**1.** Будет ли логичным следующее рассуждение: *Если губернатор не имеет соответствующего авторитета или если он не желает принимать на себя ответственность, то порядок не будет восстановлен и волнения не прекратятся до тех пор, пока участникам волнений это не надоест, и власти не начнут примирительные действия. Следовательно, если губернатор не желает взять на себя ответственность и участникам волнений это не надоест, то волнения не прекратятся.*

Решение:

Введем обозначения для простых высказываний:

А = «Губернатор не имеет соответствующего авторитета»;

В = «Губернатор не желает принимать на себя ответственность»;

С = «Порядок не будет восстановлен»;

D = «Волнения не прекратятся»;

E = «Участникам волнений это не надоест»;

F = «Власти не начнут примирительные действия».

Формализуем посылку и заключение:

А ∨ В → (С&D ↔ E&F) – посылка;

В&Е → D – заключение.

Нужно определить, соблюдается ли логическое следствие.

Для того чтобы логическое следование соблюдалось, формула

(А ∨ В → (С&D ↔ E&F)) → (B&E → D)

должна быть тавтологией.

Упростим полученное выражение, применяя равносильные преобразования:

(А ∨ В → (С&D ↔ E&F)) → (B ∨ E → D) =

¬(¬(А ∨ В) ∨ С&D&E&F ∨ ¬(С&D)&¬(E&F)) ∨ ¬(B&E) ∨ D =

= (А ∨ В)&¬(С&D&E&F ∨ ¬(С&D)&¬(E&F)) ∨ ¬B ∨ ¬E ∨ D =

= (А ∨ В)&¬(С&D&E&F)&¬(¬(С&D)&¬(E&F)) ∨ ¬B ∨ ¬E ∨ D =

= (А ∨ В)&(¬С ∨ ¬D ∨ ¬E ∨ ¬F)&(С&D ∨ E&F) ∨ ¬B ∨ ¬E ∨ D =

= (А ∨ В)&(¬С&С&D ∨ ¬D&С&D ∨ ¬E&С&D ∨ ¬F&С&D ∨ ¬С&E&F ∨ ¬D&E&F ∨ ¬E&E&F ∨ ¬F&E&F) ∨ ¬B ∨ ¬E ∨ D = (А ∨ В)& (0 ∨ 0 ∨ ¬E&С&D ∨ ¬F&С&D ∨ ¬С&E&F ∨ ¬D&E&F ∨ 0∨ 0) ∨ ¬B ∨ ¬E ∨ D = (А ∨ В)&(¬E&С&D ∨ ¬F&С&D ∨ ¬С&E&F ∨ ¬D&E&F) ∨ ¬B ∨ ¬E ∨ D = А&¬E&С&D ∨ А&¬F&С&D ∨ А&¬С&E&F ∨ А&¬D&E&F ∨ В&¬E&С&D ∨ В&¬F&С&D ∨ В&¬С&E&F ∨ В&¬D&E&F ∨ ¬B ∨ ¬E ∨ D = = F ∨ ¬B ∨ ¬E ∨ D = ¬B ∨ D ∨ ¬E ∨ F.

Значение формулы не равно 1, формула не является тавтологией. Следовательно, заключение не выводимо из посылки, и рассуждение не будет логичным.

**2.** Провести исследование булевой функции

*f*(x,y,z) = (zx ∨ (z ⊕ x))((x ∨ y)(z ⊕ y)):

a) построить таблицу функции; ответ записать в виде набора значений, упорядоченного в соответствии с лексикографическим порядком набора аргументов;

b) построить СДНФ этой функции; ответ записать, упорядочив элементарные конъюнкции в лексикографическом порядке;

c) упростить полученное выражение с помощью метода минимизирующих карт, ответ записать в виде минимальной ДНФ;

d) построить многочлен Жегалкина исходной функции;

e) построить таблицу двойственной функции; ответ записать в виде упорядоченного набора значений;

f) построить СКНФ двойственной функции; ответ записать, упорядочив элементарные дизъюнкции в лексикографическом порядке;

g) проверить исходную функцию на принадлежность основным классам замкнутости T0, T1, L, M, S;

h) Выразить отрицание h(x) = ¬x и конъюнкцию g(x,y) = x∧y через функцию *f*(x,y,z) и ее отрицание.

