Список вопросов

1- 45 вопросы надо выбрать правильный ответ и дать обоснование ответа письменно

1. Синтаксис языков описывается:

(стр. 4)

2. Грамматика языка реализуется:

(стр. 5 начало)

3. G = (T, V, P, S0),

(стр. 8 опред. 1)

4. Языком, порождаемым грамматикой G = (T, V, P, S0), называется

(стр. 9 опред. 2)

5. α = defgabck : | α | = ?

(считаешь длину цепочки, состоящей из терминальных символов)

6. Грамматика: S→0|0A A→1B B→0|0A

(стр. 10 Классификация Хомского. Регулярная грамматика)

7. Язык: {0(01)n | n³0}

8. Распознаватель: {0(01)n | n³0}

(См. таб. Хомского. КА - для регулярного языка)

9. КА = (Q, Σ, σ, q_0, F),

(стр. 12)

10. Лемма о накачке доказывает

(стр.15 Регулярность или нерегулярность языка)

11. КА задается

(Диаграммой и таблицей см. стр. 14-15)

12. Конфигурация КА

(стр. 13 опр. 5)

13. L(КА) распознаваем КА

Языком L(КА), распознаваемым КА называется множество входных цепочек, $L(КА) = \{\omega \in \Sigma^* \mid (q_0,\omega) \vdash^k (q, \varepsilon)$, где $q \in F\}$

14. Правила КС-грамматики имеют вид

$A \to \alpha$, где α- цепочка(последовательность) символов(терминальных и нетерминальных).

15. Бесполезные символы КС-грамматики 1. устраняются 2. Недостижимы

Недостижимые, не порождающие, цепные

16. G = (T, V, P, S), в нормальной форме Грейбах, если

Определение 7. КС грамматика G = (T, V, P, S) называется грамматикой в нормальной форме Грейбах, если в ней нет ε-правил, т.е. правил вида $A \to \varepsilon$, и каждое правило из P отличное от $S \to \varepsilon$, имеет вид $A \to a\alpha$, где $a \in T$, $\alpha \subset V^*$.

Также полезно представлять грамматику в нормальной форме Хомского, что позволяет упростить рассмотрение ее свойств.

(стр. 32)

17. Нормальная форма Хомского позволяет упростить рассмотрение свойств:

Не будет цепных правил, левой рекурсии, длинных правил, ε-правил.

18. G = (T, V, P, S) в нормальной форме Хомского, если каждое правило из P имеет один из следующих видов:

Определение 8. КС грамматика G = (T, V, P, S) называется грамматикой в нормальной форме Хомского, если каждое правло из P имеет один из следующих видов:

1. $A \to BC$, где $A, B, C \in V$;
2. $A \to a$, где $a \in T$;
3. $S \to \varepsilon$, если $\varepsilon \in L(G)$, причем S не встречается в правых частях правил.

(стр. 32)

19. Нетерминал КС-грамматики *рекурсивен*

Определение. Нетерминал КС-грамматики называется *рекурсивным*, если $A \Rightarrow^+ \alpha A \beta$, для некоторых α и β. Если $\alpha = \varepsilon$, то A называется *леворекурсивным*, если $\beta = \varepsilon$, то A называется *праворекурсивным*. Грамматика, имеющая хотя бы один леворекурсивный нетерминал, называется *леворекурсивной*. Грамматика, имеющая хотя бы один праворекурсивный, нетерминал называется *праворекурсивной*.

(стр. 32)

20. МП = (Q, å, Г, d, q0, z0, F)

21.Конфигурация МП-автомата:

22. Детерминированным МП-автомат

23. 1. FIRST(A) =

24. Конфигурация “1-предсказывающего” алгоритма разбора имеет вид (x, Xα^, π), где x – неиспользуемая часть входной цепочки, Xα – цепочка в магазине

25. Управляющая таблица применяется для распознавания

26. Для LR(k) языка можно построить анализатор

27. В LR(k) разборе применяются операции

28. goto задается

29. Для грамматики предшествования строится

30. LR(k)-алгоритм разбора описывается

31. G = (T, V, P, S) - грамматика типа 0 если

32. Конфигурация машины Тьюринга

33. G = (T, V, P, S) – *неукорачивающаяся грамматика*, если каждое правило из P имеет вид α → β, где

34. G = (T, V, P, S) - *контекстно-зависимая (КЗ)*, если каждое правило из P имеет вид α → β, где

35. *Грамматику типа 1* можно определить как

36. Любая регулярная грамматика является

37. Любая КС-грамматика является грамматикой типа

38. Любая КС-грамматика является грамматикой типа

39. Любая КЗ-грамматика является грамматикой типа

40. Любая неукорачивающая грамматика является грамматикой типа

41. Каждый регулярный язык является

42. Каждый КС-язык является

43. Каждый КЗ-язык является языком типа

44. Определите понятие семантики языков программирования

45. Простейшие переводы описываются:

46- 83 дать определение или привести пример письменно

46. Определите синтаксически управляемый перевод и входную и выходную цепочки СУ-схемы

47.Определите L-атрибутную и S-атрибутную транслирующие грамматики

48. Определите копирующее правило вычисления атрибутов

49. В каком случае АТ-грамматика имеет форму простого присваивания?

