Задание 11
Атрибутные грамматики

Следующий за синтаксическим анализом этап в процессе компиляции является генерация кода. В основе этого этапа лежат вычисления по дереву разбора, которые описывают с помощью атрибутных грамматик. Мы не будем детально изучать эту тему, а изучим лишь частный случай атрибутных грамматик (с синтезируемыми атрибутами).

Определение 1. КС-грамматика G называется *атрибутной с ситнезируемыми атрибутами*, если каждому нетерминалу поставлен в соответствуие набор переменных (атрибутов), и при этом для каждого правила грамматики

$$X_0 \to u_0 X_1 u_1 X_2 \dots u_{n-1} X_n u_n, X_i \in N, u_i \in \Sigma^*$$

Задан набор правил вычисления некоторых атрибутов

$$X_0[\mathsf{attr}] = f(X_1[\mathsf{attr}_1],\, X_2[\mathsf{attr}_2], \dots,\, X_n[\mathsf{attr}_n]);$$

здесь $X_i[\mathsf{attr}_i]$ – значение атрибута attr_i для нетерминала X_i , а f – некоторая функция. Набор правил вычислений атрибутов называют *атрибутной схемой*.

Пример 1. Грамматика G задана правилами

$$S \to 1D \mid 0,\; D \to 1D \mid 0D \mid 1 \mid 0$$

и порождает язык двоичных записей натуральных чисел. Определим атрибутную схему для этой грамматики

$S \to 0$	$D \to 1$	$D \to 0$	$S \to 1D$
$S[\mathsf{val}] = 0$	$D[\mathsf{val}] = 1$	$D[\mathsf{val}] = 0$	$S[\mathsf{val}] = D[\mathsf{ord}] + D[\mathsf{val}]$
	$D[\mathsf{ord}] = 2$	$D[\mathsf{ord}] = 2$	

$D_0 \to 1D_1$	$D_0 \to 0D_1$
$D_0[\mathsf{val}] = D_1[\mathsf{ord}] + D_1[\mathsf{val}]$	$D_0[\mathsf{val}] = D_1[\mathsf{val}]$
$D_0[\mathsf{ord}] = 2 \times D_1[\mathsf{ord}]$	$D_0[\mathsf{ord}] = 2 \times D_1[\mathsf{ord}]$

В случае, если правило содержит несколько одинаковых нетерминалов мы нумеруем их вхождение и различаем атрибуты как в случае двух последних правил. Нетерминал S имеет единственный атрибут val, а нетерминал D – атрибуты val и ord. Приведённая атрибутная схема вычисляет значение числа по его двоичной записи. Атрибут ord равен 2^l, где l – длина слова выведенного из D, атрибут val равен значению числа, двоичная запись которого выведена из нетерминала.

Приведём пример вычисление атрибутов для слова 1101. Атрибуты вычисляются снизу вверх.

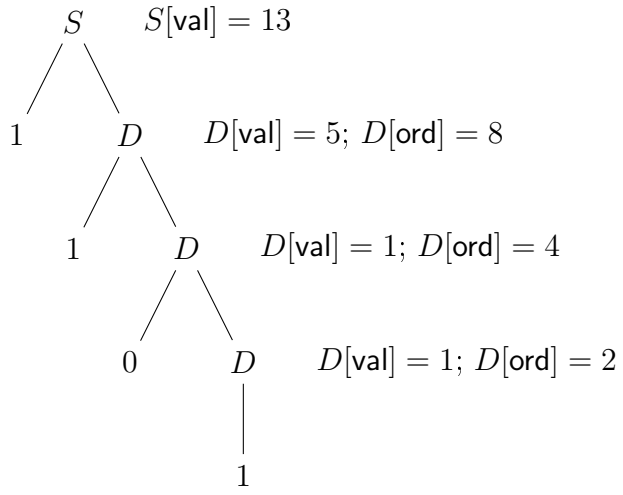

Рис. 1: вычисление атрибутов.

Задача 1. Грамматика $\mathsf{RExpr} = \langle\{E, T\}, \{a, b, +, ^*, (,)\}, P, E\rangle$ имеет множество правил P :

$$E \to T + E \mid TE \mid T \qquad T \to (E) \mid T^* \mid a \mid b$$

и порождает регулярные выражения над алфавитом $\{a, b\}$.