Решение:

а) Таблица функции *f*(x,y,z) = (zx ∨ (z ⊕ x))((x ∨ y)(z ⊕ y)):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | x | y | z | zx | z⊕x | zx∨(z⊕x) | x∨y | z⊕y | (x∨y)(z⊕y) | (zx∨(z⊕x))((x∨y)(z⊕y)) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |

b) Построим СДНФ по таблице функции.

СДНФ – дизъюнкция элементарных конъюнкций (минтермов, конституент единицы). В СДНФ входят наборы переменных, на которых значение функции равно 1, причем, если значение переменной равно 1, то она входит в минтерм без отрицания, а если 0 – то с отрицанием. Для данной функции единичные значения определяются на наборах 101 (5) и 110 (6):

*f*СДНФ(x,y,z) = x ͞y z ∨ x y ͞z.

c) Упростим полученное выражение с помощью метода минимизирующих карт.

Строим для функции минимизирующую карту:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ͞x | ͞y | ͞z | ͞x ͞y | ͞x ͞z | ͞y ͞z | ͞x ͞y ͞z |  |
| ͞x | ͞y | z | ͞x ͞y | ͞x z | ͞y z | ͞x ͞y z |  |
| ͞x | y | ͞z | ͞x y | ͞x ͞z | y ͞z | ͞x y ͞z |  |
| ͞x | y | z | ͞x y | ͞x z | y z | ͞x y z |  |
| x | ͞y | ͞z | x ͞y | x ͞z | ͞y ͞z | x ͞y ͞z |  |
| x | ͞y | z | x ͞y | x z | ͞y z | x ͞y z | + |
| x | y | ͞z | x y | x ͞z | y ͞z | x y ͞z | + |
| x | y | z | x y | x z | y z | x y z |  |

Отметим в последнем столбце те конъюнкции, которые входят в СДНФ данной функции. Вычеркнем неотмеченные строки, затем вычеркнем в остальных строках (действуя по столбцу) те элементы, которые попали в вычеркнутые строки.

*f*МДНФ(x,y,z) = x ͞y z ∨ x y ͞z.

Упростим СДНФ, используя карту Карно.

На карте Карно отыскиваются квадраты и/или прямоугольники, в которые вписаны только единичные значения логической функции. Эти квадраты и прямоугольники должны быть образованы соприкасающимися единичными ячейками карты.

Целью минимизации является покрытие всех единичных значений логической функции в ячейках карты Карно минимальным количеством квадратов и прямоугольников максимальной площади (количество клеток в фигурах должно быть четным). При этом площадь квадратов и прямоугольников должна быть 2r, где r∈0…(n – 1), n – количество аргументов логической функции.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| yz  х | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |

На карте Карно нет смежных единичных ячеек, поэтому *f*МДНФ = *f*СДНФ:

*f*МДНФ(x,y,z) = x ͞y z ∨ x y ͞z.

d) Построим многочлен Жегалкина исходной функции по ее СДНФ с помощью равносильных преобразований:

Р(x,y,z) = x ͞y z ∨ x y ͞z = x(y ⊕ 1)z ⊕ xy(z ⊕ 1) = xyz ⊕ xz ⊕ xyz ⊕ xy =

= xz ⊕ xy.

e) Построим таблицу двойственной функции.

Пусть *f* – некоторая булева функция. Тогда двойственной к *f* (обозначается *f*\*) называют функцию, таблица истинности которой получается из таблицы истинности *f* заменой всех 0 на 1, а всех 1 на 0.

Такую операцию называют *инвертированием* таблицы истинности.

Инвертирование происходит не только со значениями функции, но и со значениями аргументов. Поскольку в таблице истинности записывают значения переменных в лексикографическом порядке, после построения её нужно «перевернуть», то есть переставить все строки в обратном порядке.

