Проект по Формальным языкам. Многострочные комментарии.

Юра Худяков

22 октября 2020 г.

Я решил поддержать возможность многострочных комментариях в парсерах пролога.

Синтаксис многострочных комментариев: $(* \; text \; *)$

При этом необходим контроль вложенности: $text$ может быть любым, и также содержать вложенные многострочные комментарии, при условии, что будет соблюдаться парность скобок.

Например, $(* \; (* \; hello \; world \; *) \; *)$ является корректным комментарием, а

$$
\begin{aligned}
&(* \; (* \; hello \; world \; *) \\
&(* \; hello \; world \; *) \; *) \\
&(* \; hello \; world \\
&*)
\end{aligned}
$$

не являются (на каждой строке отдельный комментарий)

Несмотря на то, что в примерах выше рассматриваются однострочные варианты комментариев, многострочность поддерживается (на этот раз, это единый комментарий):

$$
\begin{aligned}
&(* \\
&hello \\
&(* \; (* \; ooo \; *) \; *) \\
&(* \\
&\\
&\\
&text \; *) \\
&world \\
&*)
\end{aligned}
$$

Я реализовал поддержку этого синтаксиса в 3 парсерах

1. Парсер, основанный на рекурсивном спуске

   В этом парсере используется лексер Alex, поэтому поддержка реализована следующим образом:

   (a) Лексер разбивает строку на токены, и среди них есть токен открывающей комментирующей скобки ($*$, закрывающей комментирующей скобки $*$), и токен ошибки, который принимает любой символ

   (b) При запуске парсера проводится удаление всех комментариев отдельной функцией removeComments, которая поддерживает количество открытых на данный момент комментирующих скобок. Если количество открытых скобок положительно, то всё, что между ними, опускается, а иначе сохраняется. Функция возвращает Maybe список токенов без многострочных комментариев.

   Если при удалении комментариев обнаруживаются непарные комментирующие скобки, возвращается Nothing.

   (c) Если вернулось Nothing, то выводится сообщение об ошибке

   Иначе токены без многострочных комментариев передаются дальше, где сначала ищутся оставшиеся ошибки, а затем запускается сам парсер.

2. Парсер, основанный на yacc.

   В нём также используется лексер Alex, поэтому реализация такая же, как в пункте 1.

3. Парсер, основанный на Парсер-Комбинаторах.

   Имеет встроенную поддержку многострочных комментариев с использованием Text.Parsec.Language