Комментарии:
Это собрание ответов на вопросы к экзамену, собранные из конспекта В.Е.Зайцева и конспектов с очных лекций.
Эксклюзивная информация с лекций
В части материала звездочка отображается штрихом

Оглавление

1. Предмет информатики.

Информатика: наука об осуществляемой преимущественно с помощью автоматических средств целесообразной обработке информации, рассматриваемой как представление знаний и сообщений в технических, экономических и социальных областях.

2. Информация и сообщения. Интерпретация сообщений.

Информация и сообщения

Сведение — множественное число термина «Информация».

Информация передается посредством сообщения, и наоборот, сообщение — то, что несет информацию. Сообщение является материальным носителем информации.

Интерпретация:

$\varphi = N \rightarrow I$
$N - множество\ сообщений$
$I - информация$

Информация i ,которая передается сообщением n, устанавливается с помощью правила интерпретации, которое представляет собой отображение рассматриваемого множества сообщений B в множестве сведений.

Сигналы:

Вопрос: все что мы видим, слышим — аналоговый сигнал.К сигналам относят магнитные потоки, токи, колебания, звуки и тд.Аналоговые компьютеры работают на переменном токе.Пример: часы со стрелками, стрелочки еле заметно двигаются. Они демонстрируют непрерывное течение аналогового времени. Но в них пружины → они и дискретные тоже.

3. Знаки и символы.

Атомарным знаком или буквой называется элемент какого-либо упорядоченного конечного непустого множества графически (или каким-либо иным образом) отличимых друг от друга литер (предметов, сущностей, объектов, членов), называемого алфавитом.
Составным знаком или словом называется конечная последовательность знаков (неважно, атомарных или составных). Множество слов тоже можно рассматривать как набор знаков (алфавит или словарь).

Символ - знак со смыслом

4. Кодирование.

Кодом называется правило, описывающее отображение $C': A \to A'$ алфавита A на другой набор знаков A'.
Так же стоит сказать, что если мы имели отображение φ, то теперь мы используем $\varphi_1 = C^{-1} \circ \varphi$, где C^{-1} – отображение, обратное C. Если известно C или C^{-1}, то говорят, что известен ключ кода C.

Кодирование, при котором каждый образ является отдельным знаком, называется шифрованием.

1. ASCII - таблица семибитного кода.(1966)
‘‘‘‘‘‘

порядок знаков алфавита: служебные символы → пробел → 15 разделителей → десятичные цифры 0-9 → 7 разделителей → 26 заглавных латинских букв → → 6 доп. Символов → 26 малых латинских букв.
‘‘‘‘‘‘

2. КОИ-7 — таблица ASCII(1966), где вместо малых латинских букв, поместили русские.
3. EBCDIC(первая 8-битная кодировка, по русски ДКОИ)
4. КОИ-8, русские буквы размещены фонетически. То есть гашение старшего бита дает вместо АБВГД → ABVGD.
//8-битные удобнее 7-битных, однако мы все так же можем запомнить один алфовит.
5. Unicode(первая 16-битная кодировка, 1988). Позволяет кодировать 65536 знаков вместо 256 8-битных кодов.
‘‘‘‘‘‘

Интересен факт, что 29000 позиций кодировки не заняты, а 6000 зарезервированы, для использования программистами.
‘‘‘‘‘‘

Утверждение: Произвольный конечный набор знаков (алфавит) может быть закодирован знаками набора $\omega = (b_0, b_1)$
Доказательство:
Знаки исходного набора могут быть закодированы знаками стандартного алфавита w_m, где m – число знаков в исходном наборе. Пусть $b_i \in w_m$. Сопоставим ему код $b_0 b_1 \ldots b_{(i+1)} \nabla$

5. Системы счисления

Система счисления — это способ числовой интерпретации цифровых сообщений.

Позиционная система счисления — это полиномиальный способ числовой интерпретации слов над цифровым алфавитом. Коэффициентами многочлена являются малые целые числа, означающие уже а приори (вручную) проинтерпретированные цифры.

Позиционная система задается одним натуральным числом $p \in N$ — основанием системы счисления — которое однозначно определяет алфавит $Ap = \{0, 1, \ldots, p - 1\}$ и функцию интерпретации

wiki: Позиционная систе́ма счисле́ния (позиционная, поме́стная нумерация) — система счисления, в которой значение каждого числового знака (цифры) в записи числа зависит от его позиции (разряда) относительно десятичного разделителя.

Логический сдвиг представляет собой перемещение всех разрядов машинного слова влево или вправо на число разрядов, заданное вторым операндом. При этом освобождающиеся разряды заполняются нулями, а разряды, выходящие за пределы, теряются.

При циклическом сдвиге разряды машинного слова считаются закольцованными, образуя двусторонний список, т. е. С одной стороны уходят, с другой приходят.
При арифметическом сдвиге сдвиг влево — умножение, вправо — деление. В первом случае «знакомый» разряд двигается влево, а младшие теряются.
При сдвиге вправо знаковый неизменный, а старшие разряды выпадают.

Слова Зайцева: Арифметический сдвиг — сдвиг влево(в урну выпадает 2 справа член) (вправо в 2 справа член добавляются 0).

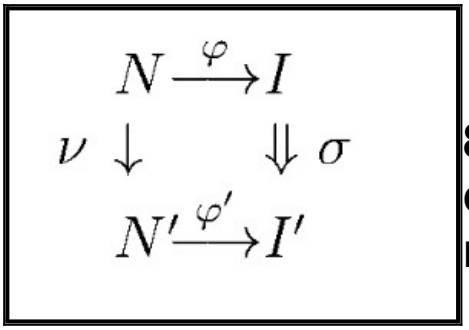
6. Обработка сообщений.

Обработка сообщений определяется правилом обработки ν, которое представляет собой отображение $\nu: N \to N'$, где N – множество исходных сообщений, а N' – множество сообщений, получающихся в результате обработки.

Примером обработки сообщений является кодирование (см. п. 1.5). Другие примеры обработки сообщений: перевод с одного языка на другой, чтение вслух, стенографирование, перепечатка текста на машинке, редактирование текста, решение системы уравнений и т. д.

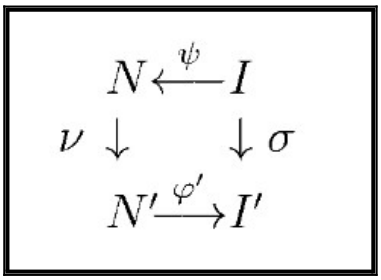
Обработка сообщений никогда не осуществляется «мгновенно», а всегда требует определенного времени, которым, как правило, нельзя пренебречь. Зависимость от времени приводит к понятию эффективности правила обработки сообщений, которая измеряется скоростью процесса обработки по сравнению со скоростями других процессов.

7. Обработка информации.



8. Автоматизация обработки информации.