50. Расширить восходящий анализатор действиями по выполнению перевода.

51. Представление в магазине и вычисление унаследованных и синтезированных атрибутов символов грамматики в S-Атрибутном ДМП-процессоре.

52. Случай АТ-грамматика с формой простого присваивания.

53. Приведите процедуру преобразования произвольной АТ-грамматики в форму простого присваивания. - этого нет в методичке походу

54. Преобразование деревьев под управлением СУ-схем

55. Приведите алгоритм построения управляющей таблицы для транслирующей грамматики цепочного перевода, входной грамматикой которого является LL(1)-грамматика.

56. Определите транслирующую грамматику.

57. Опишите, каким образом представляются в магазине и вычисляются унаследованные и синтезированные атрибуты символов грамматики в L-атрибутном ДМП-процессоре.

58. Какие СУ-схемы называются простыми.

59. Приведите процедуру преобразования ДМП-процессора в L-атрибутный ДМП-процессор.

с.132 его книжки

60. Определите активную цепочку. Ее входную и операционную части.…

61. Как программируются процедуры в методе рекурсивного спуска, реализующего перевод, описываемый L-атрибутной транслирующей грамматикой.

62. Определите входную и выходную грамматики транслирующей грамматики

63. Как программируются процедуры в методе рекурсивного спуска, реализующие перевод, описываемый транслирующей грамматикой.

64. Как граф зависимостей используется при вычислении атрибутов

65. С какой целью осуществляется переименование имен атрбутов нетерминальных символов из левых частей правил вывода L-атрибутной транслирующей грамматики при реализации вывода L-атрибутного перевода методом рекурсивного спуска.

66. Как связаны вывод цепочки и дерево вывода активной цепочки

67. С какой целью осуществляется переименование имен атрбутов нетерминальных символов из левых частей правил вывода L-атрибутной транслирующей грамматики при реализации вывода L-атрибутного перевода методом рекурсивного спуска

68. Определите атрибутную транслирующую грамматику.

69. Опишите, каким образом представляются в магазине и вычисляются унаследованные и синтезированные атрибуты символов грамматики в L-атрибутном ДМП-процессоре

70. Какие атрибуты называются унаследованными, синтезированными.

71. С какой целью осуществляется переименование имен атрбутов нетерминальных символов из левых частей правил вывода L-атрибутной транслирующей грамматики при реализации вывода L-атрибутного перевода методом рекурсивного спуска.

72. Определите различия между унаследованными и синтезированными атрибутами

73. Определите представление в магазине и вычисление унаследованных и синтезированных атрибутов символов грамматики в S-Атрибутном ДМП-процессоре.

74. Определите правило вычисления атрибутов.

75. Приведите алгоритм построения управляющей таблицы для транслирующей грамматики цепочного перевода, входной грамматикой которого является LL(1)-грамматика

76. Как строится граф зависимостей.

77. Случай АТ-грамматика с формой простого присваивания.

78. Типы ошибок при тестировании АТ-грамматики.

79. Приведите процедуру преобразования ДМП-процессора в L-атрибутный ДМП-процессор.

80. Этапы построения транслирующей грамматики

81. Определите представление в магазине и вычисление унаследованных и синтезированных атрибутов символов грамматики в S-Атрибутном ДМП-процессоре.

82. Виды объектов включаемые в выходную цепочку при переводе.

83. Определите Latрибутную и S-атрибутную транслирующие грамматики.

Ответы

46.Определите синтаксически управляемый перевод и входную и выходную цепочки СУ-схемы

Схема синтаксически управляемого (СУ-схема) перевода представляет собой

систему, порождающую пары цепочек, принадлежавших переводу. Неформально схема синтаксически управляемого перевода представляет собой грамматику, в которой каждому правилу приписан элемент перевода. При порождении цепочки каждый раз, когда правило участвует в этом процессе, элемент перевода генерирует часть выходной цепочки, соответствующей части входной цепочки, порождённой этим правилом.