Тогда таблица значений двойственной функции будет иметь вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | x | y | z | *f*\* |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 |

f) Построим СКНФ двойственной функции по таблице ее значений.

СКНФ – конъюнкция элементарных дизъюнкций (макстермов, конституент нуля). В СКНФ входят наборы переменных, на которых значение функции равно 0, причем, если значение переменной равно 0, то она входит в макстерм без отрицания, а если 1 – то с отрицанием. Для данной функции нулевые значения определяются на наборах 001 (1) и 010 (2):

*f*\*СКНФ(x,y,z) = (x ∨ y ∨ ͞z)(x ∨ ͞y ∨ z).

g) Проверим исходную функцию на принадлежность основным классам замкнутости T0, T1, L, M, S.

Так как *f*(0,0,0) = 0, функция принадлежит к классу T0.

Так как *f*(1,1,1) = 0, функция не принадлежит к классу T1.

Так как *f*(1,1,0) > *f*(1,1,1), функция не принадлежит к классу М.

Так как *f*(0,0,0) = *f*(1,1,1), функция не принадлежит к классу S.

Так как полином функции P(x,y,z)=xz ⊕ xy содержит конъюнкции, функция не принадлежит к классу L.

Заполним таблицу Поста:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Т0 | Т1 | M | S | L |
| *f*(x, y, z) | + | – | – | – | – |

h) Выразим отрицание h(x) = ¬x и конъюнкцию g(x,y) = x∧y через функцию *f*(x,y,z) и ее отрицание.

Функция *f* не принадлежит классу монотонных функций М.

Найдем пару наборов, на которой нарушается монотонность, это наборы (110) и (111).

h(х) = *f*(1,1,х) = ¬х, так как h(0) = *f*(110) = 1; h(1) = *f*(111) = 0.

Таким образом, *f*(x,y,z) = h(x) = ¬x.

Для построения конъюнкции из функции *f*СДНФ(x,y,z) = x ͞y z ∨ x y ͞z зафиксируем переменную z = 0.

Тогда *f*(x, y, 0) = x∧͞y∧0 ∨ x∧y∧1 = x∧y = g(x, y).

**3.** Привести формулу логики предикатов сначала в ПНФ, затем в СНФ:

F = ¬∃x(¬∀y(G(*f*(y))) ⊕ Q(x))

Решение:

Приведем формулу в ПНФ:

1) продвинем отрицание до элементарных формул, пользуясь правилами двойственности кванторов ¬∃xQ(x) = ∀x¬Q(x) и ¬∀xQ(x) = ∃x¬Q(x), и устраним логическую связку ⊕, пользуясь равносильностью ¬(А(х) ⊕ В(х)) = А(х) ↔ В(х):

F = ¬∃x(¬∀y(G(*f*(y))) ⊕ Q(x)) = ∀x¬(¬∀y(G(*f*(y))) ⊕ Q(x)) = ∀x(∃y¬(G(*f*(y))) ↔ Q(x));

2) вынесем кванторы в префикс, не нарушая их очередности:

F = ∀x(∃y¬(G(*f*(y))) ↔ Q(x)) = ∀x∃y(¬G(*f*(y)) ↔ Q(x));

3) приведем формулу к КНФ:

F = ∀x∃y(¬G(*f*(y)) ↔ Q(x)) =

= ∀x∃y[(¬G(*f*(y)) → Q(x))&(Q(x) → ¬G(*f*(y)))] =

= ∀x∃y[(G(*f*(y)) ∨ Q(x))&(¬Q(x) ∨ ¬G(*f*(y)))] – ПНФ.

Приведем формулу в СНФ.

Так как перед квантором существования находится квантор всеобщности, удалим квантор существования с заменой связанной им переменной на функциональный терм, аргументом которого является переменная, связанная квантором всеобщности, y = *f*(x):

F = ∀x∃y[(G(*f*(y)) ∨ Q(x))&(¬Q(x) ∨ ¬G(*f*(y)))] =

= ∀х[(G(*f*(*f*(x))) ∨ Q(x))&(¬Q(x) ∨ ¬G(*f*(*f*(x))))] – СНФ.