Заменим на прошлой диаграмме φ обратным отображением $\psi = \varphi^{-1}$, получим :



Автоматизация обработки информации заключается в выполнении σ или $\varphi^{-1} \circ \nu \circ \varphi$ ' при помощи физических устройств .

Для автоматизации обработки сообщений необходимо иметь физическое устройство, работа которого «моделировала» бы выполнение отображения ν . Для этого необходимо располагать тремя физическими представлениями:

1. физическим представлением для множества исходных сообщений N — его мы будем обозначать буквой D и называть множеством исходных данных ;

2. физическим представлением для множества результирующих сообщений N ′ — его мы будем обозначать буквой D′ и называть множеством результирующих данных (результатов);

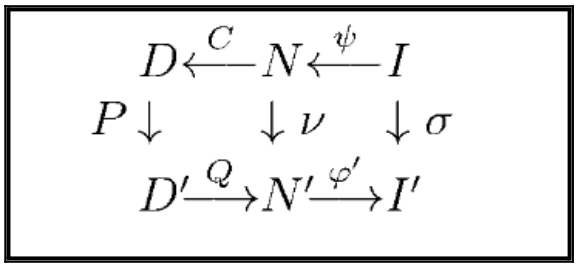
3. физическим представлением правила обработки ν , которое должно быть преобразованием, или последовательностью преобразований, оперирующих над D и дающих в результате элемент D′, — его мы будем обозначать буквой P ; преобразование P должно быть физически реализуемым, т. е. выполняться на некотором физическом устройстве.

Необходимо автоматизировать выполнение трех отображений:

1. отображения кодирования C , которое позволяет представить (абстрактное) сообщение из N с помощью элемента множества данных D, т. е ., с помощью конкретного состояния некоторого физического устройства;

2. отображения P , которое переводит элемент D в элемент D′—конкретное состояние еще одного физического устройства;

3. отображения декодирования Q, которое позволяет проинтерпретировать результат —конкретное состояние физического устройства, представляющее элемент из D′,переведя его в соответствующее сообщение из множества N′.



Из диаграммы следует ν = C ◦ P ◦ Q , где C , P и Q должны выполняться автоматом .

9. Конструктивное описание процесса обработки дискретных сообщений.

Чтобы отображение P могло служить основой для автоматизации обработки сообщений, оно должно задавать некоторый **способ построения** сообщения d′= P (d), исходя из сообщения d ∈ D. Если D — конечное множество, то отображение P может быть задано таблицей, в которой перечисляются все сообщения d ∈ D и соответствующие им сообщения P (d). Примером такой таблицы может служить таблица умножения (или таблица сложения) однозначных чисел в позиционной системе счисления. Если же D бесконечно или по крайней мере так велико, что задание P с помощью таблицы оказывается непрактичным, то нужно представить P в виде последовательности **элементарных шагов обработки (или тактов)**, каждый из которых состоит в выполнении одного или нескольких достаточно простых отображений, называемых операциями: P = P1 ◦ P2 ◦. . . ◦ Pk . Примеры операций: переписывание (копирование) буквы или слова, приписывание буквы к слову, приписывание слова к другому слову.

КОРОТКО ГОВОРЯ: Если D — конечная, то надо построить таблицу, если бесконечная, то P надо представить в виде элементарных шагов обработки, каждый их которых состоит в выполнении элементных отображений, называемых операциями

Первый язык программирования — машинный язык

10. Свойства алгоритмов.

Алгоритм – точно заданная последовательность правил , указывающая , каким образом можно за за конечное число шагов получить выходное сообщение определенного вида , используя входное сообщение .

1. **Массовость** — потенциальная бесконечность исходных сообщений, предназначенных для переработки алгоритмом

2. **Детерминированность** (однозначность, определенность)процесса применения алгоритма, заключающегося в последовательном выполнении дискретных действий

3. **Элементарность** каждого шага обработки, который должен быть легко выполним

4. **Результативность** алгоритма, точное описание того, что считается результатом применения алгоритма после выполнения всех требуемых шагов

5. **Сложность алгоритма** задается количеством простых операций, которые надо совершить для обработки сообщения. Это количество зависит от длины сообщения n и обычно задается некоторой функцией.

11. Сложность алгоритмов

Класс сложности	Описание
O(1)	Алгоритмы с постоянным временем выполнения, например доступ к элементу массива. При увеличении размера задачи вдвое время выполнения не меняется.
O(log n)	Алгоритмы с логарифмическим временем выполнения, например бинарный поиск. Время выполнения удваивается при увеличении размера задачи в n раз.
O(n)	Алгоритмы с линейным временем выполнения, например последова- тельный поиск или схема Горнера. При увеличении размера задачи вдвое время выполнения также удваивается.
O(n log n)	Алгоритмы с линеарифмическим временем выполнения, например, быстрая сортировка. Время выполнения увеличивается немного боль- ше, чем вдвое, при увеличении размера задачи в два раза.
O(n2)	Алгоритмы с квадратичным временем выполнения, например простые алгоритмы сортировки. Время выполнения увеличивается в 4 раза при увеличении размера задачи в 2 раза.

O(n3)	Алгоритмы с кубическим временем выполнения, например умножение матриц. Время выполнения увеличивается в 8 раз при увеличении размера задачи в 2 раза.
O(2n)	Алгоритмы с экспоненциальным временем выполнения, например задача о составлении расписания. Время выполнения увеличивается в 2^n раз при увеличении размера задачи в 2 раза.

12. Семиотические модели интерпретации дискретных сообщений

Семиотика — наука о понимании

Строя математическое понятие модели, мы исходим из того, что внешний мир (среда) состоит из объектов (предметов), которые обладают определенными свойствами (атрибутами) и находятся в определенных отношениях друг с другом. Объектам, их атрибутам и отношениям ставятся в соответствие знаки (имена), из которых строятся сообщения (напомним, что дискретное сообщение — это последовательность знаков).

Определение 1.11 . 1. Назовём k – местным отношением на множестве M совокупность r^k упорядоченных наборов из k элементов множества M вида $\langle x_1 , x_2 , ... , x_k \rangle$

Определение 1.11 . 2. Моделью называется некоторое множество M с заданным на нем наборов отношений $\langle r_1^{(k_1)} , r_2^{(k_2)} , ... , r_n^{(k_n)} \rangle$.

Иначе говоря модель M – это упорядоченная пара объектов $M = \langle M , \{ r_1^{(k_1)} , r_2^{(k_2)} , ... , r_n^{(k_n)} \} \rangle$, где $r_1^{(k_1)} , r_2^{(k_2)} , ... , r_n^{(k_n)}$ – отношения на мн – ве M .

Сигнатура (Ω) — это перечень знаков отношений .

Теория – это упорядоченная пара $T = (\Omega , A)$, A — множество аксиом , высказываний о свойствах сигнатуры Ω .