47.Определите L-атрибутную и S-атрибутную транслирующие грамматики

Определение: АТ-грамматика называется **L-атрибутной** тогда и только тогда, когда выполняются следующие условия:
1. Аргументами правила вычисления значения унаследованного атрибута символа из правой части правила вывода могут быть только унаследованные атрибуты символа из левой части и произвольные атрибуты символов из правой части, расположенные левее рассматриваемого символа.
2. Аргументами правила вычисления значения синтезированного атрибута символа из левой части правила вывода являются унаследованные атрибуты этого символа или произвольные атрибуты символов из правой части.
3. Аргументами правила вычисления значения синтезированного атрибута операционного символа могут быть только унаследованные атрибуты этого символа.

АТ-грамматика называется **S-атрибутной** тогда и только тогда, когда она является L-атрибутной и все атрибуты нетерминалов синтезированные.

48. Определите копирующее правило вычисления атрибутов

Определение - правило вычисления атрибутов называется **копирующим правилом** тогда и только тогда, когда левая часть правила — это атрибут или список атрибутов, а правая часть – константа или атрибут. Правая часть называется источником копирующего правила, а каждый атрибут из левой части – приемником копирующего правила.

Если источники нескольких копирующих правил совпадают, то их приемники можно объединить в одну левую часть. Например, правила z, w← a и x, y←z можно записать в виде: x, y, z, w← a , т. к. источнику второго правила z, согласно первому правилу, присваивается значение a. Аналогично x←y и a, b ← y можно записать как a, b, x ←y.

Определение - множество копирующих правил называется **независимым**, если источник каждого правила из этого множества не входит ни в одно из других правил множества. Если два копирующих правила независимы, их нельзя объединять

49. В каком случае АТ-грамматика имеет форму простого присваивания?

Определение - атрибутная транслирующая грамматика имеет форму **простого присваивания** тогда и только тогда, когда: атрибутов операционных символов копирующих правил независимо. Примером АТ-грамматики в форме простого присваивания является грамматика G4. **-- что это???**

50. Расширить восходящий анализатор действиями по выполнению перевода.

Для восходящего синтаксического анализатора последовательность операций переноса и свертки, выполняемых при обработке допустимых входных цепочек, можно описать с помощью транслирующей грамматики. Входной для этой транслирующей грамматики является грамматика, на основе которой построен анализатор. Для получения транслирующей грамматики в самую крайнюю правую позицию каждого i-го правила входной грамматики вставляется операционный символ {СВЕРТКА, i}.

Восходящий анализатор можно **расширить действиями по выполнению перевода**, если перевод определяется постфиксной транслирующей грамматикой. Модификация анализатора заключается в том, что операция свертки расширяется действиями, определяемыми операционными символами соответствующего правила грамматики. Это можно сделать, т. к. при обработке входной цепочки момент времени выполнения свертки для каждого правила грамматики совпадает с моментом выполнения действий по переводу для этого правила. Например, для постфиксной транслирующей грамматики цепочечного перевода:

E → E+ T{+}
E → T
T → T*P {*}
T → P
P → i{i}
P → (E) операции свертки расширяются следующим образом: СВЕРТКА_1 будет обеспечивать выдачу операционного символа {+} в выходную строку, СВЕРТКА З — выдачу операционного символа {*}, а СВЕРТКА 6 — выдачу операционного символа {i}.

Синтаксический анализатор, дополненный формальными действиями по выполнению перевода, принято называть **восходящим ДМП-процессором**.

51. Представление в магазине и вычисление унаследованных и синтезированных атрибутов символов грамматики в S-Атрибутном ДМП-процессоре.

В S-атрибутном ДМП-процессоре каждый магазинный символ имеет конечное множество полей для представления атрибутов. Так же, как и в L-атрибутном ДМП-процессоре, примем, что магазинный символ с п атрибутами представляется в магазине (п + 1)-ой ячейками, верхняя из которых содержит имя символа, а остальные — поля для атрибутов.

Поля магазинного символа, предназначенные для атрибутов, заполняются значениями атрибутов в момент вталкивания символа в магазин и не изменяются до момента выталкивания его из магазина.

В S-атрибутном ДМП-процессоре операция переноса расширяется таким образом, что значения атрибутов переносимого входного символа помещаются в соответствующие поля вталкиваемого при переносе магазинного символа. При выполнении операции свертки для правила с номером i верхние символы магазина представляют собой правую часть i-го правила вывода входной грамматики, а поля магазинных символов содержат значения атрибутов соответствующих символов грамматики. Расширенная операция свертки использует эти значения для вычисления значений всех атрибутов операционных символов, связанных с правилом вывода транслирующей грамматики, и значений всех атрибутов нетерминала из левой части правила. Значения атрибутов операционных символов используются для выдачи результатов в выходную ленту или выполнения других действий, определяемых этими символами. Атрибуты нетерминала из левой части правила вывода записываются в соответствующие поля магазинного символа, который соответствует этому нетерминалу и вталкивается в магазин во время свертки

52. Случай АТ-грамматика с формой простого присваивания.

