начинается с большой буквы — переменная, которая не может быть первым идентификатором в атоме, но может встречаться в аргументах.
 - * Корректные переменные: `X`, `XyZ`, `ABC`.

- * Некорректные переменные: `abc`, `123`, `X Y`.
- Синтаксический сахар для списков:
 - * Список это возможно пустая последовательность атомов или переменных, разделенных запятой (`,`), находящаяся в квадратных скобках (`[`, `]`).
 - Корректные списки: `[]`, `[X, Y, Z]`, `[a (b c), d, Z]`.
 - Некорректные списки: `[,]a, b, c[`
 - * Можно специфицировать список, состоящий из головы `H` и хвоста `T`: `[H|T]`. В голове может быть не только переменная, но и произвольный атом, хвост — обязательно переменная.
 - Корректные списки: `[H | T]`, `[a (b c) | T]`
 - Некорректные списки: `[H | abc]`, `[H | A b c]`
 - * Список может вкладываться в другой список: `[[X, [H | T]] | Z]`
 - * Список не может быть головой атома, но может быть аргументом.
 - Некорректное использование списка: `[X] Y :- f X Y`.
 - Корректное использование списка: `g [X] Y :- f X Y`.
- Примеры корректных определений отношений.
 - `f`.
 - `f :- g`.
 - `f :- g, h; t`.
 - `f :- g, (h; t)`.
 - `f a :- g, h (t c d)`.
 - `f (cons h t) :- g h, f t`.
- Примеры некорректных определений отношений.
 - `f` — нет точки.
 - `:- f`. — нет головы.
 - `f :-` . — нет тела.
 - `f :- g; h, .` — нет правого подвыражения у конъюнкции.
 - `f :- (g; (f))`. — несбалансированные скобки.
 - `f ()`. — пустые скобки
- Результатом должно быть консольное приложение, которое принимает на вход программу и печатает результат синтаксического анализа в файл с таким же названием и дополнительным расширением `.out`.
- Результатом синтаксического анализа является абстрактное синтаксическое дерево в случае успешного разбора и сообщение об ошибке иначе.
- Код должен быть сопровождается инструкцией по сборке и запуску. Желательно выложить его на гитхаб и сопровождать тестами.
- Пример парсера с пары выложен на гитхаб: <https://github.com/kajigor/fl-2020-hse-win/tree/3d6cae8eb08b862679de5e199e6ba4153d295d7e>