Модель M называется моделью теории T , если M и T , имеют одинаковую сигнатуру и если интерпретации φ каждого имени отношения теории , как одноименного отношения в модели , каждая аксиома теории становится истинным высказыванием .

13. Необходимость формального определения алгоритма.

Алгоритм – точно заданная последовательность правил , указывающая , каким образом можно за за конечное число шагов получить выходное сообщение определенного вида , используя входное сообщение .

Так как данное определение расплывчато, возникает необходимость формализации определения алгоритма:
Формализация понятия алгоритма реализуется с помощью построения **алгоритмических моделей.** Можно выделить три основных типа универсальных алгоритмических моделей: **рекурсивные функции** (понятие алгоритма связывается с вычислениями и числовыми функциями), **машины Тьюринга** (алгоритм представляется как описание процесса работы некоторой машины, способной выполнять лишь небольшое число весьма простых операций), **нормальные алгоритмы Маркова** (алгоритмы описываются как преобразования слов в произвольных алфавитах).

Все модели вышли из математики

14. Машины Тьюринга.

Историческая справка: «В 1936 г. аспирант Алан Тьюринг при исследовании алгоритмических проблем разрешимости (одной из знаменитых проблем Гильберта) предложил для уточнения понятия алгоритма использовать абстрактную вычислительную машину с очень простым набором операций.»

Тезис Тьюринга-Черча

Тезис говорит, о том что, всякий интуитивный алгоритм может быть выражен средствами одной из алгоритмических моделей. Доказано, что одни модели сводятся к другим. Наиболее близкой к вычислительным машинам является машина Тьюринга.

Неформальное определение:

Машина Тьюринга состоит из ограниченной с одного конца бесконечной ленты, разделенной на ячейки, и комбинированной читающей и пишущей головки, которая может перемещаться вдоль ленты от ячейки к ячейке.

В каждой ячейке ленты может быть записан один знак алфавита A, называемого **рабочим алфавитом МТ**, либо пробел (его мы будем обозначать знаком λ).

При этом головка находится в одном из конечного множества $Q = \{q_0, q_1, \ldots, q_s\}$ дискретных состояний , среди которых выделено одно начальное состояние q 0.

Определено три вида элементарных действий: сдвиг головки на одну ячейку влево (если считать, что лента МТ ограничена слева, то для крайней левой ячейки сдвиг влево не определен), сдвиг головки на одну ячейку вправо, запись в рабочую ячейку какой-либо буквы рабочего алфавита A либо пробел (при

этом буква, которая была записана в рабочей ячейке до выполнения записи, стирается).

Команда МТ описывается упорядоченной четверкой символов (q , a , v , q'), где
1. $q \in Q$ – символ текущего состояния головки .
2. $a \in A \cup \{\lambda\}$ – буква , записанная в текущей рабочей ячейке .
3. $\upsilon \in \{l , r\} \cup A \cup \{\lambda\}$ – символ выполняемого действия .
4. $q' \in Q$ – символ состояния , в которую машина переводит головку .

Математически строгое определение

Машиной Тьюринга называется упорядоченная четверка объектов $T = (A , Q , P , q_0)$, где T – символ МТ , A – конечное множество букв $(раб . алф)$, Q – конечное множество символов $(имен\ сост .)$, q_0 – имя начального состояния , $P = (q , a , \upsilon , q')$, где $q , q' \in Q$, $a \in A \cup \{\lambda\}$, $\upsilon \in \{l , r\} \cup A \cup \{\lambda\}$, определяющее три функции : **функцию выхода** $F_l : Q * \bar{A} \rightarrow \bar{A}$ $(\bar{A} = A \cup \{\lambda\})$, функцию переходов $F_t = Q * \bar{A} \rightarrow Q$, и функцию движения головки $F_\upsilon = A * \bar{A} \rightarrow \{l , r , s\}$.

Ситуацией (или состоянием ленты) называется упорядоченная пара объектов S = (z , k), где S — имя ситуации, z — сообщение, записанное на ленте, k — неотрицательное целое число, равное расстоянию (в ячейках) от края ленты до рабочей ячейки.

Конфигурацией называется упорядоченная пара объектов C = (S, q), где S — текущая ситуация, q — текущее состояние головки.

15. Нормальные алгоритмы Маркова.

Историческая справка: Алгоритмы Маркова предложены в 1950 г. нашим соотечественником академиком А. А . Марковым.

Неформальное определение

Нормальные алгоритмы Маркова по существу являются детерминистическими текстовыми заменами, которые для каждого входного слова однозначно задают вычисления и тем самым в случае их завершения порождают определенный результат.

Марковская стратегия применения правил заключается в следующем:

1. если применимо несколько правил, то берется правило, которое встречается в описании алгоритма первым;

2. если правило применимо в нескольких местах обрабатываемого слова, то выбирается самое левое из этих мест.

Строгое определение

Нормальный алгоритм Маркова (НАМ) задаётся конечной последовательностью правил (продукций , подстановок) p_1 , p_2 , . . . , p_n . Нормальный алгоритм Маркова начинает свою работу с начального входного слова w 0 , которое принадлежит $(A \cup A')*$. В процессе выполнения НАМ это слово подвергается постепенным изменениям . Такт работы алгоритма состоит в поиске правила , применимого к текущему обрабатыва – емому слову НАМ :

• поиск релевантного правила ведётся , начиная с первого правила НАМ ;
• если ни одно правило не применимо , алгоритм завершается без терминирующего правила ;
• первое подходящее правило применяется к самому левому вхождению слова из левой части правила ;
• после применения терминального правила НАМ завершается .

Тезис Маркова: любой алгоритм в алфавите А может быть представлен нормальным алгоритмом Маркова над алфавитом А.

Примерно так же, как и для МТ, можно доказать алгоритмическую неразрешимость проблемы остановки и самоприменимости.

16. Диаграммы машин Тьюринга

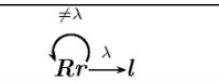
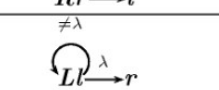
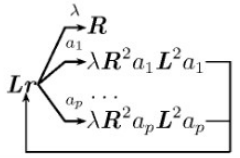
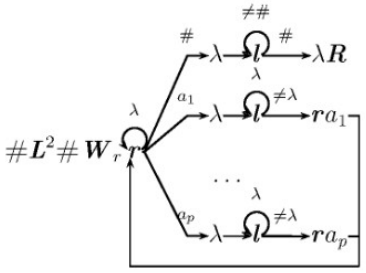
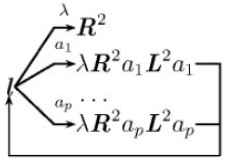
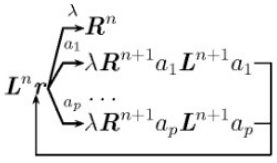
Диаграммы Тьюринга представляют одни МТ через другие, более простые МТ иным, визуально-топологическим способом, причём, как будет показано далее, этот способ не менее строг и полон, нежели "обычные"МТ.