Определение - атрибутная транслирующая грамматика имеет **форму простого присваивания** тогда и только тогда, когда: атрибутов операционных символов копирующих правил независимо. Примером АТ-грамматики в форме простого присваивания является грамматика G4.

При определении **АТ-грамматики** в правила вычисления атрибутов записывались в виде операторов присваивания, левые части которых представляют собой атрибут или список атрибутов, а правые части — функцию, использующую в качестве аргументов значения некоторых атрибутов. Простейший случай функции из правой части оператора присваивания — тождественная или константная функция, например a ← b или x, y←1.

Определение - правило вычисления атрибутов называется копирующим правилом тогда и только тогда, когда левая часть правила — это атрибут или список атрибутов, а правая часть – константа или атрибут. Правая часть называется

источником копирующего правила, а каждый атрибут из левой части – приемником копирующего правила.

Если источники нескольких копирующих правил совпадают, то их приемники можно объединить в одну левую часть. Например, правила z, w← a и x, y←z можно записать в виде: x, y, z, w← a , т. к. источнику второго правила z, согласно первому правилу, присваивается значение a. Аналогично x←y и a, b ← y можно записать как a, b, x ←y.

Если в качестве источника копирующего правила используется константная функция, являющаяся процедурой-функцией без параметров (например, функция GETNEW из **АТ-грамматики**), то такие атрибутные правила объединять нельзя, т. к. два разных вызова функции без параметров могут давать разные значения.

53. Приведите процедуру преобразования произвольной АТ-грамматики в форму простого присваивания.

1. Для каждой функции $f(x_1, x_2,..., x_n)$, входящей в правило вычисления атрибутов, связанное с некоторым правилом вывода грамматики, создать соответствующий ей операционный символ $\{f\}$, который определяется следующим образом:

   $\{f\}_{x_1,x_2,...,x_n,p}$ УНАСЛЕДОВАННЫЕ $x_1, x_2,..., x_n$

   СИНТЕЗИРОВАННЫЙ p

   $p \leftarrow f(x_1, x_2,..., x_n)$

2. Для каждого некопирующего правила $z_1, z_2,..., z_b \leftarrow f(y_1, y_2,..., y_n)$ связанного с некоторым правилом вывода грамматики, найти символы $a_1,..., a_n$, которые не содержатся в этом правиле вывода, и вставить в его правую часть символ $\{f\}_{a_1,a_2,...,a_n,r}$. Заменить некопирующее правило на следующие (n + 1) копирующих правил:

   $a_i \leftarrow y_i$для каждого аргумента y_i

   $z_1, z_2,..., z_m \leftarrow r$

   При включении в правило вывода операционного символа необходимо соблюдать следующие условия:

   - операционный символ должен располагаться *правее* всех символов правой части правила вывода, атрибутами которых являются аргументы $y_1, y_2,..., y_n$;

- операционный символ должен располагаться левее всех символов правой части правила вывода, атрибутами которых служат $z_1, z_2, ..., z_m$;
- С учетом предыдущих ограничений операционный символ может быть вставлен в любое место правой части правила вывода, но предпочтение следует отдать самой левой из возможных позиций, т. к. это позволяет упростить реализацию синтаксического анализатора

3. Два копирующих правила, соответствующие одному и тому же правилу вывода, необходимо объединить, если источник одного из них входит в другое. Это достигается удалением правила с лишним источником и объединением его приемников с приемниками оставшегося правила.

Если в качестве источника копирующего правила используется константная функция, являющаяся процедурой-функцией без параметров (например, функция GETNEW из АТ-грамматики), то такие атрибутные правила объединять нельзя, т. к. два разных вызова функции без параметров могут давать разные значения. *(Пример на странице 442 книги, не методички)*

54. Преобразование деревьев под управлением СУ-схем

Преобразование деревьев при помощи СУ-схемы:

Описание алгоритма:

1.Пусть Данный шаг применяется к внутренней вершине n Дерева D, имею шей k прямых потомков $n_1, \dots, n_k$.

1.1 Устранить из множества вершин $n_1, \dots, n_k$ листья, помеченные терминальными символами или ε.

1.2 Пусть A → α – правило входной грамматики G_i, соответствую шее вершине n и ее прямым потомкам, т. е. A – метка вершины n, и α образуется конкатенацией меток вершин $n_1, \dots, n_k$. Выбрать из R некоторое правило вида A → α, β (если таких правил несколько, то выбор произволен).