1. Постройте для грамматики RExpr LR(1)-анализатор[1].

2. Дополните грамматику RExpr до атрибутной так, чтобы она вычисляла атрибуты firstpos, lastpos и nullable согласно алгоритму их вычисления при построении ДКА по РВ. Считайте атрибуты firstpos, lastpos и nullable у терминалов заданными: на практике перед вычислением атрибутов происходит препроцессинг, во время которого могут быть заданы атрибуты терминалов, если это требуется.

Указание: Вычисление атрибута нужно определять через описание функции, которую можно описать на языке программирования или псевдокоде. Пример вычисления атрибута nullable для правила $E_0 \to T + E_1$:

```
E0[nullable] = function( T[nullable], E1[nullable]){
    if( T[nullable] or E1[nullable]){
        return True;
    } else{ return False; }
}
```

3*. Добавьте в атрибутную схему вычисление атрибута followpos.

4. С помощью анализатора постройте дерево разбора для РВ $(a+ab)^*+ab^*$ и вычислите атрибуты firstpos, lastpos, nullable (*followpos) согласно атрибутной схеме (предварительно задав атрибуты у терминалов).

[1] Анализатор для этой грамматики довольно громоздкий. Постройте его с помощью программы, например на сайте http://lrk.umeta.ru, используйте программу для выполнения последующих пунктов.

Язык разметки web-страниц HTML был разработан так, что бы его код было легко разбирать интерпретатором. Код на HTML представляет собой правильное скобочное выражение, в котором скобки имеют имена и называются тегами: <tag> ... </tag>. Внутри открывающей скобки <tag> могут быть атрибуты; в общем случае (открывающий) тег имеет вид <name attr1="value" attr2="value" ... attrN=" value">, т. е. сначала идёт имя тега, а потом перечисляются атрбиуты.

При интерпретации HTML-кода правильная скобочная последовательность тегов интерпретируется как дерево: если один тег вложен в другой, то внешний тег — родитель внутреннего. Так, в случае кода <tag1> ... <t2>....</t2> <t3>....</t3>...</tag1
тег tag1 является родителем тегов t2 и t3.

Рассмотрим тег , который является контейнером для разделения содержимого страницы. Будем рассматривать два атрибута: align и style. В случае вложенных тегов, атрибут align либо задан, например , либо наследуется от родителя. Атрибут style позволяет задавать (CSS) стиль элемента и устроен более сложно, чем align. Мы сосредоточимся только на задании с помощью атрибута style цвета фона у контейнера . В случае вложенных , если цвет фона не задан, то он наследуется у родителей.

Задача 2. Постройте по коду на рис2 дерево html документа. Определите значение атрибута align у каждого из узлов div дерева, а также определите цвет фона элемента. Проверьте себя, сохранив текст ниже в файле с расширением .html и открыв файл в браузере.

Комментарий. В HTML-документе код документа окружается тегом <html>, внутри тега <head> находятся вспомогательные данные (такие как заголовки), а содержимое документа находится в теге <body>. Внутри тега <style> описывается стиль элементов документа, в нашем коде там указан базовый стиль для тегов : наличие границы, отступы и размер в процентах относительно размера тега-родителя. Браузер интерпретирует документ HTML как дерево, точнее модель документа называется DOM (Document Object Model).

```
<html>
<head>
  <style>
   div{border: 1px solid  black; padding:1px;
        margin: 1px; width:40%; height:40%;}
  </style>
</head>
<body>
<div style="background-color:lightblue; width:500px;
      height:500px;" align="center">1
  <div style="background-color:blue;" align="left">
    2
    <div align="right">
      3
      <div style="background-color:gray;" align="center">
        4
      </div>
    </div>
    <div>
      5
    </div>
  </div>
  <div>
    6
  </div>
</div>
</body>
</html>
```

Рис. 2: HTML-код для задачи 2

Задача 3*. Докажите, что любую LL(1)-грамматику можно превратить в PEG тривиальной заменой $|$ на $/$ в любом порядке и $\to$ на $\leftarrow$