При этом рассматриваемая МТ будет описана через **элементарные МТ**, т. е. Такие, которые уже нельзя описать через более простые МТ, так как каждая из них выполняет всего одно элементарное действие и останавливается.

Элементарные МТ над алфавитом Ap = (a1 , a2 , ..., ap) определяются следующими программами:

- Элементарная МТ l (сдвиг головки на одну ячейку влево)
- Элементарная МТ r (сдвиг головки на одну ячейку вправо)
- Элементарные МТ λ, a 1, . . ., a p (запись соответствующего знака на ленту).

  Пример, диаграмм Тьюринга:

Символ	Действие	Диаграмма
$\boldsymbol{R}$	$[(\lambda)w\lambda\rangle \overset{*}{\Rightarrow} [\lambda w(\lambda)\lambda\rangle$	
$\boldsymbol{L}$	$[\lambda w(\lambda)\lambda\rangle \overset{*}{\Rightarrow} [(\lambda)w\lambda\rangle$	
$\mathfrak{R}$	$[(\lambda)w_1\lambda w_2\lambda\ldots\lambda w_n\lambda\rangle \overset{*}{\Rightarrow}$ $\overset{*}{\Rightarrow} [\lambda w_1\lambda w_2\lambda\ldots\lambda w_n(\lambda)\lambda\rangle$	
$\mathfrak{L}$	$[\lambda w_1\lambda w_2\lambda\ldots\lambda w_n(\lambda)\lambda\rangle \overset{*}{\Rightarrow}$ $\overset{*}{\Rightarrow} [(\lambda)w_1\lambda w_2\lambda\ldots\lambda w_n\lambda\rangle$	
$\boldsymbol{K}$	$[\lambda w(\lambda)\lambda\rangle \overset{*}{\Rightarrow} [\lambda w\lambda w(\lambda)\lambda\rangle$	
$\boldsymbol{W}_r$	$[(\lambda)w\lambda\rangle \overset{*}{\Rightarrow} [\lambda\lambda\ldots(\lambda)\rangle$	
$\boldsymbol{W}_l$	$[\lambda w(\lambda)\rangle \overset{*}{\Rightarrow} [(\lambda)\lambda\ldots\lambda\rangle$	
$\boldsymbol{V}$	$[\lambda w_1\lambda w_2(\lambda)\lambda\rangle \overset{*}{\Rightarrow} [\lambda w_2(\lambda)\lambda\rangle$	
$\boldsymbol{I}$	$[\lambda w(\lambda)\lambda\rangle \overset{*}{\Rightarrow} [\lambda w\lambda w^{-1}(\lambda)\lambda\rangle$	
$\boldsymbol{K}_n$	$[\lambda w_1\lambda w_2\lambda\ldots\lambda w_n(\lambda)\lambda\rangle \overset{*}{\Rightarrow}$ $\overset{*}{\Rightarrow} [\lambda w_1\lambda w_2\lambda\ldots\lambda w_n\lambda w_1(\lambda)\lambda\rangle$	

17. Моделирование машин Тьюринга.

Определение Рассмотрим 2 МТ $T = (A, Q, P, q_0)$ и $T' = (A', Q', P', q_0')$.
Будем говорить, что машина T' моделирует машину T, и обозначать это $T' \simeq T$, если выполняются следующие условия:
1. Указан способ кодирования знаков (букв)
алфавита A знаками (буквами или слова − ми) алфавита A' $C: A \to A'$;
2. Каждому состоянию $q \in Q$ машины T поставлено в соответствие некоторое состояние $q' \in Q'$ машины T', т. е. определено отображение $Q \to Q'$.
3. Если C_0 — начальная конфигурация машины T, то ее образ C_0' — начальная конфигурация машины T';
4. Если машина T из начальной конфигурации C 0 после конечного числа тактов (переходов от одной конфигурации к другой) останавливается в конфигурации C_k, то машина T' из начальной конфигурации C_0', являющейся образом C_0, после конечного числа тактов также останавливается в конфигурации C_k', являющейся образом C_k;
5. Если из начальной конфигурации C_0 машина T пробегает (конечную или беско − нечную) последовательность конфигураций $C_0, C_1, \ldots, C_k, \ldots$, то каждая последовательность конфигураций пробегаемая машиной T' из начальной конфигурации C_0', содержит в качесте подпоследовательности последовательность конфигураций $C_0', C_1', \ldots, C_k', \ldots$, где
C_i' − образы конфигураций C_i, машины T.

18. Эквивалентность программ и диаграмм.

Теорема 2.2 .6 . Каждой программе P , задающей МТ $T = (A , Q , P , q_0)$, можно эффективным образом сопоставить диаграмму D , образованную символами элементарных МТ , так , чтобы МТ , определяемая этой диаграммой , моделировала бы машину T .

Доказательство

2.2 .6 .1

Построение диаграммы D . Каждой строке $(q , a , v , q') \in P$ поставим в соответствие на диаграмме символы $\cdot v \cdot$. Для $v = s$ ничего ставить не надо , ведь пустые действия на диаграмме не отображаются . В этом случае окончанием работы машины , определяемой диаграммой , будет правая точка предыдущего элемента диаграммы .

Поставим еще одну точку $(\bullet)$, соответствующую начальному состоянию диаграммы , и соединим ее стрелками → с левыми точками всех символов v , соответствующих строкам вида $(q_0 , a_i , v , q) \in P$. Для каждой пары строк $(q , a_k , v , q') \in P$ и $(q' , a_j , v' , q'') \in P$ соединим стрелкой → правую точку символа v , соответствующего строке $(q , ak , v , q') \in P$, с левой точкой символа v' , соответствующего строке $(q' , a_j , v' , q'') \in P$ и надпишем над стрелкой букву a_j .

В полученной таким образом диаграмме произведем необходимые упрощения , описанные в конце п . 2 .2 .3 . Диаграмма D построена .

2.2.6.2

Доказательство того, что машина TD , определяемая диаграммой D, моделирует машину T. Для этого достаточно показать, что выполнены условия 17 вопроса . Условие (1) выполняется потому, что алфавиты машин T и T_D совпадают. Состояниям машины TD соответствуют точки на диаграмме D. По построению диаграммы каждому состоянию q машины T соответствует одна или две точки на диаграмме D. В последнем случае сопоставим состоянию q правую точку соответствующего символа на диаграмме. Таким образом, условие (2) выполнено. Условия (3)–(5) выполнены, так как диаграмма D определяет ту же последовательность элементарных действий, что и программа P , за исключением того, что у машины T_D могут оказаться лишние «пустые» действия, соответствующие переходам от левых точек символов v к правым.