Переставить оставшиеся прямые потомки вершины n (если они есть) согласно соответствию между вхождениями не терминалов в α и β. (Поддеревья, корнями которых служат эти потомки, переставляются вместе с ними).

1.3 Добавить в качестве прямых потомков вершины n листья с метками так, чтобы метки всех ее прямых потомков образовали цепочку β.

1.4 Применить Шаг 1 к прямым потомкам вершины n, не являвшимся листьями, в порядке слева направо.

2. Повторять Шаг 1 рекурсивно, начиная с корня дерева D.

Результирующим деревом будет D' .

*Если нужен пример алгоритма, то страница 94 в методичке, пример 5 3

55. Приведите алгоритм построения управляющей таблицы для транслирующей грамматики цепочного перевода, входной грамматикой которого является LL(1)-грамматика.

Алгоритм построения управляющей таблицы для транслирующей грамматики цепочечного перевода, входной грамматикой которой является LL(1)- грамматика Вход: Транслирующая грамматика цепочечного перевода GT = (N, ∑i , ∑a , P, S), входная грамматика которой является LL(1)- грамматика. Выход : Корректная управляющая таблица M для грамматики GT. Описание алгоритма: Управляющая таблица M определяется на множестве (N ∪ ∑i ∪ ∑a ∪ {⊥}) X ∑i ∪ {ε}) по следующим правилам: Если A→yβ - правило вывода грамматики GT, где y ∈ ∑a* , а y — либо ε, либо цепочка, начинающаяся с терминала или нетерминала, то M{A, a) = (β, y) для всех a≠ ε, принадлежащих множеству FIRST(β). Если ε ∈ FIRST(β), то M(A, b) = (β, y) для всех b ∈ FOLLOW (A). Заметим, что при вычислении FIRST(β) операционные символы, входящие в цепочку β, вычеркиваются. M(X, a) = ВЫДАЧА(X) для всех x ∈ ∑a И a ∈ ∑i ∪ {ε}. M(a, a) = ВЫБРОС для всех a ∈ ∑i . M (⊥ , ε) = допуск В остальных случаях M(X, a) = ОШИБКА для X ∈ (N ∪ ∑i ∪ ∑a ∪ {⊥}) и a ∈ ∑i ∪ { ε }. Построим управляющую таблицу для транслирующей грамматики, описывающей перевод инфиксных арифметических выражений в ПОЛИЗ. Эта транслирующая грамматика получена из входной LL(1)-грамматики G1 путем включения в нее операционных символов {i}, {+} и {*}: E →E ′(5)T′ →* P{*} T′ E ′ →+ T{+} E′6)T′ →ε E ′ →ε (7)P →i {i} T →PT′ (8)P →(E) Управляющая таблица должна содержать 14 строк, помеченных символами из множества N∪∑i ∪ ∑a ∪ {⊥}, и 6 столбцов, помеченных символами из множества ∑i ∪ { ε }. Построение управляющей таблицы для строк, отмеченных символами из множества N∪∑i ∪ {⊥}, ничем не отличается от построения таблицы для соответствующей входной LL(1)-грамматики (табл. 1.1), а строки управляющей таблицы, отмеченные операционными символами, содержат значения ВЫДАЧА{{X}), где X ∈{{i}, {+},{*}}. Заметим, что элементы таблицы, соответствующие строкам, помеченным операционными символами, для всех столбцов одинаковые, т. к. действия, выполняемые ДМП-процессором в случае, когда верхним символом магазина является операционный символ, не зависят от входного

символа.

Таблица 1.1. Управляющая таблица М для транслирующей грамматики, описывающей перевод инфиксных арифметических выражений в ПОЛИЗ

	i	(	)	+	*	ε
E	TE',ε	TE',ε				
E'			ε, ε	*P{*} T', ε		ε, ε
T	PE',ε	PE',ε				
T'			ε, ε	ε, ε	*P{*} T', ε	ε, ε
P	i{i},ε	(E),ε				
i	ВЫБРОС					
(		ВЫБРОС				
)			ВЫБРОС			
+				ВЫБРОС		
*					ВЫБРОС	
⊥						ДОПУСК
{i}	ВЫДАЧА({i})					
{+}	ВЫДАЧА({+})					
{*}	ВЫДАЧА({*})					
Начальное содержимое магазина - E⊥						