**4.** Машина Тьюринга имеет алфавит из трех символов {2,1,\*} (символ \* означает отсутствие символа на ленте), два состояния {q0, q2}, из которых q0 – начальное состояние, q2 – конечное. Символ R означает сдвиг читающей головки вправо по ленте, L – влево, E – головка остается на месте. В начальный момент головка указывает на крайний левый символ записи. Команды машины задаются набором:

q02 → q01R,

q01 → q01L,

q0\* → q21E.

Какой результат даст машина на наборе {22122}?

Решение:

Начальное состояние МТ имеет вид:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 2 | 2 | 1 | 2 | 2 | \* | \* | \* |

В начальный момент МТ обозревает крайний левый символ записи и находится в состоянии *q0*:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 2 | 2 | 1 | 2 | 2 | \* | \* | \* |
|  |  | *q*0 |  |  |  |  |  |  |  |

Находим в программе команду с левой частью *q*02. Это команда *q*02 → *q*01R. Она означает, что в рассматриваемую ячейку записывается 1, головка сдвигается на одну ячейку вправо и устанавливается в состояние *q*0. В результате применения этой команды имеем:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 1 | 2 | 1 | 2 | 2 | \* | \* | \* |
|  |  |  | *q*0 |  |  |  |  |  |  |

Снова обозревается символ 2 и МТ находится в состоянии *q*0. В обозреваемую ячейку записывается 1, головка сдвигается на одну ячейку вправо и остается в состоянии *q*0. Получим:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 1 | 1 | 1 | 2 | 2 | \* | \* | \* |
|  |  |  |  | *q*0 |  |  |  |  |  |

Теперь обозревается символ 1. Находим в программе команду с левой частью *q*01. Это команда *q*01 → *q*01L. Она означает, что в рассматриваемую ячейку записывается 1, головка сдвигается на одну ячейку влево и устанавливается в состояние *q*0. В результате применения этой команды имеем:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 1 | 1 | 1 | 2 | 2 | \* | \* | \* |
|  |  |  | *q*0 |  |  |  |  |  |  |

Снова обозревается символ 1 и МТ находится в состоянии *q*0. В обозреваемую ячейку записывается 1, головка сдвигается на одну ячейку влево и остается в состоянии *q*0. Получим:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 1 | 1 | 1 | 2 | 2 | \* | \* | \* |
|  |  | *q*0 |  |  |  |  |  |  |  |

Снова обозревается символ 1 и МТ находится в состоянии *q*0. В обозреваемую ячейку записывается 1, головка сдвигается на одну ячейку влево и остается в состоянии *q*0. Получим:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | 1 | 1 | 1 | 2 | 2 | \* | \* | \* |
|  | *q*0 |  |  |  |  |  |  |  |  |

Теперь обозревается символ \*. Находим в программе команду с левой частью *q*0\*. Это команда *q*0\* → *q*21Е. Она означает, что в рассматриваемую ячейку записывается 1, головка остается на месте, и МТ переходит в состояние *q*2. В результате применения этой команды имеем:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | 1 | 1 | 1 | 1 | 2 | 2 | \* | \* | \* |
|  | *q*2 |  |  |  |  |  |  |  |  |

Состояние *q*2 – заключительное, поэтому машина заканчивает работу.

Получено выходное слово {111122}.

**5.** Пусть A = {0,1,2,3}. Построить нормальный алгоритм Маркова, преобразующий слово так, чтобы сначала шли все четные цифры 0 и 2, а затем – все нечетные.

Решение:

Добавим к алфавиту дополнительный символ \*, отмечающий конец слова. Сначала символ \* находится перед словом, а затем перемещается к его концу. Перемещение символа \* от начала к концу слова – это замена пар \*0, \*1, \*2, \*3 на пары 0\*, 1\*, 2\* и 3\*.