19. Эквивалентность диаграмм и программ.

Теорема 2.2 .8 . Каждой диаграмме Тьюринга D может быть эффективным образом сопоставлена программа P так, что МТ, описываемая этой программой, моделирует МТ, описываемую диаграммой D.

Доказательство

Построение программы P . Заменим на диаграмме D все символы неэлементарных МТ их диаграммами, продолжая такую замену до тех пор, пока на диаграмме не останутся только обозначения элементарных МТ. В полученной диаграмме перенумеруем одинаковые символы элементарных МТ, снабдив их числовыми индексами. Перенумеруем также все точки, соответствующие выходным состояниям. Каждому символу элементарной МТ сопоставим программу этой машины из п. 2 .2 .3, пометив каждое состояние этой программы дополнительными индексами j и v , где v совпадает с названием соответствующей элементарной МТ, а j — номер, присвоенный этой элементарной МТ. Например, элементарной машине rj (j-е вхождение элементарной МТ r в диаграмме) будет сопоставлена программа:

$$\begin{array}{ll} (q_0^{v,j}, \lambda, r, q_1^{v,j}) & (q_1^{v,j}, \lambda, s, q_1^{v,j}) \\ (q_0^{v,j}, a_1, r, q_1^{v,j}) & (q_1^{v,j}, a_1, s, q_1^{v,j}) \\ \cdots & \cdots \\ (q_0^{v,j}, a_p, r, q_1^{v,j}) & (q_1^{v,j}, a_p, s, q_1^{v,j}) \end{array}$$

Точке с индексом j сопоставим программу вида

$$\begin{array}{l} (q_j, \lambda, s, q_j) \\ (q_j, a_1, s, q_j) \end{array}$$

$$\begin{array}{l} \cdots \\ (q_j, a_p, s, q_j) \end{array}$$

Соединим все программы, сопоставленные символам элементарных МТ и точкам, в одну и произведем в полученной программе следующие изменения:

• если на диаграмме символы элементарных МТ v_j и v_k соединены стрелкой a → , то строку $\left(q_1^{(v,j)}, a, s, q_1^{(v,j)}\right)$ заменим на $\left(q_1^{(v,j)}, a, a, q_0^{(v,k)}\right)$, т . е . останов машины v_j заменим на перезапись буквы рабочей ячейки с последующим переходом прямо в начальное состояние машины v_k , минуя конечное состояние машины v_j ;
• если на диаграмме между символами элементарных МТ v_j и v_k стоит точка , то проведем вышеуказанную замену для всех строк вида $\left(q_i, a, s, q_i\right)$, тем самым предотвращается остановка машины на промежуточных этапах , изображаемых точками — « пустыми » машинами ;
• введем новое начальное состояние q_0 для генерируемой программы $\left(\text{оно соответствует крайней левой точке диаграммы}\right)$ и добавим к программе пустые командные строки $\left(q_0, \lambda, \lambda, q_0\right), \left(q_0, a_i, a_i, q_0\right)\left(i = 1, 2, \ldots, p\right)$, где a_i — символы элементарных МТ , непосредственно следующих за крайней левой точкой диаграммы . Эти строки являются модифицированной программой элементарной машины « • » , в которой остановка заменена перезаписью буквы в рабочей ячейке , т . е . реализуют холостой ход конструируемой программы ;
• перенумеруем все состояния программы , за исключением нового q_0 произвольным образом , введя сплошную нумерацию состояний . например , в лексикографическом возрастании индексов : $q_i^{(x,j)} \rightarrow q_k$.

Постройте сами такое отображение индексов по схеме l , r , s , λ , ai × N^2 → N .

Проверка условий (1)–(5) определения 2.2 .5

Условие (2) выполняется по построению программы P . Условия (3)–(5) выполняются потому, что обе МТ выполняют над сообщением, записанным на ленте (по построению программы P) одни и те же элементарные действия в одной и той же последовательности (дополнительные элементарные действия имеют вид (q, a, a, q′), т. е. не меняют ситуации на ленте).

20. I теорема Шеннона. Доказательство

Теорема Для любой машины Тьюринга $T = \left(A_p, Q, P, q_0\right)$ с множеством состояний $Q = \left\{q_0, q_1, \ldots, q_s\right\}$ можно эффективным способом построить машину Тьюринга $T' = \left(A_r, \left\{\alpha, \beta\right\}, P', q'_0\right)$, моделирующую машину T и имеющую всего 2 состояния : α , β . Рабочий алфавит A_r содержит $r = 3\left(p+1\right)\left(s+1\right)+p$ букв .

Док-во громоздкое и не очень понятное, строится на «предположим».

Если хотите понять — читайте методичку, там написано более менее.

20. I теорема Шеннона. Доказательство

Теорема Для любой машины Тьюринга $T = (A_p , Q , O , q_0)$ можно эффективным образом построить машину Тьюринга $T'' = (A_1 , Q'' , P'' , q_0'')$, моделирующую машину T и имеющую однобуквенный алфавит $A_1 = \{ | \}$.

ужас, а по факту каждой букве присваиваем количество | по ее номеру и доказываем работу этого через написание диаграмм элементарных действий.

22. Вычислимые функции.

Определение
Функцией из множества L в множество A'_t называется правило , которое каждому слову $u \in L$ ставит в соответствие единнственное слово $w \in A'_t$, называемое значением функции f на слову u .

Определение
Функия $f : (A'_s)^n \rightarrow A'_t$ называется **вычислимой по Тьюрингу** , если сущестует МТ с рабочим алфавитом A_p, содержащим алфавиты A_s и A_t, такая что функция $f_T : (A'_s)^n \rightarrow A'_t$, определяемая этой МТ , совпадает с f $(f_t = f)$.

Определение
Предикат $p : (A'_s)^n \rightarrow \{И , Л\}$ называется **вычислимым по Тьюрингу** , если существует МТ T_p с рабочим алфавитом $A_t = A_s \cup \{И , Л\}$, которая вычисляет p и удовлетворяет слудующим условиям :
1. Если $X = (u_1 , u_2 , ... , u_n) \notin Def (p)$, то машина T_p никогда не останавливается .
2. Если $X = (u_1 , u_2 , ... , u_n) \in Def (p)$, и $p (u_1 , u_2 , ... , u_n) = И$, то машина T_p получает И .
3. Если $X = (u_1 , u_2 , ... , u_n) \in Def (p)$, и $p (u_1 , u_2 , ... , u_n) = Л$, то машина T_p получает Л .

23. Нормированные вычисления. Теорема о нормированной вычислимости. Доказательство

Определение

Функция называется нормированно вычислимой по Тьюрингу , если существует МТ T_f , которая вычисляет f и при это удовлетворяет следующим требованиям :

1. Если $X = (u_1 , u_2 , ... , u_n) \notin \mathrm{Def}\,(f)$, то машина T_f после применения к X никогда не останавливается ;
2. Если $X = (u_1 , u_2 , ... , u_n) \in \mathrm{Def}\,(f)$, то T_f вычисляет значении функции .