Начальное содержимое магазина — E⊥ Для входной цепочки i +i * i ДМП-процессор проделает следующую последовательность тактов: (i +i * i , E⊥, ε) (i +i * i , TE'⊥, ε) (i +i * i , PT'E'⊥, ε) (i +i * i , {I} T'E'⊥, ε (+ i * i , {i}T'E'⊥, ε) (+ i * i , {i}T'E'⊥, {i}) (+i * i , {i}E'⊥, {i}) (+i * i , +T {+} E'⊥,{i}) (i * i , T {+} E'⊥,{i}) (i * i , PT '{+} E'⊥, {i}) (i * i , i {i} T '{+} E'⊥, {i}) (*i , {i} T '{+} E'⊥, {i}) (* i , T '{+} E'⊥, {i} {i}) (* i , *P{*}T'{+}E'⊥, {i} {i}) (i , P{*}T'{+}E'⊥, {i} {i}) (i , i {i} {*}T'{+}E'⊥, {i} {i}) (ε , {i}{*}T'{+}E'⊥, {i} {i}) (ε , {i}{*}T'{+}E'⊥, {i} {i}) (ε , T'{+}E'⊥, {i} {i}{i}{*}) (ε , {+}E'⊥, {i} {i}{i}{*}) (ε , E'⊥, {i} {i}{i}{*}{+}) (ε , ⊥, {i} {i}{i}{*}{+})

ДМП-процессор можно использовать в качестве базового процессора для других видов синтаксически управляемого перевода, если операцию выдачи операционного символа в выходную ленту заменить операциями вызова соответствующих семантических процедур.

56. Определите транслирующую грамматику.

Транслирующая грамматика – это КС-грамматика, множество терминалов которой разбито на два множества: множество входных и множество операционных символов.

Дадим формальное определение транслирующей грамматики.

Определение: транслирующей грамматикой называется пятерка объектов $G^T = (N, \sum i, \sum a, P, S)$, где — словарь входных символов, $\sum a$ — словарь операционных символов, N — нетерминальный словарь, $S \in N$ — начальный символ транслирующей грамматики, P — конечное множество правил вывода вида $A \rightarrow \alpha$, в которых $A \in N$, а $\alpha \in (\sum I \cup \sum a \cup N)^*$.

58. Какие СУ-схемы называются простыми.

СУ-схема T = (N, ∑, Δ, R, S) называется **простой**, если для каждого правила A → α, β ∈ R соответствующие друг другу вхожд%ения не терминалов встречаются в α и β в одном и том же порядке.

Примечание - перевод, определяемый простой СУ-схемой, называется **простым синтаксически управляемым переводом**. Соответствие нетерминалов в выводимой паре простой СУ-схемы определяется порядком, в котором эти нетерминалы появляются в цепочках, поэтому для нее легко построить транслятор, представляющий собой преобразователь с магазинной памятью. По этой причине простые СУ-переводы образуют важный класс переводов.

60. Определите активную цепочку. Ее входную и операционную части.

Цепочки языка, порождаемого транслирующей Грамматикой, называют активными цепочками. Входной частью активной цепочки называется последовательностью входных символов, полученная из активной цепочки после удаления всех операционных символов. Операционной частью активной цепочки называется последовательность операционных символов, полученная путем вычёркивания из активной цепочки всех входных символов. Например, последовательность i + i * i является входной Частью активной цепочки i {i} + i {i} * i {i} {*} {+}, а последовательностью {i}{i}{i}{*}{+} - ее операционной частью

61. Как программируются процедуры в методе рекурсивного спуска, реализующего перевод, описываемый L-атрибутной транслирующей грамматикой.

Для того, чтобы методом рекурсивного спуска можно было выполнять Latрибутный перевод, введем в процедуры распознавания нетерминальных символов параметр для каждого атрибута этого символа. При этом если параметру процедуры соответствует унаследованный атрибут нетерминального символа, то вызов этой процедуры производится с фактическим параметром — значением унаследованного атрибута, а в случае синтезированного атрибута в качестве фактического параметра используется переменная, которой должно быть присвоено значение синтезированного атрибута в момент выхода из процедуры. Для того чтобы процедуры обеспечивали правильную передачу параметров для унаследованных и синтезированных атрибутов, они должны быть написаны на языке программирования, который обеспечивает способы передачи параметров "вызов по значению" (для унаследованных атрибутов) и "вызов по ссылке" (для синтезированных атрибутов). Замечание. Язык программирования Pascal поддерживает оба метода передачи параметров, а в языке С имеется единственный механизм передачи параметров — "вызов по значению". Эффект вызова по ссылке в языке С может быть получен, если использовать в качестве параметров указатели.