Введем следующие подстановки и получим нормальный алгоритм Маркова, преобразующий слово так, чтобы сначала шли все четные цифры 0 и 2, а затем – все нечетные:

Λ → \*, \*0 → 0\*, \*1 → 1\*, \*2 → 2\*, \*3 → 3\*,

10 → 01, 12 → 21, 30 → 03, 32 → 23, 1\* → •1, 3\*→ •3.

Проверим работу алгоритма на конкретном слове (шаги по перемещению символа \* к концу слова опущены):

30211203\* ⇒ 03211203\* ⇒ 02311203\* ⇒ 02312103\* ⇒ 02321103\* ⇒

02231103\* ⇒ 02231013\* ⇒ 02230113\* ⇒ 02203113\* ⇒ 02203113.

**6.** Построить конечный автомат с входным алфавитом {0,1}, который допускает все цепочки, в которых перед и после каждой единицы стоит 0.

Решение:

Пусть автомат, например, допускает цепочки вида 0(10)n, n > 0: 010, 0101010, 01010101010 и т.д.

Обозначим состояния автомата следующим образом:

q1 – начальное состояние;

q2 – в начальном состоянии поступил 0;

q3 – после 0 поступила 1;

q4 – после 1 поступил 0;

q5 – «ошибочное» состояние: переход в него осуществляется, если в начальном состоянии поступила 1, или после 0 поступил 0, или после 1 поступила 1, или в начальном состоянии поступил пустой символ или символ, не относящийся к алфавиту {0,1};

q6 – заключительное состояние.

Таблица переходов:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | – |
| q1 | q2 | q5 | q5 |
| q2 | q5 | q3 | q5 |
| q3 | q4 | q5 | q5 |
| q4 | q5 | q3 | q6 |

«–» – отсутствие символа или символ, не принадлежащий алфавиту {0, 1}; обозначение конца цепочки.

Проверим работу автомата.

a(t) – входной символ;

q(t) – исходное состояние;

q(t + 1) – последующее состояние.

Пусть задана цепочка 0101010:

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг | a(t) | q(t) | q(t + 1) |
| 0 | 0 | q1 | q2 |
| 1 | 1 | q2 | q3 |
| 2 | 0 | q3 | q4 |
| 3 | 1 | q4 | q3 |
| 4 | 0 | q3 | q4 |
| 5 | 1 | q4 | q3 |
| 6 | 0 | q3 | q4 |
| 7 | – | q4 | q6 |

q6 – заключительное состояние, следовательно, цепочка допускается.

Пусть задана цепочка 0101011:

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг | a(t) | q(t) | q(t + 1) |
| 0 | 0 | q1 | q2 |
| 1 | 1 | q2 | q3 |
| 2 | 0 | q3 | q4 |
| 3 | 1 | q4 | q3 |
| 4 | 0 | q3 | q4 |
| 5 | 1 | q4 | q3 |
| 6 | 1 | q3 | q5 |

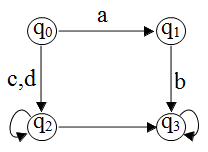
q5 – не заключительное состояние, переходов из него нет, поэтому цепочка не допускается.

**7.** Описать конечный автомат, распознающий язык, заданный регулярным выражением:

(ab + (c + d)\*)\*

Решение:

Граф конечного (недетерминированного) автомата, распознающего язык, заданный данным регулярным выражением, имеет вид:



**8.** Построить порождающую грамматику для языка L = {anbnсn, n > 0}.

Решение:

Выражения an, bn и сn означают повторение n раз символов a, b и с. Таким образом, язык L состоит из цепочек вида abc, aabbcc и т.д.

Ввиду того, что КС-грамматика является порождающей, она задает алгоритм порождения цепочек языка. Порождение здесь задается не только присоединением цепочек справа или слева имеющейся цепочки, но и вставкой цепочки внутрь имеющейся. Вставка производится заменой нетерминальных символов в цепочке на цепочку, которая стоит в правой части некоторого правила, в левой части которого находится этот нетерминал.