Теорема 1 Всякая ВТ – функция является НВТ – функцией , причем соответствующая МТ не использует никаких вспомогательных букв .

Теорема 2 Для любой машины Тьюринга можно эффективным образом построить машину T_N , которая имеет ленту , не ограниченную только с одного конца , и которая моделирует машину .

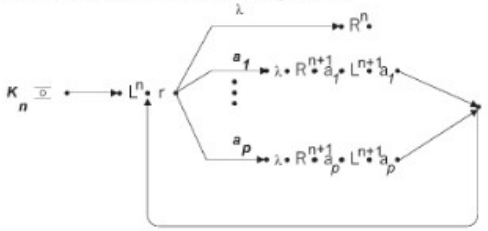
Анрил 7.5 страниц докозательство

24. Теорема о композиции. Доказательство.

▷Из теоремы 4.3.3 следует, что ВТ-функции $g_i\,(i = 1, 2, \ldots, m)$ и h являются НВТ-функциями. Пусть $T_{g_i}\,(i = 1, 2, \ldots, m)$ и T_h — МТ, реализующие нормированное вычисление функций $g_i\,(i = 1, 2, \ldots, m)$ и h соответственно. Построим МТ K_n, реализующую действие

$$|\lambda w_1 \lambda w_2 \ldots \lambda w_n(\lambda)\lambda > \overset{K_n}{\Longrightarrow}{}^* |\lambda w_1 \lambda w_2 \ldots \lambda w_n \lambda w_1(\lambda)\lambda >$$

Машина K_n описывается диаграммой



Докажем, что машина T, определяемая диаграммой

$$T = .T_{g_1} K^n_{n+1} T_{g_2} \ldots K^n_{n+1} T_{g_m} K_{(m-1)n+m} K_{(m-2)n+m} \ldots K_{(m-m)n+m} T_h$$

вычисляет функцию f, т.е. реализует действие

$$|\lambda u_1 \lambda u_2 \ldots \lambda u_n(\lambda)\lambda > \overset{T}{\Longrightarrow}{}^* |\lambda z \lambda f(u_1, u_2, \ldots, u_n)(\lambda)\lambda >,$$

где $z \in \tilde{A}^*_p$ (A_p — рабочий алфавит машин T_{g_i} $(i = 1, 2, \ldots, m)$ T_h, K_{n+1}, $K_{(m-j)n+m}$ $(j = 1, 2, \ldots, m)$, T; $A_s \subseteq A_p, A_t \subseteq A_p, A_r \subseteq A_p$).

Имеем:

$|\lambda u_1 \lambda u_2 \ldots \lambda u_n(\lambda)\lambda > \overset{T_{g_i}}{\Longrightarrow}{}^* |\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1(\lambda)\lambda > \overset{K^n_{n+1}}{\Longrightarrow}{}^*$

$|\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1 \lambda u_1 \lambda u_2 \lambda \ldots \lambda u_n(\lambda)\lambda > \overset{T_{g_2}}{\Longrightarrow}{}^*$

$|\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1 \lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_2(\lambda)\lambda > \Longrightarrow{}^* \ldots \overset{T_{g_m}}{\Longrightarrow}{}^*$

$|\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1 \lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_2 \ldots \lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_m(\lambda)\lambda > \overset{K_{(m-1)n+m}}{\Longrightarrow}{}^*$

$|\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1 \lambda u_1 \ldots \lambda u_n \lambda g_2 \ldots \lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_m \lambda g_1(\lambda)\lambda > \Longrightarrow{}^+$

$\ldots \overset{K_{(m-m)n+m}}{\Longrightarrow}{}^* |\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1 \lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_2 \ldots \lambda u_1 \lambda u_2 \ldots$

$\lambda u_n \lambda g_m \lambda g_1 \lambda g_2 \ldots \lambda g_m(\lambda)\lambda > \overset{T_h}{\Longrightarrow}{}^* |\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_1 \lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_2 \ldots$

$\lambda u_1 \lambda u_2 \ldots \lambda u_n \lambda g_m \lambda g_1 \lambda g_2 \ldots \lambda g_m \lambda h(g_1, g_2, \ldots, g_m)(\lambda)\lambda >=$

$|\lambda z \lambda f(u_1, u_2, \ldots, u_n)(\lambda)\lambda >$

так как по условию $h(g_1(u_1, u_2, \ldots u_n), g_2(u_1, u_2, \ldots u_n), \ldots, g_m(u_1, u_2, \ldots u_n)) = f(u_1, u_2, \ldots, u_n)$.

Теорема доказана. □

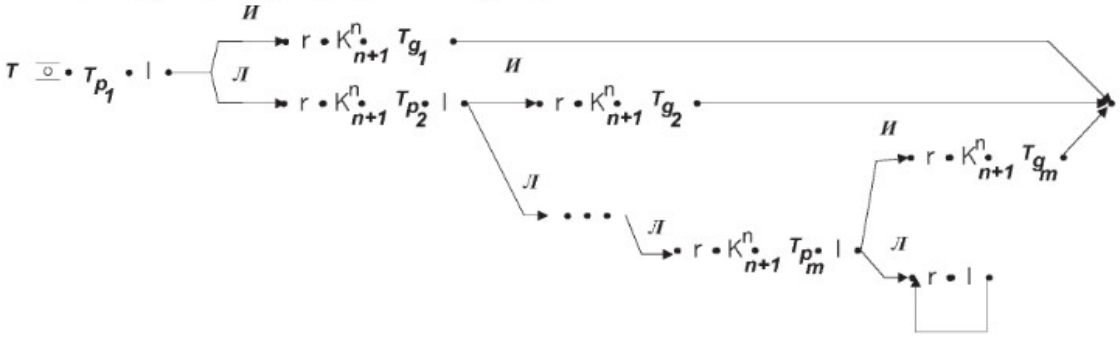
25. Теорема о ветвлении. Доказательство.

Теорема (О ветвлении) Пусть n – местная функция определяется равенством :

$$f(u_1,u_2,\ldots,u_n)=\begin{cases} g_1(u_1,u_2,\ldots,u_n), \text{если } p_1(u_1,u_2,\ldots,u_n)=\text{И} \\ g_2(u_1,u_2,\ldots,u_n), \text{если } p_2(u_1,u_2,\ldots,u_n)=\text{И} \\ \vdots \\ g_m(u_1,u_2,\ldots,u_n), \text{если } p_m(u_1,u_2,\ldots,u_n)=\text{И} \end{cases}$$

где $g_i : (A'_s)^n \to A'_t \ (i = 1, 2, \ldots, n)$ – ВТ функции , а $p_i : (A'_s)^n \to \{И, Л\}$ – Вычислимые предикат . Тогда f является ВТ – функцией , причем МТ , вычисляющая функцию f может быть эффективно построена из МТ , вычисляющих g_i и предикаты $p_i \ (i = 1, 2, \ldots, m)$.