62. Определите входную и выходную(?) грамматики транслирующей грамматик

Транслирующей грамматикой называется пятерка объектов G T = (N, ∑i, ∑a, P, S), где — словарь входных символов, ∑a — словарь операционных символов, N — нетерминальный словарь, S ϵ N — начальный символ транслирующей грамматики, P — конечное множество правил вывода вида A → α, в которых A ϵ N, а α ∈ (∑l ∪ ∑a ∪ N)* . Транслирующая грамматика *G*0 *T* , описывающая перевод инфиксных арифметических выражений в ПОЛИЗ, содержит следующие правила: (1) E → E + T {+} (4) T → P (2) E → T (5) P → i {i} (3) T → T * P {*} (6) P → (E) Грамматика, полученная из транслирующей грамматики путем вычеркивания всех операционных символов, называется входной грамматикой для этой транслирующей грамматики. Язык, порождаемый входной грамматикой, называется входным языком.

При проектировании перевода следует руководствоваться следующими общими соображениями:

1. *Описание перевода строится последовательно для конструкций входного языка в порядке их усложнения, начиная с простейших конструкций, например: выражение, оператор присваивания, оператор Цикла. (???)*

2.*Выходная цепочка (результат перевода) включает в себя объекты (сущности) трех видов: (???)*

• объекты, передаваемые в выходную цепочку из входной цепочки;

•объекты, не изменяющиеся в процессе перевода (терминалы выходного языка);
•объекты, генерируемые во время перевода.

3. Транслирующая грамматика определяет порядок применения операционных символов (элементов перевода), строящих выходную цепочку, атрибуты же используются только для передачи значений символов из одних правил в другие.

4. Операционный символ, включаемый в правило вывода грамматики, может генерировать выходную цепочку до тех пор, пока в неё не должен быть включён объект, перемещаемый из входной цепочки и недоступный в данном правиле грамматики.

Проектирование описания перевода некоторой (одной!) конструкции входного языка выполняется в несколько этапов. Неформальное описание перевода. Неформальное описание перевода представляет собой пару цепочек, первая из которых является конструкцией входного языка, а вторая - ее представлением в выходном языке. При разработке описания перевода рекомендуется: строить описание перевода на основе описания синтаксиса и семантики входного языка, учитывая все возможные форматы представления входной конструкции (например, if-then и if-then-else); конструкции, перевод которых уже описан и которые являются частью данной конструкции, считать терминальными; после построения неформального описания перевода выделить в нем символы, передаваемые из входной цепочки, и символы, генерируемые при его построении

64. Как граф зависимостей используется при вычислении атрибутов

Написано не особо понятно, но интуитивно: строится граф зависимости атрибутов, и, если в нем отсутствуют циклы, то атрибутная грамматика является корректной.

68. Определите атрибутную транслирующую грамматику.

Атрибутная транслирующая грамматика - это транслирующая грамматика, обладающая следующими дополнительными свойствами:

1.Каждый символ грамматики (входной, нетерминальный и операционной) имеет конечное множество атрибутов, и каждый атрибут имеет (возможно, бесконечное) множество допустимых значений.

2.Все атрибуты нетерминальных и операционных символов делятся на синтезированные и унаследованные.

3.Унаследованные атрибуты подчиняются следующим правилам:

• каждому вхождению унаследованного атрибута в правую часть правила вывода сопоставляется правило, позволяющее вычислить значение этого атрибута как функцию некоторых других атрибутов символов, входящих в левую или правую часть данного правила;

• для каждого унаследованного атрибута начального символа грамматики задается начальное значение.

4.Правила вычисления значений синтезированных атрибутов определяются следующим образом:

• каждому вхождению синтезированного атрибута не терминального символа в левую Часть правила вывода сопоставляется правило, позволяющее вычислить значение этого атрибута как функцию некоторых других атрибутов символов, входящих в левую или правую часть данного правила;

• каждому синтезированному атрибуту операционного символа сопоставляется правило, позволяющее вычислить значение этого атрибута как функцию некоторых других атрибутов данного символа

. • каждому синтезированному атрибуту операционного символа сопоставляется правило, позволяющее вычислить значен

70. Какие атрибуты называются унаследованными, синтезированными.

Термин "унаследованный" означает то, что значение атрибута зависит от значений атрибутов предка символа или атрибутов его соседей в дереве вывода. Например, в грамматике G3T, значение атрибута r символа {ОТВЕТ} равно значению атрибута не терминала E (соседа слева), порождающего все выражение. Оно может быть вычислено только после того, как будут определены значения всех синтезированных атрибутов не терминалов

Атрибуты, значения которых вычисляются при движении по дереву вывода снизу вверх, традиционно называют синтезированными. Термин "синтезированный" подчёркивает, что значение атрибута синтезируется из значений атрибутов потомков.

(возможно здесь стоит привести конкретные примеры, но так как автор не шарит за этот вопрос, а сами примеры довольно объемные - примеры можно взять в книжке Семенова в главах 5.4.1 и 5.4.2)

74. Определите правило вычисления атрибутов.