Зададим КС-грамматику для данного языка. Из одной цепочки языка можно получить другую, присоединяя к первой справа символ с и заменяя символы ab на aabb. Например, имеется цепочка aabbcc, из нее можно получить цепочку aaаbbbccс по правилам порождения S → Sс и ab → aabb.

Есть еще специальный случай цепочки, которая не дробится на более мелкие – это цепочка аbс. Для ее порождения введем правило S → abс.

Итак, грамматика языка имеет правила:

S → Sс, ab → aabb и S → abс.

Зададим порождение цепочки aaabbbccc: S ⇒ Sс ⇒ aaabbbccc.

**9.** Описать язык, который определяет КС грамматика S::= 1 | S0S. Удовлетворяет ли она условию однозначности ветвления по первому символу?

Решение:

Грамматика языка в форме Бэкуса-Наура:

<буква>::= <цифра>|(<буква><цифра><буква>).

<буква>::= S, <цифра>::= 1.

Грамматика удовлетворяет условию однозначности ветвления по первому символу, так как L(<буква>) ≠ L(<цифра>), то есть L(<буква>) ∩ L(<цифра>) = ∅.

**10.** Для грамматики, заданной следующими правилами вывода, построить эквивалентный ей конечный автомат:

S → 0S|0D, C → 0D|0S|1, D → 1C|1S|0.

Решение:

Для грамматики, заданной следующими правилами вывода,

S → 0S|0D, C → 0D|0S|1, D → 1C|1S|0,

построим эквивалентный ей конечный автомат.

Граф автомата имеет четыре вершины, три из которых помечены нетерминальными символами S, С и D, а третья вершина, помеченная символом T (terminal), – это заключительное состояние.

Начальное состояние – это вершина S, соответствующая аксиоме грамматики S.

Каждому правилу грамматики ставится в соответствие команда конечного автомата:

1) S0 → S: в начальном состоянии при подаче на вход терминального символа 0 автомат остается в состоянии S;

2) S0 → D: из начального состояния при подаче на вход терминального символа 0 автомат переходит в состояние D;

3) C0 → D: из состояния C при подаче на вход терминального символа 0 автомат переходит в состояние D;

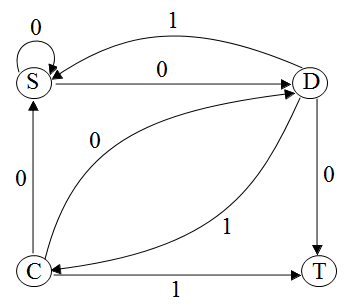
4) C0 → S: из состояния C при подаче на вход терминального символа 0 автомат переходит в начальное состояние S;

5) С1 → Т: из состояния C при подаче на вход терминального символа 1 автомат переходит в конечное состояние Т;

6) D1 → C: из состояния D при подаче на вход терминального символа 1 автомат переходит в состояние C;

7) D1 → S: из состояния D при подаче на вход терминального символа 1 автомат переходит в начальное состояние S;

8) D0 → Т: из состояния D при подаче на вход терминального символа 0 автомат переходит в конечное состояние Т.



Получившийся недетерминированный конечный автомат распознает цепочки символов, заданные правилами вывода данной грамматики.

**11.** Дана инфиксная скобочная форма записи арифметического выражения:

(a – b\*c)/(d + e + f).

Перевести ее в постфиксную форму.

Решение:

Постфиксной называют форму записи алгебраических выражений, в которой знаки операций расположены после операндов.

Постфиксная форма записи арифметического выражения (a – b\*c)/(d + e + f):

(a – b\*c)/(d + e + f) = abc\*– de + f +/.

Сначала идет оператор \*, относящийся к операндам b и c. Затем идет оператор –, относящийся к операндам a и b\*c. После этого осуществляется сложение операндов d и e и d + e и f. Последний символ в постфиксе – символ /, задающий деление операнда abc\*– на операнд de + f +.