▷Пусть T_{g_i} и T_{p_i} $(i = 1, 2, \ldots, m)$ — МТ, реализующие нормированное вычисление функций g_i и предикатов p_i $(i = 1, 2, \ldots, m)$ соответственно (возможность эффективного построения таких МТ для любых ВТ-функций следует из теоремы 2.4.3). Машина T, вычисляющая функцию f, определяется диаграммой:



В случае, когда все p_i $(i = 1, 2, \ldots, m)$ имеют значение Л, функция f не определена; поэтому в этом случае предусмотрено, чтобы машина T никогда не останавливалась. Проверка того, что машина T действительно вычисляет функцию f, производится аналогично соответствующей проверке, проведенной при доказательстве теоремы 2.5.1. Теорема доказана. □

2.6.3 Следствие из теоремы о ветвлении

Следствие. Пусть n-местная функция $f : (A_s^*)^n \to A_t^*$ определяется равенством:

$$f(u_1,u_2,\ldots,u_n)=\begin{cases} g(u_1,u_2,\ldots,u_n), \text{если } p(u_1,u_2,\ldots,u_n)=\text{И} \\ h(u_1,u_2,\ldots,u_n), \text{если } p(u_1,u_2,\ldots,u_n)=\text{Л} \end{cases}$$

где $g : (A_s^*)^n \to A_t^*$ и $h : (A_s^*)^n \to A_t^*$ — ВТ-функции, а $p : (A_s^*)^n \to \{И, Л\}$ — ВТ-предикат. Тогда f является ВТ-функцией, причем МТ, вычисляющая функцию f, может быть эффективно построена из МТ, вычисляющих функции g и h и предикат p.

26. Теорема о цикле. Доказательство.

Теорема Пусть $g : (A'_s)^n \to A'_s$ – ВТ функция , вычисляемая машиной T_g , а $p : (A'_s)^n \to \{И , Л\}$ – ВТ предикат , вычислимый машиной T_p. Тогда функция $f : (A'_s) \to A'_s$, определяемая вычислительным процессом .

$g = g_0$, $g_0 \ni A'_s$

$f(u_1 , u_2 , ... , u_n) = g(u_1 , u_2 , ... , u_{(j-1)} , g , u_{(j+1)} , ... , u_n)$, пока $p(u_1 , u_2 , ... , u_n) = И$, тоже является ВТ – функцией , причем МТ , вычисляющая функцию f , может быть эффективно построена из T_g , T_p .

27. Схемы машин Тьюринга. Нисходящая разработка.

Систематическое проектирование и разработка МТ и диаграмм МТ

Если мы хотим составить алгоритм , вычисляющий функцию f , ее надо представить в виде более простых функций . Расписываем машину T , на много маленьких T_j.

Сводим большую программу на много маленьких .

Далее в методичке идет пример Евклида по вычислению НОД .

Определние Понятие схемы машины Тьюринга определим следующим образом , отражающим нисходящий процесс конструирования :

1. символ • составляет схему .
2. если S_1 , S_2 , . . . , S_n — схемы , то их последовательная композиция S_1 , S_2 . . . S_n является схемой ;
3. если C_1 , C_2 , . . . , C_n — схемы МТ , вычисляющих предикаты P_1 , P_2 , . . . , P_n соответственно , и S 1 , S 2 , . . . , Sn — схемы , то и конструкции являютсся схемами .
4. символы элементарных машин Тьюринга составляют схемы ;
5. никакие другие объекты схемами не являются .

28. Теорема Бойма-Джакопини-Миллса. Доказательство

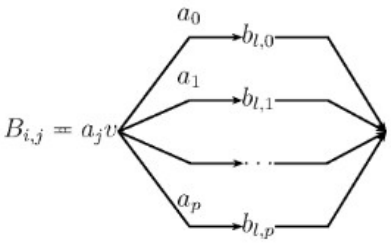
Теорема Бойма – Якопини – Миллса Для любой машины Тьюринга T можно эффективно построить машину Тьюринга S , которая является структурной , т . е . диаграмма является схемой , и которая моделирует машину T .

Доказательство этой теоремы состоит в преобразовании каждой части диаграммы машины T в одну из трех структур, приведенных в определении схемы.

Пусть машина T имеет алфавит $A = \{a_1, \ldots, a_p\}$ и набор состояний $Q = \{q_0, \ldots, q_s\}$. Тогда алфавит машины T' будет содержать помимо алфавита A еще $(p+1)(s+1)$ знаков $b_{i,j}$.

Каждой команде из программы машины T', не являющейся терминальной (q_i, a_j, s, q_i), поставим в соответствие некоторую структурную машину B_{ij} по следующему правилу:

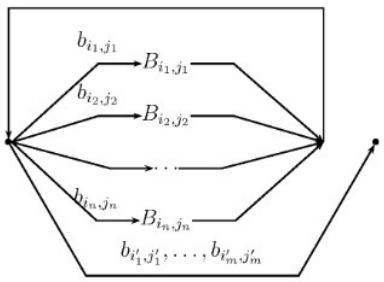
1. Если команда машины T имеет вид (q_i, a_j, a_k, q_l), то ей соответствует машина перезаписи буквы
$$B_{i,j} = b_{l,k}.$$
2. Если команда машины T имеет вид (q_i, a_j, v, q_l), $v \in \{l, r\}$, то ей соответствует машина движения



3. Если команда машины T имеет вид (q_i, a_j, s, q_l) $(l \neq i)$, то ей соответствует стоп-машина
$$B_{i,j} = b_{l,j}.$$

Таким образом, каждый знак $b_{i,j}$ соответствует определяющей паре (q_i, a_j) исходной машины T и неявно содержит в себе информацию о номере состояния, в котором она находится. Таким образом, сгенерирована матрица инциденций B_{ij}, отражающая наличие связей всех состояний (точкомест) в диаграмме.

Теперь можно построить диаграмму машины T':



Здесь введена сплошная (от 1 до n) нумерация для пар (i, j) таких, что пара (q_i, a_j) определяет некоторую команду машины T, не являющуюся терминальной. Аналогично была введена сплошная (от 1 до m) нумерация для пар (i, j), для которых пара (q_i, a_j) определяет терминальную команду машины T. Сам порядок, в котором перенумерованы пары, не важен.

Исходя из построенной диаграммы, можно заметить, что машина T' является структурной, т. к. структурными являются все машины $B_{i,j}$.

Осталось показать, что машина T' моделирует машину T.