Сопоставим каждому правилу вывода Грамматики правило вычисления значения атрибута не терминала из левой Части правила, которому соответствует не терминальная вершина дерева вывода. Рассмотрим, например, вершину T дерева, изображённого на рис. 5.4, а. Правило вывода, примененное к этой вершине, имеет вид: T → T * P. Оно определяет, Что значение подвыражения, порожденного не терминалом из левой Части правила, равно значению подвыражений, порожденных символами P и T из правой Части правила, которые являются прямыми потомками вершины T. Если p - атрибут символа T из левой Части правила, а q и r - атрибуты символов T и P из правой части правила, то правило вычисления значения атрибута p будет иметь вид: p ⟵ q * r, где символ '⟵' - знак операции присваивания. Таким образом, начиная с атрибутов

входных символов и поднимаясь по Дереву от кроны к корню, можно определить все значения атрибутов Нетерминалов из левых Частей правил вывода

76. Как строится граф зависимостей.

Граф зависимостей R(D) представляет собой ориентированный Граф, который строится для некоторого дерева вывода D следующим образом:

1) узлами R(D) являются пары (X, a), где X - узел дерева D, а a - атрибут символа, служащего меткой узла X;

2) дуга из узла ($X1$, $a1$) в узел ($X2$, $a2$) проводится, если семантическое правило, вычисляющее значение атрибута $a1$, непосредственно использует значение атрибута $a1$.

Для построения узлов графа зависимостей необходимо последовательно рассмотреть вершины дерева D и для каждой из них построить столько узлов, сколько атрибутов имеет символ, которым помечена эта вершина. Например, для корня дерева, обозначенного символом N с одним атрибутом, нужно в граф включить один узел, поместив его парой (N, $V4$). Для определения дуг графа необходимо использовать семантические правила.

78. Типы ошибок при тестировании АТ-грамматики.

Тестирование АТ-грамматики. Тестирование АТ-грамматики заключается в моделировании работы построенного по ней процессора, выполняющего перевод. При большом числе тестов и/или большой их длине эта работа может быть выполнена только при использовании соответствующих средств автоматизации. Для простых языковых конструкций, когда длина теста и их число невелики, а тестирование выполняется отдельно для каждой конструкции, начиная с простейших, работа эта не только необходима, но и реально выполнима.

Тестирование АТ-Грамматики позволяет определять два вида ошибок:

ошибки в структуре выходной цепочки (неверный порядок следования символов в выходной цепочке);

ошибки в значениях символов выходной цепочки, что связано с ошибками в передаче значений атрибутов.

При тестировании реального описания перевода желательно использовать комплексные тесты, определяющие оба вида ошибок.

83. Определите Latрибутную и S-атрибутную транслирующие грамматики.

Определение: АТ-грамматика называется L-атрибутной тогда и только тогда, когда выполняются следующие условия:

1. Аргументами правила вычисления значения унаследованного атрибута символа из правой части правила вывода могут быть только унаследованные атрибуты символа из левой части и произвольные атрибуты символов из правой части, расположенные левее рассматриваемого символа.

2. Аргументами правила вычисления значения синтезированного атрибута символа из левой части правила вывода являются унаследованные атрибуты этого символа или произвольные атрибуты символов из правой части.

3. Аргументами правила вычисления значения синтезированного атрибута операционного символа могут быть только унаследованные атрибуты этого символа.

Является ли произвольная АТ-грамматика L-атрибутной, можно проверить, независимо исследуя каждое правило вывода и каждое правило вычисления значения атрибута. Примером L-атрибутной транслирующей грамматики является грамматика G A .

При выполнении условия 1 определения унаследованные атрибуты каждой вершины дерева вывода зависят (непосредственно или косвенно) только от атрибутов входных символов, расположенных в дереве левее данной вершины, что позволяет использовать L-атрибутные грамматики в нисходящих синтаксических анализаторах. Условия 2 и 3 введены с целью сделать АТграмматику корректной.

В L-атрибутной транслирующей грамматике атрибуты символов A, B и C из правила вывода A →BC можно вычислять в следующем порядке:

унаследованные атрибуты символа A; унаследованные атрибуты символа B; синтезированные атрибуты символа B; унаследованные атрибуты символа C; синтезированные атрибуты символа C; синтезированные атрибуты символа A.

Определение: АТ-грамматика называется S-атрибутной тогда и только тогда, когда она является L-атрибутной и все атрибуты нетерминалов синтезированные.

Ограничения, накладываемые на L-атрибутную транслирующую грамматику, позволяют вычислять значения атрибутов в процессе нисходящего анализа входной цепочки. Нисходящий детерминированный анализатор для LL(1)- грамматик требует, чтобы L-атрибутная транслирующая грамматика, описывающая перевод, имела форму простого присваивания.