Конфигурации
$$\lfloor \lambda a_{l_1} \ldots \underset{q_i}{a_{l_k}} \ldots a_{l_n} \lambda \rangle$$
машины T поставим в соответствие конфигурацию
$$\lfloor \lambda a_{l_1} \ldots \underset{q^*}{b_{i,l_k}} \ldots a_{l_n} \lambda \rangle$$

машины T', где состояние q^* может быть любым. Тогда начальной конфигурацией машины T' будет

$$[\lambda a_{l_1} \dots \underset{q}{b_{i_{\text{НАЧ}},l_{k_{\text{НАЧ}}}}} \dots a_{l_n} \lambda\rangle,$$

соответствующая начальной конфигурации машины T (здесь q — начальное состояние T').

Рассмотрим выполнение нетерминальной команды (q_i, a_j, v, q_l), где $v \in A \cup \{\lambda\} \cup \{l, r, s\}$, а $i \neq l$.

Если $v \in A \cup \{\lambda\}$, то машина T' просто заменяет в своей рабочей ячейке знак $b_{i,j}$ на $b_{l,k}$. Если $v \in \{l, r\}$, то машина машина T' сначала считает из ячейки букву $b_{i,j}$, восстановит в ней букву a_j, сдвигается в направлении v и в новой рабочей ячейке записывает букву $b_{l,j'}$, если до этого в ней была записана буква $a_{j'}$. Если $v = s$, то машина T' заменяет в рабочей ячейке букву $b_{i,j}$ на $b_{l,j}$. Наконец, если команда (q_i, a_j, v, q_l) является терминальной ($l = i, v = s$), то машина T' сразу же выходит из цикла и завершает работу.

Таким образом, теорема Бойма-Якопини-Миллса доказана.

29. Универсальная машина Тьюринга. Построение*.

Определение . Машина Тьюринга , способная выполнить любой алгоритм , сымитировав работу соотвествующей МТ , называется **универсальной** .

В общем случае под универсальностью понимают способность некоторой системы моделировать работу любой другой системы, когда описание последней подается ей на вход в закодированном виде. Это определение не является вполне строгим, поскольку не указывается, какие способы кодирования являются допустимыми.

Алгоритм работы УМТ

Напомним, что универсальной машиной Тьюринга (УМТ) для алфавита A называется такая машина U , на которой может быть промоделирована любая МТ над алфавитом A.

На ленту УМТ записывается программа моделируемой МТ и исходное сообщение моделируемой МТ. УМТ по состоянию и текущему знаку МТ находит на своей ленте команду моделируемой МТ, выясняет, какое действие нужно выполнить, и выполняет его.

Представление программы УМТ

Моделируемая машина T :

• рабочий алфавит A_p ,

• состояния q_0 , q_1 , ... , q_s ,

• команды $(q_i , a_j , v_{ij} , q_k)$, где i , k = 0 , ... , n ; j = 1 , ... , p ;

$v_{ij} \in \{a_1 , a_2 , ... , a_p , l , r , h\}$ Универсальная машина U :

• алфавитBp = b_1 , b_2 , ... , b_p ,

• дополнительные знаки $\{l , r , h , + , - , O\}$

Для команды (q_i, a_j, v_{ij}, q_k) запись выглядит как

$$\begin{cases} b_j v_{ij} +^{k-i}, \text{если } k > i; \\ b_j v_{ij} O, \text{если } k = i; \\ b_j v_{ij} -^{i-k}, \text{если } k < i. \end{cases}$$

$+^{k-i}$ означает знак $+$, повторённый $k - i$ раз.

Слово-программа моделируемой машины: $cw_0 cw_1 \dots cw_n \S$, где w_i — слово с записью подряд всех правил состояния q_i.

- Слова групп команд разных состояний отделяются друг от друга маркером c.
- Вся программа заканчивается маркером §.

Интерпретация моделируемой МТ

1. Поиск команды для выполнения

(a) Запоминаем обозреваемую букву a_j размножением состояний

(b) Заменяем знак a_j на его зеркальную пару b_j

(c) Ищем слово w_i , содержащее запись команды

(d) Ищем запись команды для знака a_j

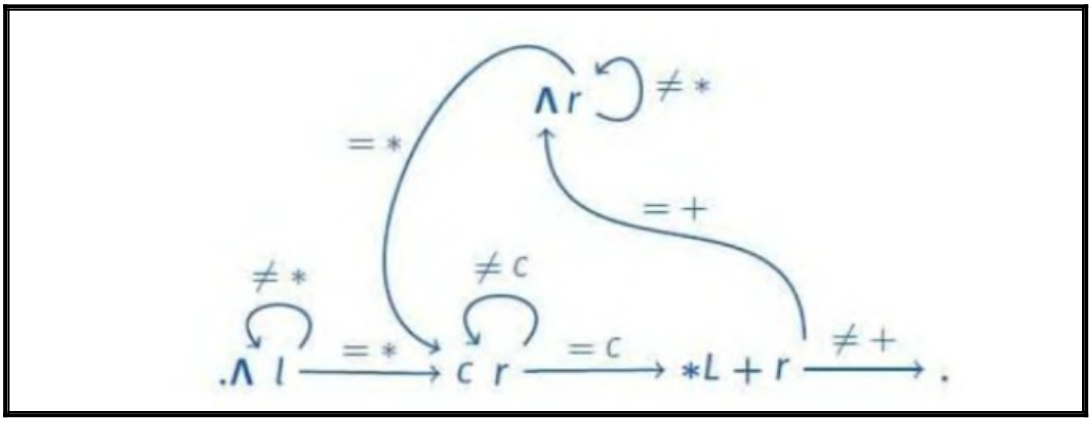
$$cw_0 cw_1 \dots \overbrace{*b_0 v_{i0} + + \dots \underbrace{b_j}_{\dagger} v_{ij} - - - \dots}^{w_j} cw_s \S a_{t1} a_{t2} \dots \overbrace{b_j}^{a_j} \dots a_{tw} \Lambda\Lambda \dots$$

2. Изменение текущего состояния моделируемой МТ

(a) Сдвигаемся на один знак вправо , пропуская v_{ij}

(b) По описанию сдвига пропускаем соответствующее количество маркеров c и ставим знак текущего состояния *

(c) Возвращаемся к знаку описания vij действия



3. Выполнение действия моделируемой МТ

(a) Ищем ячейку ленты , на которой находится головка моделируемой МТ

(b) Выполняем считанное ранее действие , запись или сдвиг , Если при сдвиге головка попала на символ § , отделяющий программу моделируемой МТ от обрабатываемого сообщения , то моделируемая МТ зашла за левый край ленты ,

4. Переход на выполнение нового такта

Остановка моделируемой МТ

Если при сдвиге маркера текущего состояния (шаг 2.2) происходит переход на символ §, то следующим состоянием будет являться состояние остановки моделируемой МТ.

В таком случае УМТ нужно выполнить действие моделируемой машины, а потом остановиться.